MICROSAR FlexRay Transceiver Driver

Technical Reference

Version 2.01.00

Author	Andreas Herkommer
Status	Released

1 Document Information

1.1 History

Date	Version	Remarks
2010-02-17	1.00.00	Creation of document
2011-01-31	1.00.01	Minor clarifications
2012-01-09	1.00.02	ESCAN00056538 Added chapter about SchM pre configuration
2012-02-01	1.00.03	Minor rework of document
2014-01-13	2.00.00	Rework for ASR4
2015-05-18	2.01.00	ESCAN00083011 AR4-830: Extend Support for module initialization ESCAN00077239 AR3-2679: Description BCD-coded return-value of XXX_GetVersionInfo() in TechRef

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR_SWS_FlexRayTransceiver.pdf	1.2.1
[2]	AUTOSAR_SWS_DET.pdf	2.2.1
[3]	AUTOSAR_SWS_DEM.pdf	2.2.0
[4]	AUTOSAR_BasicSoftwareModules.pdf	1.0.0
[5]	TJA1080_DevSpec_N1C1.pdf	-
[6]	NXP - ApplicationHints_Rev 3_TJA1080.pdf	
[7]	TechnicalReference_Asr_SchM.pdf	-

Table 1-2 Reference documents

1.3 Scope of the Document

This technical reference describes the specific use of the Generic FlexRay transceiver driver. Note that the substrings “__Your_Trcv__” and “__YOUR_TRCV__” are just placeholders for the real name of the Transceiver (e.g. Tja1080).

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

Component History

Component Version	New Features
01.00.00	Initial Version
01.01.00	Adapt MainFunction for usage with IdentityManagerConfig Minor Bugfixes
1.01.01	ESCAN00048742 Missing START_SEC_CONST_32BIT in FrTrcv_30___Your_Trcv___MemMap.inc
1.01.02	ESCAN00049012 "Dem_" prefix is missing for FRTRCV_E_FR_NO_TRCV_CONTROL ESCAN00049932 Wakeup detection is checked even though the option is not enabled in GENy
1.01.03	ESCAN00053416 AR3-2069: Remove non-SchM code for critical section handling
1.01.04	ESCAN00071791 The MISRA justifications are not implemented according WI_MISRAC2004_PES.pdf
2.00.00	ESCAN00072928 Support ASR4

Table 1-3 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*:	3, 4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	FrTRCV_30___YOUR_TRCV___VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	FRTRCV_30___YOUR_TRCV___MODULE_ID	71 decimal (according to ref. [4])

* For the precise AUTOSAR Release 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.

Info

Replace the placeholders __Your_Trcv__ and __YOUR_TRCV__ with the according name of the used transceiver. This must be done in the source code and the BSWMD Files.

__YOUR_TRCV__ is used for definitions in upper case (e.g. TJA1080).

__Your_Trcv__ is used for variables in camel case (e.g. Tja1080).

In the bswmd file the “Vendor API Infix” is set to Generic by default. Set this to the same value than __Your_Trcv__

