

Document Title	SWS_CANInterface: 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_CANInterface	5
1.1	Specification Item ECUC_CanIf_00248	5
1.2	Specification Item ECUC_CanIf_00525	7
1.3	Specification Item ECUC_CanIf_00528	13
1.4	Specification Item ECUC_CanIf_00531	18
1.5	Specification Item ECUC_CanIf_00683	24
1.6	Specification Item ECUC_CanIf_00685	29
1.7	Specification Item ECUC_CanIf_00789	35
1.8	Specification Item ECUC_CanIf_00791	41
1.9	Specification Item ECUC_CanIf_00819	46
1.10	Specification Item ECUC_CanIf_00845	52
1.11	Specification Item SWS_CANIF_00040	53
1.12	Specification Item SWS_CANIF_00142	55
1.13	Specification Item SWS_CANIF_00168	60
1.14	Specification Item SWS_CANIF_00294	61
1.15	Specification Item SWS_CANIF_00312	69
1.16	Specification Item SWS_CANIF_00316	75
1.17	Specification Item SWS_CANIF_00318	81
1.18	Specification Item SWS_CANIF_00323	83
1.19	Specification Item SWS_CANIF_00329	89
1.20	Specification Item SWS_CANIF_00334	96
1.21	Specification Item SWS_CANIF_00339	102
1.22	Specification Item SWS_CANIF_00344	109
1.23	Specification Item SWS_CANIF_00349	115
1.24	Specification Item SWS_CANIF_00356	121
1.25	Specification Item SWS_CANIF_00382	128
1.26	Specification Item SWS_CANIF_00401	134
1.27	Specification Item SWS_CANIF_00407	140
1.28	Specification Item SWS_CANIF_00413	147
1.29	Specification Item SWS_CANIF_00422	153
1.30	Specification Item SWS_CANIF_00429	160
1.31	Specification Item SWS_CANIF_00432	165
1.32	Specification Item SWS_CANIF_00437	171
1.33	Specification Item SWS_CANIF_00438	178
1.34	Specification Item SWS_CANIF_00439	183
1.35	Specification Item SWS_CANIF_00440	188
1.36	Specification Item SWS_CANIF_00449	194
1.37	Specification Item SWS_CANIF_00450	201
1.38	Specification Item SWS_CANIF_00455	206
1.39	Specification Item SWS_CANIF_00456	213

1.40	Specification Item	SWS_CANIF_00542	218
1.41	Specification Item	SWS_CANIF_00543	223
1.42	Specification Item	SWS_CANIF_00544	228
1.43	Specification Item	SWS_CANIF_00550	233
1.44	Specification Item	SWS_CANIF_00551	238
1.45	Specification Item	SWS_CANIF_00556	243
1.46	Specification Item	SWS_CANIF_00558	249
1.47	Specification Item	SWS_CANIF_00559	254
1.48	Specification Item	SWS_CANIF_00560	259
1.49	Specification Item	SWS_CANIF_00563	264
1.50	Specification Item	SWS_CANIF_00564	269
1.51	Specification Item	SWS_CANIF_00661	274
1.52	Specification Item	SWS_CANIF_00678	281
1.53	Specification Item	SWS_CANIF_00688	284
1.54	Specification Item	SWS_CANIF_00690	290
1.55	Specification Item	SWS_CANIF_00691	296
1.56	Specification Item	SWS_CANIF_00692	301
1.57	Specification Item	SWS_CANIF_00695	306
1.58	Specification Item	SWS_CANIF_00696	311
1.59	Specification Item	SWS_CANIF_00697	316
1.60	Specification Item	SWS_CANIF_00698	321
1.61	Specification Item	SWS_CANIF_00700	327
1.62	Specification Item	SWS_CANIF_00703	332
1.63	Specification Item	SWS_CANIF_00709	338
1.64	Specification Item	SWS_CANIF_00711	345
1.65	Specification Item	SWS_CANIF_00712	350
1.66	Specification Item	SWS_CANIF_00720	355
1.67	Specification Item	SWS_CANIF_00724	358
1.68	Specification Item	SWS_CANIF_00737	363
1.69	Specification Item	SWS_CANIF_00765	370
1.70	Specification Item	SWS_CANIF_00793	375
1.71	Specification Item	SWS_CANIF_00794	381
1.72	Specification Item	SWS_CANIF_00795	386
1.73	Specification Item	SWS_CANIF_00797	391
1.74	Specification Item	SWS_CANIF_00798	396
1.75	Specification Item	SWS_CANIF_00799	402
1.76	Specification Item	SWS_CANIF_00800	408
1.77	Specification Item	SWS_CANIF_00801	413
1.78	Specification Item	SWS_CANIF_00803	418
1.79	Specification Item	SWS_CANIF_00804	423
1.80	Specification Item	SWS_CANIF_00807	429
1.81	Specification Item	SWS_CANIF_00811	435
1.82	Specification Item	SWS_CANIF_00818	441

1.83	Specification Item SWS_CANIF_00822	448
1.84	Specification Item SWS_CANIF_00823	454
1.85	Specification Item SWS_CANIF_00824	460
1.86	Specification Item SWS_CANIF_00826	465
1.87	Specification Item SWS_CANIF_00827	470
1.88	Specification Item SWS_CANIF_00858	475
1.89	Specification Item SWS_CANIF_00870	480
1.90	Specification Item SWS_CANIF_00879	486
1.91	Specification Item SWS_CANIF_00882	492
1.92	Specification Item SWS_CANIF_00887	492
1.93	Specification Item SWS_CANIF_00891	499
1.94	Specification Item SWS_CANIF_00893	504
1.95	Specification Item SWS_CANIF_00894	505
1.96	Specification Item SWS_CANIF_00895	507
1.97	Specification Item SWS_CANIF_00900	508
1.98	Specification Item SWS_CANIF_00901	509
1.99	Specification Item SWS_CANIF_91002	516

1 SWS_CANInterface

1.1 Specification Item ECUC_CanIf_00248

Trace References:

none

Content:

Container Name	CanIfTxPduCfgCanIfTxPduCfg		
Description	<p>This container contains the configuration (parameters) of a transmit CAN L-PDU. It has to be configured as often as a transmit CAN L-PDU is needed.</p> <p>The SHORT-NAME of "CanIfTxPduConfig" container represents the symbolic name of Transmit L-PDU.</p> <p>This L-SDU consumes a meta data item of type CAN_ID_32.</p>		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included parameters:

Included Parameters	
Parameter Name	SWS Item ID
CanIfTxPduCanId	ECUC_CanIf_00592
CanIfTxPduCanIdMask	ECUC_CanIf_00823
CanIfTxPduCanIdType	ECUC_CanIf_00590
CanIfTxPduId	ECUC_CanIf_00591
CanIfTxPduPnFilterPdu	ECUC_CanIf_00773
CanIfTxPduReadNotifyStatus	ECUC_CanIf_00589
CanIfTxPduTriggerTransmit	ECUC_CanIf_00840
CanIfTxPduTruncation	ECUC_CanIf_00845
CanIfTxPduType	ECUC_CanIf_00593
CanIfTxPduUserTriggerTransmitName	ECUC_CanIf_00842
CanIfTxPduUserTxConfirmationName	ECUC_CanIf_00528
CanIfTxPduUserTxConfirmationUL	ECUC_CanIf_00527
CanIfTxPduBufferRef	ECUC_CanIf_00831
CanIfTxPduRef	ECUC_CanIf_00603

Included containers:

Included Containers		
Container Name	Multiplicity	Scope / Dependency
CanIfTTTxFrameTriggering	0..1	<p>CanIfTTTxFrameTriggering is specified in the SWS TTCAN Interface and defines Frame trigger for TTCAN transmission.</p> <p>This container is only included and valid if TTCAN is supported by the controller, enabled (see CanIfSupportTTCAN, ECUC_CanIf_00675), and a joblist is used.</p>

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76711: [CanIf] Discarding parts of a PDU without an error is no good idea

Problem description:

SWS_CANIF_00894 specifies that CanIf shall just ignore parts of a passed PDU that exceed the configured size. This is not a very good idea in most cases.

I see only one use case where such an approach is valid: If a gateway takes in also extended PDUs, but routes only the original size. But I doubt this is a very good use case.

I see two possible solutions for this problem:

1. Return E_NOT_OK if the PDU exceeds the configured size.
2. Return E_OK but throw a runtime error.

Agreed solution:

Add new parameter to "CanIfTxPduCfg" as following:

Name CanIfTxPduTruncation

Description Enables/disables truncation of PDUs that exceed the configured size.

Multiplicity 1

Type EcucBooleanParamDef

Default Value true

Post-Build Variant Value true

Value Configuration Class

Pre-compile time X VARIANT-PRE-COMPILE

Link time X VARIANT-LINK-TIME

Post-build time X VARIANT-POST-BUILD

Scope / Dependency scope: ECU

~SWS_CANIF_00894 When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is enabled,

CanIf shall transmit as much data as possible and discard the rest.

+SWS_CANIF_????? When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is disabled, CanIf shall report the runtime error CANIF_E_TXPDU_LENGTH_EXCEEDED and return E_NOT_OK without further actions.

7.27.2 Runtime Errors create Table with entry

Type: Message length was exceeding the maximum length.

Related error code: CANIF_E_TXPDU_LENGTH_EXCEEDED

Value: 90

–Last change on issue 76711 comment 22–

BW-C-Level:

Application	Specification	Bus
1	3	3

1.2 Specification Item ECUC_CanIf_00525

Trace References:

none

Content:

Name	CanIfDispatchUserCtrlBusOffNameCanIfDispatchCfg.CanIfDispatchUserCtrlBusOffName		
Parent Container	CanIfDispatchCfg		
Description	This parameter defines the name of <User_ControllerBusOff>. This parameter depends on the parameter CANIF_USERCTRLBUSOFF_UL. If CANIF_USERCTRLBUSOFF_UL equals CAN_SM the name of <User_ControllerBusOff> is fixed. If CANIF_USERCTRLBUSOFF_UL equals CDD, the name of <User_ControllerBusOff> is selectable.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	–		
maxLength	32		
minLength	1		
regularExpression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Scope / Dependency	scope: ECU dependency: CANIF_DISPATCH_USERCTRLBUSOFF_CanIfDispatchUserCtrlBus OffUL		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
 Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
 Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
 What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
 Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
 Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirma-

tionUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModelIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModelIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If

CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.3 Specification Item ECUC_CanIf_00528

Trace References:

none

Content:

Name	CanIfTxPduUserTxConfirmationNameCanIfTxPduCfg.CanIfTxPduUserTxConfirmationName		
Parent Container	CanIfTxPduCfg		
Description	<p>This parameter defines the name of the <User_TxConfirmation>.</p> <p>This parameter depends on the parameter CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduUserTxConfirmationUL.</p> <p>If CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduUserTxConfirmationUL equals CAN_TP, CAN_NM, PDUR, XCP, CAN_TSYN, J1939NM or J1939TP, the name of the <User_TxConfirmation> is fixed. If CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduUserTxConfirmationUL equals CDD, the name of the <User_TxConfirmation> is selectable.</p>		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
maxLength	32		
minLength	1		
regularExpression	-		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Scope / Dependency	scope: ECU		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-

patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-

gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.4 Specification Item ECUC_CanIf_00531

Trace References:

none

Content:

Name	CanIfDispatchUserValidateWakeupEventNameCanIfDispatchCfg.CanIfDispatchUserValidateWakeupEventName
Parent Container	CanIfDispatchCfg
Description	This parameter defines the name of <User_ValidateWakeupEvent>. This parameter depends on the parameter CANIF_USERVALIDATEWAKEUPEVENT_UL. CANIF_USERVALIDATEWAKEUPEVENT_UL equals ECUM the name of <User_ValidateWakeupEvent> is fixed. CANIF_USERVALIDATEWAKEUPEVENT_UL equals CDD, the name of <User_ValidateWakeupEvent> is selectable. If parameter CANIF_WAKEUP_CHECK_VALIDATION_API is disabled, no <User_ValidateWakeupEvent> API can be configured.
Multiplicity	0..1
Type	EcucFunctionNameDef

Default value	-		
maxLength	32		
minLength	1		
regularExpression	-		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Scope / Dependency	scope: ECU dependency: CANIF_WAKEUP_CHECK_VALIDATION_API, CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_CanIfDispatchUserValidateWakeupEvent UL		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_Delnit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
 Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
 Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
 What will be the initial value of CanId then? Does the CanId initial value matter at all

in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.5 Specification Item ECUC_CanIf_00683

