VECTOR

MICROSAR Classic CANTSYN

Technical Reference

Version 6.2.0

Authors	visssf, vismno, vistra, viscpi, vissi, visbfk, visjwe, visgig, istaehle
Status	Released

Document Information

History

Author	Date	Version	Remarks
visssf	2014-11-05	1.0.0	Initial version
visssf	2014-12-05	1.1.0	Minor corrections
visssf	2017-05-10	3.1.0	Support multiple Time Domains in Tx state machine
vismno	2017-07-03	3.2.0	Debounce Time introduction
vistra	2017-08-03	3.3.0	Immediate Time Synchronization
viscpi, vistra	2017-10-06	3.4.0	New DET codes and return type for API SetTransmissionMode changed Message type compatibility
vissi	2018-04-04	3.5.0	Updated AUTOSAR architecture Updated referenced documents Updated list of unsupported features
vissi	2018-05-22	3.6.0	Improved exclusive area description Support Measurement (MC Data)
vissi	2018-08-31	3.6.1	Updated chapter 4.1 to new template Updated AUTOSAR architecture Updated referenced documents
visssf	2019-01-15	3.7.0	MISRA-C:2012 compliance
vistra, vissi	2019-03-06	3.7.1	Updated used services Updated Det error codes Minor improvements
vissi	2019-06-14	4.0.0	Updated table of supported AUTOSAR features
vissi	2019-12-13	4.1.0	Updated Component History
visbfk, vistra	2020-03-30	5.0.0	Updated table of services used by the CANTSYN, chapter 3.1.1 updated CanTSyn is not initialized by EcuM
visssf	2020-08-05	5.1.0	Support Helix QAC 2019-2
visjwe	2020-10-19	5.2.0	Support Extended message format for Synchronized Time Bases (CAN-FD)
visjwe, visgig, istaehle	2022-04-20	6.0.0	ASR memmap include structure ALL SLP Advanced MC Distribution for CAN Product name updated to MICROSAR Classic
visssf	2023-04-12	6.2.0	Post-Build Variant Support

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	Specification of Time Synchronization over CAN	R19-11
[2]	AUTOSAR	Specification of Time Synchronization over CAN	R4.2.2
[3]	AUTOSAR	List of Basic Software Modules	R19-11
[4]	AUTOSAR	Specification of Default Error Tracer	R19-11
[5]	AUTOSAR	Specification of RTE Software	R19-11
[6]	AUTOSAR	Specification of Synchronized Time-Base Manager	R19-11
[7]	AUTOSAR	Specification of CAN Interface	R19-11
[8]	AUTOSAR	Specification of CRC Routines	R19-11

Scope of the Document

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

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 Introduction

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

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile, post-build-selectable	
Vendor ID:	CANTSYN_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CANTSYN_MODULE_ID	161 decimal (according to ref. [3])

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

The CanTSyn module handles the distribution of time information over CAN busses.