2.1 Architecture Overview

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

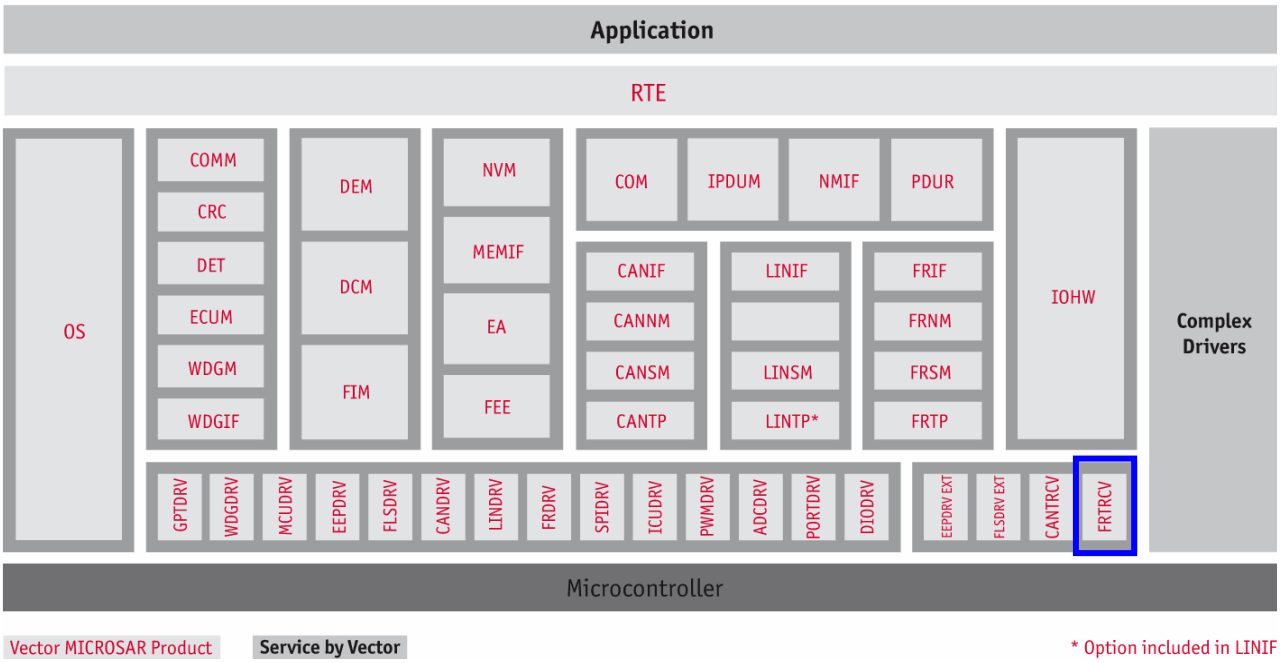

Figure 2-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the FlexRay Transceiver Driver. These interfaces are described in chapter 6.

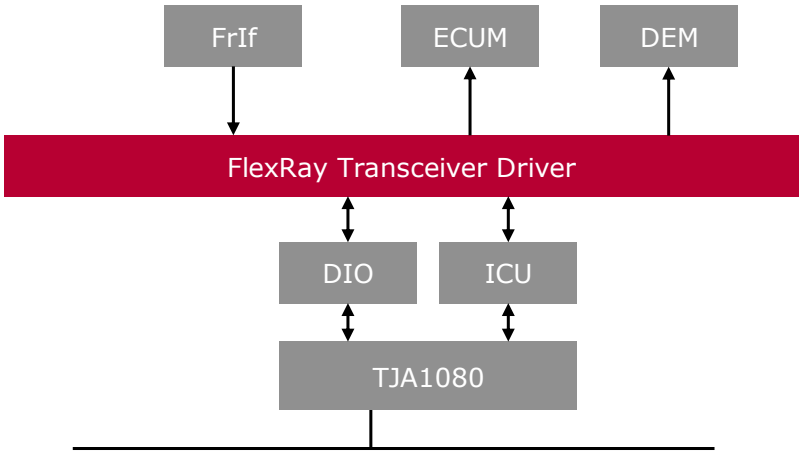

Figure 2-2 Interfaces to adjacent modules of the FlexRay Transceiver Driver

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE.

The FlexRay Transceiver Driver does not have any service ports, therefore no connection to the RTE exists.

3 Functional Description

3.1 Initialization

3.1.1 High-Level Initialization

The function `FrTrcv_30___Your_Trcv___InitMemory` initializes all necessary memory variables for the Transceiver Driver. This function has to be called first after power on or reset.

The Transceiver Driver is initialized by calling the `FrTrcv_30___Your_Trcv___TrcvInit` service with the corresponding index for each transceiver.

The default operation mode of the transceiver after Init is pre-defined in GENy during configuration process.

3.1.2 Low-Level Initialization

The user is responsible to initialize transceiver relevant I/O-ports and SPI interfaces. This can be done, e.g. by configuring the Port Module accordingly.

3.2 States

After initialization the transceiver is in a predetermined state which has been configured in GENy.

