MICROSAR CAN Transceiver driver

Technical Reference

Tja1043

Version 4.2.1

Authors	Timo Vanoni, Eugen Stripling, Andreas Weinrauch, Jan Hammer
Status	Released

1 Document Information

1.1 History of template

Author	Date	Version	Remarks
Timo Vanoni	2010-04-21	1.00.00	Creation

Table 1-1 History of the template

1.2 History of document

Author	Date	Version	Remarks
Timo Vanoni	2010-05-10	1.00.00	First implementation
Timo Vanoni	2011-05-02	1.01.00	Add support for multiple identity configuration
Eugen Stripling	2011-08-03	2.00.00	Implementation according to ASR3.2.1
Eugen Stripling	2012-01-23	2.01.00	Adapted according to R13 changes: New used service CanIf_TrcvWakeupIndication
Eugen Stripling	2013-02-06	3.00.00	Add support for ASR3.2.2 Add support for ASR4.0.3
Eugen Stripling	2014-10-31	4.00.00	AUTOSAR3 removed, Post-build selectable
Eugen Stripling	2015-07-29	4.00.01	ESCAN00081501
Eugen Stripling	2015-12-14	4.01.00	Chapter 8.3.3 added, chapter 3.2 adapted
Andreas Weinrauch	2016-12-06	4.02.00	Update to new CI template
Jan Hammer	2017-07-12	4.02.01	Add support for TCAN1043x-Q1 Add support for TJA1902A CAN2

Table 1-2 History of the document

1.3 Reference Documents

No.	Title	Version
[1]	AUTOSAR_SWS_CAN_TransceiverDriver.pdf	3.0.0
[2]	AUTOSAR_SWS_DET.pdf	2.2.1
[3]	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0
[4]	TJA1043.pdf	Rev. 5 - 23 May 2016
[5]	TCAN1043-Q1 datasheet.pdf	SLLSEV0 –JULY 2016
[6]	TJA1902A_Datasheet.pdf	Rev. 1.00 — 2 September 2016

Table 1-3 Reference documents

1.4 Scope of the Document

This technical reference describes the specific use of the CAN transceiver driver CanTrcv_30_Tja1043.

Please note
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module CANTRCV as specified in [1]. The CAN transceiver driver provides an abstraction layer for the used CAN transceiver hardware. It offers a hardware independent interface to the upper layer components.

Supported AUTOSAR Release:	4.0.3	
Supported Configuration Variants:	Pre-compile, Post-build selectable	
Vendor ID:	CANTRCV_30_TJA1043_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CANTRCV_30_TJA1043_MODULE_ID	[ModuleID] (according to ref. **[3]**)
Instance ID:	CANTRCV_30_TJA1043_INSTANCE_ID	Specified as pre-compile parameter in the Generation Tool.

2.1 Supported Hardware

Although the API includes the infix “Tja1043” this component supports several kinds of hardware:

- NXP Tja1043/T/TK
- TI TCAN1043x-Q1
- NXP Tja1902A [Only CAN2 of this Multi-Chip Module can be controlled by this driver]

Please note
The TJA1902A contains three fully independent transceivers in a single Multi-Chip Module (MCM), with this driver only one of them is supported (CAN2).

2.2 Pinning Information

The following table shows the relationship between the names of the DIO-pins modelled in the ECUC-configuration and the corresponding ones mentioned in the datasheet.

ECUC	NXP Tja1043/T/TK	TI TCAN1043x-Q1	NXP Tja1902A [CAN2]
PinEN	EN	EN	C2_EN
PinSTB	STB_N	nSTB	C2_STB_N
PinERR	ERR_N	nFAULT	C2_ERR_N

Table 2-1

3 Functional Description

3.1 Features

The features listed in this chapter cover the complete functionality specified in [1].

The supported and not supported features are presented in the following two tables.

Supported features
CAN Transceiver initialization.
CAN Transceiver control via DIO.
Detection of wakeup.
Getting and setting operation mode of CAN transceiver.

Table 3-1 Supported SWS features

Additional supported MICROSAR features
MICROSAR Identity Manager using Post-build selectable.

Table 3-2 Additional supported MICROSAR features

Not supported features
CAN Transceiver control via SPI.
CAN Transceiver self diagnostics.

Table 3-3 Not supported SWS features

The CAN transceiver driver provides service functions for initialization, operation mode change and operation mode detection of the used CAN transceiver hardware. Optional service functions and callback functions are provided to detect wakeup by bus events and report them to the upper layer components.