1.1 Architecture Overview

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

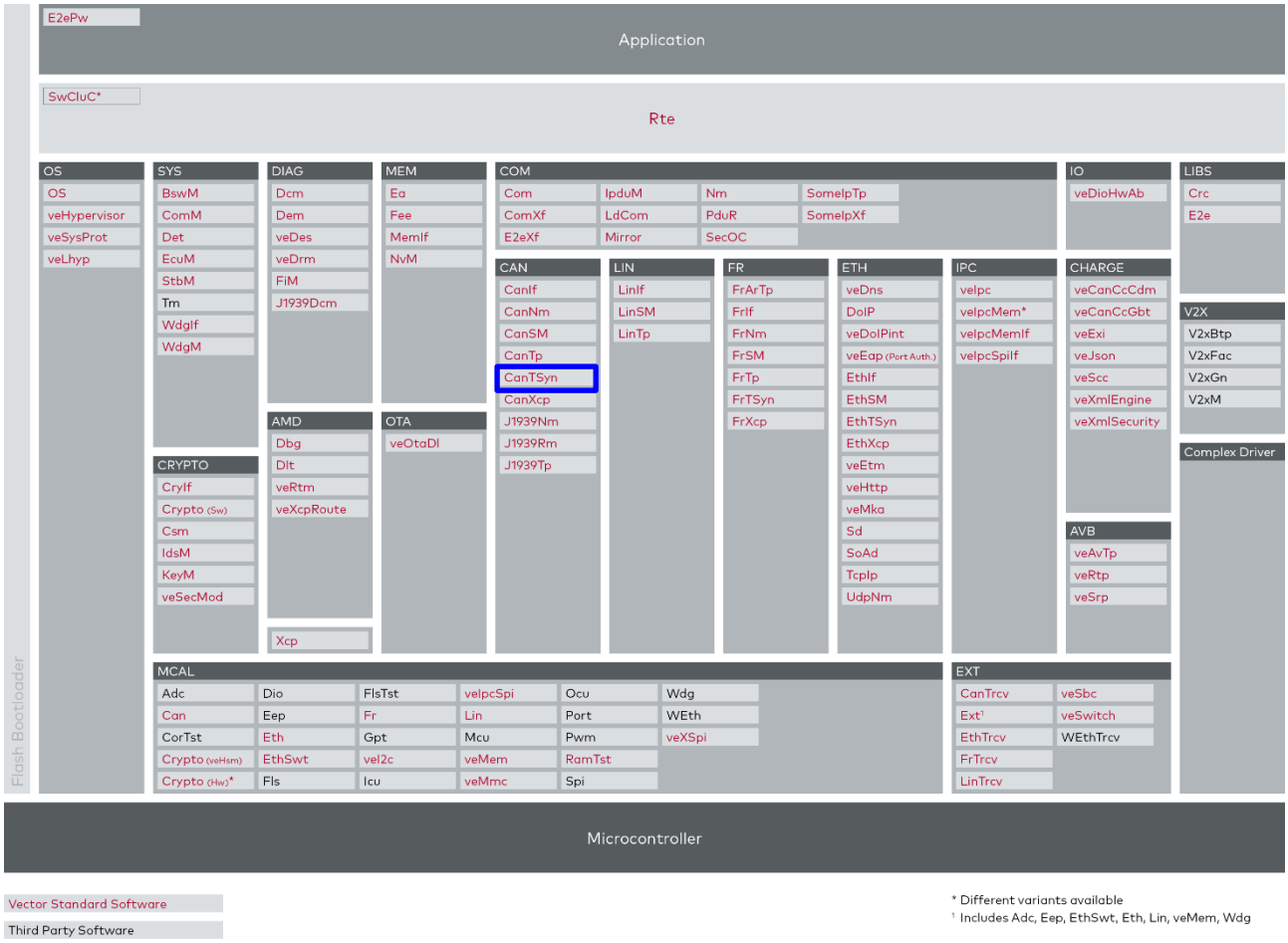

Figure 1-1 AUTOSAR Architecture Overview

The next figure shows the interfaces to adjacent modules of the CANTSYN. These interfaces are described in chapter 4.

