MICROSAR FlexRay Transceiver Driver

Technical Reference

Tja1082

Version 2.00.00

Status	Released

Document Information

History

Date	Version	Remarks
2014-05-15	1.00.00	Creation of document
2015-05-27	1.00.01	ESCAN00078929: Missing explanation of API FrTrcv_30_Tja1082_GetVersionInfo ESCAN00077241 AR3-2679: Description BCD-coded return-value of XXX_GetVersionInfo() in TechRef
2016-11-04	1.01.00	Support of AUTOSAR 3
2017-08-25	2.00.00	Rework for SafeBsw

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_FlexRayTransceiverDriver.pdf	1.5.0
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	2.2.1
[3]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	2.2.0
[4]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	1.0.0
[5]	NXP	TJA1082.pdf	Rev.6
[6]	NXP	TJA1083.pdf	Rev.1

Scope of the Document

This technical reference describes the general use of the FlexRay Transceiver Driver basis software for Tja1082 or Tja1083. Please refer to your Release Notes to get a detailed description of the platform (Host, CC, Compiler, Transceiver) your Vector FlexRay Bundle has been configured for.

Caution
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

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.00.00	ESCAN00075721 Creation of component
1.01.00	ESCAN00092491 Support AR3
2.00.00	STORY-1874 Create Safe BSW Transceiver Driver for Tja1082

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module FlexRay Transceiver Driver as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	FlexRay Transceiver Driver_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	FlexRay Transceiver Driver_MODULE_ID	71 decimal (according to ref. see [4])

* For the precise AUTOSAR Release 4.x please see the release specific documentation.

The FlexRay Transceiver Driver provides hardware independent access to control connected Transceivers in a generic way. It offers the functionality to control the mode of operation of connected Transceivers as well as to determine their current state, e.g. if events like wake up or bus errors happened.

The Transceiver itself is a hardware device, which mainly transforms the logical 1/0 signals of the FlexRay Controller to the bus compliant electrical levels, currents and timings.

2.1 Supported Devices

There are devices that are compatible to the Tja1082 regarding the host side interface. The devices, supported by this Transceiver Driver, are:

Device	Device Data Sheet	Version
TJA1082	TJA1082.pdf	Rev. 6
TJA1083	TJA1083.pdf	Rev. 1

2.2 Architecture Overview

The following figure shows where the FlexRay Transceiver Driver is located in the AUTOSAR architecture.