3.3 Main Function

The Transceiver Driver has one task `FrTrcv_30___Your_Trcv___MainFunction` which has to be called cyclically. This task is responsible for polling all connected transceivers and perform action if so required. Please note that this main function will not be present if polling is not used and the call cycle time is configured as 0.

3.4 Error Handling

3.4.1 Development Error Reporting

Development errors are reported to DET using the service Det_ReportError(), (specified in [2]), if this feature is enabled in GENy.

The reported FlexRay Transceiver Driver ID is 71.

The reported service IDs identify the services which are described in 6.4. The following table presents the service IDs and the related services:

Service ID	Service
0	`FrTrcv_30___Your_Trcv___Init()`
1	`FrTrcv_30___Your_Trcv___SetTransceiverMode()`
5	`FrTrcv_30___Your_Trcv___GetTransceiverMode()`
6	`FrTrcv_30___Your_Trcv___GetTransceiverWUReason()`
7	`FrTrcv_30___Your_Trcv___GetVersionInfo()`

Service ID	Service
10	FrTrcv_30___Your_Trcv___DisableTransceiverWakeup()
11	FrTrcv_30___Your_Trcv___EnableTransceiverWakeup()
12	FrTrcv_30___Your_Trcv___ClearTransceiverWakeup()
13	FrTrcv_30___Your_Trcv___MainFunction()
14	FrTrcv_30___Your_Trcv___Cbk_WakeupByTransceiver()
15	FrTrcv_30___Your_Trcv___DisableTransceiverBranch
16	FrTrcv_30___Your_Trcv___EnableTransceiverBranch
8	FrTrcv_30___Your_Trcv___GetTransceiverError

Table 3-1 Mapping of service IDs to services

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

Error Code		Description
0x01	BUSTRCV_30___YOUR_TRCV___E_FR_INVALID_TRCVIDX	The Transceiver Driver was called with an invalid transceiver Index.
0x10	BUSTRCV_30___YOUR_TRCV___E_FR_UNINIT	A Transceiver Driver service was called without initializing the module first by calling FrTrcv_TrcvInit.
0x20	BUSTRCV_30___YOUR_TRCV___E_FR_INVALID_POINTER	A Transceiver Driver service was called with a zero pointer as parameter.

Table 3-2 Errors reported to DET

3.4.2 Production Code Error Reporting

Production code related errors are reported to DEM using the service Dem_ReportErrorStatus() (specified in [3]).

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

Error Code	Description
FRTRCV_E_FR_NO_TRCV_CONTROL	If communication with the transceiver does not work as intended

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

The static files are not to be modified

File Name	Description	
FrTrcv_30___Your_Trcv__.c	Source code of Transceiver Driver.	
FrTrcv_30___Your_Trcv__.h	API definitions of the Transceiver Driver.	
FrTrcv_30___Your_Trcv___Cbk.h	API definitions of call-back services	

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files can be modified if necessary, e.g. GENy is not used for configuration.

File Name	Description	
FrTrcv_30___Your_Trcv___Cfg.c	Parameter Configuration source file for Transceiver Driver. Can be modified if GENy is not used.	
FrTrcv_30___Your_Trcv___Cfg.h	Parameter Configuration header file for Transceiver Driver. Can be modified if GENy is not used.	

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 FlexRay Transceiver Driver and illustrates their assignment among each other.

Memory Mapping Sections / Compiler Abstraction Definitions	FRTRCV_30___YOUR_TRCV___CONST	FRTRCV30___YOUR_TRCV___VAR_NOINIT	FRTRCV30___YOUR_TRCV___CODE
FRTRCV_30___YOUR_TRCV___START_SEC_CONST_UNSPECIFIED	■		
FRTRCV_30___YOUR_TRCV___START_SEC_VAR_NOINIT_UNSPECIFIED		■	
FRTRCV_30___YOUR_TRCV___START_SEC_CODE			■

Table 4-3 Compiler abstraction and memory mapping

4.3 Data Consistency

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.

4.4 Adaptation of FrTrcv_30___Your_Trcv__.c

Depending on the number of used transceiver or how your transceiver is connected, adaptation of the file `FrTrcv_30___Your_Trcv__.c` is required. The following subchapters highlight some details.

