MICROSAR Classic FRTSYN

Technical Reference

Version 6.0.0

Authors	visssf, visnmo, vistra, viscpi, vissi, visbfk, visjwe, istaehle
Status	Released

Document Information

History

Author	Date	Version	Remarks
visssf	2014-12-05	1.0.0	Initial version
visssf	2016-03-22	2.1.0	Support TriggerTransmit API according to AUTOSAR 4.2.2
visssf	2017-05-15	3.1.0	Support multiple Time Domains in Tx state Machine
visnmo	2017-07-03	3.2.0	Debounce Time introduction
vistra	2017-08-04	3.3.0	Immediate Time Synchronization
viscpi, vistra	2017-10-06	3.4.0	New DET error codes added Message type compatibility
vissi	2018-04-03	3.5.0	Updated AUTOSAR architecture Updated referenced documents Updated list of unsupported features
vissi	2018-08-31	3.6.0	Improved exclusive area description Support Measurement (MC Data) Updated chapter 4.1 to new template Updated AUTOSAR architecture Updated referenced documents
vissi, vistra	2019-02-07	3.7.0	MISRA-C:2012 compliance Corrected Det error codes Updated used services
vissi	2019-11-04	4.0.0	Updated table of supported AUTOSAR features
visbfk, vistra	2020-03-30	5.0.0	Updated table of services used by the FRTSYN, chapter 3.1.1 updated FrTSyn is not initialized by EcuM
visssf	2020-08-05	5.1.0	Support Helix QAC 2019-2
visjwe, istaehle	2022-05-09	6.0.0	ASR memmap include structure ALL SLP Product name updated to MICROSAR Classic

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_TimeSyncOverFlexRay.pdf	4.4.0
[2]	AUTOSAR	AUTOSAR_SWS_TimeSyncOverFlexRay.pdf	4.2.2
[3]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	4.2.2
[4]	AUTOSAR	AUTOSAR_SWS_DefaultErrorTracer.pdf	4.2.2
[5]	AUTOSAR	AUTOSAR_SWS_Rte.pdf	4.2.2
[6]	AUTOSAR	AUTOSAR_SWS_SynchronizedTimeBaseManager.pdf	4.4.0
[7]	AUTOSAR	AUTOSAR_SWS_FlexRayInterface.pdf	4.4.0
[8]	AUTOSAR	AUTOSAR_SWS_CRCLibrary.pdf	4.2.2

Scope of the Document

This technical reference describes the general use of the Time Synchronization over FlexRay.

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
1.0.0	Initial creation
2.1.0	Support TriggerTransmit API according to AUTOSAR 4.2.2
3.1.0	Support multiple Time Domains in Tx state machine
3.2.0	Debounce Time introduction
3.3.0	Immediate Time Synchronization
3.4.0	Minor improvements
3.5.0	Module refactored
3.6.0	Support Measurement (MC Data)
3.7.0	MISRA-C:2012 compliance
4.0.0	Module rework and introduction of sub units for functional safety
5.0.0	Enhance precision of Global Time according to AUTOSAR 4.4.0
5.1.0	Rework QAC justifications for Helix QAC 2019-2

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	FRTSYN_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	FRTSYN_MODULE_ID	163 decimal (according to ref.[3])

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

The FrTSyn module handles the distribution of time information over FlexRay busses.

2.1 Architecture Overview

The following figure shows where the FRTSYN is located in the AUTOSAR architecture.