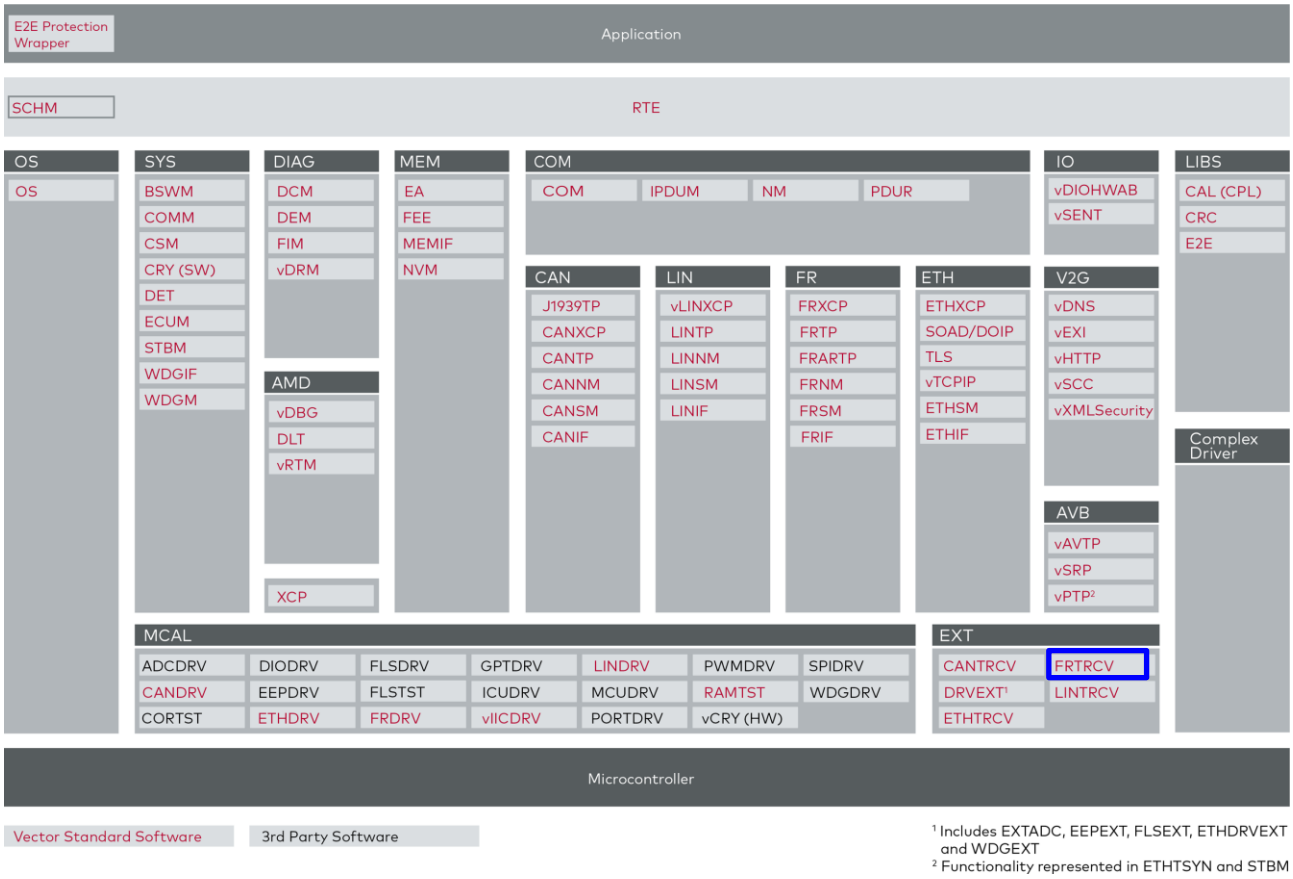

Figure 2-1 AUTOSAR 4.x Architecture Overview

3 Functional Description

3.1 Features

The AUTOSAR standard functionality is specified in [1].

3.1.1 Deviations

3.1.1.1 AUTOSAR 4

The general environment of the driver follows the AUTOSAR 4.1.2 specification.

The following features are not supported:

Not supported AUTOSAR Standard Conform Features
"Active Star" Mode (not supported by HW)
Debugging Support (AUTOSAR Debugging Concept)
Init Pointer
Branch Handling (not supported by HW)
Configurable DEM APIs
Initialization Retries
Usage of TM module for busy waiting

3.1.2 Limitations

3.1.2.1 Error indication

Currently only simple error mode is supported and hence no detailed error analysis is possible.

If your Transceiver is not connected to the SPI it should be externally wired such that it is automatically in simple error indication mode.

If your Transceiver is connected to the SPI use the Chip Select line and the Clock line to bring your Transceiver into simple error indication mode before initializing the SPI Interface. Keep your Chip Select line as DIO port instead of using it for the SPI interface. This way the Transceiver can be safely kept in the simple error indication mode. Refer to the device datasheet for timing limitations.

3.2 Initialization

3.2.1 High-Level Initialization

The Transceiver Driver is initialized by calling the FrTrcv_30_Tja1082_Init. The default operation mode of the Transceiver after Init is pre-defined in the configuration tool during configuration process.

If a startup code is used that does not initialize the memory please call FrTrcv_30_Tja1082_InitMemory first.

3.2.2 Low-Level Initialization

The user is responsible to initialize all I/O-ports used by the Transceiver before any Transceiver Driver function is called. The same holds true for the ICU module, if it is used.

3.3 States

The Transceiver supports two states that can be set:

- `FRTRCV_TRCVMODE_NORMAL`
- `FRTRCV_TRCVMODE_STANDBY`

After initialization the Transceiver is in a predetermined state which has been configured in the configuration tool..