Trace References:

none

Content:

Name	CanIfDispatchUserCtrlModeIndicationNameCanIfDispatchCfg.CanIfDispatchUserCtrlModeIndicationName		
Parent Container	CanIfDispatchCfg		
Description	This parameter defines the name of <User_ControllerModelIndication>. This parameter depends on the parameter CANIF_USERCTRLMODEINDICATION_UL. If CANIF_USERCTRLMODEINDICATION_UL equals CAN_SM the name of <User_ControllerModelIndication> is fixed. If CANIF_USERCTRLMODEINDICATION_UL equals CDD, the name of <User_ControllerModelIndication> is selectable.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	–		
maxLength	32		
minLength	1		
regularExpression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: CANIF_DISPATCH_USERCTRLMODEINDICATION_CanIfDispatchUserCtrlModeIndicationUL		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOff-
 Name
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUser-
 ConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDis-
 patchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-
 patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> Can-
 IfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDis-
 patchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> Can-
 IfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
 Name":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
 CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
 the API <User_TxConfirmation>() has to be configured via parameter
 CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():

If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.6 Specification Item ECUC_CanIf_00685

Trace References:

none

Content:

Name	CanIfDispatchUserTrcvModeIndicationNameCanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationName
Parent Container	CanIfDispatchCfg

Description	This parameter defines the name of <User_TrvcModeIndication>. This parameter depends on the parameter CANIF_USERTRCVMODEINDICATION_UL. If CANIF_USERTRCVMODEINDICATION_UL equals CAN_SM the name of <User_TrvcModeIndication> is fixed. If CANIF_USERTRCVMODEINDICATION_UL equals CDD, the name of <User_TrvcModeIndication> is selectable.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
maxLength	32		
minLength	1		
regularExpression	-		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Scope / Dependency	scope: ECU dependency: CANIF_DISPATCH_USERTRCVMODEINDICATION_CanIfDispatch UserTrcvModeIndicationUL		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.

Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"

(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.7 Specification Item ECUC_CanIf_00789

Trace References:

none

Content:

Name	CanIfDispatchUserClearTrcvWufFlagIndicationNameCanIfDispatchCfg.CanIfDispatchUserClearTrcvWufFlagIndicationName		
Parent Container	CanIfDispatchCfg		
Description	This parameter defines the name of <User_ClearTrcvWufFlagIndication>. If CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_CanIfDispatchUserClearTrcvWufFlagIndicationUL equals CAN_SM the name of <User_ClearTrcvWufFlagIndication> is fixed. If it equals CDD, the name is selectable. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	–		
maxLength	–		
minLength	–		
regularExpression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	

Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Scope / Dependency	scope: ECU dependency: CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_CanIfDispatchUserClearTrcvWufFlagIndicationUL, CANIF_PUBLIC_PN_SUPPORT		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
 Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
 Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
 What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
 Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
 Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirma-

tionUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModelIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModelIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

- ~SWS_CANIF_00800
- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If

CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.8 Specification Item ECUC_CanIf_00791

Trace References:

none

Content:

Name	CanIfDispatchUserCheckTrcvWakeFlagIndicationNameCanIfDispatchCfg.CanIfDispatchUserCheckTrcvWakeFlagIndicationName		
Parent Container	CanIfDispatchCfg		
Description	This parameter defines the name of <User_ClearCheckTrcvWufWakeFlagIndication>. If CANIF_DISPATCH_USERCHECKTRCVWAKEFLAGINDICATION_CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CANIF_PUBLIC_PN_SUPPORT CanIfPublicPnSupport equals False, this parameter shall not be configurable.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	-		
maxLength	-		
minLength	-		
regularExpression	-		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Scope / Dependency	scope: ECU dependency: CANIF_DISPATCH_USERCHECKTRCVWAKEFLAGINDICATION_UL, CANIF_PUBLIC_PN_SUPPORT		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.
The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.
Should this callback use CANIF_E_PARAM_CONTROLLER error code?
Or should each relevant API use the same error code?
–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===
if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).
 # # endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

- CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName
- CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
- CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
- CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
- CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
- CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
- CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
- CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
- CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
- CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
- CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
- CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
- CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
- CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> Can-

IfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
 Name":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
 CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
 the API <User_TxConfirmation>() has to be configured via parameter
 CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
 If CanIfDispatchUserTrcvModeIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.9 Specification Item ECUC_CanIf_00819

Trace References:

none

Content:

Name	CanIfDispatchUserConfirmPnAvailabilityNameCanIfDispatchCfg.CanIfDispatchUserConfirmPnAvailabilityName
Parent Container	CanIfDispatchCfg
Description	This parameter defines the name of <User_ConfirmPnAvailability>. If CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_CanIfDispatchUserConfirmPnAvailability UL equals CAN_SM the name of <User_ConfirmPnAvailability> is fixed. If it equals CDD, the name is selectable. If CANIF_PUBLIC_PN_SUPPORT equals False, this parameter shall not be configurable.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
maxLength	–
minLength	–

regularExpression	-		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	-	
Scope / Dependency	scope: ECU dependency: CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_CanIf DispatchUserConfirmPnAvailabilityUL, CANIF_PUBLIC_PN_SUPPORT		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
 Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
 Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
 What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDis-

patchUserCtrlModeIndicationName
 CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
 Name":
 ~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.10 Specification Item ECUC_CanIf_00845

Trace References:

none

Content:

Name	CanIfTxPduTruncationCanIfTxPduCfg.CanIfTxPduTruncation		
Parent Container	CanIfTxPduCfg		
Description	Enables/disables truncation of PDUs that exceed the configured size.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76711: [CanIf] Discarding parts of a PDU without an error is no good idea

Problem description:

SWS_CANIF_00894 specifies that CanIf shall just ignore parts of a passed PDU that exceed the configured size. This is not a very good idea in most cases.

I see only one use case where such an approach is valid: If a gateway takes in also extended PDUs, but routes only the original size. But I doubt this is a very good use case.

I see two possible solutions for this problem:

1. Return E_NOT_OK if the PDU exceeds the configured size.
2. Return E_OK but throw a runtime error.

Agreed solution:

Add new parameter to "CanIfTxPduCfg" as following:

Name CanIfTxPduTruncation
 Description Enables/disables truncation of PDUs that exceed the configured size.
 Multiplicity 1
 Type EcucBooleanParamDef
 Default Value true
 Post-Build Variant Value true
 Value Configuration Class
 Pre-compile time X VARIANT-PRE-COMPILE
 Link time X VARIANT-LINK-TIME
 Post-build time X VARIANT-POST-BUILD
 Scope / Dependency scope: ECU

~SWS_CANIF_00894 When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is enabled, CanIf shall transmit as much data as possible and discard the rest.

+SWS_CANIF_????? When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is disabled, CanIf shall report the runtime error CANIF_E_TXPDU_LENGTH_EXCEEDED and return E_NOT_OK without further actions.

7.27.2 Runtime Errors create Table with entry
 Type: Message length was exceeding the maximum length.
 Related error code: CANIF_E_TXPDU_LENGTH_EXCEEDED
 Value: 90
 –Last change on issue 76711 comment 22–

BW-C-Level:

Application	Specification	Bus
1	3	3

1.11 Specification Item SWS_CANIF_00040

Trace References:

none

Content:

API function	Description
Can_SetControllerMode	This function performs software triggered state transitions of the CAN controller State machine.
Can_Write	This function is called by CanIf to pass a CAN message to Can Drv for transmission.
Det_ReportRuntimeError	Service to report runtime errors. If a callout has been configured then this callout shall be called.
SchM_Enter_CanIf_<ExclusiveArea>	Invokes the SchM_Enter function to enter a module local exclusive area.
SchM_Exit_CanIf_<ExclusiveArea>	Invokes the SchM_Exit function to exit an exclusive area.

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

1.12 Specification Item SWS_CANIF_00142

Trace References:

SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00361

Content:

Module	Imported Type
Can_GeneralTypes	CanTrcv_TrcvModeType
	CanTrcv_TrcvWakeupModeType
	CanTrcv_TrcvWakeupReasonType
	Can_ControllerStateType
	Can_ErrorStateType
	Can_HwHandleType
	Can_HwType
	Can_IdType
	Can_PduType
	Can_ReturnType
Can_StateTransitionType	
ComStack_Types	IcomConfigIdType
	IcomSwitch_ErrorType
	PduIdType
	PduInfoType
EcuM	EcuM_WakeupSourceType

Module	Imported Type
Std_Types	Std_ReturnType
	Std_VersionInfoType

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77329: [Can] Removed type Can_StateTransitionType still used in Can_SetControllerMode function definition

Problem description:

With CP_R4.3.0 the type definition for Can_StateTransitionType has been removed from SWS_CANDriver.

In the function definition of Can_SetControllerMode (SWS_Can_00230) this type is still used in the signature syntax.

Within mentioned function calls (e.g. "Can_SetControllerMode(CAN_CS_STARTED)" in SWS_Can_00261) the enumeration values of type Can_ControllerStateType are used.

It shall be clarified if the type of the second function parameter in the syntax entry of SWS_Can_00230 shall be changed from Can_StateTransitionType to Can_ControllerStateType.

Agreed solution:

=== CanDrv ===

~SWS_Can_00230: Change Can_StateTransitionType to Can_ControllerStateType.

~SWS_Can_00222: Remove Can_StateTransitionType from Imported types

=== CanIf ===

~SWS_CANIF_00142: Remove Can_StateTransitionType from Can_GeneralTypes

Replace Can_StateTransitionType with Can_ControllerStateType of function call Can_SetControllerMode in chapters

- 9.11 (figure and table)

- 9.13 (figure)

–Last change on issue 77329 comment 12–

BW-C-Level:

Application	Specification	Bus
1	4	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

Problem description:

While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:

- * type change from enumeration to extra_literal
- * Remove range element CAN_OK
- * Remove range element CAN_NOT_OK
- * Assign value "0x02" to range element "CAN_BUSY"
- * Description: Overlaid return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode

Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)

Return value:

Std_ReturnType

E_OK: request accepted

E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup

Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)

Return value:

Std_ReturnType

E_OK: API call has been accepted

E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write

Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)

Return value:

Std_ReturnType

E_OK: Write command has been accepted

E_NOT_OK: development error occurred

CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that

can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_Can_00048

~SWS_Can_00089

7.11.5 Return Values

~SWS_Can_00198

~SWS_Can_00199

~SWS_Can_00200

~SWS_Can_00216

~SWS_Can_00217

~SWS_Can_00218

~SWS_CAN_00219

~SWS_CAN_00505

~SWS_CAN_00506

~SWS_Can_00212

=== CanIf ===

Adapt API Can_Write() to new signature:

* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"

* Figure 9.1 "Transmission request with a single CAN Driver"

* Figure 9.2 "Transmission request with multiple CAN Drivers"

* Figure 9.5 "Transmit confirmation with buffering"

* Figure 9.6 "Transmit Cancelation"

* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:

* Figure 9.11: Start CAN network

* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

Note between SWS_CANIF_00162 and SWS_CANIF_00319

Table in chapter 9.7 Trigger Transmit Request

Table in chapter 9.11 Start CAN network

=== CanTrcv ===

Adapt API Can_SetControllerMode() to new signature:

- * 9.3 De-Initialization (SPI Synchronous)
- * 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===

Adapt API Can_CheckWakeup() to new signature:

- * Figure 42 CAN controller wake up by interrupt
- * Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===

Adapt API Can_Write() to new signature:

- * Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:

- * Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_TtCanIf_00071

=== TTCanDrv ===

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

- ~SWS_TtCan_00014
- ~SWS_TtCan_00018
- ~SWS_TtCan_00022
- ~SWS_TtCan_00026
- ~SWS_TtCan_00059
- ~SWS_TtCan_00078
- ~SWS_TtCan_00112

=== XCP ===

Adapt API Can_Write() to new signature:

- * Figure 5: Xcp on Can Transmit
- Last change on issue 77952 comment 22-

BW-C-Level:

Application	Specification	Bus
1	4	1

1.13 Specification Item SWS_CANIF_00168

Trace References:

none

Content:

If the Data Length Check rejects a received L-PDU (see SWS_CANIF_00026), CanIf shall report **development runtime** error code CANIF_E_INVALID_DATA_LENGTH to the Det_ReportRuntimeError() service of the DET module.

