MICROSAR Classic EthTSyn

Technical Reference

Global Time Synchronization over Ethernet

Version 17.0.0

Authors	vislje, vissem, visfaz
Status	Released

Document Information

History

Author	Date	Version	Remarks
vislje	2014-09-23	1.0.0	Creation of document
vislje	2014-11-11	1.0.1	Add new API
vissem	2015-01-12	1.0.2	ESCAN00080452 Modification of Chapter “Configuration”
vissem	2015-02-26	1.1.0	Added Configuration for SW-Timestamping
vissem	2015-07-10	1.2.0	ESCAN00083936
vissem	2015-10-23	2.0.0	ESCAN00085377: FEAT-1529: Support Ethernet Switches for Ethernet Time Sync
vissem	2016-03-15	2.1.0	ESCAN00085303 Added Chapters: > Boundary Clock > AlwaysAsCapable > Announce > Source Port Identity Check > Correction Action
vissem	2016-05-13	2.2.0	ESCAN00089686 Added Chapter: > Flexible Pdelay configuration
vissem	2016-11-25	3.0.0	FEATC-248: FEAT-1998: Support of HW Time Stamping for Switch for EthTSyn (SysService_AsrTSynEth)
vissem	2017-03-07	4.0.0	FEATC-383: FEAT-2279: Time Synchronization acc. AR 4.3 for EthTSyn
vissem	2017-04-12	4.0.1	Fixed review findings
vissem	2017-07-05	5.0.0	STORYC-1213, STORY-128
vissem	2018-02-22	6.0.0	Formal rework
vissem	2018-08-14	6.0.1	Fixed review findings
vissem	2018-11-16	7.0.0	STORYC-6413: [AVB] Implementation of 802.1Qbv extensions for EthTSyn
vissem	2019-01-31	7.0.1	Added EthIf restriction for STORYC-6413
vissem	2019-01-31	8.0.0	Added Switch management limitations
vissem	2019-03-22	9.0.0	Adapted SyncSentCbk and FollowUpSentCbk
vissem	2019-07-25	10.0.0	Switch management API acc. to AUTOSAR 4.4
vissem	2019-11-22	11.0.0	ESCAN00104993

vissem	2020-04-03	12.0.0	StbM interaction acc. to AUTOSAR 4.4
vissem, visfaz	2020-08-04	12.1.0	Time validation, Optimization of switch time synchronization algorithm, Bridge modes Transparent/Boundary clock, Switch and Host as one time-aware system
vissem	2020-10-02	13.0.0	Updated documentation after code refactoring
visfaz	2020-12-14	13.1.0	Support of debounce time
vissem	2021-01-21	13.2.0	Extended CRC support for bridges
alefarth	2022-03-15		Product name updated to MICROSAR Classic.
visfaz	2022-09-05	15.0.0	Support for FollowUp correction field threshold
vissem	2022-10-10		Added new bridge mode: Asymmetric Transparent clock
			Support of rate correction for sync reception delay
			Support of time domain specific destination MAC address
vissem	2022-01-30	15.1.0	Added configuration hints for Asymmetric Transparent clock
			Support API to retrieve Port Parameter Statistics
vissem	2023-03-02	15.2.0	Use global time for Pdelay response
vissem	2023-07-13	16.0.0	Remove special handling (SwtSyncFrame) to get the host port ingress timestamp
			Describe limitation for time validation with enabled switch management
vissem	2023-11-30	16.1.0	Updated static source file list. Added new Service and Error IDs
vissem	2024-01-15	17.0.0	Enhance TLV processing

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	Specification of Time Synchronization over Ethernet	R4.4.0
[2]	AUTOSAR	Specification of Time Synchronization over Ethernet	R19-11
[3]	AUTOSAR	Specification of Time Synchronization over Ethernet	R21-11
[4]	AUTOSAR	Specification of Time Synchronization over Ethernet	R22-11
[5]	AUTOSAR	Specification of Default Error Tracer	R4.3.0
[6]	AUTOSAR	Specification of Diagnostic Event Manager	R4.3.0
[7]	AUTOSAR	Specification of Synchronized Time-Base Manager	R4.4.0
[8]	AUTOSAR	Specification of Synchronized Time-Base Manager	R19-11
[9]	IEEE	IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks	2011
[10]	IEEE	IEEE 1588-2008: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems	2008
[11]	Vector	TechnicalReference EthIf.pdf	see delivery
[12]	Vector	TechnicalReference StbM.pdf	see delivery
[13]	Vector	TechnicalReference_EthSwt_<Driver>.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 [7]), the EthIf (see [11]) 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 Introduction