3.2 Initialization

After power on the CAN transceiver hardware has to be initialized. Therefore the CAN transceiver driver provides two service functions.
The function `CanTrcv_30_Tja1043_InitMemory` initializes all necessary values for the transceiver driver. This function has to be called first after a power-on or reset.
The function `CanTrcv_30_Tja1043_Init` initializes all CAN transceiver channels which are selected by the generation tool. Each CAN transceiver channel is switched into the operating mode: NORMAL. Note that `CanTrcv_30_Tja1043_InitMemory` has to be called before the function `CanTrcv_30_Tja1043_Init` is called.
If a wakeup event was pending before the initialization, the CAN transceiver driver does store this event and notifies it by calling of `EcuM_SetWakeupEvent` function.

3.3 Set operation mode

The operation mode of the CAN transceiver hardware is changed by service function `CanTrcv_30_Tja1043_SetOpMode`. It can be switched from normal into standby mode, from standby into sleep mode and from standby or sleep mode into normal mode. Mode change from normal into sleep mode or from sleep into standby mode is not supported by this function corresponding to the specification.

If the function is called with the same operation mode as already set, no mode change occurs and `E_OK` is returned.

3.4 Get operation mode

To retrieve the current operation mode of a specified CAN transceiver hardware, the service function `CanTrcv_30_Tja1043_GetOpMode` has to be called.

3.5 Get version info

The service function `CanTrcv_30_Tja1043_GetVersionInfo` can be called to get the version info of the software module. This function must be enabled in the configuration tool by setting the checkbox **Version Info Api**.
The version of the CAN transceiver driver module can be acquired in two different ways. Calling the function `CanTrcv_30_Tja1043_GetVersionInfo` will return the version of the module in the structure `Std_VersionInfoType` which additionally includes the VendorID and the ModuleID. Accessing the version defines which are specified in the header file `CanTrcv_30_Tja1043.h`:

Autosar Revision:

```
CANTRCV_30_TJA1043_AR_RELEASE_MAJOR_VERSION
CANTRCV_30_TJA1043_AR_RELEASE_MINOR_VERSION
CANTRCV_30_TJA1043_AR_RELEASE_PATCH_VERSION
```

Module Version:

```
CANTRCV_30_TJA1043_SW_MAJOR_VERSION
CANTRCV_30_TJA1043_SW_MINOR_VERSION
CANTRCV_30_TJA1043_SW_PATCH_VERSION
```

3.6 Wakeup by bus event detection

If wakeup by bus detection is enabled in the configuration tool the callback function `CanTrcv_30_Tja1043_CheckWakeup` has to be called by the lower layer in case of a wakeup. This function checks the specified CAN transceiver channel. If the transceiver hardware is in sleep or standby mode and a wakeup by bus event is detected the function returns `E_OK`, otherwise `E_NOT_OK`.

!

Caution
For determination whether a wakeup event occurred or not the flag "Wake" of the transceiver is evaluated. If the "Wake" flag is set then the reason for the wakeup event may be one of the following:

- local wakeup via WAKE-pin,
- remote wakeup by bus or
- power-on.

The CAN transceiver driver does not distinguish between them. Hence the API `CanTrcv_30_Tja1043_GetBusWakeupReason` returns only `CANTRCV_30_TJA1043_WU_BY_BUS` in case of occurrence of a wakeup event.

This behavior applies for the following hardware:

- NXP Tja1043/T/TK (see [4] chapter 7.2.4 Wake flag)
- TI TCAN1043x-Q1 (see [5] chapter 9.3.3 Wake-Up Request Flag)
- NXP Tja1902A (see [6] chapter 7.2.2.4 Wake flag)

3.6.1 Get bus wakeup reason

The service function `CanTrcv_30_Tja1043_GetBusWakeupReason` returns the reason which caused the wakeup. Please pay attention to caution mentioned in chapter 3.6 Wakeup by bus event detection.

3.6.2 Set wakeup mode

The service function `CanTrcv_30_Tja1043_SetWakeupMode` sets the wakeup mode which is required by the CAN interface.

3.6.3 Development Error Reporting

Development errors are reported to DET using the service `Det_ReportError`, if the pre-compile parameter `CANTRCV_30_TJA1043_DEV_ERROR_DETECT == STD_ON`.

The reported CANTRCV instance ID has to be specified in the configuration tool. For details please refer to chapter 7.1 Configuration with DaVinci Configurator 5.