3.4 Main Functions

The Transceiver Driver has one FrTrcv_30_Tja1082_MainFunction which has to be called cyclically. This task is responsible for polling all connected Transceivers and perform action if so required.

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled.

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported FlexRay Transceiver Driver ID is 71.

The reported service IDs identify the services which are described in [1]. The following table presents the service IDs and the related services:

Service ID	Service
-	`FrTrcv_30_Tja1082_InitMemory()`
0	`FrTrcv_30_Tja1082_Init()`
13	`FrTrcv_30_Tja1082_MainFunction()`
7	`FrTrcv_30_Tja1082_GetVersionInfo()`
1	`FrTrcv_30_Tja1082_SetTransceiverMode()`
5	`FrTrcv_30_Tja1082_GetTransceiverMode()`
6	`FrTrcv_30_Tja1082_GetTransceiverWUReason()`
12	`FrTrcv_30_Tja1082_ClearTransceiverWakeup()`
15	`FrTrcv_30_Tja1082_DisableTransceiverBranch()`
16	`FrTrcv_30_Tja1082_EnableTransceiverBranch()`
8	`FrTrcv_30_Tja1082_GetTransceiverError()`

Service ID	Service
14	FrTrcv_30_Tja1082_CheckWakeupByTransceiver()

Table 3-1 Service IDs

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

Error Code	Description
0x01	FRTRCV_30_TJA1082_E_FR_INVALID_TRCVIDX
0x03	FRTRCV_30_TJA1082_E_FR_INVALID_MODE
0x10	FRTRCV_30_TJA1082_E_FR_UNINIT
0x11	FRTRCV_30_TJA1082_E_FR_TRCV_NOT_STANDBY
0x12	FRTRCV_30_TJA1082_E_FR_TRCV_NOT_NORMAL
0x15	FRTRCV_30_TJA1082_E_FR_TRCV_NULL_PTR
0x16	FRTRCV_30_TJA1082_E_FR_NO_CONTROL_TRCV

Table 3-2 Errors reported to DET

3.5.2 Production Code Error Reporting

By default, production code related errors are reported to the DEM using the service Dem_ReportErrorStatus() as specified in [3], if production error reporting is enabled.

If another module is used for production code error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service Dem_ReportErrorStatus().

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

Error Code	Description
FRTRCV_30_TJA1082_E_FR_ERRN_TRCV**x**	If the Transceiver signals an error condition the Transceiver Driver will notify the DEM with this error code.

Table 3-3 Errors reported to DEM

4 Integration

This chapter gives necessary information for the integration of the MICROSAR FlexRay Transceiver Driver into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the FlexRay Transceiver Driver contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
FrTrcv_30_Tja1082.c	Source code of Transceiver Driver.
FrTrcv_30_Tja1082.h	API definitions of the Transceiver Driver.
FrTrcv_30_Tja1082_Cbk.h	Call-back header of the Transceiver Driver

Table 4-1 Static files

4.1.2 Dynamic Files

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

File Name	Description
FrTrcv_30_Tja1082_Cfg.c	Parameter Configuration source file for Transceiver Driver.
FrTrcv_30_Tja1082_Cfg.h	Parameter Configuration header file for Transceiver Driver.

Table 4-2 Generated files

4.2 Critical Sections

The FlexRay Transceiver Driver calls service functions of upper layers in order to prevent interruption of critical sections (e.g. accessing Transceiver pins).

These service functions have to be provided (normally by the Schedule Manager) and configured accordingly. The following critical areas are used:

4.2.1 FRTRCV_30_TJA1082_EXCLUSIVE_AREA_0

This exclusive area is used to lock transceiver functionality against interruption by each other. They may be interrupted by application functionality or time critical CAT1 interrupts.