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

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])

The EthTSyn module provides the functionality of time synchronization defined by the gPTP (generalized Precision Time Protocol) IEEE 802.1AS-2011 (see [9]). 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 [10]) 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

1.1 Architecture Overview

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

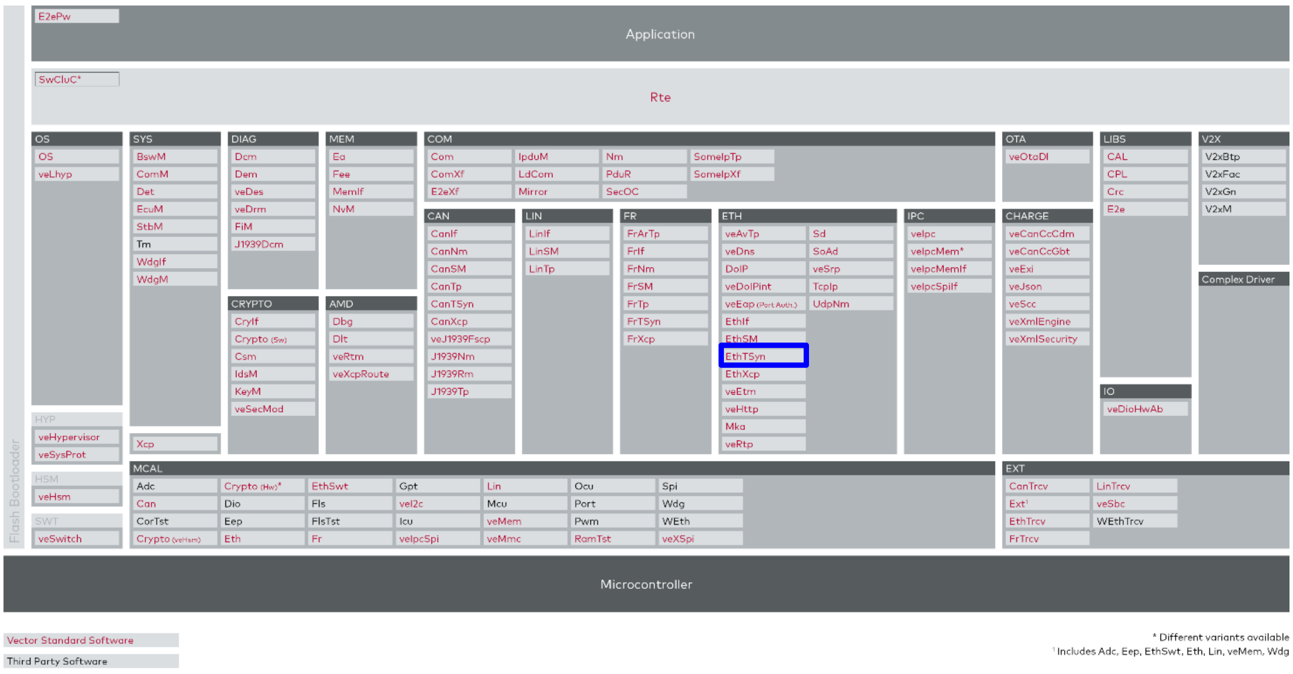


Figure 1-1 AUTOSAR 4.3 Architecture Overview

Figure 1-2 shows the interfaces to adjacent modules of the EthTSyn. These interfaces are described in chapter 4.