The reported service IDs identify the services which are described in 6.1. The following table presents the service IDs and the related services:

Service ID	Service
0x00	CanTrcv_30_Tja1043_Init
0x01	CanTrcv_30_Tja1043_SetOpMode
0x02	CanTrcv_30_Tja1043_GetOpMode
0x03	CanTrcv_30_Tja1043_GetBusWuReason
0x05	CanTrcv_30_Tja1043_SetWakeupMode
0x07	CanTrcv_30_Tja1043_CheckWakeup
0x04	CanTrcv_30_Tja1043_GetVersionInfo
0x06	CanTrcv_30_Tja1043_MainFunction

Table 3-4 Mapping of service IDs to services

The errors reported to DET are described in the following table:

Error Code		Description
0x01	CANTRCV_30_TJA1043_E_INVALID_TRANSCEIVER	Invalid channel index is used in function argument list.
0x02	CANTRCV_30_TJA1043_E_PARAM_POINTER	Invalid pointer NULL_PTR is used in function argument list.
0x11	CANTRCV_30_TJA1043_E_UNINIT	CAN transceiver hardware is not initialized.
0x21	CANTRCV_30_TJA1043_E_TRCV_NOT_STANDBY	CAN transceiver hardware is not in standby mode.
0x22	CANTRCV_30_TJA1043_E_TRCV_NOT_NORMAL	CAN transceiver hardware is not in normal operation mode.
0x23	CANTRCV_30_TJA1043_E_PARAM_TRCV_WAKEUP_MODE	The requested wakeup mode is not valid.
0x24	CANTRCV_30_TJA1043_E_PARAM_TRCV_OPMODE	The requested operating mode is not supported by the underlying transceiver hardware.
0x40	CANTRCV_30_TJA1043_E_NO_TRCV_CONTROL	If the CAN transceiver is not under control, which means the transceiver does remain in an invalid state, this production error is raised.

Table 3-5 Errors reported to DET

4 Integration

This chapter gives necessary information for the integration of the MICROSAR CANTRCV into an application environment of an ECU. This component is only applicable on CAN transceiver hardware NXP Tja1043.

4.1 Scope of Delivery

The delivery of the CANTRCV contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
`CanTrcv_30_Tja1043.h`	Header file which has to be included by higher layers.
`CanTrcv_30_Tja1043.c`	Implementation

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool [config tool].

File Name	Description
`CanTrcv_30_Tja1043_Cfg.h`	Header file contains type definitions and external data declarations. It is included by `CanTrcv_30_Tja1043.h`.
`CanTrcv_30_Tja1043_Cfg.c`	Contains the configuration data.
`CanTrcv_GeneralTypes.h`	Header file with all CAN Transceiver types specified by SWS. This file can be included by `Can_GeneralTypes.h`.

Table 4-2 Generated files

4.2 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction definitions defined for the CANTRCV and illustrates their assignment among each other.

Memory Mapping Sections \ Compiler Abstraction Definitions	CANTRCV_30_TJA1043_VAR	CANTRCV_30_TJA1043_APPL_VAR	CANTRCV_30_TJA1043_CONST	CANTRCV_30_TJA1043_CODE	CANTRCV_30_TJA1043_APPL_CODE
CANTRCV_30_TJA1043_START_SEC_CODE CANTRCV_30_TJA1043_STOP_SEC_CODE				■	■
CANTRCV_30_TJA1043_START_SEC_CONST_UNSPECIFIED CANTRCV_30_TJA1043_STOP_SEC_CONST_UNSPECIFIED			■		
CANTRCV_30_TJA1043_START_SEC_VAR_NOINIT_UNSPECIFIED CANTRCV_30_TJA1043_STOP_SEC_VAR_NOINIT_UNSPECIFIED	■	■			

Table 4-3 Compiler abstraction and memory mapping

4.3 Data consistency

The CAN transceiver driver calls service functions of upper layers in order to prevent interruption when accessing the CAN transceiver pins.

These service functions have to be provided by the components VStdLib, Schedule Manager or OSEK OS depending on which of these components is used for interrupt disable / restore handling. The component for interrupt control handling has to be selected in the configuration tool.

4.4 Exclusive Areas

The transceiver driver uses the following exclusive areas.

4.4.1 CANTRCV_EXCLUSIVE_AREA_0

This section ensures consistent handling of CanTrcv hardware via the DIO interface.

> Protects: atomic / consistent reading of DIO-pins and setting of them.

> Used in: `CanTrcv_30_Tja1043_Init()`, `CanTrcv_30_Tja1043_SetOpMode()`, `CanTrcv_30_Tja1043_GetOpMode()`, `CanTrcv_30_Tja1043_CheckWakeup()` and `CanTrcv_30_Tja1043_MainFunction()`.