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

1.14 Specification Item SWS_CANIF_00294

Trace References:

none

Content:

API function	Description
Can_CheckWakeup	This function checks if a wakeup has occurred for the given controller.
Can_SetBaudrate	This service shall set the baud rate configuration of the CAN controller. Depending on necessary baud rate modifications the controller might have to reset.
Can_SetIcomConfiguration	This service shall change the Icom Configuration of a CAN controller to the requested one.
CanNm_RxIndication	Indication of a received PDU from a lower layer communication interface module.
CanNm_TxConfirmation	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
CanSM_CheckTransceiverWakeFlagIndication	This callback function indicates the CanIf_CheckTrcvWakeFlag API process end for the notified CAN Transceiver.
CanSM_ClearTrcvWufFlagIndication	This callback function shall indicate the CanIf_ClearTrcvWuf Flag API process end for the notified CAN Transceiver.
CanSM_ConfirmPnAvailability	This callback function indicates that the transceiver is running in PN communication mode.
CanSM_ControllerBusOff	This callback function notifies the CanSM about a bus-off event on a certain CAN controller, which needs to be considered with the specified bus-off recovery handling for the impacted CAN network.
CanSM_ControllerModeIndication	This callback shall notify the CanSM module about a CAN controller mode change.
CanSM_CurrentIcomConfiguration	This service shall inform about the change of the Icom Configuration of a CAN network.

API function	Description
CanSM_TransceiverModeIndication	This callback shall notify the CanSM module about a CAN transceiver mode change.
CanTp_RxIndication	Indication of a received PDU from a lower layer communication interface module.
CanTp_TxConfirmation	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
CanTrcv_CheckWakeFlag	Requests to check the status of the wakeup flag from the transceiver hardware.
CanTrcv_CheckWakeup	Service is called by underlying CANIF in case a wake up interrupt is detected.
CanTrcv_GetBusWuReason	Gets the wakeup reason for the Transceiver and returns it in parameter Reason.
CanTrcv_GetOpMode	Gets the mode of the Transceiver and returns it in OpMode.
CanTrcv_SetOpMode	Sets the mode of the Transceiver to the value OpMode.
CanTrcv_SetWakeupMode	Enables, disables or clears wake-up events of the Transceiver according to TrcvWakeupMode.
Det_ReportError	Service to report development errors.
EcuM_ValidateWakeupEvent	After wakeup, the ECU State Manager will stop the process during the WAKEUP VALIDATION state/sequence to wait for validation of the wakeup event. This API service is used to indicate to the ECU Manager module that the wakeup events indicated in the sources parameter have been validated.
J1939Nm_RxIndication	Indication of a received PDU from a lower layer communication interface module.
J1939Nm_TxConfirmation	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
J1939Tp_RxIndication	Indication of a received PDU from a lower layer communication interface module.
J1939Tp_TxConfirmation	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
PduR_CanIfRxIndication	Indication of a received PDU from a lower layer communication interface module.
PduR_CanIfTxConfirmation	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
Xcp_CanIfRxIndication	Indication of a received PDU from a lower layer communication interface module.
Xcp_CanIfTxConfirmation	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76404: [Det] Clarifications on runtime errors

Problem description:

There are several uncertainties/problems in the SWS DET:

1. According to SWS_Det_00180, the callouts should have the same signatures as the corresponding DET functions, but they are void(void) (SWS_Det_00181, SWS_Det_00184, SWS_Det_00187).

2. Section 8.2.3.1 does not describe how the instance ID is passed to DET.
3. Configuration of header files for all three error type callouts are missing.
4. Why does the development error callout reside in DetNotification, while the other two callouts reside in DetGeneral?
5. The limitation in section 4.1 regarding "supervisor mode" does not really make sense. It is assumed that the DET is ignorant regarding the call context, and the software receiving DET callbacks (like DLT or the implementers of the callouts) need to take care of resolving the calling context, if necessary (e.g. in multi-core environments).
6. SWS_Det_00302 defines several runtime errors. But apart from DET_E_CANNOT_REPORT, it is unclear in which situation these errors could be reported by DET: For errors reported by BSW, the DET has no means to validate anything that could lead to such an error. And for SWCs, the modeling already takes care that DET_E_WRONG_MODULE and DET_E_WRONG_INSTANCE cannot occur, while the other two errors can also not be checked by DET without further configuration.
7. Det_ReportTransientFault (SWS_Det_01003) shall return the return value of a configured callout. But what shall happen if more than one callout exists, and the return different values?
8. SWS_Det_00052: The only API that can result in DET_E_PARAM_POINTER is Det_GetVersionInfo (as the error description mentions correctly). Please reformulate this requirement and move it to section 8.1.3.6 "Det_GetVersionInfo".
–Last change on issue 76404 comment 13–

Agreed solution:

1.
~change SWS_Det_00181/184/187 such that signatures match the APIs
~Figures 3,5, and 7 to be corrected (return missing)
5. remove from 4.1. the sentence: "It is assumed that the whole Basic Software runs in supervisor mode or the switch to supervisor mode is done by a system call within the error reporting function of the DET module."
6. remove SWS_Det_00302 and SWS_Det_00303 and all included errors
7. change SWS_Det_01003 (Return Value-Part only): "Std_ReturnType" If no callout exists it shall return E_OK, otherwise it shall return the value of the configured callout. In case several callouts are configured the logical or (sum) of the callout return values shall be returned. Rationale: since E_OK=0, E_OK will be only returned if all are E_OK, and for multiple error codes there is a good chance to detect several of them.
8. change SWS_Det_00052 from "in case a null pointer error occurs." to "in case a null pointer error occurs in Det_GetVersionInfo." Do not move the requirement, since otherwise the section 7.7 would be empty, but add the following sentence to 8.1.3.6: "In case a null pointer is passed, DET_E_PARAM_POINTER is returned,

see SWS_Det_00052."

–Last change on issue 76404 comment 30–

BW-C-Level:

Application	Specification	Bus
1	4	1

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_Delnit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
the name of the function has to be configured via parameter
CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
If CanIfDispatchUserTrcvModeIndicationUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
<User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
If CanIfDispatchUserValidateWakeupEventUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
If CanIfDispatchUserConfirmPnAvailabilityUL is set to
CDD, the name of the service has to be configurable via parameter
CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
to CDD, the name of the service has to be configurable via parameter
CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
to CDD, the name of the service has to be configurable via parameter
CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.15 Specification Item SWS_CANIF_00312

Trace References:

none

Content:

Caveats of CanIf_SetControllerMode():

- The CAN Driver module must be initialized after Power ON.
- The CAN Interface module must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== Ehtlf ===

~8.3 Function definitions

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

Ethlf_Init, Ethlf_GetVersionInfo

~8.3.2 Ethlf_SetControllerMode

-[SWS_Ethlf_00038]

~8.3.3 Ethlf_GetControllerMode

-[SWS_Ethlf_00044]

~8.3.8 Ethlf_CheckWakeup

-[SWS_Ethlf_00249]

~8.3.9 Ethlf_GetPhysAddr

-[SWS_Ethlf_00066]

~8.3.10 Ethlf_SetPhysAddr

-[SWS_Ethlf_00138]

~8.3.11 Ethlf_UpdatePhysAddrFilter

-[SWS_Ethlf_00144]

~8.3.12 Ethlf_GetPortMacAddr

-[SWS_Ethlf_00195]

~8.3.13 Ethlf_GetArITable

-[SWS_Ethlf_00201]

8.3.14 Ethlf_GetBufferLevel

-[SWS_Ethlf_00207]

~8.3.15 Ethlf_GetDropCount

-[SWS_Ethlf_00213]

~8.3.16 Ethlf_StoreConfiguration

-[SWS_Ethlf_00218]

~8.3.17 Ethlf_ResetConfiguration

-[SWS_Ethlf_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 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 Commu-

nication 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_CANIF_00316

Trace References:

none

Content:

Caveats of CanIf_GetControllerMode:

- The CanDrv must be initialized after Power ON.
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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 CanIfRxSduld, 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_ControllerModelIndication()
- SWS_CANIF_00709 CanIf_TrvcModelIndication()
- 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_ControllerModelIndication>()

=== 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_ModelIndication(): 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_CANIF_00318

Trace References:

none

Content:

The service `CanIf_Transmit()` shall map the parameters of the data structure call `Can_Write()` with the hardware transmit handle corresponding to the provided `TxPduId` and a `Can_PduType` structure where:

- the L-SDU handle (`swPduHandle` is set to the `CanTxPduId`) refers to (`CanID`, `HTH/HRH` of the CAN Controller) used in the corresponding `CanIf_TxConfirmation()` call
- and the length is set to the value provided as `PduInfoPtr` which specifies length and data pointer of the Transmit Request `Ptr->SduLength`, possibly reduced according to `SWS_CANIF_00894`
- `id` is set to the CAN ID associated with the `TxPduId`
- `sdu` is set to the pointer provided as `PduInfoPtr->SduDataPtr`

to the corresponding `CanDrv` and call the function `Can_Write(Hth, *PduInfo)`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77098: [CanIf] Explicitly state which length to use for Transmission

Problem description:

Description/Motivation:

[SWS_CANIF_00894]

" When `CanIf_Transmit()` is called with `PduInfoPtr->SduLength` exceeding the maximum length of the PDU referenced by `TxPduId`, `CanIf` shall transmit as much data as possible and discard the rest."

[SWS_CANIF_00894] explicitly says what to do if `PduInfoPtr->SduLength` exceeds the configured length. But I found no explicit requirement what to do otherwise (i.e. `PduInfoPtr->SduLength` is not exceeding the maximum length of the PDU referenced by `TxPduId`)

Was there already a decision?

Agreed solution:

Replace `SWS_CANIF_00318` by:

[SWS_CANIF_00318] `CanIf_Transmit()` shall call `Can_Write()` with the hardware transmit handle corresponding to the provided `TxPduId` and a `Can_PduType` structure where:

- * swPduHandle is set to the CanTxPduId used in the corresponding CanIf_TxConfirmation() call
 - * length is set to the value provided as PduInfoPtr->SduLength, possibly reduced according to SWS_CANIF_00894
 - * id is set to the CAN ID associated with the TxPduId
 - * sdu is set to the pointer provided as PduInfoPtr->SduDataPtr
- Last change on issue 77098 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.18 Specification Item SWS_CANIF_00323

Trace References:

none

Content:

Caveats of CanIf_Transmit():

- During the call of this API 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.
- CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()
 "

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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_TrcvModelIndication()
 -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_ControllerModelIndication>()

=== 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_ModelIndication(): 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

1.19 Specification Item SWS_CANIF_00329

Trace References:

none

Content:

Caveats of CanIf_ReadRxPduData() :

During the call of this API 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.

This API must not be shall not be used for CanIfRxSduld, which are defined to receive multiple CAN-Ids (range reception).

CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization 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.

=== Ehtlf ===

~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_GetArTable
-[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 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

1.20 Specification Item SWS_CANIF_00334

Trace References:

none

Content:

Caveats of CanIf_ReadTxNotifyStatus(): CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable

-[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 CanIfRxSduld, 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_ModelIndication(): 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

1.21 Specification Item SWS_CANIF_00339

Trace References:

none

Content:

Caveats of CanIf_ReadRxNotifStatus():

- CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization 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.

=== EthIf ===

~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_GetArTable

-[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 `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

1.22 Specification Item SWS_CANIF_00344

Trace References:

none

Content:

Caveats of CanIf_SetPduMode(): CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable

-[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 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_TrvcModeIndication()

-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_ModelIndication(): 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

1.23 Specification Item SWS_CANIF_00349

Trace References:

none

Content:

Caveats of CanIf_GetPduMode(): CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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 CanIfRxSduld, 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_ControllerModelIndication()
- SWS_CANIF_00709 CanIf_TrvcModelIndication()
- 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_ControllerModelIndication>()

=== 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_ModelIndication(): 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

1.24 Specification Item SWS_CANIF_00356

Trace References:

none

Content:

Caveats of CanIf_SetDynamicTxId() :

CanIf must be initialized after Power ON.

This function may shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

```
"
...
[SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
"
```

Such requirement 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.

=== EthIf ===

~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_GetArITable
-[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_TrvcModeIndication()

-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

1.25 Specification Item SWS_CANIF_00382

Trace References:

SRS_Can_01126

Content:

If an L-PDU is requested to be transmitted via a PDU channel mode (refer to [REF sec_3a_PduChannelModes]), which equals CANIF_OFFLINE, the CanIf shall report the **development runtime** error code CANIF_E_STOPPED to the Det_Report**Error Runtime Error()** service of the DET and CanIf_**TranmsitTransmit()** shall return E_NOT_OK.

