VECTOR

MICROSAR EthTSyn

Technical Reference

Global Time Synchronization over Ethernet

Version 5.0.0

Authors	Jeroen Laverman, Michael Seidenspinner
Status	Released

Document Information

History

Author	Date	Version	Remarks
Jeroen Laverman	2014-09-23	1.0.0	Creation of document
Jeroen Laverman	2014-11-11	1.0.1	Add new API
Michael Seidenspinner	2015-01-12	1.0.2	ESCAN00080452 Modification of Chapter “Configuration”
Michael Seidenspinner	2015-02-26	1.1.0	Added Configuration for SW-Timestamping
Michael Seidenspinner	2015-07-10	1.2.0	ESCAN00083936
Michael Seidenspinner	2015-10-23	2.0.0	ESCAN00085377: FEAT-1529: Support Ethernet Switches for Ethernet Time Sync
Michael Seidenspinner	2016-03-15	2.1.0	ESCAN00085303 Added Chapters: > Boundary Clock > AlwaysAsCapable > Announce > Source Port Identity Check > Correction Action
Michael Seidenspinner	2016-05-13	2.2.0	ESCAN00089686 Added Chapter: > Flexible Pdelay configuration
Michael Seidenspinner	2016-11-25	3.0.0	FEATC-248: FEAT-1998: Support of HW Time Stamping for Switch for EthTSyn (SysService_AsrTSynEth)
Michael Seidenspinner	2017-03-07	4.0.0	FEATC-383: FEAT-2279: Time Synchronization acc. AR 4.3 for EthTSyn
Michael Seidenspinner	2017-04-12	4.0.1	Fixed review findings
Michael Seidenspinner	2017-07-05	5.0.0	STORYC-1213, STORY-128

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_TimeSyncOverEthernet.pdf	V4.3.0
[2]	AUTOSAR	AUTOSAR_SWS_DefaultErrorTracer.pdf	V4.3.0
[3]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	V4.3.0
[4]	AUTOSAR	AUTOSAR_SWS_SynchronizedTimeBaseManager.pdf	V4.3.0

[5]	IEEE	IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks	2011
[6]	IEEE	IEEE 1588-2008: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems	2008
[7]	Vector	TechnicalReference_IpBase.pdf	see delivery
[8]	Vector	TechnicalReference_EthIf.pdf	see delivery

Scope of the Document

This technical reference describes the general use of the EthTSyn basis software. The EthTSyn can only be used in conjunction with the StbM (see [4]), the EthIf (see [8]) and the Eth basis software module which is also part of the delivery.

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.xx	Created Beta version
1.01.xx	SW-Timestamp support
2.00.xx	Time-Aware-Bridge support
2.01.xx	Boundary Clock support
2.02.xx	Flexible Pdelay configuration
3.00.xx	Switch Timestamp support
4.00.xx	AR 4.3 Support
5.00.xx	Immediate Time Sync support
5.01.xx	P3 release

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release*:	4.3.0	
Supported Configuration Variants:	Pre-compile	
Vendor ID:	EthTSyn_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	EthTSyn_MODULE_ID	164 decimal (according to ref. [1])

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

The EthTSyn module provides the functionality of time synchronization defined by the gPTP (generalized Precision Time Protocol) IEEE 802.1AS-2011 (see [5]). The gPTP uses Ethernet communication for transmission of time stamped Ethernet frames to achieve time synchronization to a master clock.

Caution
When HW Timestamping is activated, the EthTSyn uses a hardware timer / counter for determination of frame ingress and
egress timestamps. This hardware timer / counter module is provided as special PTP (see [6]) feature of some Ethernet controller devices. These additional Ethernet frame timing features are accessed by the EthTSyn over an API extension of the EthIf.

The EthTSyn module offers the functionality to:

- Provide global network time as master clock role
- Provide global network time to the StbM for synchronization as slave clock role
- Use of Hardware or Software for Timestamping

2.1 Architecture Overview

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

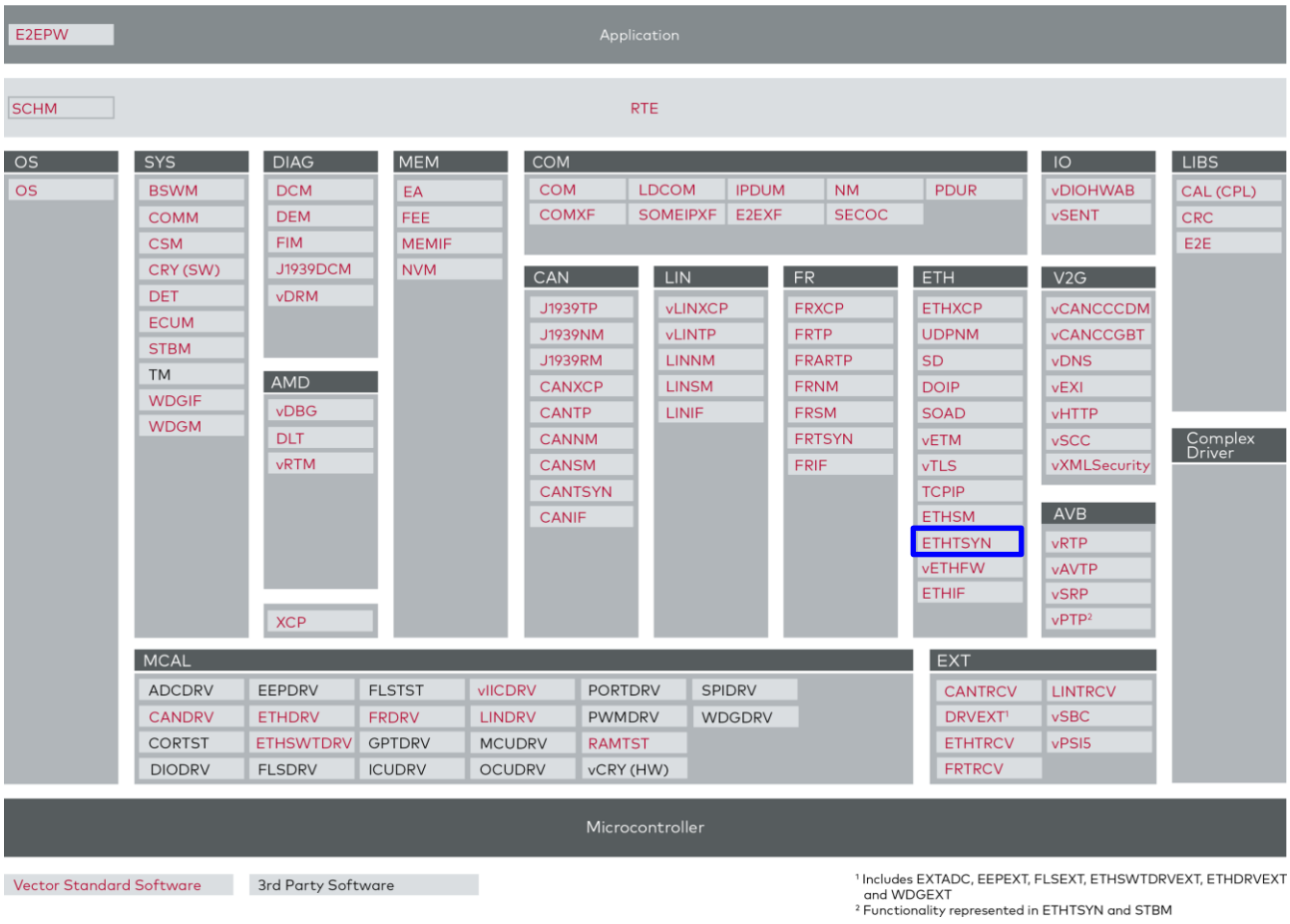

Figure 2-1 AUTOSAR 4.2 Architecture Overview

Figure 2-2 shows the interfaces to adjacent modules of the EthTSyn. These interfaces are described in chapter 5.