4.3 The Software Timers

In order to access the Transceiver, a certain timing is required by the Transceiver Driver software. To generate this timing a software timer callback function named `Appl_FrTrcv_30_Tja1082_Wait` is used. This function has to be implemented by the user to generate the correct timing. The following implementation is an example and has to be completed to generate the correct timing:

Example

```
#include "FrTrcv_30_Tja1082.h"
#include "FrTrcv_30_Tja1082_Cbk.h"


FUNC(void, FRTRCV_30_TJA1082_CODE) Appl_FrTrcv_30_Tja1082_Wait(uint8 TimerIndex)
{
  switch(TimerIndex)
  {
    case kFrTrcv_30_Tja1082_SetMode:
      /* Insert code to delay for td(norm-stb) and td(stb-norm)
         Please refer to the device datasheet for correct numbers */
    break;

    case kFrTrcv_30_Tja1082_ModeChange:
      /* Insert code to delay until the Mode change is completed by
         the Transceiver device */
    break;
  }
}
```

To verify correct Timing, measurement of the delay with an Oscilloscope is recommended. Please take into consideration to map this function to the same context as the Transceiver Driver. Otherwise a call of this function might cause an context switch which would increase the delay considerably.

Used Timer: `kFrTrcv_30_Tja1082_SetMode`

This timer is used to make certain the ERRN pin is stable in the requested mode. If this delay is not used, it might be possible that an error might trigger a wake up notification.

Used Timer: `kFrTrcv_30_Tja1082_ModeChange`

This timer is used to make certain that the Transceiver device completed the mode change until code execution is continued.

5 API Description

The AUTOSAR Transceiver Driver provides the following services:

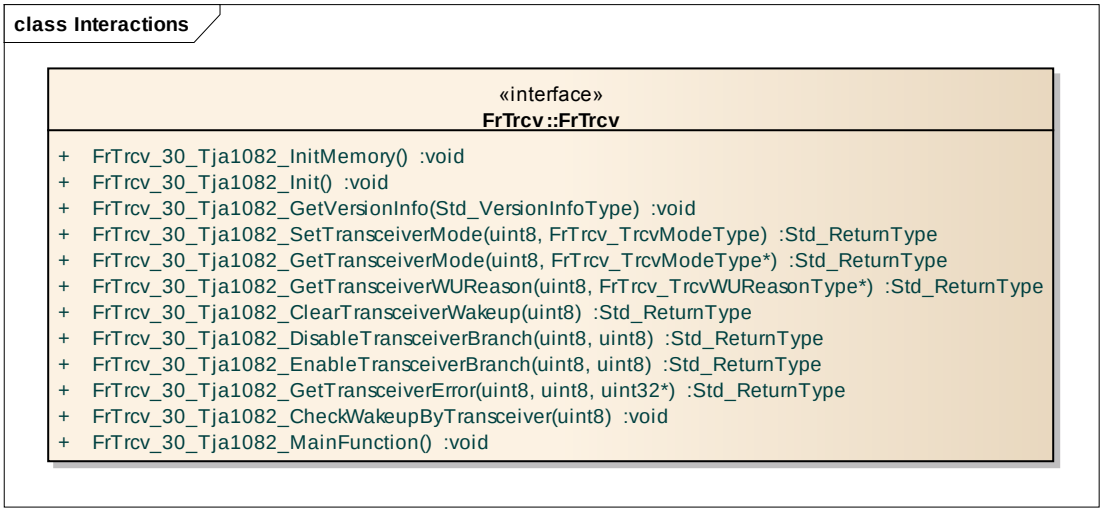

Figure 5-1 Interface Overview FlexRay Transceiver Driver

5.1 Type Definitions

The types defined by the FlexRay Transceiver Driver are described in this chapter.

Type Name	C-Type	Description	Value Range
FrTrcv_TrcvModeType	uint8	Defines all possible Transceiver modes	FRTRCV_TRCVMODE_UNKNOWN Temporary state before initalization
			FRTRCV_TRCVMODE_NORMAL Normal operation mode
			FRTRCV_TRCVMODE_STANDBY Standby operation mode
FrTrcv_ TrcvWUReasonType	uint8	The reason for the last recent wakeup	FRTRCV_WU_NOT_SUPPORTED The Transceiver does not support any information for the wake up reason.
			FRTRCV_WU_BY_BUS The Transceiver has detected that the bus has caused the wake up of the ECU.
			FRTRCV_WU_INTERNALLY The transceiver has detected that the "wake up" is due to an internal mode change.
			FRTRCV_WU_RESET The Transceiver has detected that the "wake up" is due to

Type Name	C-Type	Description	Value Range
			an ECU reset.