> Exclude call of CanTrcv driver APIs: `CanTrcv_30_Tja1043_Init()`, `CanTrcv_30_Tja1043_SetOpMode()`, `CanTrcv_30_Tja1043_GetOpMode()`, `CanTrcv_30_Tja1043_CheckWakeup()` and `CanTrcv_30_Tja1043_MainFunction()`
>
> AND
>
> In fact the instructions within the exclusive area must be executed atomic without any interruption / delay in between.

> Duration: Used in calls to DIO module – SHORT duration

5 Dependencies to other components

5.1 Dio driver

The CAN transceiver driver performs hardware access by calling service functions of the lower layer component Dio driver:

> Function `Dio_WriteChannel` is used to set the logical level of the channel pins the CAN transceiver hardware is connected to.

> Function `Dio_ReadChannel` is used to get the logical level of the channel pins the CAN transceiver hardware is connected to.

> The Dio driver has to provide the pin assignment for the CAN transceiver hardware pins EN, STB and ERR. These pins are referred by the CAN transceiver driver using the symbolic names specified in the configuration tool.

5.2 Icu driver

The CAN transceiver driver performs hardware access by calling service functions of the lower layer component Icu driver:

> Function `Icu_DisableNotification()` is used by the transceiver driver to disable the ICU notification after wakeup event notification.

> Function `Icu_EnableNotification()` is used by the transceiver driver to enable the ICU notification during the transition into the STANDBY mode.

Please note
The following chapter applies only to this use case:
- the used CAN controller does not support the feature "wakeup by CAN bus", i.e. the CAN controller can not detect incoming CAN messages and generate a so-called wakeup-interrupt
- the ECU shall wake up and start its own communication due to detected communication on the CAN bus

The proposal described in this chapter enables the user to support the use-case by using an additional µC I/O port to generate the wakeup information via the ICU driver. This I/ O port is connected either parallel to the CAN Rx port or is directly attached to the CAN transceivers ERR port (depending on the used CAN transceiver). The following steps have to be done for every CAN channel that shall be able to wake up via the CAN bus:

5.2.1 ICU configuration

Add an ICU channel

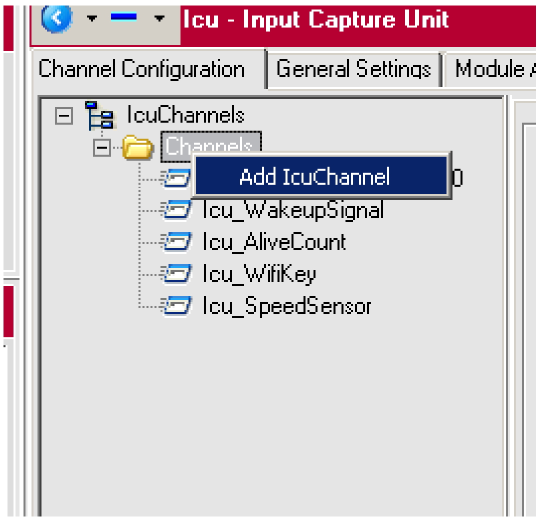
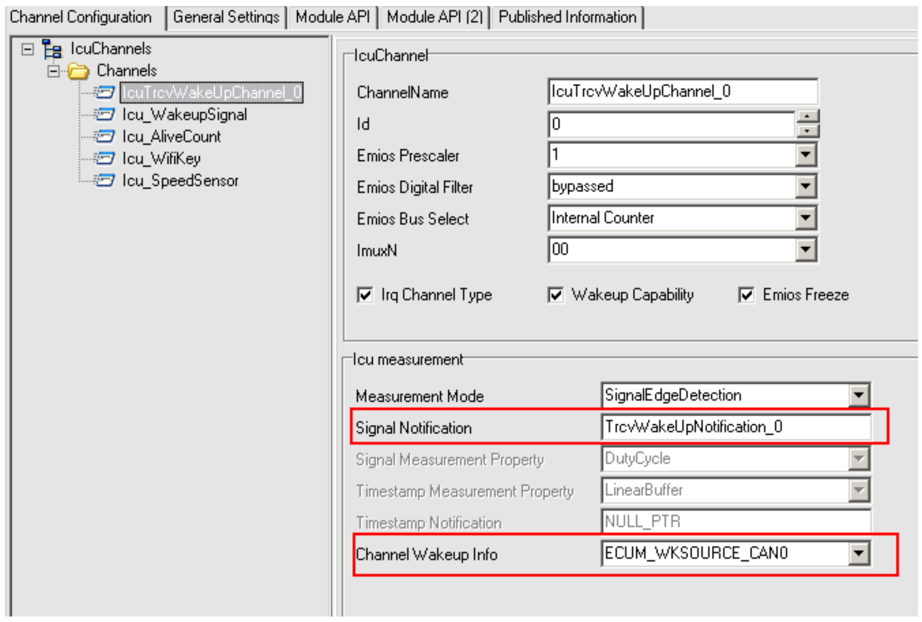


Figure 5-1 Add an ICU channel

The user has to configure the ICU channel and to add a signal notification function.