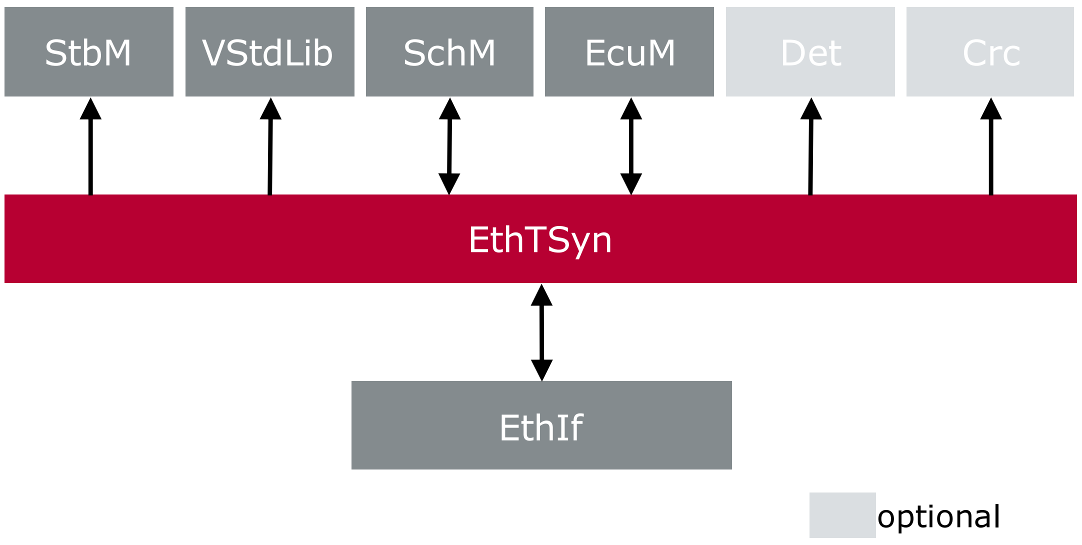


Figure 1-2 Interface to adjacent modules of EthTSyn

2 Functional Description

2.1 Features

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

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

> Table 2-1 Supported AUTOSAR standard conform features

> Table 2-2 Not supported AUTOSAR standard conform features

> Table 2-3 Deviations of supported AUTOSAR standard conform features

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

> Table 2-5 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
VLAN Support
Handling of different Virtual Local Time sources
Debounce Time
Pdelay Protocol for Latency Calculation
Sync and Follow_Up acc. to IEEE 802.1As
Sync and Follow_Up acc. to AUTOSAR
Acting as Time Master
Acting as Time Slave
Time measurement with Switches
Error Classification

Table 2-1 Supported AUTOSAR standard conform features

2.1.1 Deviations

2.1.1.1 AUTOSAR 4.4.0 Deviations

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

Category	Description
API	EthTSyn_TrcvLinkStateChg: No error is reported to DET when called with invalid parameter ‘`CtrlIdx`’
Config	EthTSynGlobalTimeSecureTmacLength
Config	EthTSynGlobalTimeMinMsgGap
Config	EthTSynTxTmacCalculated
Config	EthTSynRxTmacValidated

Functional	Re-initialization is not supported

Table 2-2 Not supported AUTOSAR standard conform features

The following deviations of [1] Specification of Time Synchronization over Ethernet apply to the EthTSyn.

Category	Description
API	EthTSyn_RxIndication: The parameters ‘`PhysAddrPtr`’ and ‘`DataPtr`’ are of type ‘`uint8*`’ rather than ‘`const uint8*`’
Config	Master parameters for AUTOSAR TLV configuration (EthTSynTLVFollowUpOFSSubTLV, EthTSynTLVFollowUpStatusSubTLV, EthTSynTLVFollowUpTimeSubTLV, EthTSynTLVFollowUpUserDataSubTLV) are named according to AUTOSAR R22-11 (see [4]) (EthTSynTxSubTLVOFS, EthTSynTxSubTLVStatus, EthTSynTxSubTLVTime, EthTSynTxSubTLVUserData)
Functional	A time master always transmits the UserData TLV when enabled (in accordance with AUTOSAR R22-11)
Functional	Acting as Time Slave: The sync reception delay is calculated according to AUTOSAR R21-11 (see [3])
Functional	Debounce Time: The debounce time is linked to a physical port
Functional	Initialization: Pdelay is initialized with the preconfigured value rather than zero