Table 5-1 Type definitions

The following structures are used in the configuration file.

FrTrcv_30_Tja1082_GenConfigType

This structure contains general configuration information per channel

Struct Element Name	C-Type	Description
FrTrcv_InitState	FrTrcv_TrcvModeType	Contains the initial Transceiver state after initialization
FrTrcv_WakeupSourceRef	EcuM_WakeupSourceType	Contains the Wakeup bit mask used to trigger an wake up event in the EcuM
FrTrcv_WakeupSupported	uint8	Defines whether wake up is supported on this channel
FrTrcv_WakeupPolling	uint8	Defines whether wake up event shall be polled
FrTrcv_ChannelUsed	uint8	Defined whether this channel is used at all

Table 5-2 FrTrcv_30_Tja1082_GenConfigType

FrTrcv_30_Tja1082_ChannelType

This structure contains DIO pin name configuration information per channel

Struct Element Name	C-Type	Description
TrcvPinSTBN	Dio_ChannelType	STBN pin name
TrcvPinERRN	Dio_ChannelType	ERRN pin name

Table 5-3 FrTrcv_30_Tja1082_ChannelType

5.2 Services provided by FlexRay Transceiver Driver

5.2.1.1 FrTrcv_30_Tja1082_InitMemory: Initialization of Transceiver Driver

FrTrcv_30_Tja1082_InitMemory

Prototype	
`void FrTrcv_30_Tja1082_InitMemory( void );`	
Parameters [in/out/both]	
`Void`	-
Return code	
`Void`	-
Service ID	
Service ID	-
Functional Description	
Initialization of the Transceiver Driver memory in case no start-up code is used that zeroes out the memory.	
Preconditions	
None.	
Postconditions	
The Transceiver Driver memory is initialized.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

Table 5-4 FrTrcv_30_Tja1082_InitMemory

5.2.1.2 FrTrcv_30_Tja1082_Init: Initialization of Transceiver Driver

FrTrcv_30_Tja1082_Init

Prototype	
`void FrTrcv_30_Tja1082_Init( void );`	
Parameters [in/out/both]	
`Void`	-
Return code	
`Void`	-
Service ID	
Service ID	0
Functional Description	
Initialization of the Transceiver Driver module as well as the physical Transceiver itself.	

Preconditions
The I/O ports, used to access the Transceiver, have to be initialized!
Postconditions
The Transceiver will be initialized to the configured operation state.
Particularities and Limitations
> Call context: task level > Not re-entrant > Synchronous

Table 5-5 FrTrcv_30_Tja1082_Init

5.2.1.3 FrTrcv_30_Tja1082_MainFunction: Main Function of Transceiver Driver

FrTrcv_30_Tja1082_MainFunction

Prototype	
`void FrTrcv_30_Tja1082_MainFunction( void );`	
Parameters [in/out/both]	
–	
Return code	
`void`	-
Service ID	
Service ID	13
Functional Description	
Main function of the Transceiver Driver for one instance. This service polls the respective Transceiver for any wake up events. In case a wake up is detected and notifications are allowed the ECU Manager is notified via `EcuM_SetWakeupEvent`.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
If enabled a callback in case of a wake-up event is triggered.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

Table 5-6 FrTrcv_30_Tja1082_MainFunction

5.2.1.4 FrTrcv_30_Tja1082_GetVersionInfo: Read Version Information of the Driver

FrTrcv_30_Tja1082_GetVersionInfo

Prototype	
`void FrTrcv_30_Tja1082_GetVersionInfo( P2VAR(Std_VersionInfoType, AUTOMATIC, FRTRCV_APPL_DATA) versioninfo );`	
Parameters [in/out/both]	
`Versioninfo [out]`	Pointer to the location where the Version information shall be stored.
Return code	
`void`	-
Service ID	
Service ID	7
Functional Description	
FrTrcv_30_Tja1082_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component. The versions are BCD-coded.	
Preconditions	
The API is only available if the feature is enabled.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