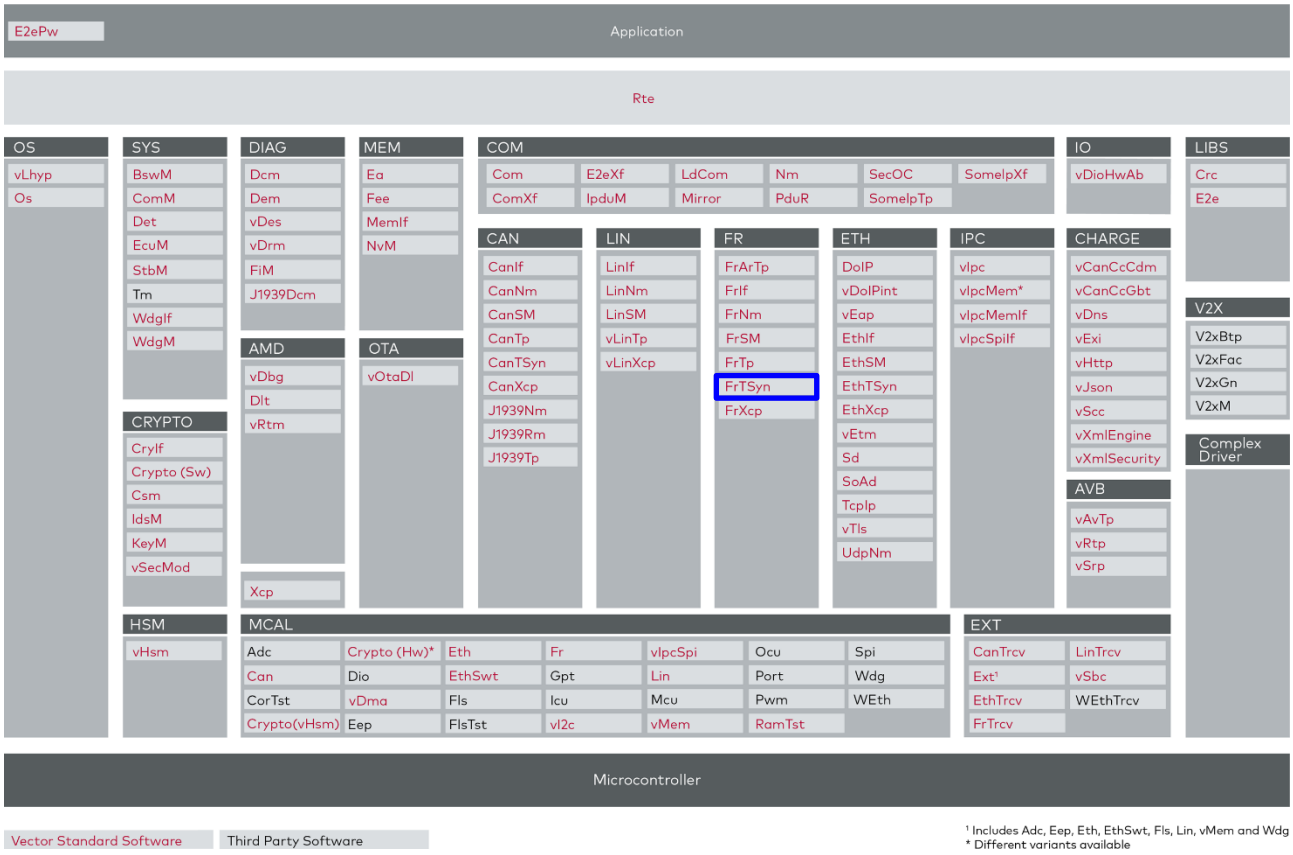

Figure 2-1 AUTOSAR Architecture Overview

The next figure shows the interfaces to adjacent modules of the FRTSYN. These interfaces are described in chapter 5.