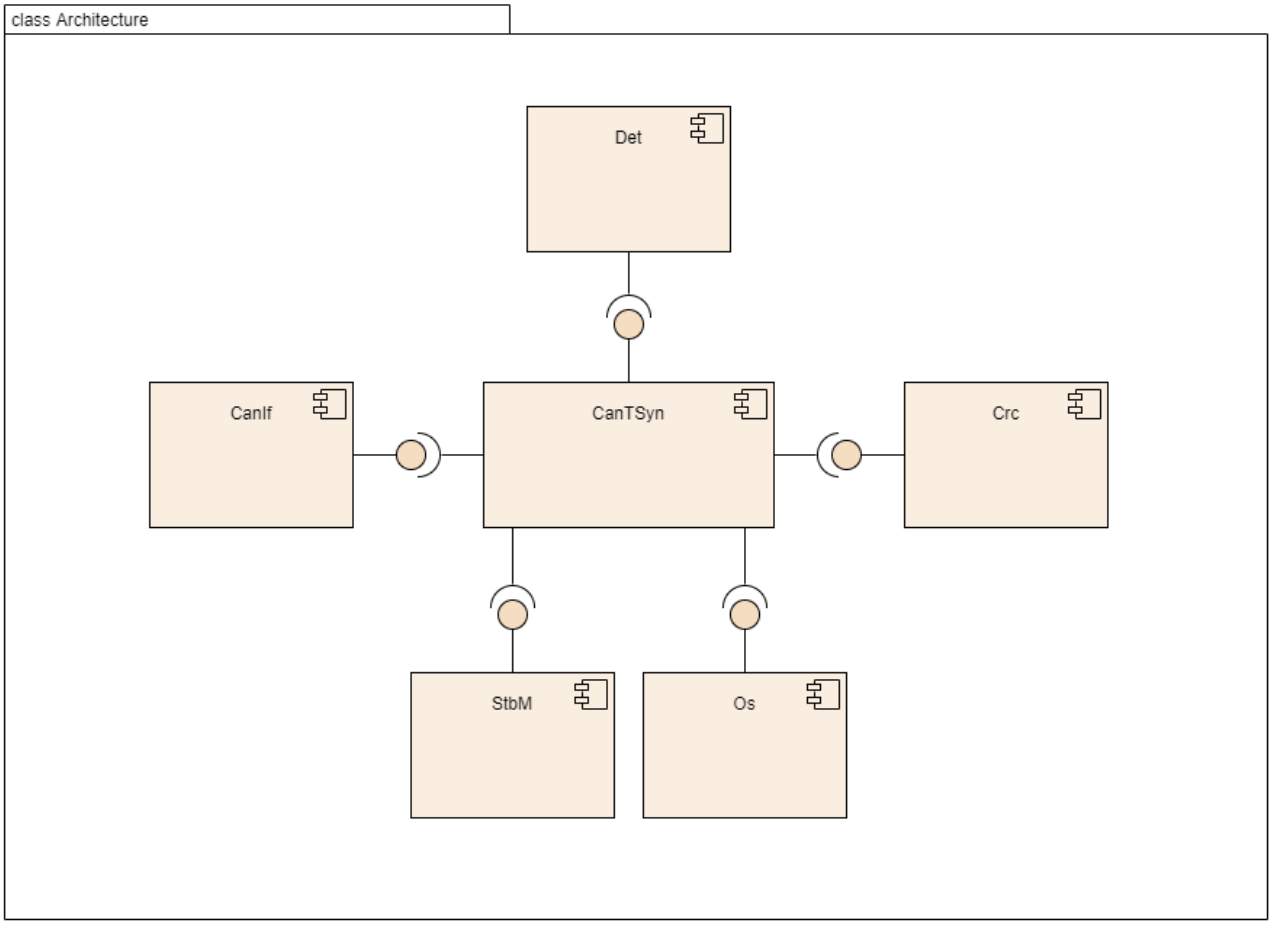

Figure 1-2 Interfaces to adjacent modules of the CANTSYN

2 Functional Description

2.1 Features

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

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

> Table 2-1 Supported AUTOSAR standard conform features

> Table 2-2 Not supported AUTOSAR standard conform features

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

> Table 2-3 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 CAN
Validation and disassembling of Time Synchronization Messages on CAN
Standard Message Format (SYNC, FUP)
Offset Message Format (OFS, OFNS)
Extended Message Format for Synchronized Time Bases (CAN FD PDU)
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
Configuration variant Post-Build

Table 2-1 Supported AUTOSAR standard conform features

2.1.1 Deviations

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

Category	Description
Config	CRC validation option CRC_OPTIONAL
Functional	Extended Message Format for Offset Time Bases (CAN FD PDU)
Functional	User Data in Offset Message Format for Offset Time Bases

Category	Description
Functional	SGW status in Offset Message Format for Offset Time Bases
Functional	Message Authentication
Functional	Minimum Message Gap

Table 2-2 Not supported AUTOSAR standard conform features

2.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond the AUTOSAR Standard
Memory Initialization
Message type compatibility
Multi-Partition, Multi-Core and Mixed ASIL support

Table 2-3 Features provided beyond the AUTOSAR standard

2.1.2.1 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 CanTSyn provides an additional function to initialize the relevant variables of the CanTSyn. See also chapters 2.2 and 4.2.2 for details.

2.1.2.2 Message Type Compatibility

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

2.1.3 Limitations

2.1.3.1 Multi-Core/Multi-Partition

`OsResources` cannot span over multiple cores. Therefore, `OsResouces` cannot be used as exclusive area impl. mechanism for CanTSyn in a multi-core CanTSyn configuration. Please consider Critical Sections.

2.2 Initialization

The Time Synchronization over CAN is initialized in three steps:

- `CanTSyn_InitMemory()`: Initializes the RAM variables. On platform where RAM is not initialized to zero by the startup code, this function has to be called first before `CanTSyn_PreInit()`
- `CanTSyn_PreInit()`. Called by the EcuM before the Os is started.
- `CanTSyn_Init()`. Called by the BSW Mode Manager (BswM) for every configured partition after Os has started.