Table 5-7 FrTrcv_30_Tja1082_GetVersionInfo

5.2.1.5 FrTrcv_30_Tja1082_SetTransceiverMode: Set the Transceiver in the requested mode

FrTrcv_30_Tja1082_SetTransceiverMode

Prototype	
`Std_ReturnType FrTrcv_30_Tja1082_SetTransceiverMode ( uint8 FrTrcv_TrcvIdx, FrTrcv_TrcvModeType FrTrcv_TrcvMode );`	
Parameters [in/out/both]	
`FrTrcv_TrcvIdx [in]`	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.
`FrTrcv_TrcvMode [in]`	This parameter describes the mode the Transceiver shall be set in. It can have one of the following values: ■ `FRTRCV_TRCVMODE_NORMAL` ■ `FRTRCV_TRCVMODE_STANDBY`

Return code	
Std_ReturnType	The service returns E_NOT_OK if the Transceiver could not be set to the requested mode, otherwise E_OK is returned.
Service ID	
Service ID	1
Functional Description	
This service sets the Transceiver in the requested mode. If NORMAL mode is requested the wake up reason is set to FRTRCV_WU_INTERNALLY.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

Table 5-8 FrTrcv_30_Tja1082_SetTransceiverMode

5.2.1.6 FrTrcv_30_Tja1082_GetTransceiverMode: Get the current Transceiver mode

FrTrcv_30_Tja1082_GetTransceiverMode

Prototype	

Std_ReturnType FrTrcv_30_Tja1082_GetTransceiverMode
(
  uint8 FrTrcv_TrcvIdx,
  FrTrcv_TrcvModeType *FrTrcv_TrcvModePtr
);	

Parameters [in/out/both]	
FrTrcv_TrcvIdx [in]	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.
FrTrcv_TrcvModePtr [out]	This parameter describes the current Transceiver mode. It can have one of the following values: ■ FRTRCV_TRCVMODE_NORMAL ■ FRTRCV_TRCVMODE_STANDBY
Return code	
Std_ReturnType	The service returns E_NOT_OK if the Transceiver status could not be determined, otherwise E_OK is returned.
Service ID	
Service ID	5

Functional Description
This service determines the current Transceiver mode.
Preconditions
The Transceiver Driver module must be initialized.
Postconditions
None.
Particularities and Limitations
> Call context: task level > Not re-entrant > Synchronous

Table 5-9 FrTrcv_30_Tja1082_GetTransceiverMode

5.2.1.7 FrTrcv_30_Tja1082_GetTransceiverWUReason: Get the wake up reason

FrTrcv_30_Tja1082_GetTransceiverWUReason

Prototype	
`Std_ReturnType FrTrcv_30_Tja1082_GetTransceiverWUReason ( uint8 FrTrcv_TrcvIdx, FrTrcv_TrcvWUReasonType *FrTrcv_TrcvWUReasonPtr );`	
Parameters [in/out/both]	
`FrTrcv_TrcvIdx [in]`	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.
`FrTrcv_TrcvWUReasonPtr [out]`	This parameter contains the wake up reason of the last wake-up event. It can have one of the following values: ■ `FRTRCV_WU_POWER_ON` ■ `FRTRCV_WU_BY_BUS` ■ `FRTRCV_WU_INTERNALLY`
Return code	
`Std_ReturnType`	The service returns `E_NOT_OK` if the wake up reason could not be determined, otherwise `E_OK` is returned.
Service ID	
Service ID	6
Functional Description	
This service determines the wake up reason of the last wake up event. It can be used after an `EcuM_SetWakeupEvent` callback to determine if the wake up event happened locally or was triggered by the bus.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	

Particularities and Limitations
> Call context: task level > Not re-entrant > Synchronous