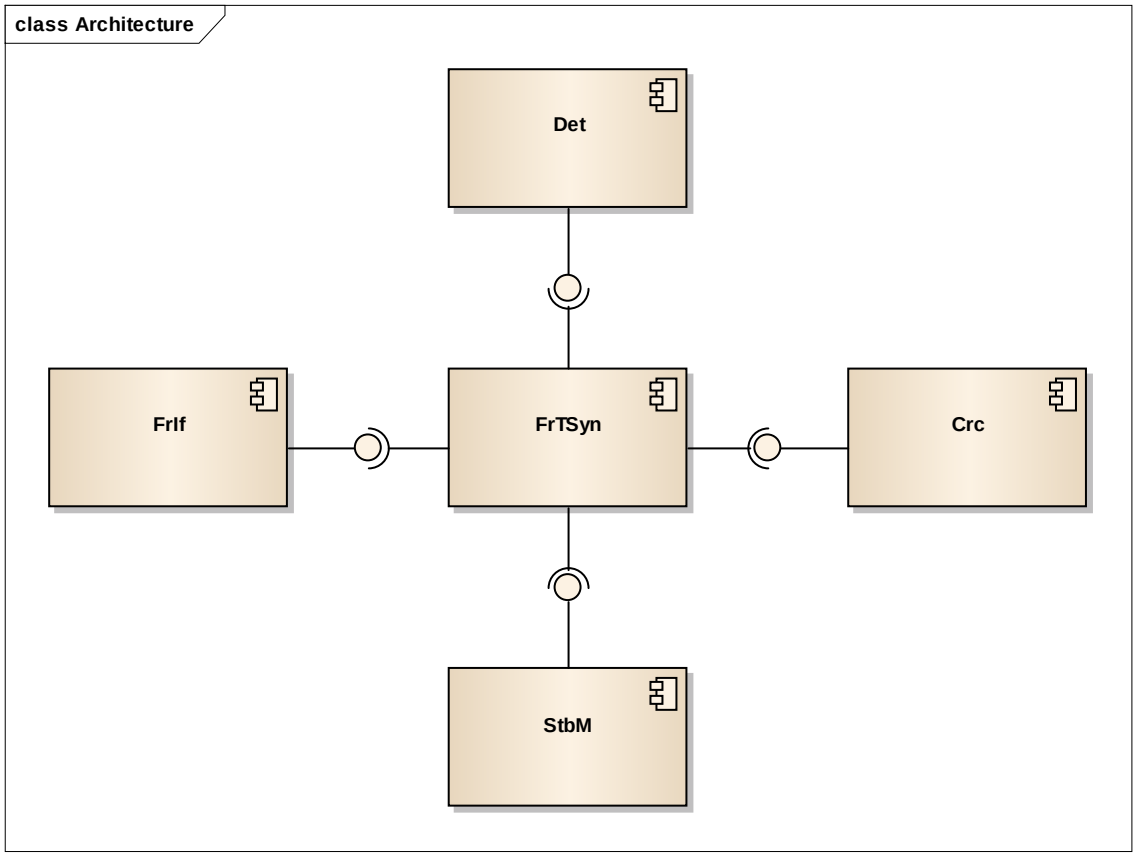

Figure 2-2 Interfaces to adjacent modules of the FRTSYN

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the FRTSYN.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further FRTSYN functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Time Master, Time Slave and Time Gateway support
Calculation and assembling of Time Synchronization Messages on FlexRay
Validation and disassembling of Time Synchronization Messages on FlexRay
Standard Message Format (SYNC)
Offset Message Format (OFS)
User Data support in standard message format
CRC calculation and CRC validation
Enabling and disabling of network access
Configurable Debounce Time
Immediate Time Synchronization
Timeout handling
Configuration variant Pre-Compile

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
Configuration variant Post-Build
CRC validation option CRC_OPTIONAL
User Data in Offset Message Format for Offset Time Bases
SGW status in Offset Message Format for Offset Time Bases
Message Authentication

Not Supported AUTOSAR Standard Conform Features
Minimum Message Gap

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Memory Initialization
Message Type Compatibility

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.2.1 Memory Initialization

AUTOSAR expects the startup code to automatically initialize RAM. Not every startup code of embedded targets reinitializes all variables correctly. It is possible that the state of a variable may not be initialized as expected. To avoid this problem the Vector AUTOSAR FrTSyn provides an additional function to initialize the relevant variables of the FrTSyn. See also chapters 3.2 and 5.2.2 for details.

3.1.2.2 Message Type Compatibility

Message types of Offset messages have been changed after [2]. FrTSyn provides the additional parameter `FrTSynMessageCompatibility` to configure the used message types.

3.1.3 Limitations

There are no known limitations.

3.2 Initialization

The Time Synchronization over FlexRay is initialized by calling `FrTSyn_Init()`. This is done by the BSW Mode Manager (BswM).

On platforms, where RAM is not initialized to zero by the startup code, the function `FrTSyn_InitMemory()` has to be called first and then a call to `FrTSyn_Init()` can be realized.

3.3 States

The FrTSyn is operational after initialization. It implements state machines for the transmission and reception of Time Synchronization messages.

3.3.1 Message transmission states

> FRTSYN_STATE_SEND_WAITING_FOR_SYNC_SEND

If the GLOBAL_TIME_BASE bit is set, a time master transmits SYNC messages according to a configured cycle time or immediately, if the corresponding Time Base has been changed.

> FRTSYN_STATE_SEND_WAITING_FOR_SYNC_TRIGGER_TRANSMIT

After transmission of the SYNC message the time master waits for the call of the TriggerTransmit API. When the TriggerTransmit API is called or a timeout occurs the master resets its state and sends the next SYNC message.

3.3.2 Message reception states

> FRTSYN_STATE_RECEIVE_WAITING_FOR_SYNC

After initialization a time slave is waiting for the reception of a SYNC message.

> FRTSYN_STATE_RECEIVE_SYNC_RECEIVED

After reception of a SYNC message a time slave changes its state to indicate that the next MainFunction has to handle the received message. After disassembling the message, the time slave will reset its state and wait for the next SYNC message.

3.4 Main Functions

The `FrTSyn_MainFunction()` triggers the transmission of Time Synchronization messages and handles received Time Synchronization messages. Depending on the configuration cyclic and immediate transmission is possible.

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 [4], if development error reporting is enabled (i.e. pre-compile parameter `FRTSYN_DEV_ERROR_DETECT==STD_ON`).