Additionally the wakeup source for the CAN channel which shall be handled via this ICU channel have to be chosen.

In dependency of the provided ICU configuration options it is possible to configure the “IcuDefaultStartEdge”. This option should be configured to ICU_FALLING_EDGE, because the wakeup event is provided by the Rx pin and/ or the ERR pin via a level change from recessive to dominant.



IcuChannel

ChannelName	IcuTrcvWakeUpChannel_0
Id	0
Emios Prescaler	1
Emios Digital Filter	bypassed
Emios Bus Select	Internal Counter
ImuxN	00

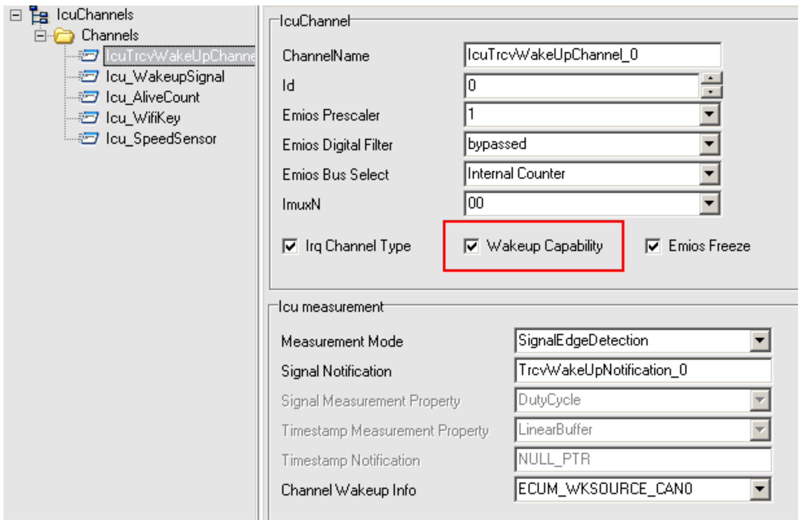
☑ Irq Channel Type ☑ Wakeup Capability ☑ Emios Freeze

Icu measurement

Measurement Mode	SignalEdgeDetection
Signal Notification	TrcvWakeUpNotification_0
Signal Measurement Property	DutyCycle
Timestamp Measurement Property	LinearBuffer
Timestamp Notification	NULL_PTR
Channel Wakeup Info	ECUM_WKSOURCE_CAN0

Figure 5-2 ICU channel configuration for wakeup via transceiver

If the transceiver shall also wake up the ECU and not only the CAN channel then it is necessary to activate the “Wakeup Capability” for this ICU channel.



IcuChannels
- Channels
  - IcuTrcvWakeUpChanne
  - Icu_WakeupSignal
  - Icu_AliveCount
  - Icu_WifiKey
  - Icu_SpeedSensor

IcuChannel

ChannelName	IcuTrcvWakeUpChannel_0
Id	0
Emios Prescaler	1
Emios Digital Filter	bypassed
Emios Bus Select	Internal Counter
ImuxN	00

✓ Irq Channel Type ✓ Wakeup Capability ✓ Emios Freeze

Icu measurement

Measurement Mode	SignalEdgeDetection
Signal Notification	TrcvWakeUpNotification_0
Signal Measurement Property	DutyCycle
Timestamp Measurement Property	LinearBuffer
Timestamp Notification	NULL_PTR
Channel Wakeup Info	ECUM_WKSOURCE_CAN0

Figure 5-3 ICU wakeup capability

5.2.2 Implementation of the signal notification function

The user has to implement the signal notification function as shown in the following example:

Example

```
void Icu_TrcvWakeUpNotification_0(void)
{

  /* inform the EcuM about the wakeup event, the parameter
   * is the configured transceiver wakeup source */
  EcuM_CheckWakeup(ECUM_WKSOURCE_CAN0);
}
```

Please note
If no wakeup validation is used for the configured wakeup source, then it is possible to call `EcuM_SetWakeupEvent()` instead of `EcuM_CheckWakeup()`.