Table 5-10 FrTrcv_30_Tja1082_GetTransceiverWUReason

5.2.1.8 FrTrcv_30_Tja1082_ClearTransceiverWakeup: Clear pending wake up events

FrTrcv_30_Tja1082_ClearTransceiverWakeup

Prototype	
`Std_ReturnType FrTrcv_30_Tja1082_ClearTransceiverWakeup( uint8 FrTrcv_TrcvIdx );`	
Parameters [in/out/both]	
`FrTrcv_TrcvIdx [in]`	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.
Return code	
`Std_ReturnType`	The service returns `E_NOT_OK` if wake up events could not be disabled, otherwise `E_OK` is returned.
Service ID	
Service ID	12
Functional Description	
This service clears pending wake up events. Furthermore the wake up reason is reset to FRTRCV_WU_RESET.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

Table 5-11 FrTrcv_30_Tja1082_ClearTransceiverWakeup

5.2.1.9 FrTrcv_30_Tja1082_GetTransceiverError: Read current Transceiver error

FrTrcv_30_Tja1082_GetTransceiverError

Prototype	
`Std_ReturnType FrTrcv_30_Tja1082_GetTransceiverError ( uint8 FrTrcv_TrcvIdx, uint8 FrTrcv_BranchIdx, uint32* FrTrcv_BusErrorState );`	
Parameters [in/out/both]	
`FrTrcv_TrcvIdx [in]`	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.

FrTrcv_BranchIdx [in]	This zero based index identifies the Transceiver branch. On Transceivers without branches this parameter will be ignored.
FrTrcv_BusErrorState [out]	Pointer to variable where the error status word will be stored.
Return code	
Std_ReturnType	The service returns E_OK if the function call was successful, otherwise E_NOT_OK.
Service ID	
Service ID	8
Functional Description	
This service return error information should an error be notified to the DEM.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

Table 5-12 FrTrcv_30_Tja1082_GetTransceiverError

5.2.1.10 FrTrcv_30_Tja1082_DisableTransceiverBranch: Disable an individual branch

FrTrcv_30_Tja1082_DisableTransceiverBranch

Prototype	

Std_ReturnType FrTrcv_30_Tja1082_DisableTransceiverBranch
(
  uint8 FrTrcv_TrcvIdx,
  uint8 FrTrcv_BranchIdx
);	

Parameters [in/out/both]	
FrTrcv_TrcvIdx [in]	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.
FrTrcv_BranchIdx [in]	Index of the branch to disable.
Return code	
Std_ReturnType	The service returns E_NOT_OK always.
Service ID	
Service ID	5

Functional Description
This service is intended to disable an individual branch of Active Star Transceivers. As the TJA1082 is a node transceiver and does not support branches, this API does not perform anything but return E_NOT_OK.
Preconditions
The Transceiver Driver module must be initialized.
Postconditions
None.
Particularities and Limitations
> Call context: task level > Not re-entrant > Synchronous

Table 5-13 FrTrcv_30_Tja1082_DisableTransceiverBranch

5.2.1.11 FrTrcv_30_Tja1082_EnableTransceiverBranch: Disable an individual branch

FrTrcv_30_Tja1082_EnableTransceiverBranch

Prototype	
`Std_ReturnType FrTrcv_30_Tja1082_EnableTransceiverBranch ( uint8 FrTrcv_TrcvIdx, uint8 FrTrcv_BranchIdx );`	
Parameters [in/out/both]	
`FrTrcv_TrcvIdx [in]`	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.
`FrTrcv_BranchIdx [in]`	Index of the branch to enable.
Return code	
`Std_ReturnType`	The service returns `E_OK` always.
Service ID	
Service ID	5
Functional Description	
This service is intended to enable an individual branch of Active Star Transceivers. As the TJA1082 is a node transceiver and does not support branches, this API does not perform anything but return E_OK.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	

Particularities and Limitations
> Call context: task level > Not re-entrant > Synchronous

Table 5-14 FrTrcv_30_Tja1082_DisableTransceiverBranch

5.3 Services used by FlexRay Transceiver Driver

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

Component	API
DET (optional)	`Det_ReportError`
DEM	`Dem_SetEventStatus`
ECU Manager	`EcuM_SetWakeupEvent`
Dio	`Dio_WriteChannel` `Dio_ReadChannel`

Table 5-15 Services used by the FlexRay Transceiver Driver

5.4 Callback Functions

This chapter describes the callback functions that are implemented by the FlexRay Transceiver Driver and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `FrTrc_30_Tja1082_Cbk.h` by the FlexRay Transceiver Driver.