Table 2-3 Deviations of supported AUTOSAR standard conform features

2.1.1.2 IEEE 802.1AS-2011 Deviations

The following deviations of [9] “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
10.3 Best master clock selection and announce interval setting state machines	Neither the Best Master Clock Algorithm (BMCA) nor the announce interval setting state machine are supported.
10.4 Message attributes	Signaling messages are not supported
10.5.2.2 Header field specifications	All Announce message specific bits of the flags1 field are neither set on transmission nor evaluated on reception. The bits are: > leap61

Section in Document	Limitation
	> leap59 > currentUtcOffsetValid > timeTraceable > frequencyTraceable
10.5.2.2 Header field specifications	The ptpTimescale bit of the flag1 field is set for all messages, if message compliance is enabled.
11.4 Message Format	Header Correction Field: > Only supported in FollowUp messages > Fractional nanoseconds are not supported and ignored on reception > Value is interpreted as unsigned integer 64 (in accordance with [1])
11.4 Message Format	Following fields of the FollowUp information TLV are always set to "0" on transmission and never evaluated on reception: > cumulativeScaledRateOffset > gmTimeBaseIndicator > lastGmPhaseChange > scaledLastGmFreqChange
11.2 State machines	Cyclic transmission of Announce Messages is optional 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.
14.7.9 syncReceiptTimeoutCount	This counter is not support and therefore always returned as '0'
14.7.10 announceReceiptTimeoutCount	This counter is not support and therefore always returned as '0'

Table 2-4 Deviations to IEEE 802.1AS-2011: Timing and Synchronization for Time-Sensitive Applications in Bridged Local Networks

2.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard.

Features Provided Beyond the AUTOSAR Standard
Announce reception support on a slave port
Announce transmission for master ports
AsCapable calculation based on Pdelay protocol
AUTOSAR TLV handling according to R22-11 (see also [4])
Backup master functionality for bridges
Bridge operation mode: Asymmetric transparent clock (EthTSyn only handles forwarding of the FollowUp message)
Bridge operation mode: Boundary Clock

Bridge operation mode: Transparent Clock
Correction field threshold for a FollowUp received on a slave port
Optional callback notification after successful transmission of a Sync/Follow_Up message
Port Parameter Statistics according to chapter 14.7 of [9]
Source port identity check (static configured master identity) on a slave port
Time domain specific configuration of the destination MAC address
Time synchronization of Ethernet switches
Time validation acc. to [2]
Usage of global times for Pdelay responder (see also 2.1.7.1)
Usage of static Pdelay but still sending cyclic Pdelay_Req
Weighted average calculation for Pdelay value

Table 2-5 Features provided beyond the AUTOSAR standard

2.1.3 Limitations

2.1.3.1 General Limitations

The EthTSyn can only be used in conjunction with:

> MICROSAR Classic EthIf

> MICROSAR Classic StbM

2.1.3.2 Switch management limitations

When switch management mode is used, the following limitations apply.

2.1.3.2.1 Message limitations

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

2.1.3.2.2 AUTOSAR message format limitations

> A Transparent Clock does not add or remove AUTOSAR Sub-Tlvs

2.1.3.2.3 Slave port limitations

> In a bridge configuration, Announce can't be used for EndStation slave ports

2.1.3.2.4 Master port limitations

> Immediate time synchronization is only applicable when acting as time master or as Boundary Clock

2.1.3.2.5 Time validation limitations

> In a bridge configuration, time validation can only be used when the host and the switch are considered as one time-aware system (see also 2.1.4.1)
  > Rational: The user of time validation needs to be able to convert the collected Pdelay timing data into global times. This is only possible, when the Pdelay times are based on the virtual local time of the StbM. In addition, all timestamps used for time validation need to be compared and hence, taken at the same device/point (Pdelay timestamps are always taken from the switch so same needs to be true for Sync messages).