6 API Description

6.1 Services provided by CANTRCV

The CANTRCV API consists of services, which are realized by function calls.

6.1.1 CanTrcv_30_Tja1043_InitMemory

Prototype	
`void` **`CanTrcv_30_Tja1043_InitMemory`** `( void )`	
Parameter	
-	-
Return code	
`void`	none
Functional Description	
This function initializes the memory and needed values of the CAN transceiver driver.	
Particularities and Limitations	
> This function must be called before any other functionality of the CAN transceiver driver. > Configuration Variant(s): -	
Expected Caller Context	
This function must be called from task level and is not reentrant.	

Table 6-1 CanTrcv_30_Tja1043_InitMemory

6.1.2 CanTrcv_30_Tja1043_Init

Prototype	
`void` **`CanTrcv_30_Tja1043_Init`** `(CanTrcv_30_Tja1043_ConfigType* ConfigPtr)`	
Parameter	
`ConfigPtr[in]`	Pointer to the `CanTrcv_30_Tja1043_Config` struct. If multiple configurations are available, the active configuration can be selected by using the related `CanTrcv_30_Tja143_Config_<IdentityName>` structure.
Return code	
`void`	none
Functional Description	
This function initializes all channels of the CAN Transceiver driver which are configured in the configuration tool. This function has the service id 0x00.	
Particularities and Limitations	
> The function `CanTrcv_30_Tja1043_InitMemory` must be called before the function `CanTrcv_30_Tja1043_Init` can be called. > This function must be called before any other service functionality of the Transceiver driver. > The DIO driver must be initialized. > Configuration Variant(s): -	
Expected Caller Context	
This function must be called from task level and is not reentrant.	

Table 6-2 CanTrcv_30_Tja1043_Init

6.1.3 CanTrcv_30_Tja1043_SetOpMode

Prototype	
`Std_ReturnType` **`CanTrcv_30_Tja1043_SetOpMode`** `(uint8 CanTrcvIndex, CanTrcv_TrcvModeType OpMode )`	
Parameter	
`CanTrcvIndex[in]`	Index of the selected transceiver channel to which API call has to be applied.
`OpMode[in]`	Requested mode to which the transceiver state has to be changed.
Return code	
`E_OK / E_NOT_OK`	`E_OK`: is returned if the transceiver has accepted the request to change to the requested mode. `E_NOT_OK`: is returned if the transceiver state cannot be accepted or the parameter is out of the allowed range. The previous state has not been changed.
Functional Description	
Sets the CAN transceiver to the requested operation mode. These operation modes are: > `CANTRCV_TRCVMODE_NORMAL` > `CANTRCV_TRCVMODE_STANDBY` > `CANTRCV_TRCVMODE_SLEEP` If return code is E_OK the notification function `CanIf_30_Tja1043_TrcvModeIndication` will be called if mode change is completed. Note that the notification will occur in context of `CanTrcv_30_Tja1043_SetOpMode`. This function has the service id 0x01.	
Particularities and Limitations	
> The CAN transceiver driver must be initialized. > Not each transition from one mode in any other is allowed. For these limitations please refer to chapter Set operation mode. > Configuration Variant(s): -	
Expected Caller Context	
This function can called from task or interrupt level and is not reentrant.	

Table 6-3 CanTrcv_30_Tja1043_SetOpMode

6.1.4 CanTrcv_30_Tja1043_GetOpMode

Prototype	
`Std_ReturnType` **`CanTrcv_30_Tja1043_GetOpMode`** `( uint8 CanTrcvIndex, CanTrcv_TrcvModeType *OpMode )`	
Parameter	
`CanTrcvIndex [in]`	Index of the selected transceiver channel to which API call has to be applied.
`OpMode [out]`	Pointer to buffer where the current operation mode is stored.
Return code	
`E_OK / E_NOT_OK`	`E_OK`: is returned if the operation mode was detected. `E_NOT_OK`: is returned if transceiver operation mode is not initialized.
Functional Description	
Stores the current operation mode of the selected CAN transceiver to `OpMode`. These operation modes are: > `CANTRCV_CANTRCV_NORMAL` > `CANTRCV_CANTRCV_STANDBY` > `CANTRCV_CANTRCV_SLEEP` The mode is determined from status of PINs of the transceiver hardware. This function has the service id 0x02.	
Particularities and Limitations	
> The CAN transceiver driver must be initialized. > If a mode change was requested before, the reported operation mode may not be valid until the mode change is completed and `CanIf_30_Tja1043_TrcvModeIndication` was called. > Configuration Variant(s): -	
Expected Caller Context	
This function can be called from task or interrupt level and is not reentrant.	