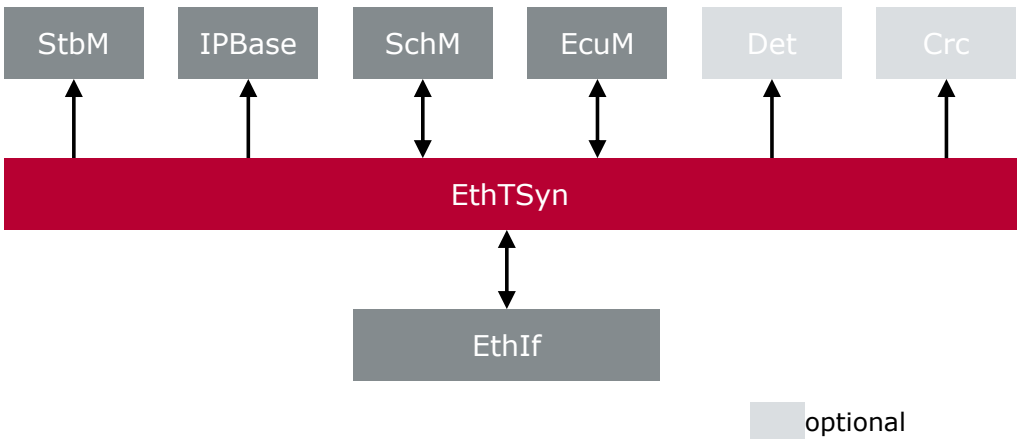

Figure 2-2 Interface to adjacent modules of EthTSyn

3 Functional Description

3.1 Features

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

3.1.1 IEEE 802.1AS-2011 Conformance

The following features specified in [5] “IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks” are supported:

Supported features
Clock Master Role
Clock Slave Role

Table 3-1 Feature conformance to IEEE 802.1AS-2011

3.1.2 General Limitations

The following general limitations apply to the EthTSyn.

Not supported features
Best Master Clock Algorithm (BMCA) support

Table 3-2 General Limitations

3.1.3 IEEE 802.1AS-2011 Deviations

The following deviations of [5] “IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks” apply to the EthTSyn.

Section in Document	Limitation
6.3.3.4 Data Types	- TimeInterval Format is not supported - ScaledNs Format is not supported - UScaledNs Format is not supported - ExtendedTimestamp Format is not supported
8.2.2 Timescale Epoch	gPTP domain epoch is not supported
8.2.3 Timescale UTC Offset	UTC Offset is not supported
8.6 Time-aware system characterization	No Time-aware system characterization is supported
10.4 Message Attributes	Correction field in General Messages is not supported
11.4 Message Format	Header Correction Field is not supported
11.2 State machines	Announce Messages are transmitted optionally following to FollowUp Messages in clock master role
11.2.16 MDPdelay-Req state machine	The loss of allowedLostResponses number of responses is tolerated. The standard allows allowedLostResponses + 1 missing responses before setting AsCapable = false.

Table 3-3 Deviations to IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks

3.1.4 AUTOSAR 4.3.0 Deviations

The following deviations of [1] “AUTOSAR_SWS_TimeSyncOverEthernet.pdf” apply to the EthTSyn module.

Section in Document	Limitation
7.3 Debounce Time	Debounce Time is not supported
7.6.1.1 and 7.7.1.1 Runtime Error detection	Master-Slave conflict detection is not supported

Table 3-4 Deviations to AUTOSAR 4.3.0: Specification of Global Time Synchronization over Ethernet

3.1.5 Time-Aware-Bridge (Switch Management)

This feature can be turned on when an EthSwt is included in the configuration by using the configuration parameter `EthTSynEnableSwitchManagement`. The EthTSyn module will then manage all gPTP traffic of the EthSwt. If the Host CPU itself should take Part in the time synchronization, an extra EthTSyn-Port (End-Station Port) for the Host CPU has to be configured. The following constraints apply for the configuration of the End-Station Port:

- If a Switch-Slave Port is configured, the End-Station Port only can be a Slave Port as well.
- When only Master-Ports are configured for the Switch, an End-Station Master-Port has to be present as well since the Switch itself cannot act as Time Master. In this case, all Switch Master-Ports will inherit the configuration parameters of the End-Station Master-Port.

Figure 3-1 provides an overview of the general structure when an EthSwt is used. The EthTSyn module is running on the Host CPU. The EthSwt is connected with the Host CPU e.g. via MII using the determined management Port of the EthSwt.

Note
The EthSwt has to run in the so called ‘Managed Mode’ such that all PTP traffic is trapped and forwarded to the Host CPU only. The normal forwarding rules do not apply to the PTP traffic.

Figure 3-2 provides an overview of the sequence for PTP frames received on an EthSwt port. Figure 3-3 provides an overview of the sequence applying to PTP frames transmitted by the EthTSyn on an EthSwt port.