2.3 States

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

2.3.1 Message transmission states

> CANTSYN_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.

> CANTSYN_STATE_SEND_WAITING_FOR_SYNC_TX_CONFIRMATION

After transmission of the SYNC message the time master waits for the TX confirmation. If a timeout occurs while waiting for the TX confirmation the master resets its state and sends the next SYNC message.

> CANTSYN_STATE_SEND_WAITING_FOR_FOLLOW_UP_SEND

If the TX confirmation for the SYNC message is received before a timeout occurs the time master sends the FUP message.

> CANTSYN_STATE_SEND_WAITING_FOR_FOLLOW_UP_TX_CONFIRMATION

After transmission of the FUP message the time master waits for the TX confirmation. When the TX confirmation is received or a timeout occurs the master resets its state and sends the next SYNC message.

2.3.2 Message reception states

> CANTSYN_STATE_RECEIVE_WAITING_FOR_SYNC

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

> CANTSYN_STATE_RECEIVE_WAITING_FOR_FOLLOW_UP

After reception of a SYNC message a time slave is waiting for a FUP message. When the message was received or the configured follow up timeout time is expired, the time slave will reset its state and wait for the next SYNC message.

2.4 Main Functions

The `CanTSyn_MainFunction()` triggers the transmission of Time Synchronization Messages and monitors timeouts for correct handling of the RX and TX state machines. Depending on the configuration cyclic and immediate transmission is possible.

2.4.1 Main Functions with Multi-Partition configurations

For configurations where CanTSyn is mapped to more than one partition, a `CanTSyn_MainFunction_<OsApplication>()` per partition is generated. Each of these generated main functions have to be mapped to the correct partition.

2.5 Error Handling

2.5.1 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 `CANTSYN_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 CANTSYN ID is 161.

The reported service IDs identify the services which are described in 4.2. The following table presents the service IDs and the related services:

Service ID	Service
0x01	CanTSyn_Init
0x02	CanTSyn_GetVersionInfo
0x03	CanTSyn_SetTransmissionMode
0x42	CanTSyn_RxIndication
0x40	CanTSyn_TxConfirmation
0x06	CanTSyn_MainFunction
0xC0	CanTSyn_PreInit

Table 2-4 Service IDs

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

Error Code	Description
0x01	CANTSYN_E_INVALID_PDUID
0x02	CANTSYN_E_UNINIT
0x03	CANTSYN_E_NULL_POINTER
0x04	CANTSYN_E_INIT_FAILED
0x05	CANTSYN_E_PARAM
0x06	CANTSYN_E_INV_CTRL_IDX

Table 2-5 Errors reported to DET

2.5.2 Production Code Error Reporting

No production error codes are currently used by CanTSyn.

3 Integration

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

3.1 Embedded Implementation

The delivery of the CANTSYN consists out of these files:

File Name	Description	Integration Tasks
CanTSyn.c	Main implementation file of the CanTSyn.	-
CanTSyn.h	Main header file of the CanTSyn.	-
CanTSyn_Cbk.h	Header file that contains the prototypes of callback functions of the CanTSyn.	-
CanTSyn_Types.h	Header file that contains the type definitions of the CanTSyn.	-
CanTSyn_Cfg.c	Generated file that contains definitions of structures in pre-compile-time and post-build variant.	-
CanTSyn_Cfg.h	Generated file that contains declarations of structures in pre-compile-time and post-build variant.	-

Table 3-1 Implementation files

3.2 Critical Sections

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

`CANTSYN_EXCLUSIVE_AREA_0`

The CanTSyn 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 CanTSyn 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 `CANTSYN_EXCLUSIVE_AREA_0` inside the RTE / SchM configuration needs to be set to `OS_RESOURCE` or `CUSTOM`.

If `OS_RESOURCE` is selected the CAN 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].

3.3 Memory Sections

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

4 API Description

For an interface overview please see Figure 1-2.