4.4.1 Dio pin configuration

For access of Dio controlled pins the correct name must be known. This name can be configured in the structure `FrTrcv_30___Your_Trcv___Channel`. For each transceiver there must be one entry defining the names for the I/O pins used to access the transceiver.

4.4.2 FrTrcv_30___Your_Trcv___SetTransceiverMode

This method is used to set the respective transceiver into the requested mode. Configuring the transceiver for a different operation mode is done by setting the I/O ports to a certain state, reflecting the requested operation mode. The given example is made for the TJA1080. If a Transceiver is used that is connected via SPI a certain command word might be written to the Transceiver.

4.4.3 FrTrcv_30___Your_Trcv___GetTransceiverMode

This method is used to read back the current mode of the transceiver. The given example is made for the TJA1080. Depending on the used Transceiver either I/O ports are read or a status word is read back via SPI.

4.4.4 Timers

As the used transceiver hardware may have some timing constraints that must be met, the transceiver driver sometimes needs to wait some time until the next request to the hardware can be made.

An application function which handles this wait states is declared in FrTrcv_30___Your_Trcv___Cbk.h and has to be implemented by the user. To enable or disable this callback function and all predefined timers, you have to specify the following constant in the FrTrcv_30___Your_Trcv__.h:

```
# define FRTRCV_30___YOUR_TRCV___USE_TIMERS                    STD_ON
```

Declaration:

```
FUNC(void, FRTRCV_30___YOUR_TRCV___CODE) Appl_FrTrcv_30___Your_Trcv___Wait(uint8 delay);
```

The parameter `TimerIndex` is used to distinguish between the timers needed by the transceiver driver. `TimerIndex` is represented by a symbolic constant that is defined in the FrTrcv_30___Your_Trcv__.h. The following table lists all available timer indexes:

Timer Index (symbolic constant)	Wait Time	Description
`kFrTrcv_30___Your_Trcv___SetMode`	~ 80µs	This timer is called by the transceiver driver in function SetMode() in order to delay the EN line by the required time.

Table 4-4 Timer indexes and their wait times

In the used example a while loop is used to reach a delay time of 80µs because of timing restrictions of transceiver and interrupt locking time. Of course a timer interrupt can be used to achieve the delay time but the timing restrictions of the transceiver shall be met.

Example of the implementation

```
FUNC(void, FRTRCV_30___YOUR_TRCV___CODE)
Appl_FrTrcv_30___Your_Trcv___Wait(uint8 TimerIndex);
{
    uint32 timer;

    switch (TimerIndex)
    {
      case kFrTrcv_30___Your_Trcv___SetMode:
        timer = 100; /* Delay 80 us */
```

```
        break;

      default:
        timer = 0;
        break;
    }

    while (timer > 0)
    {
       timer--;
    }
}
```

4.4.5 FrTrcv_30___Your_Trcv___Cbk_WakeupByTransceiver

This service is the call-back notification in case a wake up is detected. This service is typically called by the ICU module in case an interrupt is registered on a port pin.

4.4.6 Interrupt Enable/Disable Handling

The Transceiver Driver provides the possibility to enable/disable Transceiver Interrupts based on the selected mode. For this the compiler switch:

```
#define FRTRCV_30___YOUR_TRCV___USE_ICU                    STD_ON
```

must be set in the code. When enabled the call-backs:

```
FUNC(void, FRTRCV_30___YOUR_TRCV___APPL_CODE)
Appl_FrTrcv_30___Your_Trcv___EnableIcuNotification(uint8 FrTrcv_TrcvIdx);
FUNC(void, FRTRCV_30___YOUR_TRCV___APPL_CODE)
Appl_FrTrcv_30___Your_Trcv___DisableIcuNotification(uint8 FrTrcv_TrcvIdx);
```

are called and can be used to modify the interrupt possibility of the Transceiver interrupt pin. This is used to disable the RXD/RXEN Interrupt in Normal Operation mode to prevent false wake up Interrupts. It is not required if the Transceiver Driver is used in polling mode.