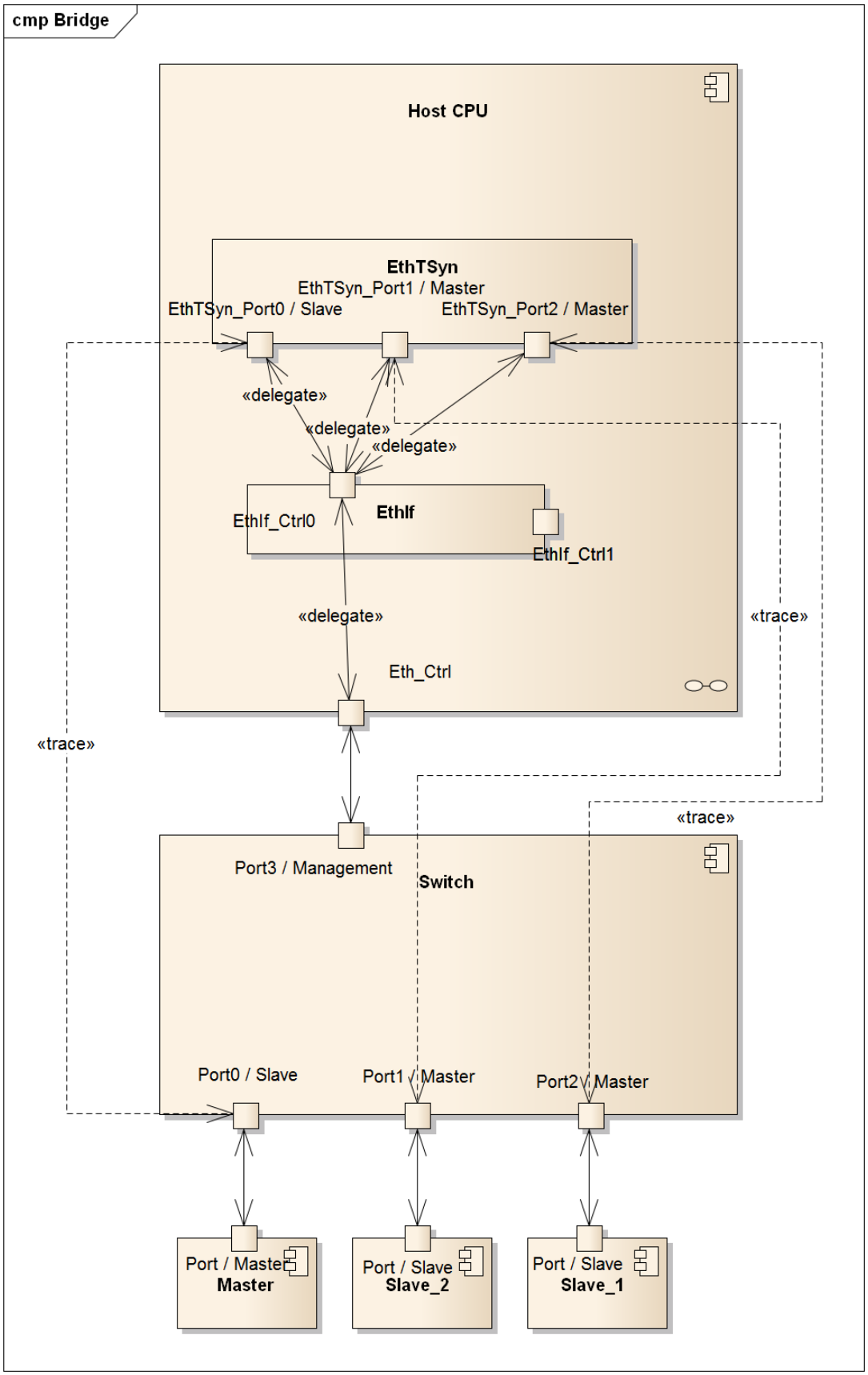

Figure 3-1 Bridge Overview

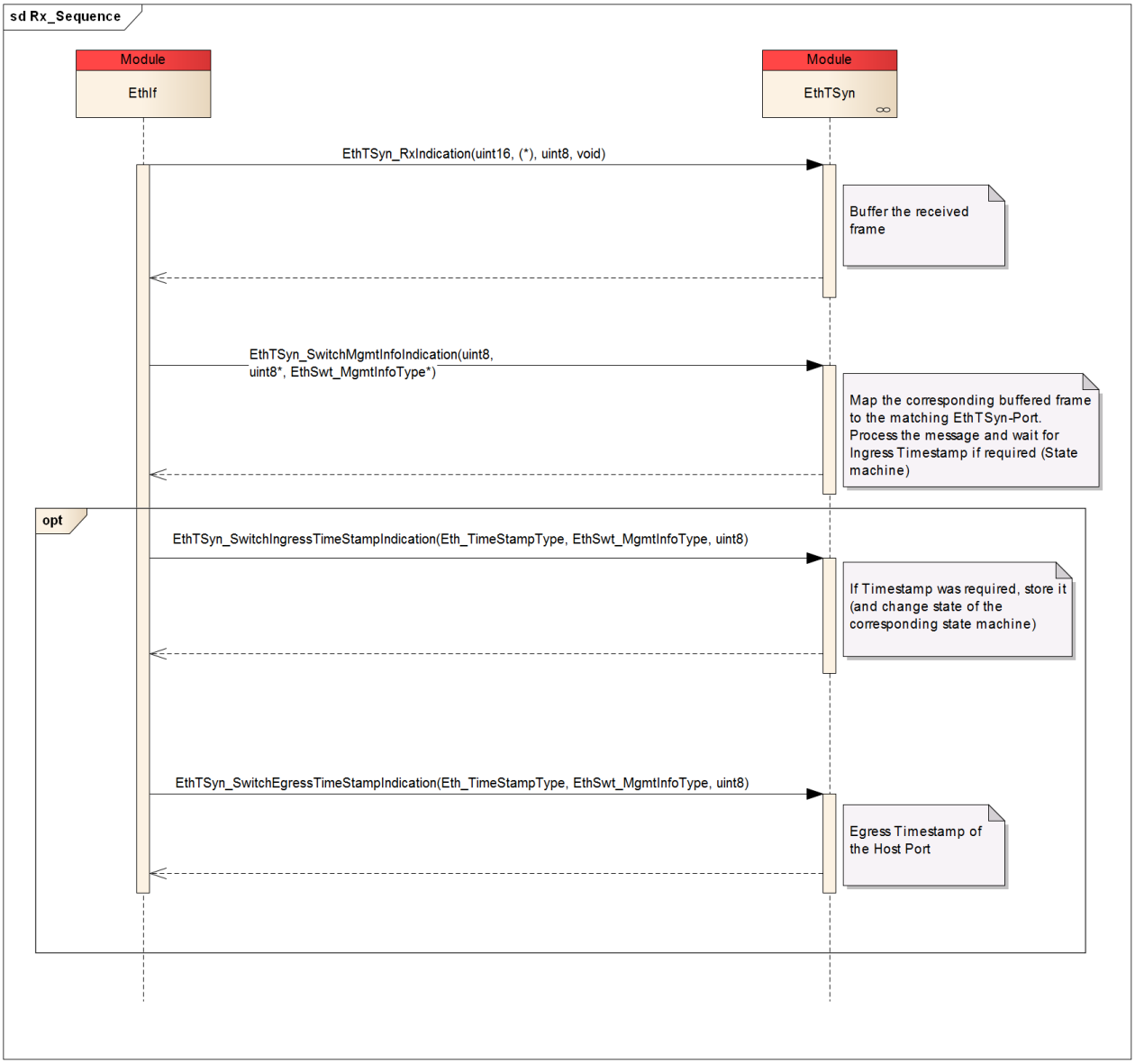

Figure 3-2 Sequence diagram of the switch management reception path

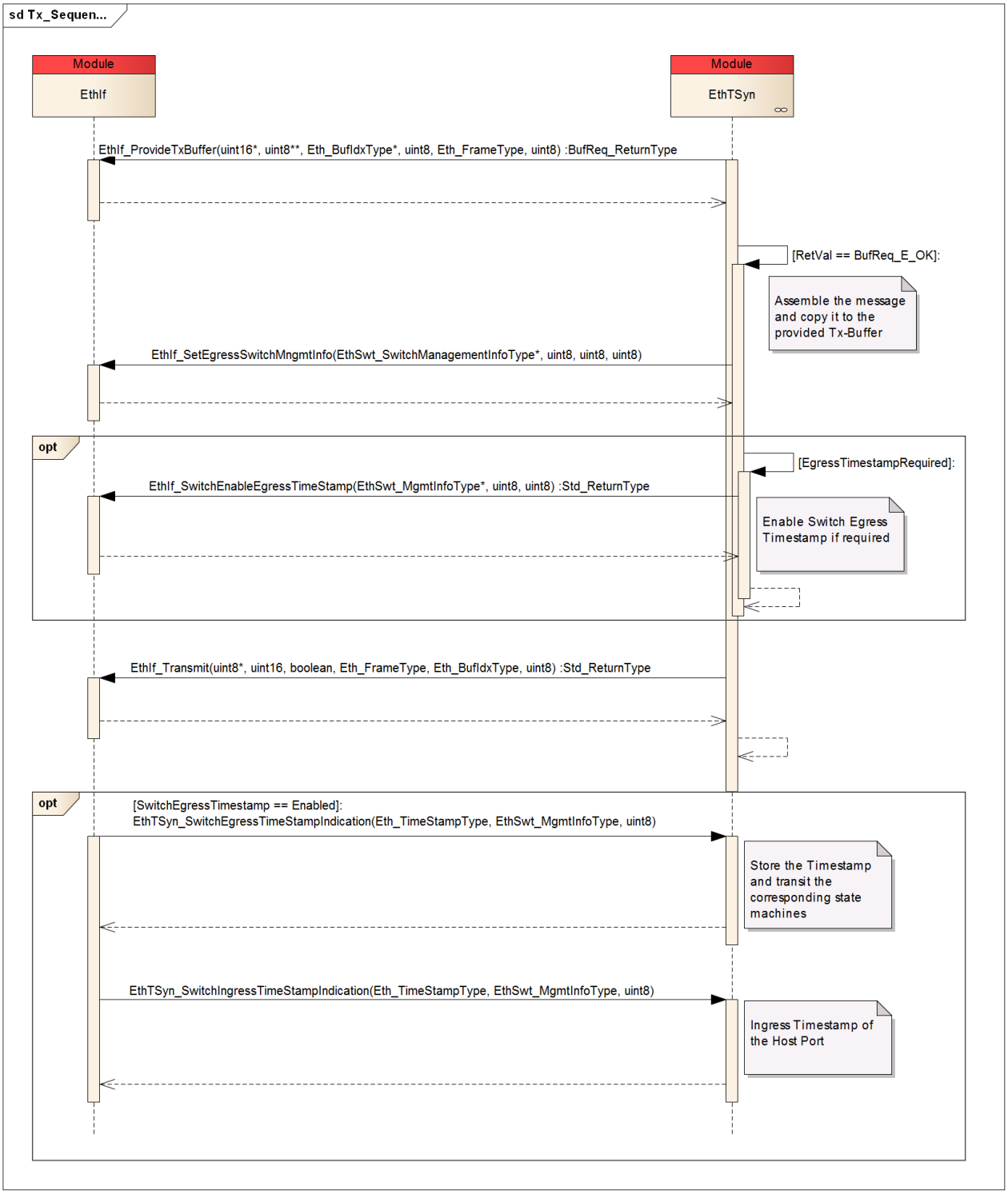

Figure 3-3 Sequence diagram of the switch management transmission path

3.1.5.1 Switch Timestamping

Whether the message timestamps from the Ethernet Switch or the Host CPU are used can be decided by the configuration parameter `EthTSynSwitchMgmtSwitchTimestampSupport`.

