Document Title	SWS_TTCANDriver: 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_TTCANDriver

1.1 Specification Item SWS_TtCan_00014

Trace References:

none

Content:

If development error detection for the Ttcan module is enabled: The function Can_TTGetControllerTime() shall raise the error CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the parameter Can_TTGlobalTime or the parameter Can_TTLocalTime or the parameter Can_TTCycleTime or the parameter Can_TTCycelCount is a NULL pointer.

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_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed 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

1.2 Specification Item SWS_TtCan_00018

Trace References:

none

Content:

If development error detection for the Ttcan module is enabled: The function Can_TTGetMasterState() shall raise the error CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the parameter Can_TTMasterState is a NULL pointer.

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_litaral
* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed 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

1.3 Specification Item SWS_TtCan_00022

Trace References:

none

Content:

If development error detection for the Ttcan module is enabled: The function Can_TTGetNTUActual() shall raise the error CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the parameter Can_TTNTUAct is a NULL pointer.

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_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed 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

1.4 Specification Item SWS_TtCan_00026

Trace References:

none

Content:

If development error detection for the Ttcan module is enabled: The function Can_TTGetErrorLevel() shall raise the error CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the parameter Can_TTErrorLevel is a NULL pointer.

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_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed 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

1.5 Specification Item SWS_TtCan_00059

Trace References:

none

Content:

If development error detection for the Ttcan module is enabled: The function Can_TTGetSyncQuality() shall raise the error CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the parameter Can_TTClockSpeed or the parameter Can_TTGlobal TimePhase is a NULL pointer.

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_litaral
* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed 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

1.6 Specification Item SWS_TtCan_00078

Trace References:

none

Content:

If development error detection for the Ttcan module is enabled: The function Can_TTGet TimeMarkIRQStatus() shall raise the error CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the parameter Can_TT IRQStatus is a NULL pointer.

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_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed 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

1.7 Specification Item SWS_TtCan_00112

Trace References:

none

Content:

If development error detection for the Ttcan module is enabled: The function Can_TTReceive() shall raise the error CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if one of the parameter CanId, CanDlc or CanSduPtr is a NULL pointer.

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_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed 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