5 Dependencies to other components

5.1 Dependencies to Dio component

The FlexRay 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 to which the transceiver hardware is connected.
- Function Dio_ReadChannel is used to get the logical level of the channel pins to which the transceiver hardware is connected.
- The Dio Driver has to provide the pin assignment for the transceiver hardware pins EN, STB, ERRN and RXEN. These pins are referred by the FlexRay Transceiver Driver by using the symbolic names which must be specified in the FrTrcvPhy_30___Your_Trcv__.c file.

5.2 SPI Driver

Depending on the used transceiver hardware a SPI interface might be required. In this case the following interfaces should be used:

- Function Spi_WriteIB is used to set the data to be transmitted.
- Function Spi_ReadIB is used to get the formerly received data.
- Function Spi_SyncTransmit is used to transmit/receive data.

Please keep in mind, that all API services of the AUTOSAR Transceiver driver are synchronous!

6 API Description

6.1 Interfaces Overview

The AUTOSAR Transceiver Driver provides the following services:

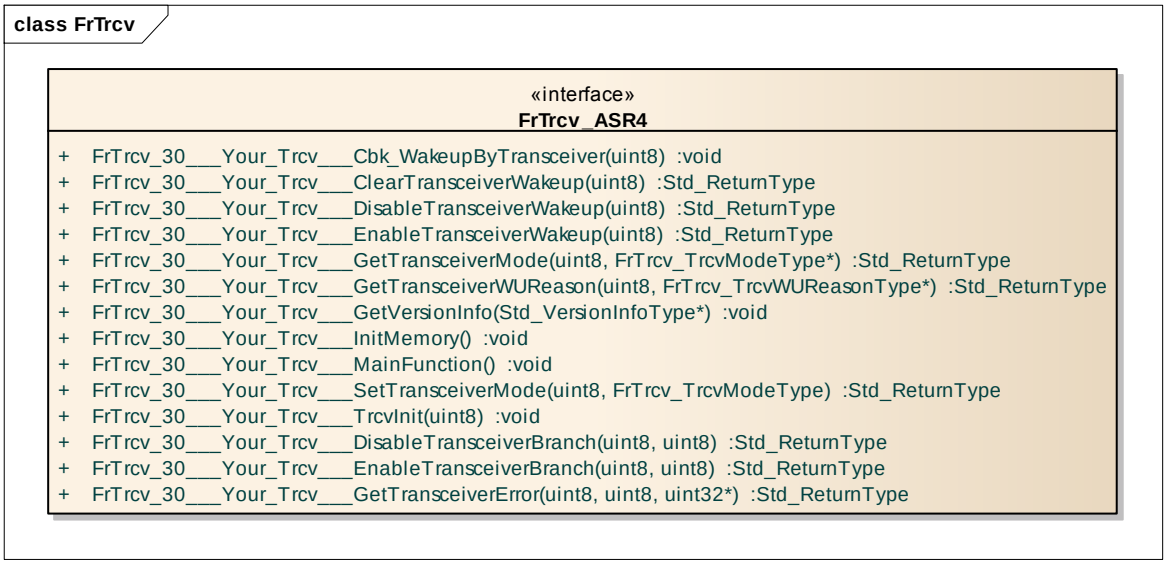

Figure 6-1 FlexRay Transceiver Driver

6.2 Type Definitions

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_TRCVMODE_SLEEP Sleep operation mode
			FRTRCV_TRCVMODE_RECEIVEONLY Receive only 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

Type Name	C-Type	Description	Value Range
			that the bus has woken up by the ECU via FrTrcv_GotoNormalMode API call.
			FRTRCV_WU_RESET The transceiver has detected that the "wake up" is due to an ECU reset.
			FRTRCV_WU_POWER_ON The transceiver has detected that the "wake up" is due to an ECU reset after power on.

Table 6-1 Type definitions

6.3 Structures

FrTrcv_30___Your_Trcv___Channel

The following structure contains the Dio pin description (in this example for the TJA1080) the transceiver driver will use to access the hardware. This structure is located in the source file FrTrcv_30___Your_Trcv__.c and must be adapted to the used Transceiver.