When Switch Timestamp Support is enabled in the EthTSyn, the timestamps have to be provided by the Ethernet Switch using the EthTSyn APIs `EthTSyn_SwitchEgressTimeStampIndication`/`EthTSyn_SwitchIngressTimeStampIndication`.

When Switch Timestamp Support is disabled in the EthTSyn, all necessary timestamps are taken when the corresponding frames are received/transmitted by the Host CPU. Therefore the switch residence time cannot be computed with disabled Switch timestamp support.

3.1.5.2 Acting as Bridge

The Time-Aware-Bridge only supports time synchronization and Peer-Delay measurement. This belongs to the gPTP message groups:

- Sync/FollowUp
- Pdelay_Req/Pdelay_Resp/Pdelay_Resp_FollowUp

3.1.5.2.1 Forwarding of Sync messages

Valid Sync messages received on the Slave-Port will be forwarded on each Master-Port of the Bridge without waiting for the reception of the corresponding FollowUp.

If no Sync message is received on the Slave-Port either after initialization or during runtime, no Sync message will be sent on any Master-Port.

3.1.5.2.2 Forwarding of FollowUp messages

When a valid FollowUp message is received on the Slave-Port it will be forwarded on each Master-Port, only if the corresponding Sync message was already transmitted on the Master-Port.

The Pdelay of the Slave-Port and the residence time of the Sync message will be added to the FollowUpCorrectionField before transmission of the FollowUp message.

3.1.5.2.3 Modification of the SourcePortIdentity for forwarded Sync/FollowUp

It is decided by configuration (`EthTSynSwitchMgmtKeepSourcePortIdentity`) if the SourcePortIdentity of forwarded Sync and FollowUp messages is modified by the bridge or not:

1. `EthTSynSwitchMgmtKeepSourcePortIdentity` is set to ‘false’: The SourcePortIdentity of each forwarded Sync and FollowUp message will be set to the PortIdentity of the bridge master port the message is transmitted on
2. `EthTSynSwitchMgmtKeepSourcePortIdentity` is set to ‘true’: The SourcePortIdentity will not be modified by the bridge

3.1.5.2.4 Pdelay

When acting as Time-Aware Bridge, the Pdelay mechanism can be configured for each Bridge-Port individually. For a detailed description see chapter 3.1.8.

3.1.5.2.5 Boundary Clock

In addition to the behavior of the Time-Aware Bridge descripted in Chapter 3.1.5 the Boundary Clock will take over the master role if a Sync-Timeout is detected on the Slave-Port of the Bridge (if the parameter `EthTSynSwitchMgmtGmCapable` is set to 'true').

3.1.5.3 Acting as Grand Master

When acting as Grand Master Sync and FollowUp messages will be transmitted on each Master-Port in the configured cycle.

3.1.6 Clock Master Role

In clock master role the EthTSyn transmits cyclic Sync and FollowUp messages from the Ethernet controller (Eth) that is referenced by the EthTSyn GlobalTimeDomain. The FollowUp message contains the *PreciseOriginTimestamp* (egress timestamp of previous Sync message) that is generated either by Hardware or Software at transmission of the Sync message.

A more detailed description of how the egress timestamp is generated can be found in [1] AUTOSAR_SWS_TimeSyncOverEthernet.pdf, Chapter 9.2.

If enabled, Announce messages providing information about the local clock properties are transmitted after a FollowUp message. This additional information is used for evaluation of BMCA (Best Master Clock Algorithm) for best master clock determination that is not supported by EthTSyn.

3.1.7 Clock Slave Role

In clock slave mode no transmission of Sync, FollowUp and Announce messages is performed. The clock slave receives the Sync message form master and stores the ingress timestamp that is either generated by HW or SW. After receiving the FollowUp message, the *PreciseOriginTimestamp* of the FollowUp message is corrected by the *CorrectionField* of the FollowUp message, the path delay time on the link and the time passed between the reception of the Sync and FollowUp message (by using the ingress timestamp of the Sync message). This corrected *PreciseOriginTimestamp* is forwarded to the StbM by using `StbM_BusSetGlobalTime()`.

To determine the path delay of message transmission on this link a path delay measurement (see chapter 3.1.8) is performed.

A more detailed description can be found in [1] AUTOSAR_SWS_TimeSyncOverEthernet.pdf, Chapter 9.3.

3.1.7.1 Announce

According to [5] IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks each Time-Master transmits cyclic Announce messages. The Announce messages are holding information about the Time-Master and the path of messages traversing the network. The Time-Slave is using this information to verify if received Sync/FollowUp messages are valid. However in an AUTOSAR environment no such Announce messages will be transmitted.

With the parameter `EthTSynEnableAnnounce` it can be decided if the information of the Announce messages is used by the Time-Slave or not.

- `EthTSynEnableAnnounce` is set to ‘false’: The information of the Announce messages will not be used by the Time-Slave and Sync/FollowUp messages with any SourcePortIdentity will be accepted (unless the Source Port Identity Check described in 3.1.7.2 is used).
- `EthTSynEnableAnnounce` is set to ‘true’: The reception of an Announce message by the Slave-Port is mandatory before synchronization is possible. The Slave-Port uses the PortIdentity with the Announce message to verify the Source Port Identity of received Sync/FollowUp messages. Furthermore the GmPresent flag is set to ‘true’ after the reception of a valid Announce.

3.1.7.2 Source Port Identity Check

With enabled source port identity check it is possible to configure a master port identity (consisting of a MAC-Address and a port number). Only Sync and FollowUp messages with the configured SourcePortIdentity will be accepted and processed by the slave. All Sync/FollowUp messages with any other SourcePortIdentity will be discarded.

3.1.8 Path Delay (Pdelay) Measurement

In each clock role (master and slave) a cyclic path delay measurement can be performed by transmitting PdelayReq messages. The neighbor node replies by transmitting PdelayResp and PdelayRespFollowUp messages.

When the path delay initiator transmits a PdelayReq message, the egress timestamp is stored. The path delay responder transmits a PdelayResp massage in return that contains the ingress timestamp of the PdelayReq message. In addition, the responder transmits a PdelayRespFollowUp message that contains the egress timestamp of the PdelayResp message. Figure 3-4 shows the sequence diagram of the path delay measurement.