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 FRTSYN ID is 163.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x01	FrTSyn_Init
0x02	FrTSyn_GetVersionInfo
0x03	FrTSyn_SetTransmissionMode
0x04	FrTSyn_MainFunction
0x41	FrTSyn_TriggerTransmit
0x42	FrTSyn_RxIndication

Table 3-4 Service IDs

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

Error Code	Description
0x01	FRTSYN_E_INVALID_PDUID
0x20	FRTSYN_E_UNINIT
0x21	FRTSYN_E_NULL_POINTER
0x22	FRTSYN_E_INIT_FAILED
0x23	FRTSYN_E_PARAM
0x24	FRTSYN_E_INV_CTRL_IDX

Table 3-5 Errors reported to DET

3.5.2 Production Code Error Reporting

No production error codes are currently used by FrTSyn.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR Classic FRTSYN into an application environment of an ECU.

4.1 Embedded Implementation

The delivery of the FRTSYN consists out of these files:

File Name	Description	Integration Tasks
FrTSyn.c	Main implementation file of the FrTSyn.	-
FrTSyn.h	Main header file of the FrTSyn.	-
FrTSyn_Cbk.h	Header file that contains the prototypes of callback functions of the FrTSyn.	-
FrTSyn_Types.h	Header file that contains the type definitions of the FrTSyn.	-
FrTSyn_Cfg.c	Generated file that contains definitions of structures in pre-compile-time variant.	-
FrTSyn_Cfg.h	Generated file that contains declarations of structures in pre-compile-time variant.	-

Table 4-1 Implementation files

4.2 Critical Sections

The FrTSyn has code sections which need protection against interrupts and OS tasks which can interrupt each other. Therefore, the FrTSyn uses one exclusive area which requires a global interrupt lock:

`FRTSYN_EXCLUSIVE_AREA_0`

The FrTSyn calls StbM services. Depending on the StbM configuration, the StbM in turn calls OS APIs like `GetCounterValue()` and `GetElapsedValue()`. According the AUTOSAR OS specification it is not allowed to call these OS APIs with disabled interrupts. Nevertheless, the FrTSyn module requires the interrupt lock to be able to guarantee high accuracy and data consistency.

Caution

If the StbM is configured to use OS APIs and the implementation method of the exclusive area is configured to `OS_INTERRUPT_BLOCKING` or `ALL_INTERRUPT_BLOCKING`, the OS may report the error `E_OS_DISABLEDINT` notified by the OS `ErrorHook()`. In that case the implementation method of the exclusive area `FRTSYN_EXCLUSIVE_AREA_0` inside the RTE / SchM configuration needs to be set to `OS_RESOURCE` or `CUSTOM`.

If `OS_RESOURCE` is selected the Flexray ISR(s) need to reference the OS Resource created by the RTE.

If `CUSTOM` is selected the SchM APIs for entering and exiting the exclusive area need to be implemented manually by using an interrupt lock mechanism but without calling OS APIs like `SuspendOSInterrupts()` or `DisableAllInterrupts()`.

Note:

The exclusive area implementation method `CUSTOM` is a MICROSAR Classic RTE extension and might not be available in other RTEs.

For general details about exclusive areas refer to [5].

4.3 Memory Sections

The FrTSyn_MemMap.h is generated by the MemMap Generator (/ActiveEcuC/MemMap). If adaptions should be done to the Memory Mapping of the FrTSyn, the changes must be configured in the MemMap Generator.

5 API Description

For an interface overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the FRTSYN are described in this chapter.

Type Name	C-Type	Description	Value Range
FrTSyn_ConfigType	struct	Post-build configuration structure	–
FrTSyn_TransmissionModeType	enum	Handles the enabling and disabling of the transmission mode	`FRTSYN_TX_OFF` Transmission disabled
			`FRTSYN_TX_ON` Transmission enabled

Table 5-1 Type definitions

5.2 Services provided by FRTSYN

5.2.1 FrTSyn_Init

Prototype	
`void` **`FrTSyn_Init`** `( const FrTSyn_ConfigType *configPtr )`	
Parameter	
`configPtr`	Pointer to selected configuration structure.
Return code	
–	-
Functional Description	
This function initializes the Time Synchronization over FlexRay.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This API should be called by the BSW Mode Manager during the startup phase. > This function has to be called before any other FrTSyn service function is called (except FrTSyn_InitMemory()).	
Expected Caller Context	
> Task context	

Table 5-2 FrTSyn_Init

5.2.2 FrTSyn_InitMemory

Prototype	
`void` **`FrTSyn_InitMemory`** `( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
Initializes the global variables in case an initializing startup code is not used. This function sets the FrTSyn into an uninitialized state.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant. > If this function is used it shall be called before any other FrTSyn function after startup.	
Expected Caller Context	
> Task context	