Struct Element Name	C-Type	Description	Value Range
TrcvPinEN	Dio_ChannelType	Dio name of the respective pin	FRTRCV_CHANNEL_EN_0
TrcvPinSTBN	Dio_ChannelType	Dio name of the respective pin	FRTRCV_CHANNEL_STBN_0
TrcvPinERRN	Dio_ChannelType	Dio name of the respective pin	FRTRCV_CHANNEL_ERRN_0
TrcvPinRXEN	Dio_ChannelType	Dio name of the respective pin	FRTRCV_CHANNEL_RXEN_0

Table 6-2 FrTrcvChannel

6.4 Services provided by FlexRay Transceiver Driver

The FlexRay Transceiver Driver API consists of services, which are realized by function calls.

6.4.1 Administrative Functions

6.4.1.1 FrTrcv_30___Your_Trcv___InitMemory: Initialization of Transceiver Driver

FrTrcv_30___Your_Trcv___InitMemory

Prototype	
void FrTrcv_30___Your_Trcv___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, FrTrcv_TrcvInit can be called	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

6.4.1.2 FrTrcv_30___Your_Trcv___Init: Initialization of Transceiver Driver

FrTrcv_30___Your_Trcv___Init

Prototype	
`void FrTrcv_30___Your_Trcv___Init( void );`	
Parameters [in/out/both]	
–	-
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	

6.4.1.3 FrTrcv_30___Your_Trcv___MainFunction: Main Function of Transceiver Driver

FrTrcv_30___Your_Trcv___MainFunction

Prototype	
`void FrTrcv_30___Your_Trcv___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 call back in case of a wake-up event is triggered.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

6.4.1.4 FrTrcv_30_Generic_GetVersionInfo: Read Version Information of the Driver

FrTrcv_30_Generic_GetVersionInfo

Prototype	
`void FrTrcv_30_Generic_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_Generic_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component. The versions are BCD-coded.	

Preconditions
None.
Postconditions
None.
Particularities and Limitations
> Call context: task level > Not re-entrant > Synchronous

Service Functions

6.4.1.5 FrTrcv_30___Your_Trcv___SetTransceiverMode: Set the transceiver to the requested mode

FrTrcv_30___Your_Trcv___SetTransceiverMode

Prototype	
`Std_ReturnType FrTrcv_30___Your_Trcv___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` ■ `FRTRCV_TRCVMODE_SLEEP` ■ `FRTRCV_TRCVMODE_RECEIVEONLY`
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.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > synchronous	

6.4.1.6 FrTrcv_30___Your_Trcv___GetTransceiverMode: Get the current Transceiver mode

FrTrcv_30___Your_Trcv___GetTransceiverMode

Prototype
`Std_ReturnType FrTrcv_30___Your_Trcv___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` ■ `FRTRCV_TRCVMODE_SLEEP` ■ `FRTRCV_TRCVMODE_RECEIVEONLY`

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

6.4.1.7 FrTrcv_30___Your_Trcv___GetTransceiverWUReason: Get the wake up reason

FrTrcv_30___Your_Trcv___GetTransceiverWUReason

Prototype
`Std_ReturnType FrTrcv_30___Your_Trcv___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 call back 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	

6.4.1.8 FrTrcv_30___Your_Trcv___DisableTransceiverWakeup: Disable wake up event notifications

FrTrcv_30___Your_Trcv___DisableTransceiverWakeup

Prototype	
Std_ReturnType FrTrcv_30___Your_Trcv___DisableTransceiverWakeup(const 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	10
Functional Description	
This service disables any wake up notifications.	
Preconditions	
The Transceiver Driver module must be initialized.	