4.1 Type Definitions

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

Type Name	C-Type	Description	Value Range
CanTSyn_ConfigType	struct	Post-build configuration structure	–
CanTSyn_TransmissionModeType	enum	Handles the enabling and disabling of the transmission mode	`CANTSYN_TX_OFF` Transmission disabled
			`CANTSYN_TX_ON` Transmission enabled

Table 4-1 Type definitions

4.2 Services provided by CANTSYN

4.2.1 CanTSyn_Init

Prototype	
`void` **`CanTSyn_Init`** `( const CanTSyn_ConfigType *configPtr )`	
Parameter	
`configPtr`	Pointer to selected configuration structure.
Return code	
-	-
Functional Description	
This function initializes the Time Synchronization over CAN.	
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 CanTSyn service function is called (except `CanTSyn_PreInit()`).	
Expected Caller Context	
> Task context	

Table 4-2 CanTSyn_Init

4.2.2 CanTSyn_InitMemory

Prototype	
`void` **`CanTSyn_InitMemory`** `( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
Initializes the global variables in case an initializing startup code is not used. This function sets the CanTSyn 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 CanTSyn function after startup.	
Expected Caller Context	
> Task context	

Table 4-3 CanTSyn_InitMemory

4.2.3 CanTSyn_PreInit

Prototype	
`void` **`CanTSyn_PreInit`** `( const CanTSyn_ConfigType *configPtr )`	
Parameter	
`configPtr`	Pointer to selected configuration structure.
Return code	
–	-
Functional Description	
This function initializes the Time Synchronization over CAN.	
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 EcuM during the startup phase. > This function has to be called before any other CanTSyn service function is called (except `CanTSyn_InitMemory()`).	
Expected Caller Context	
> Task context	

Table 4-4 CanTSyn_PreInit

4.2.4 CanTSyn_GetVersionInfo

Prototype	
`void CanTSyn_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 CanTSyn.	
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 `CanTSynVersionInfoApi`.	
Expected Caller Context	
> No restriction	

Table 4-5 CanTSyn_GetVersionInfo

4.2.5 CanTSyn_SetTransmissionMode

Prototype	
`void CanTSyn_SetTransmissionMode ( uint8 CtrlIdx, CanTSyn_TransmissionModeType Mode )`	
Parameter	
`CtrlIdx`	Index of the CAN channel
`Mode`	CANTSYN_TX_OFF: Turn TX capabilities off CANTSYN_TX_ON: Turn TX capabilities on
Return code	
-	-
Functional Description	
This API is used to turn on and off the TX capabilities of the CanTSyn.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> No restriction	

Table 4-6 CanTSyn_SetTransmissionMode

4.2.6 CanTSyn_MainFunction

Prototype	
`void CanTSyn_MainFunction ( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
Main function for cyclic and immediate part of message transmission. Handling of cyclic part of message reception	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> Task context	

Table 4-7 CanTSyn_MainFunction

4.3 Services used by CANTSYN

In the following table services provided by other components, which are used by the CANTSYN 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
CanIf	CanIf_Transmit
Crc	Crc_CalculateCRC8H2F
Det	Det_ReportError
RTE / SchM	SchM_Enter_CanTSyn_CANTSYN_EXCLUSIVE_AREA_0 SchM_Exit_CanTSyn_CANTSYN_EXCLUSIVE_AREA_0
Os	GetApplicationID

Table 4-8 Services used by the CANTSYN

4.4 Callback Functions

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

4.4.1 CanTSyn_RxIndication

Prototype	
`void` **`CanTSyn_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 4-9 CanTSyn_RxIndication

4.4.2 CanTSyn_TxConfirmation

Prototype	
`void` **`CanTSyn_TxConfirmation`** `( PduIdType TxPduId )`	
Parameter	
`TxPduId`	ID of the I-PDU that has been transmitted.
Return code	
-	-
Functional Description	
The lower layer communication interface module confirms the transmission of an I-PDU.	
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 4-10 CanTSyn_TxConfirmation

5 Configuration

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

- Configuration in DaVinci Configurator

5.1 Configuration Variants

The CANTSYN supports the configuration variants

- `VARIANT-PRE-COMPILE`
- `VARIANT-POST-BUILD-SELECTABLE`

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

6 Glossary and Abbreviations

6.1 Glossary

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

Table 6-1 Glossary

6.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CAN	Controller Area Network
CANIF	CAN Interface
CANTSYN	Time Synchronization over CAN
CRC	Cyclic Redundancy Check
DET	Development Error Tracer
ECU	Electronic Control Unit
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 6-2 Abbreviations

7 Contact

Visit our website for more information on

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

www.vector.com