5.4.1 FrTrcv_30_Tja1082_CheckWakeupByTransceiver

Prototype	
`void FrTrcv_30_Tja1082_CheckWakeupByTransceiver( uint8 FrTrcv_TrcvIdx );`	
Parameter	
`FrTrcv_TrcvIdx [in]`	This zero based index identifies the Transceiver within the context of the Transceiver driver to which the API call has to be applied.
Return code	
void	-
Functional Description	
Callback to trigger wake up detection in case of an interrupt or non periodically. If the component is not initialized this service will not generate a DET call. In this case the function will simply return.	
Particularities and Limitations	
■ Particularities, limitations, post-conditions, pre-conditions	
Expected Caller Context	

> Call context: task level > Not re-entrant > Synchronous

Table 5-16 FrTrcv_30_Tja1082_CheckWakeupByTransceiver

5.5 Configurable Interfaces

5.5.1 Notifications

At its configurable interfaces the FlexRay Transceiver Driver defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the FlexRay Transceiver Driver but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

5.5.1.1 Appl_FrTrcv_30_Tja1082_ReportErrorStatusPreFailed

Prototype	
void **Appl_FrTrcv_30_Tja1082_ReportErrorStatusPreFailed** (Dem_EventIdType eventId)	
Parameter	
eventId	Unique ID that specified the error and the Transceiver that caused it.
Return code	
void	-
Functional Description	
This notification is called when an error is detected.	
Particularities and Limitations	
> Particularities, limitations, post-conditions, pre-conditions	
Call context	
> interrupt or task context	

Table 5-17 Appl_FrTrcv_30_Tja1082_ReportErrorStatusPreFailed

5.5.1.2 Appl_FrTrcv_30_Tja1082_ReportErrorStatusPrePassed

Prototype	
void **Appl_FrTrcv_30_Tja1082_ReportErrorStatusPrePassed** (Dem_EventIdType eventId)	
Parameter	
eventId	Unique ID that specified the error and the Transceiver that caused it.
Return code	
void	-
Functional Description	
This notification is called when no error is detected.	

Particularities and Limitations
> Particularities, limitations, post-conditions, pre-conditions
Call context
> interrupt or task context

Table 5-18 Appl_FrTrcv_30_Tja1082_ReportErrorStatusPrePassed

5.5.2 Callout Functions

At its configurable interfaces the FlexRay Transceiver Driver defines callout functions. The declarations of the callout functions are provided by the BSW module, i.e. the FlexRay Transceiver Driver. It is the integrator's task to provide the corresponding function definitions. The definitions of the callouts can be adjusted to the system's needs. The FlexRay Transceiver Driver callout function declarations are described in the following tables:

5.5.3 Appl_FrTrcv_30_Tja1082_Wait

Prototype	
`void Appl_FrTrcv_30_Tja1082_Wait( uint8 TimerIndex );`	
Parameter	
`TimerIndex [in]`	This zero based index identifies the timer which shall be used. `kFrTrcv_30_Tja1082_SetMode`: Delay time until ERRN stabilized `kFrTrcv_30_Tja1082_ModeChange`: Delay time until mode change complete
Return code	
void	-
Functional Description	
Mandatory call out to delay a certain time in order to provide the Transceiver Driver with a deterministic timing.	
Particularities and Limitations	
■ Particularities, limitations, post-conditions, pre-conditions	
Expected Caller Context	
> Call context: task level > Not re-entrant > Synchronous	

Table 5-19 Appl_FrTrcv_30_Tja1082_Wait

6 Glossary and Abbreviations

6.1 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
Dio	Digital Input/Output
ECU	Electronic Control Unit
FrTrcv	FlexRay Transceiver Driver
HIS	Hersteller Initiative Software
ICU	Input Capture Unit
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 6-1 Abbreviations

7 Contact

Visit our website for more information on

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

www.vector.com