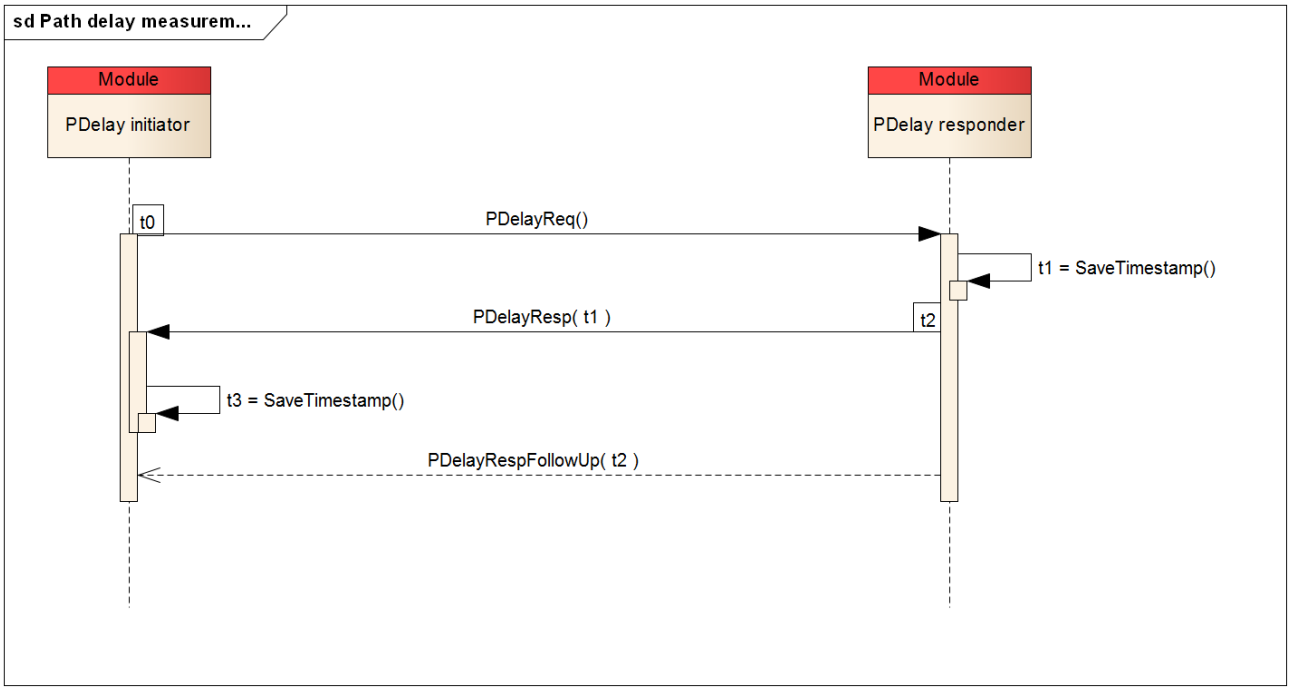

Figure 3-4 Sequence diagram of path delay measurement

3.1.8.1 Flexible Pdelay configuration

The Pdelay configuration options shown in Table 3-5 and Table 3-6 (only relevant if Pdelay Initiator is enabled) can be configured for each Port individually.

Configuration Option	Description
Global Time Tx Pdelay Req Period	The transmission period for PdelayReq messages. A value of 0 disables the Pdelay initiator functionality
Pdelay Responder	Enable or Disable the Pdelay responder functionality. A Pdelay responder will answer to received PdelayReq messages with the corresponding PdelayResp and PdelayRespFollowUp messages.
Use Static Pdelay	Enable or Disable the use of a static configured value for Pdelay. When this option is enabled, the value configured for ‘Initial Pdelay’ will always be used as Pdelay. No Pdelay calculation will be performed even if the Pdelay Initiator is enabled.
Initial Pdelay	The value initially used for Pdelay (in nanoseconds) before a valid Pdelay was calculated by the Pdelay mechanism. If the option ‘Use Static Pdelay’ is enabled, this parameter defines the static value used for Pdelay.
Always As-Capable	See 3.1.9

Table 3-5 General Pdelay configuration options

Configuration Option	Description
Pdelay Average Weight	Represents the weight factor (w) for the Pdelay calculation: $PDelay_{New} = \frac{((w-1) * PDelay_{Old}) + PDelay_{Current}}{w}$
Pdelay Neighbor Delay Threshold	The maximum valid value for Pdelay. If the threshold is exceed, the last valid Pdelay will still be used.
Pdelay Req Allowed Lost Responses	The allowed number of lost responses for a PdelayReq message. If this count is exceeded, 'AsCapable' of the Port is set to false.

Table 3-6 Pdelay initiator configuration options

Caution
If the Pdelay Initiator functionality is disabled, the configuration option 'Always AsCapable' has to be enabled.

Note
If the configuration options for both Pdelay Initiator and Use Static Pdelay are disabled, no Pdelay is used for the Port and therefore the value for Initial Pdelay has to be 0.

3.1.9 AlwaysAsCapable

When AlwaysAsCapable is set to true, AsCapable is set to true without checking for a gPTP counterpart via Pdelay-Mechanism. Hence for a master port the transmission of Sync and Follow_Up messages is possible without Pdelay measurement. For a slave port, synchronization is possible without a valid Pdelay.

3.1.10 Acting as Time-Master and Time-Slave in parallel

Each physical Port can act as Time-Master and Time-Slave in parallel, but can only have one role within the same TimeDomain. This means a port can act as Time-Slave in one TimeDomain and act as Time-Master in another TimeDomain.

3.2 Interaction with EthIf and Eth BSW module

Figure 3-5 shows a sequence diagram that presents the interaction of EthTSyn with EthIf and Eth BSW module when an event message is transmitted.

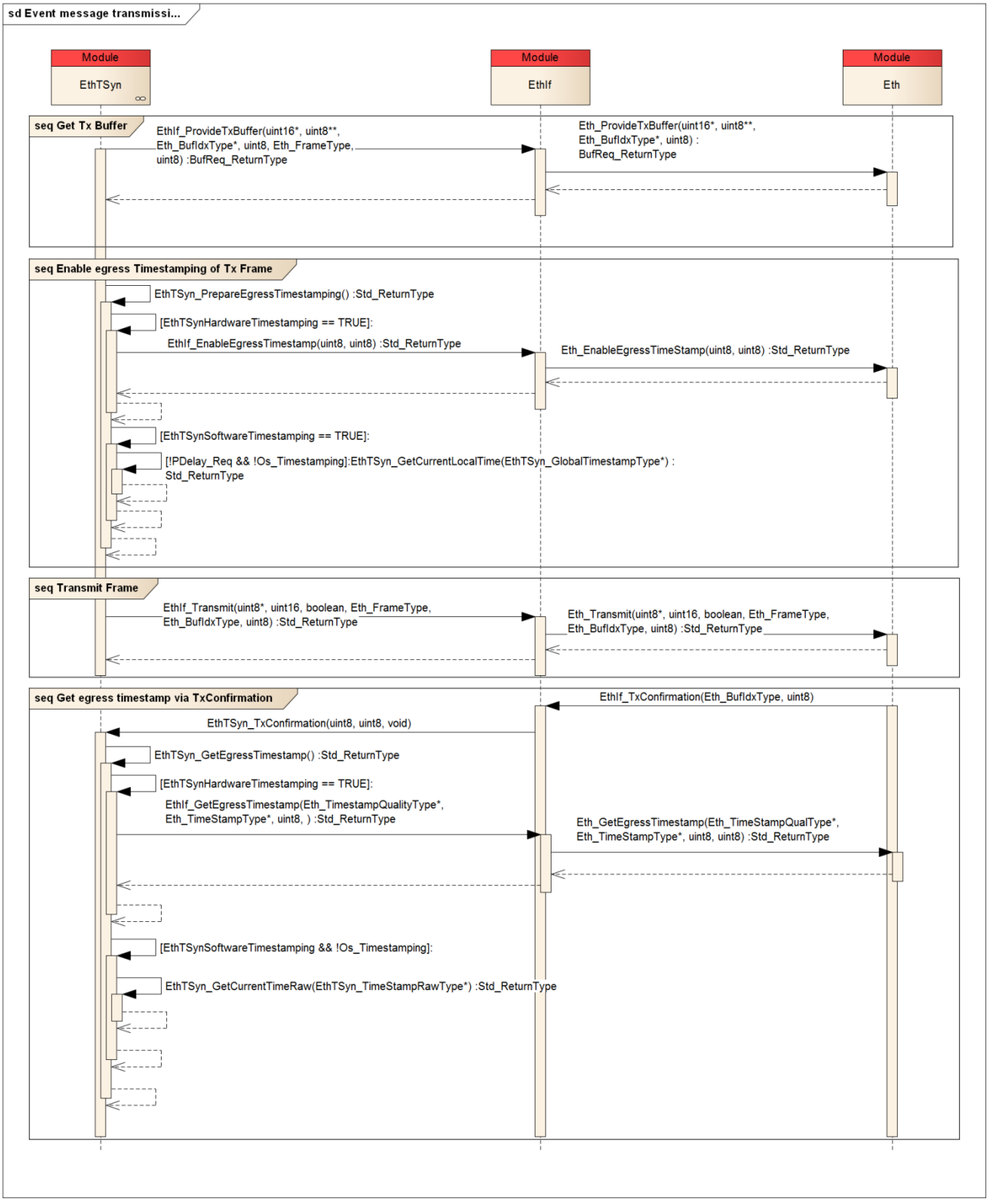

Figure 3-5 Sequence diagram of EthTSyn event message transmission

Figure 3-6 shows a sequence diagram that presents the interaction of EthTSyn with EthIf and Eth BSW module when an event message is received.