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

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"

(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

- ~SWS_CANIF_00800
- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.26 Specification Item SWS_CANIF_00401

Trace References:

none

Content:

Caveats of CanIf_CheckWakeup():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"
 ...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable
-[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 CanIfRxSduld, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329

Note: During the call 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_ModelIndication(): 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

1.27 Specification Item SWS_CANIF_00407

Trace References:

none

Content:

Caveats of CanIf_CheckValidation():

- The CAN Interface module must be initialized after Power ON.
- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The corresponding CAN controller and transceiver must be switched on via Can Trcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_Set ControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

```
"
...
[SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
"
```

Such requirement 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.

```
=== EthIf ===
~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_GetArITable

-[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 CanIfRxSduld, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329

Note: During the call 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_ModelIndication(): 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

1.28 Specification Item SWS_CANIF_00413

Trace References:

none

Content:

Caveats of CanIf_TxConfirmation():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"
 ...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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 CanIfRxSduld, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329

Note: During the call 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_ModelIndication(): 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

1.29 Specification Item SWS_CANIF_00422

Trace References:

none

Content:

Caveats of CanIf_RxIndication():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable
-[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_TrvcModeIndication()

-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

1.30 Specification Item SWS_CANIF_00429

Trace References:

SRS_BSW_00323

Content:

If parameter ControllerId of CanIf_ControllerBusOff() has an invalid value, CanIf shall report development error code CANIF_E_PARAM_CONTROLLER_CONTROLLERID to the Det_ReportError service of the DET module, when CanIf_ControllerBusOff() is called.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some

cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
 Name":

~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
 CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
 the API <User_TxConfirmation>() has to be configured via parameter
 CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
 If CanIfDispatchUserTrcvModeIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():

If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of `<User_CheckTrcvWakeFlagIndication>`. If `CanIfDispatchUserCheckTrcvWakeFlagIndicationUL` equals `CAN_SM` the name of `<User_CheckTrcvWakeFlagIndication>` is fixed. If it equals `CDD`, the name is selectable. If `CanIfPublicPnSupport` equals `False`, this parameter shall not be configurable.
 –Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.31 Specification Item SWS_CANIF_00432

Trace References:

none

Content:

Caveats of `CanIf_ControllerBusOff()`:

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The `CanIf` must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()
 "

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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_TrcvModelIndication()
 -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_ControllerModelIndication>()

=== 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_ModelIndication(): 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

1.32 Specification Item SWS_CANIF_00437

Trace References:

none

Content:

Caveats of <User_TxConfirmation>(): The call context is either on interrupt level (interrupt mode) or on task level (polling mode).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

```
"
...
[SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
"
```

Such requirement 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.

=== EthIf ===

~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_GetArItable
-[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_TrvcModeIndication()
- 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 tem-

plate <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

1.33 Specification Item SWS_CANIF_00438

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): The upper layer module, which provides this callback service, has to be configured by **CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL** (see ECUC_CanIf_00527). If no upper layer modules are configured for transmit confirmation using <User_TxConfirmation>(), no transmit confirmation is executed.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
 Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.
 –Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.34 Specification Item SWS_CANIF_00439

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If **CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL** is set to PDUR, **CANIF_TXPDU_USERTXCONFIRMATION_NAME** **CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName** must be PduR_CanIfTxConfirmation.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.
- 3) SWS_CanTrcv_91001: CanIf_DeInit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"

(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

- ~SWS_CANIF_00800
- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.35 Specification Item SWS_CANIF_00440

Trace References:

none

Content:

Caveats of <User_RxIndication>:

- Until this service 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.
- CanDrv module must be initialized after Power ON.
- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable

-[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_ModelIndication(): 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

1.36 Specification Item SWS_CANIF_00449

Trace References:

none

Content:

Caveats of <User_ControllerBusOff>():

- The CanDrv must be initialized after Power ON.
- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- This callback service is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller.
- Before re-initialization/restart during BusOff recovery is executed this callback service is performed only once in case of multiple BusOff events at CAN Controller.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

```
"
...
[SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
"
```

Such requirement 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.

=== EthIf ===

~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_GetArITable

-[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 CanIfRxSduld, 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_ModelIndication(): 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

1.37 Specification Item SWS_CANIF_00450

Trace References:

none

Content:

Configuration of <User_ControllerBusOff>(): The upper layer module which provides this callback service has to be configured by **CANIF_DISPATCH_USERCTRLBUSOFF_CanIfDispatchCfg.CanIfDispatchUserCtrlBusOffUL** (see ECUC_CanIf_00547).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized

L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():

If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in

SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.38 Specification Item SWS_CANIF_00455

Trace References:

none

Content:

Caveats of <User_ValidateWakeupEvent>:

- The CanDrv must be initialized after Power ON.
- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- This callback service is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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 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_ControllerModelIndication>()

=== 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_ModelIndication(): 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

1.39 Specification Item SWS_CANIF_00456

Trace References:

none

Content:

Configuration of `<User_ValidateWakeupEvent>()`: The upper layer module which provides this callback service has to be configured by `CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_CanIfDispatchCfg.CanIfDispatchUserValidateWakeupEventUL` (see ECUC_CanIf_00549), but:

- If no upper layer modules are configured for wake up notification using `<User_ValidateWakeupEvent>()`, no wake up notification needs to be configured. `CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_CanIfDispatchCfg.CanIfDispatchUserValidateWakeupEventUL` needs not to be configured.
- If wake up is not supported (`CANIF_CTRL_WAKEUP_SUPPORT` and `CANIF_TRCV_WAKEUP_SUPPORT` equal `FALSE`, see ECUC_CanIf_00637, ECUC_CanIf_00606), `CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_CanIfDispatchCfg.CanIfDispatchUserValidateWakeupEventUL` is not configurable.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".

The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DelNit` API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.

Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
 Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
 Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
 CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
 APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.
 The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.
 Should this callback use CANIF_E_PARAM_CONTROLLER error code?
 Or should each relevant API use the same error code?
 –Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is

```
Can_ReturnType (E_OK -> CAN_OK).  
# # endif
```

```
=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===  
CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName  
CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL  
CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName  
CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL  
CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL  
CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName  
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL  
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName  
CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL  
CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName  
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL  
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName  
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL  
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName  
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL  
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName
```

```
=== Others ===
```

```
Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"  
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":  
~SWS_CANIF_00800
```

~SWS_CANIF_00801
~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801
~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to

CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DelNit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:
 This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.
 –Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.40 Specification Item SWS_CANIF_00542

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): The name of the API <User_TxConfirmation>() which is called by CanIf shall be configured for CanIf by parameter **CANIF_TXPDU_USERTXCONFIRMATION_NAME** CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName (see ECUC_CanIf_00528).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: CanIf_DelNit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the trans-

mit request
 service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).
 # # endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===
 CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName
 CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
the API <User_TxConfirmation>() has to be configured via parameter
CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>():
If CanIfDispatchUserCtrlModelIndicationUL is set to CDD
the name of the function has to be configured via parameter
CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>():
If CanIfDispatchUserTrcvModelIndicationUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
<User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
If CanIfDispatchUserValidateWakeupEventUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be

configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.41 Specification Item SWS_CANIF_00543

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If **CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL** is set to CAN_NM, **CANIF_TXPDU_USERTXCONFIRMATION_NAME** **CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName** must be CanNm_TxConfirmation.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request

from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).
endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===
CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName
CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter

CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.42 Specification Item SWS_CANIF_00544

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If **CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL** is set to J1939TP, **CANIF_TXPDU_USERTXCONFIRMATION_NAME** **CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName** must be J1939Tp_TxConfirmation.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called". The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-

patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-

gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.43 Specification Item SWS_CANIF_00550

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL is set to CAN_TP, CANIF_TXPDU_USERTXCONFIRMATION_NAME CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName must be CanTp_TxConfirmation.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
 Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
 Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
 Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
 CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
 APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.
 The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.
 Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUser-

ConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.44 Specification Item SWS_CANIF_00551

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If **CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL** is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter **CANIF_TXPDU_USERTXCONFIRMATION_NAME**. The function parameter has to be of type **PduIdTypeCanIfTxPduCfg.CanIfTxPduUserTxConfirmationName**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeUpEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDis-

```

patchUserValidateWakeupEventName
CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOff-
Name
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUser-
ConfirmPnAvailabilityUL
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDis-
patchUserConfirmPnAvailabilityName
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-
patchUserClearTrcvWufFlagIndicationUL
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> Can-
IfDispatchUserClearTrcvWufFlagIndicationName
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDis-
patchUserCheckTrcvWakeFlagIndicationUL
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> Can-
IfDispatchUserCheckTrcvWakeFlagIndicationName
  
```

=== Others ===

```

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
~SWS_CANIF_00800
~SWS_CANIF_00801
~SWS_CANIF_00803
  
```

```

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
Name":
~SWS_CANIF_00801
~SWS_CANIF_00803
  
```

```

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
CanTrcv_CheckWakeFlag() shall be called.
  
```

```

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
the API <User_TxConfirmation>() has to be configured via parameter
CanIfTxPduUserTxConfirmationName.
  
```