Table 5-3 FrTSyn_InitMemory

5.2.3 FrTSyn_GetVersionInfo

Prototype	
`void` **`FrTSyn_GetVersionInfo`** `( Std_VersionInfoType *versioninfo )`	
Parameter	
`versioninfo`	Pointer to where to store the version information of this module.
Return code	
–	-
Functional Description	
This API can be used to get the version information of the FrTSyn.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This API is only available if enabled by the configuration parameter `FrTSynVersionInfoApi`.	
Expected Caller Context	
> No restriction	

Table 5-4 FrTSyn_GetVersionInfo

5.2.4 FrTSyn_SetTransmissionMode

Prototype	
`void` **`FrTSyn_SetTransmissionMode`** `( uint8 CtrlIdx, FrTSyn_TransmissionModeType Mode )`	
Parameter	
`CtrlIdx`	Index of the FlexRay channel.
`Mode`	FRTSYN_TX_OFF: Turn TX capabilities off FRTSYN_TX_ON: Turn TX capabilities on
Return code	
-	-
Functional Description	
This API is used to turn on and off the TX capabilities of the FrTSyn.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 5-5 FrTSyn_SetTransmissionMode

5.2.5 FrTSyn_MainFunction

Prototype	
`void FrTSyn_MainFunction ( void)`	
Parameter	
–	-
Return code	
–	-
Functional Description	
Main function for cyclic and immediate call / resp. SYNC transmission.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> Task context	

Table 5-6 FrTSyn_MainFunction

5.3 Services used by FRTSYN

In the following table services provided by other components, which are used by the FRTSYN are listed. For details about prototype and functionality refer to the documentation of the providing component [4], [5], [6], [7], [8].

Component	API
StbM	StbM_BusGetCurrentTime StbM_BusSetGlobalTime StbM_GetOffset StbM_GetTimeBaseStatus StbM_GetTimeBaseUpdateCounter StbM_GetCurrentVirtualLocalTime
FrIf	FrIf_Transmit FrIf_GetCycleLength FrIf_GetGlobalTime FrIf_GetMacrotickDuration FrIf_GetState
Crc	Crc_CalculateCRC8H2F
Det	Det_ReportError
RTE / SchM	SchM_Enter_FrTSyn_FRTSYN_EXCLUSIVE_AREA_0 SchM_Exit_FrTSyn_FRTSYN_EXCLUSIVE_AREA_0

Table 5-7 Services used by the FRTSYN

5.4 Callback Functions

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

5.4.1 FrTSyn_RxIndication

Prototype	
`void` **`FrTSyn_RxIndication`** `( PduIdType RxPduId, const PduInfoType *PduInfoPtr )`	
Parameter	
`RxPduId`	ID of the received I-PDU.
`PduInfoPtr`	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Return code	
-	-
Functional Description	
Indication of a received I-PDU from a lower layer communication interface module.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant for different PduIds. Non-reentrant for the same PduId.	
Expected Caller Context	
> No restriction	

Table 5-8 FrTSyn_RxIndication

5.4.2 FrTSyn_TriggerTransmit

Prototype	
`Std_ReturnType` **`FrTSyn_TriggerTransmit`** `( PduIdType TxPduId, PduInfoType *PduInfoPtr )`	
Parameter	
`TxPduId`	ID of the SDU that is requested to be transmitted.
`PduInfoPtr`	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLength. On return, the service will indicate the length of the copied SDU data in SduLength.
Return code	
`Std_ReturnType`	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Functional Description	
Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant for different PduIds. Non-reentrant for the same PduId.	
Expected Caller Context	
> No restriction	

Table 5-9 FrTSyn_TriggerTransmit

6 Configuration

In the FRTSYN the attributes can be configured with the following tools:

- Configuration in DaVinci Configurator

6.1 Configuration Variants

The FRTSYN supports the configuration variants

- `VARIANT-PRE-COMPILE`

The configuration classes of the FRTSYN parameters depend on the supported configuration variants. For their definitions please see the FrTSyn_bswmd.arxml file.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
DaVinci Configurator	Configuration and generation tool for MICROSAR Classic components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CRC	Cyclic Redundancy Check
DET	Development Error Tracer
ECU	Electronic Control Unit
FRIF	FlexRay Interface
FRTSYN	Time Synchronization over FlexRay
HIS	Hersteller Initiative Software
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PDU	Protocol Data Unit
RTE	Runtime Environment
SCHM	Schedule Manager
SGW	Synchronized Gateway
SRS	Software Requirement Specification
STBM	Synchronized Time-Base Manager
SWC	Software Component
SWS	Software Specification

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com