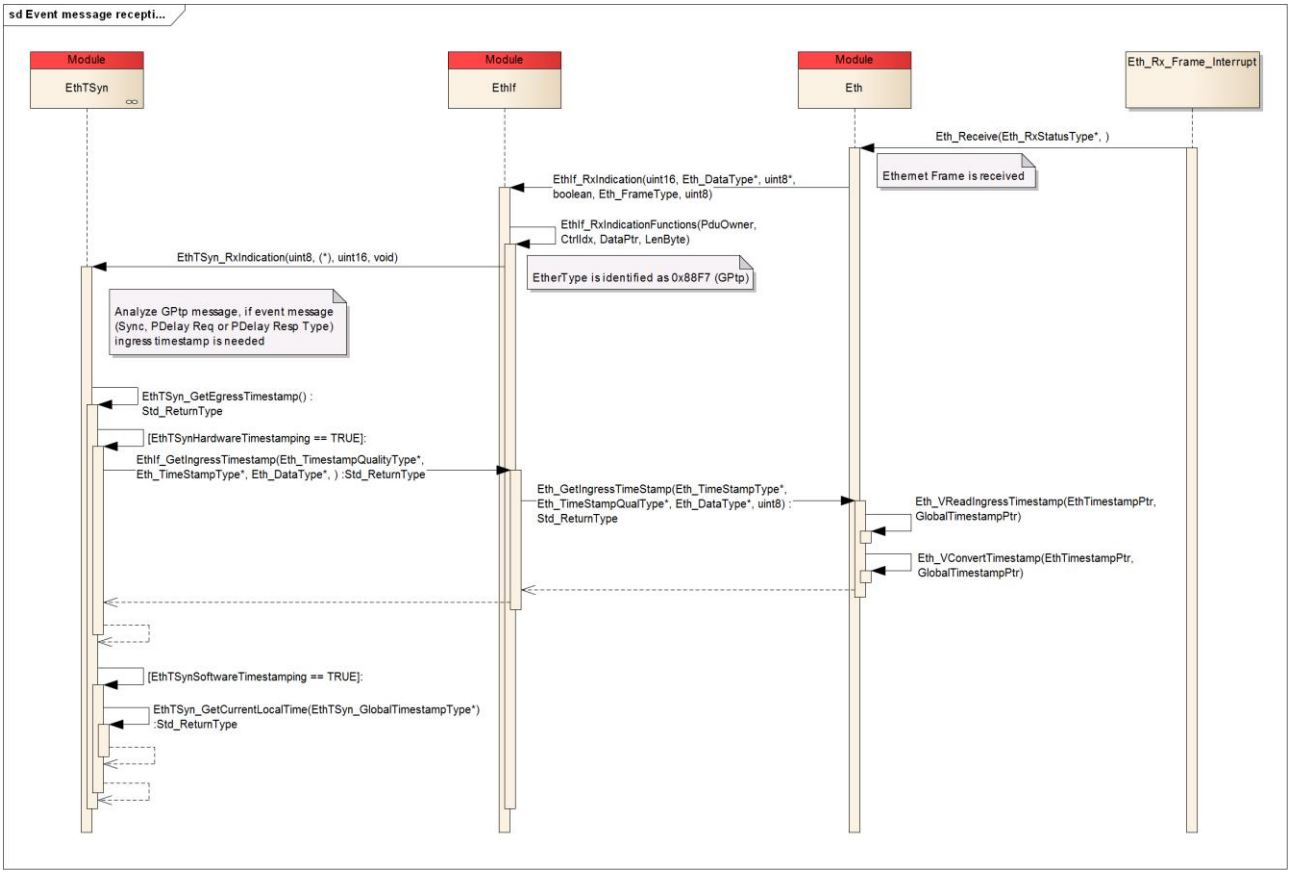

Figure 3-6 Sequence diagram of EthTSyn event message reception

3.3 Initialization

The EthTSyn is initialized by calling the `EthTSyn_InitMemory()` and `EthTSyn_Init()` services without parameter. The `EthTSyn_InitMemory()` function has to be called before `EthTSyn_Init()` to initialize used memory of the EthTSyn module.

The EthTSyn port configuration is pre-defined by Configurator Pro configuration process.

Note
Due to the fact that EthTSyn is using the EthIf and Eth BSW module they have to be initialized before executing the EthTSyn initialization process.

3.4 States

The EthTSyn is operational after initialization. The EthTSyn transmits PdelayRequest messages periodically to determine the path delay to a connected EthTSyn device. If no PdelayResponse is received, no further actions for synchronization are taken. A list of implemented state machines can be found in Table 3-7. The detailed description of each state machine is presented in IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks.

3.5 Main Functions

The EthTSyn has a `EthTSyn_MainFunction()` that handles cyclic tasks like timeout handling and processing of state machines needed for EthTSyn operation. Table 3-7 shows the state machines that are processed in an `EthTSyn_MainFunction()` cycle.

State machine	Task
`EthTSyn_ProcessSmSyncReceive()`	Processing of received Sync and FollowUp messages in EthTSyn port slave role.
`EthTSyn_ProcessSmPdelayReq()`	Periodic transmission of PdelayRequest messages and processing of received PdelayResponse and PdelayResponseFollowUp messages
`EthTSyn_ProcessSmPdelayResp()`	Processing of received PdelayRequest messages and transmission of PdelayResponse and PdelayResponseFollowUp messages.
`EthTSyn_ProcessSmSyncSend()`	Periodic transmission of Sync and FollowUp in port master role.
`EthTSyn_ProcessSmAnnounceReceive()`	Processing of received Announce messages in EthTSyn port slave role.
`EthTSyn_ProcessSmSiteSyncSync()`	Processing of Sync and FollowUp received on the slave port and to be forwarded on each master port.

Table 3-7 List of processed state machines in EthTSyn_MainFunction()

3.6 Error Handling

3.6.1 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 (i.e. pre-compile parameter `ETHTSYN_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 EthTSyn ID is 164.

The reported service IDs identify the services which are described in chapters 5.2 and 5.4. The following table presents the service IDs and the related services:

Service ID	Service
0x01	`EthTSyn_Init()`
0x02	`EthTSyn_GetVersionInfo()`
0x05	`EthTSyn_SetTransmissionMode()`
0x06	`EthTSyn_RxIndication()`
0x07	`EthTSyn_TxConfirmation()`

Service ID	Service
0x08	EthTSyn_TrcvLinkStateChg()
0x09	EthTSyn_MainFunction()
0x0A	EthTSyn_SwitchMgmtInfoIndication()
0x0B	EthTSyn_SwitchIngressTimeStampIndication()
0x0C	EthTSyn_SwitchEgressTimeStampIndication()
0x50	EthTSyn_TimestampMinusTimestamp()
0x60	EthTSyn_QualifyAnnounceMsg()
0x61	EthTSyn_Transmit()
0x62	EthTSyn_ProcessSmSyncSend()
0x63	EthTSyn_ProcessSmSiteSyncSync()
0x64	EthTSyn_ProcessReceivedSyncMsg()
0x70	EthTSyn_SwtMgmt_HandleMessageReception()
0x71	EthTSyn_SwtMgmt_ProcessMsgBuffer()