Table 6-4 CanTrcv_30_Tja1043_GetOpMode

6.1.5 CanTrcv_30_Tja1043_GetBusWuReason

Prototype	
`Std_ReturnType` **`CanTrcv_30_Tja1043_GetBusWuReason`** `(uint8 CanTrcvIndex, CanTrcv_TrcvWakeupReasonType *Reason )`	
Parameter	
`CanTrcvIndex[in]`	Index of the selected transceiver channel to which API call has to be applied.
`Reason[out]`	Pointer to buffer where the bus wake-up reason is stored.
Return code	
`E_OK / E_NOT_OK`	`E_OK`: is returned if the wake up reason was detected. `E_NOT_OK`: is returned if bus wake-up reason is not detected or feature is not supported.
Functional Description	
Stores the last wakeup reason for the channel `CanTrcvIndex` to `Reason`. These wakeup reasons are: > `CANTRCV_WU_INTERNALLY`: The wakeup was caused by setting the CAN transceiver in operation mode via `CanTrcv_30_Tja1043_SetOpMode`. > `CANTRCV_WU_ERROR`: No wakeup was detected by the transceiver and no reason is stored. The function returns `E_NOT_OK`. > `CANTRCV_WU_NOT_SUPPORTED`: No wakeup detection supported by this CAN transceiver. The function returns `E_NOT_OK`. > `CANTRCV_WU_BY_BUS`: The wakeup was caused by an external bus wakeup. > `CANTRCV_WU_RESET`: The wakeup was detected after a reset. The wake-up reason is read from state variable. No access to transceiver hardware is performed. This function has the service id 0x03.	
Particularities and Limitations	
> The CAN transceiver driver must be initialized. > The wakeup reason represents always the last detected wakeup reason. If there was more than one wakeup detected only the last one will be reported. > Configuration Variant(s): -	
Expected Caller Context	
This function can be called from task or interrupt level and is not reentrant.	

Table 6-5 CanTrcv_30_Tja1043_GetBusWuReason

6.1.6 CanTrcv_30_Tja1043_SetWakeupMode

Prototype	
`Std_ReturnType` **`CanTrcv_30_Tja1043_SetWakeupMode`** `(uint8 CanTrcvIndex, CanTrcv_TrcvWakeupModeType TrcvWakeupMode )`	
Parameter	
`CanTrcvIndex[in]`	Index of the selected transceiver channel to which API call has to be applied.
`TrcvWakeupMode[in]`	Requested wake-up mode for the transceiver channel (`CanTrcvIndex`).
Return code	
`E_OK / E_NOT_OK`	`E_OK`: is returned, if the wakeup state has been changed to the requested mode. `E_NOT_OK`: is returned, if the wakeup state change has failed or the parameter is out of the allowed range. The previous state has not been changed.
Functional Description	
Enables and disables the reporting from wakeup events on the channel `CanTrcvIndex`. > `CANTRCV_WUMODE_ENABLE`: Report wakeup events to upper layer. > `CANTRCV_WUMODE_DISABLE`: Do not report wakeup events to upper layer. > `CANTRCV_WUMODE_CLEAR`: Clear a pending wakeup event. This function has the service id 0x05.	
Particularities and Limitations	
> The CAN transceiver driver must be initialized. > If wakeup handling is not enabled in configuration tool the function always return `E_NOT_OK`. > Configuration Variant(s): -	
Expected Caller Context	
This function is called from task or interrupt level and not reentrant.	

Table 6-6 CanTrcv_30_Tja1043_SetWakeupMode

6.1.7 CanTrcv_30_Tja1043_GetVersionInfo

Prototype	
`void` **`CanTrcv_30_Tja1043_GetVersionInfo`** `(Std_VersionInfoType *VersionInfo)`	
Parameter	
`VersionInfo[out]`	Structure pointer to version information of this module.
Return code	
`void`	none
Functional Description	
Get the version info of the module and store it into the structure pointed by `VersionInfo`. This function has the service id 0x04.	
Particularities and Limitations	
> The CAN transceiver driver must be initialized. > Configuration Variant(s): `CANTRCV_30_TJA1043_GET_VERSION_INFO == STD_ON`	
Expected Caller Context	
This function can be called from task or interrupt level and is not reentrant.	