```

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
the name of the function has to be configured via parameter
CanIfDispatchUserCtrlModeIndicationName.
  
```

~[SWS_CANIF_00698] Configuration of <User_TrvcModelIndication>():
 If CanIfDispatchUserTrvcModelIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrvcModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.45 Specification Item SWS_CANIF_00556

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If `CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL` is set to XCP, `CANIF_TXPDU_USERTXCONFIRMATION_NAME` `CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName` must be `Xcp_CanIfTxConfirmation`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

- 1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: `CanIf_DelNit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?
- 5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.
- 6) Figure 9.6: There is no API `Can_CancelTransmit`.
- 7) SWS_CANIF_00382:
Please correct API name: `CanIf_Tranmsit` -> `CanIf_Transmit`
Besides, this error code is not defined: `CANIF_E_STOPPED`.
- 8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
the name of the function has to be configured via parameter
CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
If CanIfDispatchUserTrcvModeIndicationUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
<User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
If CanIfDispatchUserValidateWakeupEventUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
If CanIfDispatchUserConfirmPnAvailabilityUL is set to
CDD, the name of the service has to be configurable via parameter
CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
to CDD, the name of the service has to be configurable via parameter
CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
to CDD, the name of the service has to be configurable via parameter
CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.46 Specification Item SWS_CANIF_00558

Trace References:

none

Content:

Configuration of <User_ControllerBusOff>(): The name of the API <User_ControllerBusOff>() which will be called by CanIf shall be configured for CanIf by parameter **CANIF_DISPATCH_USERCTRLBUSOFF_NAME** **CanIfDispatchCfg.CanIfDispatchUserCtrlBusOffName** (see ECUC_CanIf_00525).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserC-

trlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():

If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.47 Specification Item SWS_CANIF_00559

Trace References:

none

Content:

Configuration of <User_ControllerBusOff>(): If `CANIF_DISPATCH_USERCTRLBUSOFF_CanIfDispatchCfg.CanIfDispatchUserCtrlBusOffUL` is set to `CAN_SM`, `CANIF_DISPATCH_USERCTRLBUSOFF_NAME` `CanIfDispatchCfg.CanIfDispatchUserCtrlBusOffName` must be `CanSM_ControllerBusOff`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

- 1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: `CanIf_DeInit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descrip-

tions like e.g. ECUC_CanIf_00528) ===
 CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName
 CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-

Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModeIndication>(): If CanIfDispatchUserTrvcModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrvcModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set

to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DelNit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not

be configurable.
 –Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.48 Specification Item SWS_CANIF_00560

Trace References:

none

Content:

Configuration of <User_ControllerBusOff>(): If **CANIF_DISPATCH_USERCTRLBUSOFF_CanIfDispatchCfg.CanIfDispatchUserCtrlBusOffUL** is set to CDD the name of the API has to be configured via parameter **CANIF_DISPATCH_USERCTRLBUSOFF_NAME**. The function parameter has to be of type **uint8CanIfDispatchCfg.CanIfDispatchUserCtrlBusOffName**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
 Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
 Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
 CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
 APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.
 The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.
 Should this callback use CANIF_E_PARAM_CONTROLLER error code?
 Or should each relevant API use the same error code?
 –Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is

Can_ReturnType (E_OK -> CAN_OK).
 # # endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800

~SWS_CANIF_00801
~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801
~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to

CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DelNit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:
 This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.
 –Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.49 Specification Item SWS_CANIF_00563

Trace References:

none

Content:

Configuration of <User_ValidateWakeupEvent>(): If **CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT** **IfDispatchCfg.CanIfDispatchUserValidateWakeupEventUL** is set to ECUM, **CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME** **CanIfDispatchCfg.CanIfDispatchUserValidateWakeupEventName** must be EcuM_ValidateWakeupEvent.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInIt API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the trans-

mit request
 service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).
 # # endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME	->	CanIfDispatchUserCtrlModeIndicationName
CANIF_DISPATCH_USERCTRLMODEINDICATION_UL	->	CanIfDispatchUserCtrlModeIndicationUL
CANIF_TXPDU_USERTXCONFIRMATION_NAME	->	CanIfTxPduUserTxConfirmationName
CANIF_TXPDU_USERTXCONFIRMATION_UL	->	CanIfTxPduUserTxConfirmationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_UL	->	CanIfDispatchUserTrcvModeIndicationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME	->	CanIfDispatchUserTrcvModeIndicationName
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL	->	CanIfDispatchUserValidateWakeupEventUL
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME	->	CanIfDispatchUserValidateWakeupEventName
CANIF_DISPATCH_USERCTRLBUSOFF_UL	->	CanIfDispatchUserCtrlBusOffUL
CANIF_DISPATCH_USERCTRLBUSOFF_NAME	->	CanIfDispatchUserCtrlBusOffName
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL	->	CanIfDispatchUserConfirmPnAvailabilityUL
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME	->	CanIfDispatchUserConfirmPnAvailabilityName
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL	->	CanIfDispatchUserClearTrcvWufFlagIndicationUL
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME	->	CanIfDispatchUserClearTrcvWufFlagIndicationName
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL	->	CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME	->	CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

- ~SWS_CANIF_00800
- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
 Name":

- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
 CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
 the API <User_TxConfirmation>() has to be configured via parameter
 CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>():
 If CanIfDispatchUserCtrlModelIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>():
 If CanIfDispatchUserTrcvModelIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be

configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.50 Specification Item SWS_CANIF_00564

Trace References:

none

Content:

Configuration of <User_ValidateWakeupEvent>(): If **CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT** is set to CDD the name of the API has to be configured via parameter **CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME**. The function parameter has to be of type **EcuM_WakeupSourceType** **CanIfDispatchCfg.CanIfDispatchUserValidateWakeupEventName**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized

L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

- ~SWS_CANIF_00800
- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():

If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in

SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.51 Specification Item SWS_CANIF_00661

Trace References:

none

Content:

If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall :

not execute their normal operation

report to the DET (using CANIF_E_UNINIT)

and return E_NOT_OK

unless the CanIf has been initialized with a preceding call of CanIf_Init().

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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 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_ControllerModelIndication>()

=== 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_ModelIndication(): 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

1.52 Specification Item SWS_CANIF_00678

Trace References:

none

Content:

If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return (CAN_NOT_OK / E_NOT_OK) to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

Problem description:

While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:

- * type change from enumeration to extra_literal
- * Remove range element CAN_OK
- * Remove range element CAN_NOT_OK
- * Assign value "0x02" to range element "CAN_BUSY"
- * Description: Overlaid return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode

Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)

Return value:

Std_ReturnType

E_OK: request accepted

E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup

Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)

Return value:

Std_ReturnType

E_OK: API call has been accepted

E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write

Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const
 Can_PduType* PduInfo)

Return value:

Std_ReturnType

E_OK: Write command has been accepted

E_NOT_OK: development error occurred

CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that
 can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_Can_00048

~SWS_Can_00089

7.11.5 Return Values

~SWS_Can_00198

~SWS_Can_00199

~SWS_Can_00200

~SWS_Can_00216

~SWS_Can_00217

~SWS_Can_00218

~SWS_CAN_00219

~SWS_CAN_00505

~SWS_CAN_00506

~SWS_Can_00212

=== CanIf ===

Adapt API Can_Write() to new signature:

* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"

* Figure 9.1 "Transmission request with a single CAN Driver"

* Figure 9.2 "Transmission request with multiple CAN Drivers"

* Figure 9.5 "Transmit confirmation with buffering"

* Figure 9.6 "Transmit Cancelation"

* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:

* Figure 9.11: Start CAN network

* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
 Note between SWS_CANIF_00162 and SWS_CANIF_00319
 Table in chapter 9.7 Trigger Transmit Request
 Table in chapter 9.11 Start CAN network

=== CanTrcv ===

Adapt API Can_SetControllerMode() to new signature:

- * 9.3 De-Initialization (SPI Synchronous)
- * 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===

Adapt API Can_CheckWakeup() to new signature:

- * Figure 42 CAN controller wake up by interrupt
- * Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===

Adapt API Can_Write() to new signature:

- * Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:

- * Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_TtCanIf_00071

=== TTCanDrv ===

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

- ~SWS_TtCan_00014
- ~SWS_TtCan_00018
- ~SWS_TtCan_00022
- ~SWS_TtCan_00026
- ~SWS_TtCan_00059

~SWS_TtCan_00078
 ~SWS_TtCan_00112

=== XCP ===

Adapt API Can_Write() to new signature:

* Figure 5: Xcp on Can Transmit

–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.53 Specification Item SWS_CANIF_00688

Trace References:

none

Content:

Caveats of <User_ControllerModeIndication>():

- The CanDrv must be initialized after Power ON.
- The call context is either on task level (polling mode).
- This callback service is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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 CanIfRxSduld, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329

Note: During the call 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_ModelIndication(): 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

1.54 Specification Item SWS_CANIF_00690

Trace References:

none

Content:

Configuration of <User_ControllerModelIndication>(): The name of <User_ControllerModelIndication>() which is called by CanIf shall be configured for CanIf by

parameter **CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME** **CanIfDispatchCfg.CanIfDispatchUserCtrlModeIndicationName** (see ECUC_CanIf_00683). This is only necessary if state transition notifications are configured via **CANIF_DISPATCH_USERCTRLMODEINDICATION_CanIfDispatchCfg.CanIfDispatchUserCtrlModeIndicationUL**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
 CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcv-

ModelIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter

CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
 If CanIfDispatchUserTrcvModeIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.55 Specification Item SWS_CANIF_00691

Trace References:

none

Content:

Configuration of `<User_ControllerModelIndication>()`: If
`CANIF_DISPATCH_USERCTRLMODEINDICATION_CanIfDispatch`
`Cfg.CanIfDispatchUserCtrlModelIndicationUL` is set to `CAN_SM`,
`CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME` `CanIfDispatchCfg.CanIf`
`DispatchUserCtrlModelIndicationName` must be `CanSM_ControllerModelIndication`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DeInit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModelIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.56 Specification Item SWS_CANIF_00692

Trace References:

none

Content:

Configuration of <User_ControllerModelIndication>(): If **CANIF_DISPATCH_USERCTRLMODEINDICATION_CanIfDispatchCfg.CanIfDispatchUserCtrlModelIndicationUL** is set to CDD the name of the function has to be configured via parameter **CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME**. The function parameter has to be of type **uint8CanIfDispatchCfg.CanIfDispatchUserCtrlModelIndicationName**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: CanIf_Delnit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify

the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
 What will be the initial value of CanId then? Does the CanId initial value matter at all
 in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value,
 E_OK is not valid.
 Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
 Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
 CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed
 to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID
 error code.
 APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER
 error code.
 The only exception is CanIf_CbkCurrentIcomConfiguration which uses
 CANIF_E_PARAM_CONTROLLERID.
 Should this callback use CANIF_E_PARAM_CONTROLLER error code?
 Or should each relevant API use the same error code?
 –Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===
 # # if Can_ReturnType is not replaced by Std_ReturnType:
 ~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized
 L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of
 Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request
 from the Transmit L-PDU Buffer immediately, before the transmit confirmation
 returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the trans-
 mit request
 service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is
 Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter

CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:
 This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.
 –Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.57 Specification Item SWS_CANIF_00695

Trace References:

none

Content:

Configuration of <User_TrvcModeIndication>(): The upper layer module which provides this callback service has to be configured by CANIF_DISPATCH_USERTRCVMODEINDICATION_CanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationUL (see ECUC_CanIf_00686), but:

- If no upper layer modules are configured for transceiver mode indications using <User_TrvcModeIndication>(), no transceiver mode indication needs to be configured. CANIF_DISPATCH_USERTRCVMODEINDICATION_CanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationUL needs not to be configured.
- If transceivers are not supported (CanInterfaceTransceiverDriver Configuration is not configured, see ECUC_CanIf_00273), CANIF_DISPATCH_USERTRCVMODEINDICATION_CanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationUL is not configurable.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-

patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-

gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.58 Specification Item SWS_CANIF_00696

Trace References:

none

Content:

Configuration of <User_TrvcModeIndication>(): The name of <User_TrvcMode Indication>() which will be called by CanIf shall be configured for CanIf by parameter **CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME** **CanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationName** (see ECUC_CanIf_00685). This is only necessary if state transition notifications are configured via **CANIF_DISPATCH_USERTRCVMODEINDICATION_CanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationUL**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOff-
 Name
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUser-
 ConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDis-
 patchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-
 patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> Can-
 IfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDis-
 patchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> Can-
 IfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
 Name":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
 CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
 the API <User_TxConfirmation>() has to be configured via parameter
 CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():

If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.59 Specification Item SWS_CANIF_00697

Trace References:

none

Content:

Configuration of <User_TrvcModelIndication>(): If **CANIF_DISPATCH_USERTRCVMODEINDICATION_IfDispatchCfg.CanIfDispatchUserTrcvModelIndicationUL** is set to CAN_SM,

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME `CanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationName` must be `CanSM_TransceiverModeIndication`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DelNit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API `Can_CancelTransmit`.

7) SWS_CANIF_00382:
Please correct API name: `CanIf_Tranmsit` -> `CanIf_Transmit`
Besides, this error code is not defined: `CANIF_E_STOPPED`.

8) Figure 9.10: Read received data
`CanIf_ReadTxNotifStatus`, `CanIf_ReadNotifStatus` should be changed to `CanIf_ReadRxNotifStatus`.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():

If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcevModelIndication>():
 If CanIfDispatchUserTrcevModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcevModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcevWufFlagIndication>():
 If CanIfDispatchUserClearTrcevWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcevWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcevWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcevWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcevWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.60 Specification Item SWS_CANIF_00698

Trace References:

none

Content:

Configuration of <User_TrcvModeIndication>(): If **CANIF_DISPATCH_USERTRCVMODEINDICATION_IfDispatchCfg.CanIfDispatchUserTrcvModeIndicationUL** is set to CDD the name of the API has to be configured via parameter **CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME**. The function parameter has to be of type **uint8CanIfDispatchCfg.CanIfDispatchUserTrcvModeIndicationName**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirma-

tionUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModelIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModelIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

- ~SWS_CANIF_00800
- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

- ~SWS_CANIF_00801
- ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If

CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.61 Specification Item SWS_CANIF_00700

Trace References:

none

Content:

If parameter ControllerId of CanIf_ControllerModelIndication() has an invalid value, CanIf shall report development error code CANIF_E_PARAM_CONTROLLER_CONTROLLERID to the Det_ReportError service of the DET module, when CanIf_ControllerModelIndication() is called.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserC-

trlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():

If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.62 Specification Item SWS_CANIF_00703

Trace References:

none

Content:

Caveats of CanIf_ControllerModeIndication():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable

-[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 CanIfRxSduld, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329

Note: During the call 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_ModelIndication(): 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

1.63 Specification Item SWS_CANIF_00709

Trace References:

none

Content:

Caveats of CanIf_TrcvModeIndication():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization 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.

=== EthIf ===

~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_GetArITable
-[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 `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_ModelIndication(): 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

1.64 Specification Item SWS_CANIF_00711

Trace References:

none

Content:

When callback `CanIf_ControllerModelIndication(ControllerId, ControllerMode)` is called, `CanIf` shall call `CanSm_ControllerModelIndication(ControllerId, ControllerMode)` of the `CanSm` (see [REF_sec_3a_User_ControllerModelIndication]) or a CDD (see SWS_CANIF_00691, SWS_CANIF_00692).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against `CanIf` SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DelNit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855. What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserC-

trlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():

If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.65 Specification Item SWS_CANIF_00712

Trace References:

none

Content:

When callback `CanIf_TrcvModeIndication(Transceiver, TransceiverMode)` is called, `CanIf` shall call `CanSM_TransceiverModeIndication(TransceiverId, TransceiverMode)` of the `CanSm` (see [REF sec_3a_User_ControllerModeIndication]) or a CDD (see SWS_CANIF_00697, SWS_CANIF_00698).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against `CanIf` SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".

The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DeInit` API.

Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`. Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.

What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descrip-

tions like e.g. ECUC_CanIf_00528) ===

- CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName
- CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
- CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
- CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
- CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
- CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
- CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
- CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
- CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
- CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
- CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
- CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
- CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
- CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
- CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
- CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-

Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set

to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DelNit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not

be configurable.
 –Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.66 Specification Item SWS_CANIF_00720

Trace References:

none

Content:

If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns (CAN_OK / E_OK) to CanIf, then CanIf_CheckWakeup() shall return E_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

Problem description:

While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:

- * type change from enumeration to extra_literal
- * Remove range element CAN_OK
- * Remove range element CAN_NOT_OK
- * Assign value "0x02" to range element "CAN_BUSY"
- * Description: Overlaid return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode

Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller,

Can_StateTransitionType Transition)

Return value:

Std_ReturnType

E_OK: request accepted

E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup

Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)

Return value:

Std_ReturnType

E_OK: API call has been accepted

E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write

Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)

Return value:

Std_ReturnType

E_OK: Write command has been accepted

E_NOT_OK: development error occurred

CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_Can_00048

~SWS_Can_00089

7.11.5 Return Values

~SWS_Can_00198

~SWS_Can_00199

~SWS_Can_00200

~SWS_Can_00216

~SWS_Can_00217

~SWS_Can_00218

~SWS_CAN_00219

~SWS_CAN_00505

~SWS_CAN_00506

~SWS_Can_00212

=== CanIf ===

Adapt API Can_Write() to new signature:

* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"

* Figure 9.1 "Transmission request with a single CAN Driver"

* Figure 9.2 "Transmission request with multiple CAN Drivers"

- * Figure 9.5 "Transmit confirmation with buffering"
- * Figure 9.6 "Transmit Cancelation"
- * Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:

- * Figure 9.11: Start CAN network
- * Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

Note between SWS_CANIF_00162 and SWS_CANIF_00319

Table in chapter 9.7 Trigger Transmit Request

Table in chapter 9.11 Start CAN network

=== CanTrcv ===

Adapt API Can_SetControllerMode() to new signature:

- * 9.3 De-Initialization (SPI Synchronous)
- * 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===

Adapt API Can_CheckWakeup() to new signature:

- * Figure 42 CAN controller wake up by interrupt
- * Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===

Adapt API Can_Write() to new signature:

- * Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:

- * Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_TtCanIf_00071

==== TTCanDrv ====

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

- ~SWS_TtCan_00014
- ~SWS_TtCan_00018
- ~SWS_TtCan_00022
- ~SWS_TtCan_00026
- ~SWS_TtCan_00059
- ~SWS_TtCan_00078
- ~SWS_TtCan_00112

==== XCP ====

Adapt API Can_Write() to new signature:

* Figure 5: Xcp on Can Transmit

–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.67 Specification Item SWS_CANIF_00724

Trace References:

none

Content:

When callback CanIf_ControllerBusOff(ControlllerId) is called, the CanIf shall call Can SM_ControllerBusOff(ControlllerId) of the CanSm (see [REF sec_3a_User_Controller ModelIndication] or a CDD (see SWS_CANIF_00559, SWS_CANIF_00560).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData). Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.
The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.
Should this callback use CANIF_E_PARAM_CONTROLLER error code?
Or should each relevant API use the same error code?
-Last change on issue 77764 comment 3-

Agreed solution:

=== Dependent on decision of # 77952 ===
if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).
 # # endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

- CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName
- CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
- CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
- CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
- CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
- CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
- CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
- CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
- CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
- CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
- CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
- CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
- CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
- CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> Can-

IfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
 Name":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
 CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
 the API <User_TxConfirmation>() has to be configured via parameter
 CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModeIndication>():
 If CanIfDispatchUserTrcvModeIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:
 - Error Code: CANIF_E_STOPPED
 - Type: Transmit requested on offline PDU channel
 - Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.68 Specification Item SWS_CANIF_00737

Trace References:

none

Content:

Caveats of CanIf_GetTxConfirmationState():

- The call context is on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization 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.

=== EthIf ===

~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_GetArITable
-[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 `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_ModelIndication(): 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

1.69 Specification Item SWS_CANIF_00765

Trace References:

none

Content:

Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfir-

mationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModelIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModelIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter

CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.70 Specification Item SWS_CANIF_00793

Trace References:

none

Content:

Caveats of <User_ClearTrcvWufFlagIndication>():

- The CanTrcv must be initialized after Power ON.
- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- This callback service is in general re-entrant for multiple CAN Transceiver usage, but not for the same CAN Transceiver.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization 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.

=== EthIf ===

~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_GetArITable

-[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_ModelIndication(): 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

1.71 Specification Item SWS_CANIF_00794

Trace References:

none

Content:

Configuration of <User_ClearTrcvWufFlagIndication>(): The upper layer module, which is called (see SWS_CANIF_00757), has to be configurable by **CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_CanIfDispatch** **Cfg.CanIfDispatchUserClearTrcvWufFlagIndicationUL** (see ECUC_CanIf_00790) if CANIF_PUBLIC_PN_SUPPORT (see ECUC_CanIf_00772) equals True.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-

patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-

gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.72 Specification Item SWS_CANIF_00795

Trace References:

none

Content:

Configuration of <User_ClearTrcvWufFlagIndication>(): The name of <User_ClearTrcvWufFlagIndication>() shall be configurable by **CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME** **CanIfDispatchCfg.CanIfDispatchUserClearTrcvWufFlagIndicationName** (see ECUC_CanIf_00789) if CANIF_PUBLIC_PN_SUPPORT (see ECUC_CanIf_00772) equals True.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.
The only exception is CanIf_CbkCurrentIcomConfiguration which uses

CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOff-

Name

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to

CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DelNit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.73 Specification Item SWS_CANIF_00797

Trace References:

none

Content:

Configuration of <User_ClearTrcvWufFlagIndication>(): If
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_CanIfDispatch
 Cfg.CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CAN_SM,
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME CanIfDispatch

`Cfg.CanIfDispatchUserClearTrcvWufFlagIndicationName` must be `CanSM_ClearTrcvWufFlagIndication`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DelNit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API `Can_CancelTransmit`.

7) SWS_CANIF_00382:
Please correct API name: `CanIf_Tranmsit` -> `CanIf_Transmit`
Besides, this error code is not defined: `CANIF_E_STOPPED`.

8) Figure 9.10: Read received data
`CanIf_ReadTxNotifStatus`, `CanIf_ReadNotifStatus` should be changed to `CanIf_ReadRxNotifStatus`.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>():

If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModelIndication>(): If CanIfDispatchUserTrvcModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrvcModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.74 Specification Item SWS_CANIF_00798

Trace References:

none

Content:

Configuration of `<User_ClearTrcvWufFlagIndication>()`: If `CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_CanIfDispatchCfg.CanIfDispatchUserClearTrcvWufFlagIndicationUL` is set to CDD, the name of the service has to be configurable via parameter `CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME` and the function parameter has to be of type `uint8CanIfDispatchCfg.CanIfDispatchUserClearTrcvWufFlagIndicationName`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

- 1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: `CanIf_DelNit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?
- 5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.
- 6) Figure 9.6: There is no API `Can_CancelTransmit`.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfir-

mationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModelIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModelIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter

CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.75 Specification Item SWS_CANIF_00799

Trace References:

none

Content:

Caveats of <User_CheckTrcvWakeFlagIndication>():

- The CanTrcv must be initialized after Power ON.
- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- This callback service is in general re-entrant for multiple CAN Transceiver usage, but not for the same CAN Transceiver.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization 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.

=== EthIf ===

~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_GetArITable

-[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_ModelIndication(): 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

1.76 Specification Item SWS_CANIF_00800

Trace References:

none

Content:

Configuration of <User_CheckTrcvWakeFlagIndication>(): The upper layer module, which is called (see SWS_CANIF_00759), has to be configurable by **CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_CanIfDispatch** **Cfg.CanIfDispatchUserCheckTrcvWakeFlagIndicationUL** (see ECUC_CanIf_00792) if CANIF_PUBLIC_PN_SUPPORT (see ECUC_CanIf_00772) equals True.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-

patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-

gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.77 Specification Item SWS_CANIF_00801

Trace References:

none

Content:

Configuration of <User_CheckTrcvWakeFlagIndication>(): The name of <User_CheckTrcvWakeFlagIndication>() shall be configurable by **CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME** **CanIfDispatch Cfg.CanIfDispatchUserCheckTrcvWakeFlagIndicationName** (see ECUC_CanIf_00791) if CANIF_PUBLIC_PN_SUPPORT (see ECUC_CanIf_00772) equals True.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DeInit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.
APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.