Postconditions
None.
Particularities and Limitations
> Call context: task level > Not re-entrant > Synchronous

6.4.1.9 FrTrcv_30___Your_Trcv___EnableTransceiverWakeup: Enable wake up event notifications

FrTrcv_30___Your_Trcv___EnableTransceiverWakeup

Prototype	
`Std_ReturnType FrTrcv_30___Your_Trcv___EnableTransceiverWakeup( 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 enabled, otherwise `E_OK` is returned.
Service ID	
Service ID	11
Functional Description	
This service enables wake up notifications.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > synchronous	

6.4.1.10 FrTrcv_30___Your_Trcv___ClearTransceiverWakeup: Clear pending wake up events

FrTrcv_30___Your_Trcv___ClearTransceiverWakeup

Prototype	
`Std_ReturnType FrTrcv_30___Your_Trcv___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	

6.4.1.11 FrTrcv_30___Your_Trcv___DisableTransceiverBranch: Disable selected Branch

FrTrcv_30___Your_Trcv___DisableTransceiverBranch

Prototype	
Std_ReturnType FrTrcv_30___Your_Trcv___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]	This zero based index identifies the branch of the selected transceiver.
Return code	
Std_ReturnType	The service returns E_NOT_OK if the branch could not be disabled, otherwise E_OK is returned.
Service ID	
Service ID	15
Functional Description	
This service enables selected transceiver branches. By default all branches are enabled. If the transceiver does not support branches, this API will do nothing.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	

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

6.4.1.12 FrTrcv_30___Your_Trcv___EnableTransceiverBranch: Enable selected Branch

FrTrcv_30___Your_Trcv___EnableTransceiverBranch

Prototype	
`Std_ReturnType FrTrcv_30___Your_Trcv___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]`	This zero based index identifies the branch of the selected transceiver.
Return code	
`Std_ReturnType`	The service returns `E_NOT_OK` if the branch could not be disabled, otherwise `E_OK` is returned.
Service ID	
Service ID	16
Functional Description	
This service enables selected transceiver branches. By default all branches are enabled. If the transceiver does not support branches, this API will do nothing.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

6.4.1.13 FrTrcv_30___Your_Trcv___GetTransceiverError: Read out detected Trcv errors

FrTrcv_30___Your_Trcv___GetTransceiverError

Prototype
`Std_ReturnType FrTrcv_30___Your_Trcv___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 branch of the selected transceiver.
FrTrcv_BusErrorState [out]	Contains the error state after return of the function
Return code	
Std_ReturnType	The service returns E_NOT_OK if the branch could not be disabled, otherwise E_OK is returned.
Service ID	
Service ID	8
Functional Description	
This service reads out detected transceiver and bus errors. If the transceiver does not support error detection this API will always return no error.	
Preconditions	
The Transceiver Driver module must be initialized.	
Postconditions	
None.	
Particularities and Limitations	
> Call context: task level > Not re-entrant > Synchronous	

6.5 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[1]	Dio_WriteChannel Dio_ReadChannel

Table 6-3 Services used by the FlexRay Transceiver Driver

[1] Depending on used Transceiver

6.6 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 FrTrcv_30___Your_Trcv___Cbk.h by the FlexRay Transceiver Driver.

6.6.1 FrTrcv_30___Your_Trcv___Cbk_WakeupByTransceiver

Prototype	
`void FrTrcv_30___Your_Trcv___Cbk_WakeupByTransceiver( 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	
Call back to trigger wake up detection in case of an interrupt or non-periodically.	
Particularities and Limitations	
■ Particularities, limitations, post-conditions, pre-conditions	
Expected Caller Context	
> Call context: task level > Not re-entrant > synchronous	

7 AUTOSAR Standard Compliance

7.1 Deviations

None.

7.2 Additions/ Extensions

7.2.1 Memory initialization

To have an independent memory initialization for this BSW module the additional function `FrTrcv_30___Your_Trcv___InitMemory` was added.

7.3 Limitations

7.3.1 Local Wake up

The template implementation of the Transceiver Driver does not differentiate between local and remote wakeup. Therefore a local wake up will also lead to a bus wake up.

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
EAD	Embedded Architecture Designer; generation tool for MICROSAR components
GENy	Generation tool for CANbedded and MICROSAR components

Table 8-1 Glossary

8.2 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
EAD	Embedded Architecture Designer
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)
Platform	Hardware including Host and Communication Controller (might also be integrated in Host) on which the communication stack is implemented.
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector-informatik.com