Table 3-8 Service IDs

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

Error Code		Description
0x01	ETHTSYN_E_TMCONFLICT	Time Master Conflict
0x02	ETHTSYN_E_TSCONFLICT	Time Slave Conflict
0x20	ETHTSYN_E_NOT_INITIALIZED	EthTSyn API was called without former initialization of the EthTSyn module
0x21	ETHTSYN_E_INIT_FAILED	EthTSyn initialization failed
0x22	ETHTSYN_E_CTRL_IDX	API called with invalid controller index
0x23	ETHTSYN_E_PARAM_POINTER	API called with invalid Pointer
0x24	ETHTSYN_E_PARAM	API called with invalid parameter
0x30	ETHTSYN_E_TX_FAILED	Transmission failed
0x31	ETHTSYN_E_GET_INGRESS_TIMESTAMP_FAILED	Unable to retrieve ingress Timestamp
0x32	ETHTSYN_E_PDELAY_REQ_MSG_DROPPED	Dropped received Pdelay Req message
0x33	ETHTSYN_E_PDELAY_RESP_MSG_DROPPED	Dropped received Pdelay Resp message
0x34	ETHTSYN_E_PDELAY_RESP_FUP_MSG_DROPPED	Dropped received Pdelay Resp FollowUp message
0x35	ETHTSYN_E_SYNC_MSG_DROPPED	Dropped received Sync message
0x36	ETHTSYN_E_FOLLOW_UP_MSG_DROPPED	Dropped received FollowUp

Error Code		Description
		message
0x37	ETHTSYN_E_ANNOUNCE_MSG_DROPPED	Dropped received Announce message
0x38	ETHTSYN_E_INV_MSG_LENGTH	Received message with an invalid Message Length (either in Header or total frame length)
0x39	ETHTSYN_E_INV_PROTOCOL_VERSION	Received message with an invalid Protocol Version
0x3A	ETHTSYN_E_INV_MSG_TYPE	Received message with an invalid Message Type
0x3B	ETHTSYN_E_INV_FRAME_TYPE	Received message with an invalid Frame Type
0x3C	ETHTSYN_E_INV_DOMAIN_NUMBER	Received message with an invalid Domain Number
0x3D	ETHTSYN_E_TRCV_DOWN	EthTSyn_RxIndication() called for inactive controller
0x3E	ETHTSYN_E_SRC_PORT_IDENT_CHECK_FAILED	Source Port Identity check failed
0x3F	ETHTSYN_E_ANNOUNCE_CHECK_FAILED	Announce check failed
0x40	ETHTSYN_SWT_MGMT_E_MSG_BUFFER_OVERFLOW	Received message while there is no free message buffer available
0x41	ETHTSYN_SWT_MGMT_E_MSG_BUFFER_PAYLOAD_OVERFLOW	Payload too big for the message buffer
0x42	ETHTSYN_SWT_MGMT_E_NO_MSG_AVAILABLE	EthTSyn_SwitchInfoIndication() called with no buffered message available
0xFF	ETHTSYN_E_INTERNAL_ERROR	Internal error occurred (usually this should not happen)

Table 3-9 Errors reported to DET

3.6.2 Production Code Error Reporting

No production error code reporting to the DEM (see [3]) is supported.

Error Code	Description
none	

Table 3-10 Errors reported to DEM

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
EthTSyn.c	Implementation of the EthTSyn core functionality
EthTSyn.h	API declaration
EthTSyn_Cbk.h	API call-back declaration
EthTSyn_CfgAccess_Int.h	Internal macro and function definition to access the configuration data
EthTSyn_Crc_Int.h	Internal macro and function definition to handle Crc calculation and validation
EthTSyn_Int.h	Internal (private) API declaration
EthTSyn_Master_Int.c	Implementation of the EthTSyn Master functionality
EthTSyn_Master_Int.h	Internal API declarations with respect to the Master functionality
EthTSyn_Pdelay_Int.c	Implementation of the EthTSyn Pdelay functionality
EthTSyn_Pdelay_Int.h	Internal API declarations with respect to the Pdelay functionality
EthTSyn_Slave_Int.c	Implementation of the EthTSyn Slave functionality
EthTSyn_Slave_Int.h	Internal API declarations with respect to the Slave functionality
EthTSyn_SwtMgmt.c	Implementation of the EthTSyn Switch Management functionality
EthTSyn_SwtMgmt_Cbk.h	Switch Management API call-back declaration
EthTSyn_SwtMgmt_Int.h	Switch Management internal (private) API declaration
EthTSyn_Types.h	Internal type definitions for EthTSyn modules
EthTSyn_Util_Int.c	Implementation of utilities used by different sub-modules of the EthTSyn
EthTSyn_Util_Int.h	Internal API declarations of utilities used by different sub-modules of the EthTSyn

Table 4-1 Static files

Do not edit manually
Static source code files must not be edited manually!

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator Pro.

File Name	Description
`EthTSyn_Cfg.h`	Generated header file for pre-compile time configuration data
`EthTSyn_Lcfg.h`	Generated header for pre-compile and link time configuration data
`EthTSyn_Lcfg.c`	Generated pre-compile and link time configuration data

Table 4-2 Generated files

Do not edit manually
Generated source code files must not be edited manually but can be influenced by changing the configuration elements in the configuration tool!

4.2 Critical Sections

To ensure data consistency and a correct function of the EthTSyn module the exclusive area ETHTSYN_EXCLUSIVE_AREA_0 has to be provided during the integration.

Considering the timing behavior of your system (e.g. depending on the CPU load of your system, priorities and interruptibility of interrupts and OS tasks and their jitter and delay times) the integrator has to choose and configure a critical section solution in such way that it is ensured that the API functions do not interrupt each other.

It is recommended to use the functions SuspendAllInterrupts() and ResumeAllInterrupts() for ETHTSYN_EXCLUSIVE_AREA_0 to ensure data consistency.

The following data operations are locked via critical section:

- Protocol state machine handling

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

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

Type Name	C-Type	Description	Value Range
EthTSyn_TransmissionModeType	uint8	Transmission mode	ETHTSYN_TX_ON, ETHTSYN_TX_OF

Table 5-1 Type definitions

5.2 Services provided by EthTSyn

5.2.1 EthTSyn_GetVersionInfo

Prototype	
void **EthTSyn_GetVersionInfo** (ETHTSYN_P2VAR(Std_VersionInfoType) VersionInfoPtr)	
Parameter	
VersionInfoPtr [out]	Pointer to a memory location where the EthTSyn version information shall be stored.
Return Code	
void	none
Functional Description	
Return the BCD-coded version information of the EthTSyn module.	
Particularities and Limitations	
none	
Pre-Conditions	
> Availability: This function is only available if EthTSynVersionInfoApi is enabled.	
Call Context	
This function can be called in any context.	

Table 5-2 EthTSyn_GetVersionInfo

5.2.2 EthTSyn_Init

Prototype
void **EthTSyn_Init** (EthTSyn_ConfigType* CfgPtr)

Parameter	
CfgPtr [in]	Pointer to post-build configuration or null pointer
Return Code	
void	none
Functional Description	
Initialization of the EthTSyn module.	
Particularities and Limitations	
This function must be called before using the module	
Pre-Conditions	
The function EthTSyn_InitMemory() must be called first	
Call Context	
This function can be called in any context.	

Table 5-3 EthTSyn_Init

5.2.3 EthTSyn_InitMemory

Prototype	
`void EthTSyn_InitMemory (void)`	
Parameter	
none	
Return Code	
void	none
Functional Description	
Memory initialization of the EthTSyn module.	
Particularities and Limitations	
This function must be called before using the module	
Pre-Conditions	
Call Context	
This function can be called in any context.	

Table 5-4 EthTSyn_InitMemory

5.2.4 EthTSyn_MainFunction

Prototype	
`void EthTSyn_MainFunction (void)`	
Parameter	
none	

Return Code	
void	none
Functional Description	
Processing of cyclic tasks of the EthTSyn module.	
Particularities and Limitations	
none	
Pre-Conditions	
The function EthTSyn_Init() must be called first	
Call Context	
This function can be called in any context.	