The only exception is CanIf_CbkCurrentIcomConfiguration which uses

CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOff-

Name

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to

CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DelNit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.78 Specification Item SWS_CANIF_00803

Trace References:

none

Content:

Configuration of <User_CheckTrcvWakeFlagIndication>(): If
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_CanIfDispatch
 Cfg.CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CAN_SM,
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME CanIfDispatch

`Cfg.CanIfDispatchUserCheckTrcvWakeFlagIndicationName` must be `CanSM_CheckTrcvWakeFlagIndication`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DelNit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API `Can_CancelTransmit`.

7) SWS_CANIF_00382:
Please correct API name: `CanIf_Tranmsit` -> `CanIf_Transmit`
Besides, this error code is not defined: `CANIF_E_STOPPED`.

8) Figure 9.10: Read received data
`CanIf_ReadTxNotifStatus`, `CanIf_ReadNotifStatus` should be changed to `CanIf_ReadRxNotifStatus`.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():

If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcevModelIndication>(): If CanIfDispatchUserTrcevModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcevModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcevWufFlagIndication>(): If CanIfDispatchUserClearTrcevWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcevWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcevWakeFlagIndication>(): If CanIfDispatchUserCheckTrcevWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcevWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.79 Specification Item SWS_CANIF_00804

Trace References:

none

Content:

Configuration of `<User_CheckTrcvWakeFlagIndication>()`: If `CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_CanIfDispatchCfg.CanIfDispatchUserCheckTrcvWakeFlagIndicationUL` is set to CDD, the name of the service has to be configurable via parameter `CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME` and the function parameter has to be of type `uint8CanIfDispatchCfg.CanIfDispatchUserCheckTrcvWakeFlagIndicationName`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: `CanIf_DelNit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?
- 5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.
- 6) Figure 9.6: There is no API `Can_CancelTransmit`.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfir-

mationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModelIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModelIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter

CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.80 Specification Item SWS_CANIF_00807

Trace References:

none

Content:

Caveats of CanIf_ClearTrcvWufFlagIndication():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable

-[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 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_TrvcModeIndication()

-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_ModelIndication(): 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

1.81 Specification Item SWS_CANIF_00811

Trace References:

none

Content:

Caveats of CanIf_CheckTrcvWakeFlagIndication():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable

-[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_ModelIndication(): 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

1.82 Specification Item SWS_CANIF_00818

Trace References:

none

Content:

Caveats of CanIf_ConfirmPnAvailability():

- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- The CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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 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

1.83 Specification Item SWS_CANIF_00822

Trace References:

none

Content:

Caveats of <User_ConfirmPnAvailability>():

- The CanTrcv must be initialized after Power ON.
- The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- This callback service is in general re-entrant for multiple CAN Transceiver usage, but not for the same CAN Transceiver.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable
-[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 CanIfRxSduld, 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_ControllerModelIndication()
- SWS_CANIF_00709 CanIf_TrvcModelIndication()
- 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_ControllerModelIndication>()

=== 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_ModelIndication(): 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

1.84 Specification Item SWS_CANIF_00823

Trace References:

none

Content:

Configuration of `<User_ConfirmPnAvailability>()`: The upper layer module, which is called (see SWS_CANIF_00753), has to be configurable by `CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_CanIfDispatchCfg.CanIfDispatchUserConfirmPnAvailabilityUL` (see ECUC_CanIf_00820) if `CANIF_PUBLIC_PN_SUPPORT` (see ECUC_CanIf_00772) equals True.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DeInit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`.
Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855.
What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API `Can_CancelTransmit`.

7) SWS_CANIF_00382:
Please correct API name: `CanIf_Tranmsit` -> `CanIf_Transmit`
Besides, this error code is not defined: `CANIF_E_STOPPED`.

8) Figure 9.10: Read received data
 CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcv-

ModelIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter

CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
 If CanIfDispatchUserTrcvModeIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_Delnit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.85 Specification Item SWS_CANIF_00824

Trace References:

none

Content:

Configuration of `<User_ConfirmPnAvailability>()`: The name of `<User_ConfirmPnAvailability>()` shall be configurable by **CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME** **CanIfDispatchCfg.CanIfDispatchUserConfirmPnAvailabilityName** (see ECUC_CanIf_00819) if `CANIF_PUBLIC_PN_SUPPORT` (see ECUC_CanIf_00772) equals True.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within `CanIf_CheckTrcvWakeFlag()` the function `CanTrcv_CheckTrcvWakeFlag()` shall be called".
The API name is `CanTrcv_CheckWakeFlag`.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: `CanIf_DeInit` API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: `CanIf_Init` initializes the `CanIds` of the dynamic Transmit L-PDUs with `CanIfTxPduType` to the value configured via `CanIfTxPduCanId`. Besides, SWS_CANIF_00855 states that `CanIfTxPduCanId` can be omitted in some cases (`CanId` is directly taken from `MetaData`).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when `CanIfTxPduCanId` is omitted according to SWS_CANIF_00855. What will be the initial value of `CanId` then? Does the `CanId` initial value matter at all in case of this configuration?

5) `Can_Write` has `Can_ReturnType` return value: `CAN_OK` is valid return value, `E_OK` is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModelIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL
 CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
 CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
 ~SWS_CANIF_00800
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 ~SWS_CANIF_00801
 ~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.86 Specification Item SWS_CANIF_00826

Trace References:

none

Content:

Configuration of <User_ConfirmPnAvailability>(): If **CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY** **IfDispatchCfg.CanIfDispatchUserConfirmPnAvailabilityUL** is set to CAN_SM, **CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME** **CanIfDispatchCfg.CanIfDispatchUserConfirmPnAvailabilityName** must be CanSM_ConfirmPnAvailability.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
 The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
 There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.
 Please create requirement ID starting with 'SWS_CANIF_'.
 Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
 Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).
 Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.
 What will be the initial value of CanId then? Does the CanId initial value matter at all

in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>(): If CanIfDispatchUserCtrlModelIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>(): If CanIfDispatchUserTrcvModelIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.87 Specification Item SWS_CANIF_00827

Trace References:

none

Content:

Configuration of <User_ConfirmPnAvailability>(): If **CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY** **IfDispatchCfg.CanIfDispatchUserConfirmPnAvailabilityUL** is set to CDD, the name of the service has to be configurable via parameter **CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME** and the function parameter has to be of type **uint8CanIfDispatchCfg.CanIfDispatchUserConfirmPnAvailabilityName**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: CanIf_DelNit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Trans-

mit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.
 Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
 Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:
 Please correct API name: CanIf_Tranmsit -> CanIf_Transmit
 Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data
 CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).
 # # endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME	->	CanIfDispatchUserCtrlModeIndicationName
CANIF_DISPATCH_USERCTRLMODEINDICATION_UL	->	CanIfDispatchUserCtrlModeIndicationUL
CANIF_TXPDU_USERTXCONFIRMATION_NAME	->	CanIfTxPduUserTxConfirmationName
CANIF_TXPDU_USERTXCONFIRMATION_UL	->	CanIfTxPduUserTxConfirmationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_UL	->	CanIfDispatchUserTrcvModeIndicationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME	->	CanIfDispatchUserTrcvModeIndicationName
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL	->	CanIfDispatchUserValidateWakeupEventUL
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME	->	CanIfDispatchUserValidateWakeupEventName
CANIF_DISPATCH_USERCTRLBUSOFF_UL	->	CanIfDispatchUserCtrlBusOffUL
CANIF_DISPATCH_USERCTRLBUSOFF_NAME	->	CanIfDispatchUserCtrlBusOffName
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL	->	CanIfDispatchUserConfirmPnAvailabilityUL
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME	->	CanIfDispatchUserConfirmPnAvailabilityName
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL	->	CanIfDispatchUserClearTrcvWufFlagIndicationUL
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME	->	CanIfDispatchUserClearTrcvWufFlagIndicationName
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL	->	CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME	->	CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
the API <User_TxConfirmation>() has to be configured via parameter
CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
the name of the function has to be configured via parameter
CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
If CanIfDispatchUserTrcvModeIndicationUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
<User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
If CanIfDispatchUserValidateWakeupEventUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.88 Specification Item SWS_CANIF_00858

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If **CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL** is set to J1939NM, **CANIF_TXPDU_USERTXCONFIRMATION_NAME** **CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName** must be J1939Nm_TxConfirmation.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelInit API.

Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation

returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> Can-

IfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
the API <User_TxConfirmation>() has to be configured via parameter
CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>():
If CanIfDispatchUserCtrlModelIndicationUL is set to CDD
the name of the function has to be configured via parameter
CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>():
If CanIfDispatchUserTrcvModelIndicationUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
<User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
If CanIfDispatchUserValidateWakeupEventUL is set to
CDD the name of the API has to be configured via parameter
CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>(): If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function

CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.89 Specification Item SWS_CANIF_00870

Trace References:

none

Content:

Caveats of CanIf_SetBaudrate():

- The call context is on task level (polling mode).
- CanIf must be initialized after Power ON.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArTable

-[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_TrvcModeIndication()
- 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_ModelIndication(): 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

1.90 Specification Item SWS_CANIF_00879

Trace References:

none

Content:

Configuration of <User_TxConfirmation>(): If **CANIF_TXPDU_USERTXCONFIRMATION_CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL** is set to **CAN_TSYN**, **CANIF_TXPDU_USERTXCONFIRMATION_NAME** **CanIfTxPduCfg.CanIfTxPduUserTxConfirmationName** must be **CanTSyn_CanIfTxConfirmation**.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

- 1) SWS_CANIF_00765: "Within **CanIf_CheckTrcvWakeFlag()** the function **CanTrcv_CheckTrcvWakeFlag()** shall be called".
The API name is **CanTrcv_CheckWakeFlag**.
- 2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.
- 3) SWS_CanTrcv_91001: **CanIf_DeInit** API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.
- 4) SWS_CANIF_00857: **CanIf_Init** initializes the **CanIds** of the dynamic Transmit L-PDUs with **CanIfTxPduType** to the value configured via **CanIfTxPduCanId**.
Besides, SWS_CANIF_00855 states that **CanIfTxPduCanId** can be omitted in some cases (**CanId** is directly taken from **MetaData**).
Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when **CanIfTxPduCanId** is omitted according to SWS_CANIF_00855.
What will be the initial value of **CanId** then? Does the **CanId** initial value matter at all in case of this configuration?
- 5) **Can_Write** has **Can_ReturnType** return value: **CAN_OK** is valid return value, **E_OK** is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.
- 6) Figure 9.6: There is no API **Can_CancelTransmit**.
- 7) SWS_CANIF_00382:
Please correct API name: **CanIf_Tranmsit** -> **CanIf_Transmit**
Besides, this error code is not defined: **CANIF_E_STOPPED**.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
 CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
 CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
 CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
 CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
 CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME" (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of

the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>():
 If CanIfDispatchUserCtrlModeIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModeIndication>():
 If CanIfDispatchUserTrcvModeIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-
 gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with

CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in
 SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function
 CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of
 <User_CheckTrcvWakeFlagIndication>. If
 CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is
 fixed. If it equals CDD, the name is selectable. If
 CanIfPublicPnSupport equals False, this parameter shall not
 be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.91 Specification Item SWS_CANIF_00882

Trace References:

none

Content:

CanIf_Transmit() shall accept a NULL pointer as PduInfoPtr->SduDataPtr, if the PDU is configured for triggered transmission: CanIfPublicCfg.CanIfPublicTxBuffering TxPduTriggerTransmit = TRUE.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #78366: [CanIf] SWS_CANIF_00882 refers to wrong configuration parameter

Problem description:

SWS_CANIF_00882 refers to CanIfPublicTxBuffering instead of CanIfTxPduTriggerTransmit. (it seems to be a mistake introduced in R4.3.0 as it was correct in R4.2.2)

Agreed solution:

Update SWS_CANIF_00882 from "CanIfPublicTxBuffering = TRUE" to "CanIfTxPduTriggerTransmit = TRUE"

BW-C-Level:

Application	Specification	Bus
1	4	1

1.92 Specification Item SWS_CANIF_00887

Trace References:

none

Content:

Caveats of <User_TriggerTransmit>(): The call context is either on interrupt level (interrupt mode) or on task level (polling mode).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (EthIf_Init). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable
-[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 `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_ModelIndication(): 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

1.93 Specification Item SWS_CANIF_00891

Trace References:

none

Content:

Configuration of <User_TriggerTransmit>(): If CanIfTxPduCfg.CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduCfg.CanIfTxPduUserTriggerTransmitName. **One function parameter has to be of type PduIdType and one of type PduInfoType*.**

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findings against CanIf SWS.

1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".
The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698
There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.
Please create requirement ID starting with 'SWS_CANIF_'.
Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId. Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData). Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855. What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.
Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request

service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserC-