Table 6-7 CanTrcv_30_Tja1043_GetVersionInfo

6.1.8 CanTrcv_30_Tja1043_CheckWakeup

Prototype	
`Std_ReturnType` **`CanTrcv_30_Tja1043_CheckWakeup`** `( uint8 CanTrcvIndex )`	
Parameter	
`CanTrcvIndex[in]`	Index of the selected transceiver channel to which API call has to be applied.
Return code	
`E_OK / E_NOT_OK`	`E_OK` a wakeup-by-bus event was detected. `E_NOT_OK` no wakeup was detected or an error occurred.
Functional Description	
This function requests the CAN transceiver driver to check for wakeups and to report them. If a wakeup was detected, the CAN Transceiver reports it by calling of `EcuM_SetWakeupEvent`. This function has the service id 0x07.	
Particularities and Limitations	
> The CAN transceiver driver must be initialized. > Do not use the return value `E_OK` for wakeup detection. A wakeup can be considered as valid only if `EcuM_SetWakeupEvent` was called for the corresponding wakeup source. > Configuration Variant(s): -	
Call context	
This function can be called from task or interrupt level and is not reentrant.	

Table 6-8 CanTrcv_30_Tja1043_CheckWakeup

6.1.9 CanTrcv_30_Tja1043_MainFunction

Prototype	
`void` **`CanTrcv_30_Tja1043_MainFunction`** `(void)`	
Parameter	
-	-
Return code	
`void`	none
Functional Description	
This service can be called from task level and periodically checks if a wakeup was detected by the underlying transceiver hardware. This function has the service id 0x06.	
Particularities and Limitations	
> The CAN transceiver driver must be initialized. > This service only has effect if Wakeup support is set to polling. > Configuration Variant(s): -	
Expected Caller Context	
This function can be called from task context and is not reentrant.	

Table 6-9 CanTrcv_30_Tja1043_MainFunction

6.2 Services used by CANTRCV

In the following table services provided by other components, which are used by the CANTRCV are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	`Det_ReportError`
DIO	`Dio_WriteChannel`
DIO	`Dio_ReadChannel`
CANIF	`CanIf_TrcvModeIndication`
ICU	`Icu_EnableNotification`
ICU	`Icu_DisableNotification`

Table 6-10 Services used by the CANTRCV

7 Configuration

7.1 Configuration with DaVinci Configurator 5

Refer to the integrated online help and parameter descriptions of Configurator 5.

8 AUTOSAR Standard Compliance

8.1 Additions/ Extensions

8.1.1 Memory initialization

To have an independent memory initialization for this BSW module the additional function `CanTrcv_30_Tja1043_InitMemory` was added.

8.2 Limitations

8.2.1 Support of SPI

This CAN Transceiver driver does not support the connection to a CAN transceiver via SPI.

8.3 Deviations

While the driver is implemented according [1], some requirements could not be fulfilled in order to ensure proper functionality. The following chapter lists these deviations.

8.3.1 Notification functions

According to SWS [1] the transceiver shall call notification functions in CanIf. As the given CanTrcvChannelId is valid only for one transceiver driver instance, it was decided to call the following notification functions instead:

SWS API	Used API
`CanIf_TrcvModeIndication`	`CanIf_30_Tja1043_TrcvModeIndication`

Table 8-1 Deviation of APIs used by CanTrcv

Within these functions recalculation of the given `CanTrcvIndex` has to be performed so that the local index of the driver matches the global index of the CanIf.

Affected requirements: CanTrcv086, CanTrcv222, CanTrcv239, CanTrcv238

8.3.2 Unused BSWMD parameters

According to SWS [1], the parameters `CanTrcvSPICommRetries` and `CanTrcvSPICommTimeout` shall have the multiplicity 1. As the SWS does not describe where to use these values, it was decided to set multiplicity to 0..1 so they do not have to be used.

Affected requirements: CanTrcv172, CanTrcv171

8.3.3 Initialization of operating mode

According to SWS [1] each CAN transceiver channel shall be initialized to operating mode which is configured by parameter `CanTrcvInitState`. Independent of configuration each CAN transceiver channel is initialized into operating mode NORMAL.

Affected requirements: CanTrcv100, CanTrcv148.

9 Glossary and Abbreviations

9.1 Glossary

Term	Description
Cfg5	DaVinci Configurator 5

Table 9-1 Glossary

9.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CANIF	CAN Interface
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
ECUC	Electronic Control Unit Configuration
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 9-2 Abbreviations

10 Contact

Visit our website for more information on

> News
> Products
> Demo software
> Support
> Training data
> Addresses

www.vector-informatik.com