Table 5-5 EthTSyn_MainFunction

5.2.5 EthTSyn_SetTransmissionMode

Prototype	
`void EthTSyn_SetTransmissionMode (uint8 CtrlIdx, EthTSyn_TransmissionModeType Mode)`	
Parameter	
CtrlIdx [in]	Index of the Ethernet controller
Mode [in]	ETHTSYN_TX_OFF ETHTSYN_TX_ON
Return Code	
void	none
Functional Description	
This API is used to turn on and off the TX capabilities of the EthTSyn.	
Particularities and Limitations	
none	
Pre-Conditions	
The function EthTSyn_Init() must be called first	
Call Context	
This function can be called in any context.	

Table 5-6 EthTSyn_SetTransmissionMode

5.3 Services used by EthTSyn

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

Component	API
Crc	Crc_CalculateCRC8H2F()
Det	Det_ReportError()
Det	Det_ReportRuntimeError()
EthIf	EthIf_EnableEgressTimestamp()
EthIf	EthIf_GetCurrentTime()
EthIf	EthIf_GetEgressTimestamp()
EthIf	EthIf_GetIngressTimestamp()
EthIf	EthIf_ProvideTxBuffer()
EthIf	EthIf_SetSwitchMgmtInfo()
EthIf	EthIf_SwitchEnableEgressTimeStamp()
EthIf	EthIf_Transmit()
StbM	StbM_BusSetGlobalTime()
StbM	StbM_GetCurrentTime()
StbM	StbM_GetCurrentTimeDiff()
StbM	StbM_GetCurrentTimeRaw()
StbM	StbM_GetOffset()
StbM	StbM_GetTimeBaseStatus()
StbM	StbM_GetTimeBaseUpdateCounter()

Table 5-7 Services used by the EthTSyn

5.4 Callback Functions

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

5.4.1 EthTSyn_RxIndication

Prototype	
void **EthTSyn_RxIndication** (uint8 CtrlIdx, Eth_FrameType FrameType, boolean IsBroadcast, ETHTSYN_P2CONST(uint8) PhysAddrPtr, ETHTSYN_P2VAR(uint8) DataPtr, uint16 LenByte)	
Parameter	
CtrlIdx [in]	Index of the Ethernet controller
FrameType [in]	Frame type of received Ethernet frame
IsBroadcast [in]	Parameter to indicate a broadcast frame
PhysAddrPtr [in]	Pointer to Physical source address (MAC address in network byte order) of received Ethernet frame

DataPtr [in]	Pointer to payload of the received Ethernet frame (i.e. Ethernet header is not provided)
LenByte [in]	Length of received data
Return Code	
void	none
Functional Description	
By this API service the EthTSyn gets an indication and the data of a received frame.	
Particularities and Limitations	
none	
Pre-Conditions	
The function EthTSyn_Init() must be called first	
Call Context	
This function can be called in any context.	

Table 5-8 EthTSyn_RxIndication

5.4.2 EthTSyn_TxConfirmation

Prototype	
`void EthTSyn_TxConfirmation (uint8 CtrlIdx, uint8 BufIdx)`	
Parameter	
CtrlIdx [in]	Index of the Ethernet controller within the context of the Ethernet Interface
BufIdx [in]	Index of the buffer resource
Return Code	
void	void
Functional Description	
Confirms the transmission of an Ethernet frame. This callback function is called by lower layer (EthIf) if a message has been transmitted by the hardware.	
Particularities and Limitations	
Pre-Conditions	
The function EthTSyn_Init() must be called first	
Call Context	
This function can be called in interrupt or task context.	

Table 5-9 EthTSyn_TxConfirmation

5.4.3 EthTSyn_TrcvLinkStateChg

Prototype	
`void EthTSyn_TrcvLinkStateChg (uint8 CtrlIdx, EthTrcv_LinkStateType TrcvLinkState)`	
Parameter	
CtrlIdx [in]	Index of the controller that changed its state
TrcvLinkState [in]	New link state of the transceiver: ETHTRCV_LINK_STATE_DOWN ETHTRCV_LINK_STATE_ACTIVE
Return Code	
void	none
Functional Description	
Allows resetting state machine in case of unexpected Link loss to avoid inconsistent Sync and Follow_Up sequences	
Particularities and Limitations	
none	
Pre-Conditions	
The function EthTSyn_Init() must be called first	
Call Context	
This function can be called in task context.	

Table 5-10 EthTSyn_TrcvLinkStateChg

5.4.4 EthTSyn_SwitchMgmtInfoIndication

Prototype	
`void EthTSyn_SwitchMgmtInfoIndication (uint8 CtrlIdx, ETHTSYN_P2CONST(uint8) DataPtr, ETHTSYN_P2CONST(EthSwt_MgmtInfoType) MgmtInfoPtr)`	
Parameter	
CtrlIdx [in]	Index of the Ethernet controller within the context of the Ethernet Interface
DataPtr [in]	Pointer to the Rx Buffer (Payload portion) to map the Info indication to the Rx frame
MgmtInfoPtr [in]	Management information
Return code	
void	none
Functional Description	
Ingress Switch management info indication redirected call to upper layers who registered for the call.	
Particularities and Limitations	

Call context
> This function can be called in task context.

Table 5-11 EthTSyn_SwitchMgmtInfoIndication

5.4.5 EthTSyn_SwitchEgressTimeStampIndication

Prototype	
`void EthTSyn_SwitchEgressTimeStampIndication (uint8 CtrlIdx, ETHTSYN_P2CONST(uint8) DataPtr, ETHTSYN_P2CONST(EthSwt_MgmtInfoType) MgmtInfoPtr, ETHTSYN_P2CONST(Eth_TimeStampType) timeStampPtr)`	
Parameter	
CtrlIdx [in]	Index of the Ethernet controller within the context of the Ethernet Interface
DataPtr [in]	Buffer Pointer to map the Timestamp indication to the received/transmitted frame
MgmtInfoPtr [in]	Management information
timeStampPtr [in]	Current timestamp
Return code	
void	none
Functional Description	
Delivers to upper layers an egress timestamp value out of the Switch where MgmtInfo refers. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	
Particularities and Limitations	
Call context	
> This function can be called in task context.	

Table 5-12 EthTSyn_SwitchEgressTimeStampIndication

5.4.6 EthTSyn_SwitchIngressTimeStampIndication

Prototype	
`void EthTSyn_SwitchIngressTimeStampIndication (uint8 CtrlIdx, ETHTSYN_P2CONST(uint8) DataPtr, ETHTSYN_P2CONST(EthSwt_MgmtInfoType) MgmtInfoPtr, ETHTSYN_P2CONST(Eth_TimeStampType) timeStampPtr)`	
Parameter	
CtrlIdx [in]	Index of the Ethernet controller within the context of the Ethernet Interface
DataPtr [in]	Buffer Pointer to map the Timestamp indication to the received/transmitted frame
MgmtInfoPtr [in]	Management information
timeStampPtr [in]	Current timestamp

Return code	
void	none
Functional Description	
Delivers to upper layers an ingress timestamp value out of the Switch where MgmtInfo refers. If the HW resolution is lower than the Eth_TimeStampType resolution resp. range, than the remaining bits will be filled with 0.	
Particularities and Limitations	
Call context	
> This function can be called in task context.	

Table 5-13 EthTSyn_SwitchIngressTimeStampIndication

6 Configuration

The EthTSyn attributes can be configured and generated with the tool DaVinci Configurator Pro.

6.1 Configuration Variants

The EthTSyn supports the configuration variants

> VARIANT-PRE-COMPILE

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

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
Configurator Pro	DaVinci Configurator Pro 5 generation tool for MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
Eth	Ethernet Driver
EthIf	Ethernet Interface
EthSwt	Ethernet Switch
EthTSyn	Time Synchronization over Ethernet
gPTP	Generalized Precision Time Protocol
HW	Hardware
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
MII	Media Independent Interface
Pdelay	Path Delay
PdelayReq	Pdelay request
PdelayResp	Pdelay response
PdelayRespFollowUp	Pdelay response follow up
StbM	Synchronized Time-Base Manager
SW	Software
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