```

trlModeIndicationUL
CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName
CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL
CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL
CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName
CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL
CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL
CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDispatchUserClearTrcvWufFlagIndicationUL
CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName
  
```

=== Others ===

```

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":
~SWS_CANIF_00800
~SWS_CANIF_00801
~SWS_CANIF_00803
  
```

```

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
(typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-
Name":
~SWS_CANIF_00801
~SWS_CANIF_00803
  
```

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TxConfirmation>() has to be configured via parameter CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModeIndication>(): If CanIfDispatchUserCtrlModeIndicationUL is set to CDD the name of the function has to be configured via parameter CanIfDispatchUserCtrlModeIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrvcModeIndication>(): If CanIfDispatchUserTrcvModeIndicationUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserTrcvModeIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTriggerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>(): If CanIfDispatchUserValidateWakeupEventUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>(): If CanIfDispatchUserConfirmPnAvailabilityUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>(): If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():

If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set to CDD, the name of the service has to be configurable via parameter CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":
 Description:

This parameter defines the name of <User_CheckTrcvWakeFlagIndication>. If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is fixed. If it equals CDD, the name is selectable. If CanIfPublicPnSupport equals False, this parameter shall not be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.94 Specification Item SWS_CANIF_00893

Trace References:

none

Content:

When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId:

- SduLength > 8 if the Can_IdType indicates a classic CAN frame
- SduLength > 64 if the Can_IdType indicates a CAN FD frame

CanIf shall report **development** **runtime** error code CANIF_E_DATA_LENGTH_MISMATCH to the Det_ReportError **RuntimeError()** service of the DET.

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

1.95 Specification Item SWS_CANIF_00894

Trace References:

none

Content:

When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and **CanIfTxPduTruncation is enabled**, CanIf shall transmit as much data as possible and discard the rest.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76711: [CanIf] Discarding parts of a PDU without an error is no good idea

Problem description:

SWS_CANIF_00894 specifies that CanIf shall just ignore parts of a passed PDU that exceed the configured size. This is not a very good idea in most cases.

I see only one use case where such an approach is valid: If a gateway takes in also extended PDUs, but routes only the original size. But I doubt this is a very good use case.

I see two possible solutions for this problem:

1. Return E_NOT_OK if the PDU exceeds the configured size.
2. Return E_OK but throw a runtime error.

Agreed solution:

Add new parameter to "CanIfTxPduCfg" as following:

Name CanIfTxPduTruncation

Description Enables/disables truncation of PDUs that exceed the configured size.

Multiplicity 1

Type EcucBooleanParamDef

Default Value true

Post-Build Variant Value true

Value Configuration Class

Pre-compile time X VARIANT-PRE-COMPILE

Link time X VARIANT-LINK-TIME

Post-build time X VARIANT-POST-BUILD

Scope / Dependency scope: ECU

~SWS_CANIF_00894 When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is enabled, CanIf shall transmit as much data as possible and discard the rest.

+SWS_CANIF_????? When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is disabled, CanIf shall report the runtime error CANIF_E_TXPDU_LENGTH_EXCEEDED and return E_NOT_OK without further actions.

7.27.2 Runtime Errors create Table with entry

Type: Message length was exceeding the maximum length.

Related error code: CANIF_E_TXPDU_LENGTH_EXCEEDED

Value: 90

–Last change on issue 76711 comment 22–

BW-C-Level:

Application	Specification	Bus
1	3	3

1.96 Specification Item SWS_CANIF_00895

Trace References:

none

Content:

If the rejected data length exceeds the configured size, CanIf shall:

- buffer the configured amount of data and discard the rest
- and report **development runtime** error code CANIF_E_DATA_LENGTH_MISMATCH to the Det_ReportError RuntimeError() service of the DET.

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

1.97 Specification Item SWS_CANIF_00900

Trace References:

none

Content:

When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is disabled, Can If shall report the runtime error CANIF_E_TXPDU_LENGTH_EXCEEDED and return E_NOT_OK without further actions.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76711: [CanIf] Discarding parts of a PDU without an error is no good idea

Problem description:

SWS_CANIF_00894 specifies that CanIf shall just ignore parts of a passed PDU that exceed the configured size. This is not a very good idea in most cases.

I see only one use case where such an approach is valid: If a gateway takes in also extended PDUs, but routes only the original size. But I doubt this is a very good use case.

I see two possible solutions for this problem:

1. Return E_NOT_OK if the PDU exceeds the configured size.
2. Return E_OK but throw a runtime error.

Agreed solution:

Add new parameter to "CanIfTxPduCfg" as following:

Name CanIfTxPduTruncation
 Description Enables/disables truncation of PDUs that exceed the configured size.
 Multiplicity 1
 Type EcucBooleanParamDef
 Default Value true
 Post-Build Variant Value true
 Value Configuration Class
 Pre-compile time X VARIANT-PRE-COMPILE
 Link time X VARIANT-LINK-TIME
 Post-build time X VARIANT-POST-BUILD
 Scope / Dependency scope: ECU

~SWS_CANIF_00894 When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is enabled, CanIf shall transmit as much data as possible and discard the rest.

+SWS_CANIF_????? When CanIf_Transmit() is called with PduInfoPtr->SduLength exceeding the maximum length of the PDU referenced by TxPduId and CanIfTxPduTruncation is disabled, CanIf shall report the runtime error CANIF_E_TXPDU_LENGTH_EXCEEDED and return E_NOT_OK without further actions.

7.27.2 Runtime Errors create Table with entry
 Type: Message length was exceeding the maximum length.
 Related error code: CANIF_E_TXPDU_LENGTH_EXCEEDED
 Value: 90
 –Last change on issue 76711 comment 22–

BW-C-Level:

Application	Specification	Bus
1	3	3

1.98 Specification Item SWS_CANIF_00901

Trace References:

none

Content:

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()`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initialization 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 initialization of the certain api's, e.g.:

"

...

[SWS_EthIf_00213] ?

Caveat: The function requires previous initialization (`EthIf_Init`). ?()

"

Such requirement 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.

=== EthIf ===

~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_GetArITable
-[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_TrvcModeIndication()

-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

1.99 Specification Item SWS_CANIF_91002

Trace References:

[SRS_Can_01168](#), [SRS_BSW_00336](#)

Content:

Service name:	CanIf_DelInitCanIf_DelInit
Syntax:	void CanIf_DelInit(void)
Service ID[hex]:	0x02
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	De-initializes the CanIf module.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77764: [CanIf] Documentation issues

Problem description:

I have the following findigns against CanIf SWS.

- 1) SWS_CANIF_00765: "Within CanIf_CheckTrcvWakeFlag() the function CanTrcv_CheckTrcvWakeFlag() shall be called".

The API name is CanTrcv_CheckWakeFlag.

2) SWS_CANIF_00551, SWS_CANIF_00692, SWS_CANIF_00698

There are now two function parameters.

3) SWS_CanTrcv_91001: CanIf_DelNit API.

Please create requirement ID starting with 'SWS_CANIF_'.

Note, SWS_CANIF_91001 is already in use.

4) SWS_CANIF_00857: CanIf_Init initializes the CanIds of the dynamic Transmit L-PDUs with CanIfTxPduType to the value configured via CanIfTxPduCanId.

Besides, SWS_CANIF_00855 states that CanIfTxPduCanId can be omitted in some cases (CanId is directly taken from MetaData).

Thus, I think requirement SWS_CANIF_00857 is not complete. It does not specify the behavior when CanIfTxPduCanId is omitted according to SWS_CANIF_00855.

What will be the initial value of CanId then? Does the CanId initial value matter at all in case of this configuration?

5) Can_Write has Can_ReturnType return value: CAN_OK is valid return value, E_OK is not valid.

Please correct SWS_CANIF_00183, SWS_CANIF_00162, tables in 9.1, 9.2, 9.6.

6) Figure 9.6: There is no API Can_CancelTransmit.

7) SWS_CANIF_00382:

Please correct API name: CanIf_Tranmsit -> CanIf_Transmit

Besides, this error code is not defined: CANIF_E_STOPPED.

8) Figure 9.10: Read received data

CanIf_ReadTxNotifStatus, CanIf_ReadNotifStatus should be changed to CanIf_ReadRxNotifStatus.

9) APIs in chapter 8.3 Function definitions use CANIF_E_PARAM_CONTROLLERID error code.

APIs in chapter 8.4 Callback notifications use CANIF_E_PARAM_CONTROLLER error code.

The only exception is CanIf_CbkCurrentIcomConfiguration which uses CANIF_E_PARAM_CONTROLLERID.

Should this callback use CANIF_E_PARAM_CONTROLLER error code?

Or should each relevant API use the same error code?

–Last change on issue 77764 comment 3–

Agreed solution:

=== Dependent on decision of # 77952 ===

if Can_ReturnType is not replaced by Std_ReturnType:

~[SWS_CANIF_00183] When CanIf calls the function Can_Write() for prioritized L-PDUs and Transmit Requests stored in CanIfTxBuffer and the return value of Can_Write() is CAN_OK, then CanIf shall remove this L-PDU or Transmit Request from the Transmit L-PDU Buffer immediately, before the transmit confirmation returns.

~[SWS_CANIF_00162] d If the call of Can_Write() returns CAN_OK the transmit request service CanIf_Transmit() shall return E_OK. c()

Chapter 9 correct in tables below of 9.1, 9.2, 9.6 that return type of Can_Write() is Can_ReturnType (E_OK -> CAN_OK).

endif

=== Refactoring (affecting multiple requirements, often also ECUC descriptions like e.g. ECUC_CanIf_00528) ===

CANIF_DISPATCH_USERCTRLMODEINDICATION_NAME -> CanIfDispatchUserCtrlModeIndicationName

CANIF_DISPATCH_USERCTRLMODEINDICATION_UL -> CanIfDispatchUserCtrlModeIndicationUL

CANIF_TXPDU_USERTXCONFIRMATION_NAME -> CanIfTxPduUserTxConfirmationName

CANIF_TXPDU_USERTXCONFIRMATION_UL -> CanIfTxPduUserTxConfirmationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_UL -> CanIfDispatchUserTrcvModeIndicationUL

CANIF_DISPATCH_USERTRCVMODEINDICATION_NAME -> CanIfDispatchUserTrcvModeIndicationName

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_UL -> CanIfDispatchUserValidateWakeupEventUL

CANIF_DISPATCH_USERVALIDATEWAKEUPEVENT_NAME -> CanIfDispatchUserValidateWakeupEventName

CANIF_DISPATCH_USERCTRLBUSOFF_UL -> CanIfDispatchUserCtrlBusOffUL

CANIF_DISPATCH_USERCTRLBUSOFF_NAME -> CanIfDispatchUserCtrlBusOffName

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_UL -> CanIfDispatchUserConfirmPnAvailabilityUL

CANIF_DISPATCH_USERCONFIRMPNAVAILABILITY_NAME -> CanIfDispatchUserConfirmPnAvailabilityName

CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_UL -> CanIfDis-

patchUserClearTrcvWufFlagIndicationUL
 CANIF_DISPATCH_USERCLEARTRCVWUFFLAGINDICATION_NAME -> CanIfDispatchUserClearTrcvWufFlagIndicationName
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL -> CanIfDispatchUserCheckTrcvWakeFlagIndicationUL
 CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME -> CanIfDispatchUserCheckTrcvWakeFlagIndicationName

=== Others ===

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_UL"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndicationUL":

~SWS_CANIF_00800

~SWS_CANIF_00801

~SWS_CANIF_00803

Rename "CANIF_DISPATCH_USERCHECKRCVWAKEFLAGINDICATION_NAME"
 (typo and outdated style) to "CanIfDispatchUserCheckTrcvWakeFlagIndication-Name":

~SWS_CANIF_00801

~SWS_CANIF_00803

~[SWS_CANIF_00765] Within CanIf_CheckTrcvWakeFlag() the function
 CanTrcv_CheckWakeFlag() shall be called.

~[SWS_CANIF_00551] Configuration of <User_TxConfirmation>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of
 the API <User_TxConfirmation>() has to be configured via parameter
 CanIfTxPduUserTxConfirmationName.

~[SWS_CANIF_00692] Configuration of <User_ControllerModelIndication>():
 If CanIfDispatchUserCtrlModelIndicationUL is set to CDD
 the name of the function has to be configured via parameter
 CanIfDispatchUserCtrlModelIndicationName.

~[SWS_CANIF_00698] Configuration of <User_TrcvModelIndication>():
 If CanIfDispatchUserTrcvModelIndicationUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserTrcvModelIndicationName.

~[SWS_CANIF_00891] Configuration of <User_TriggerTransmit>(): If
 CanIfTxPduUserTxConfirmationUL is set to CDD, the name of the API
 <User_TriggerTransmit>() has to be configured via parameter CanIfTxPduUserTrig-

gerTransmitName.

~[SWS_CANIF_00564] Configuration of <User_ValidateWakeupEvent>():
 If CanIfDispatchUserValidateWakeupEventUL is set to
 CDD the name of the API has to be configured via parameter
 CanIfDispatchUserValidateWakeupEventName.

~[SWS_CANIF_00560] Configuration of <User_ControllerBusOff>(): If
 CanIfDispatchUserCtrlBusOffUL is set to CDD the name of the API has to be
 configured via parameter CanIfDispatchUserCtrlBusOffName.

~[SWS_CANIF_00827] Configuration of <User_ConfirmPnAvailability>():
 If CanIfDispatchUserConfirmPnAvailabilityUL is set to
 CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserConfirmPnAvailabilityName.

~[SWS_CANIF_00798] Configuration of <User_ClearTrcvWufFlagIndication>():
 If CanIfDispatchUserClearTrcvWufFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserClearTrcvWufFlagIndicationName.

~[SWS_CANIF_00804] Configuration of <User_CheckTrcvWakeFlagIndication>():
 If CanIfDispatchUserCheckTrcvWakeFlagIndicationUL is set
 to CDD, the name of the service has to be configurable via parameter
 CanIfDispatchUserCheckTrcvWakeFlagIndicationName.

~[SWS_CANIF_00429]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

~[SWS_CANIF_00700]: replace CANIF_E_PARAM_CONTROLLER with
 CANIF_E_PARAM_CONTROLLERID

Table "7.27.1 Development Errors": Remove entry
 "CANIF_E_PARAM_CONTROLLER 14"

~SWS_CANIF_00382: Please correct "CanIf_Tranmsit" with "CanIf_Transmit"

For SWS_CANIF_00382, kind of reintroduce development error
 CANIF_E_STOPPED for PDU channel offline while transmit requested:

- Error Code: CANIF_E_STOPPED
- Type: Transmit requested on offline PDU channel
- Value: 70

=== TO ===

Create new ID for "[SWS_CanTrcv_91001] CanIf_DeInit()" which is in SWS_CANIF_XXXXX namespace

Figure 9.10: correct CanIf_ReadTxNotifStatus() call and return to use function CanIf_ReadRxNotifStatus()

Add CanTrcv_CheckWakeFlag() to optional interfaces (SWS_CANIF_00294).

~ECUC_CanIf_00791 "CanIfDispatchUserCheckTrcvWakeFlagIndicationName":

Description:

This parameter defines the name of

<User_CheckTrcvWakeFlagIndication>. If

CanIfDispatchUserCheckTrcvWakeFlagIndicationUL

equals CAN_SM the name of <User_CheckTrcvWakeFlagIndication> is

fixed. If it equals CDD, the name is selectable. If

CanIfPublicPnSupport equals False, this parameter shall not

be configurable.

–Last change on issue 77764 comment 20–

BW-C-Level:

Application	Specification	Bus
1	4	1