2.1.4 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 must 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 2-1 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 must 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 2-2 provides an overview of the sequence for PTP frames received on an EthSwt port. Figure 2-3 provides an overview of the sequence applying to PTP frames transmitted by the EthTSyn on an EthSwt port.

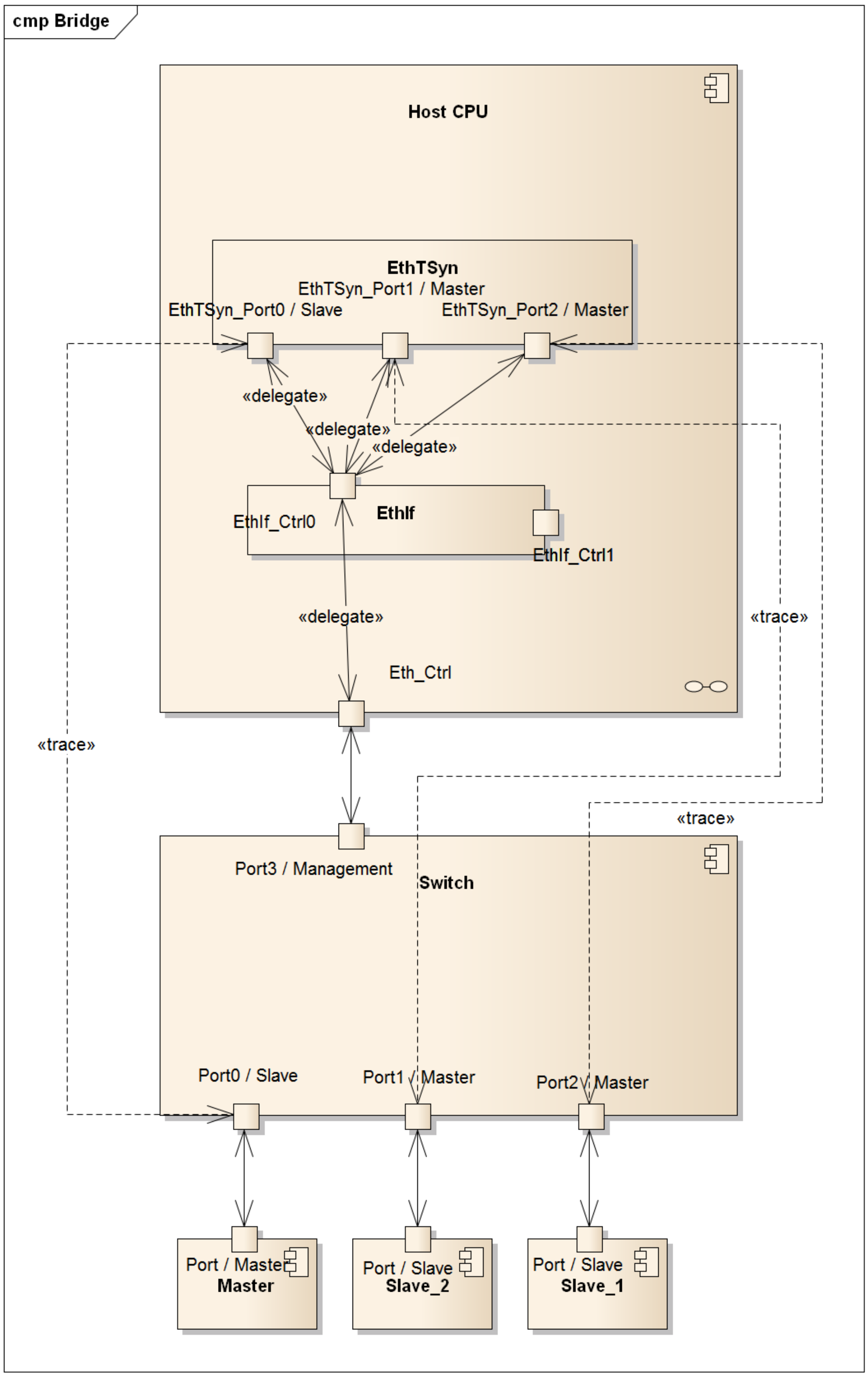


Figure 2-1 Bridge Overview

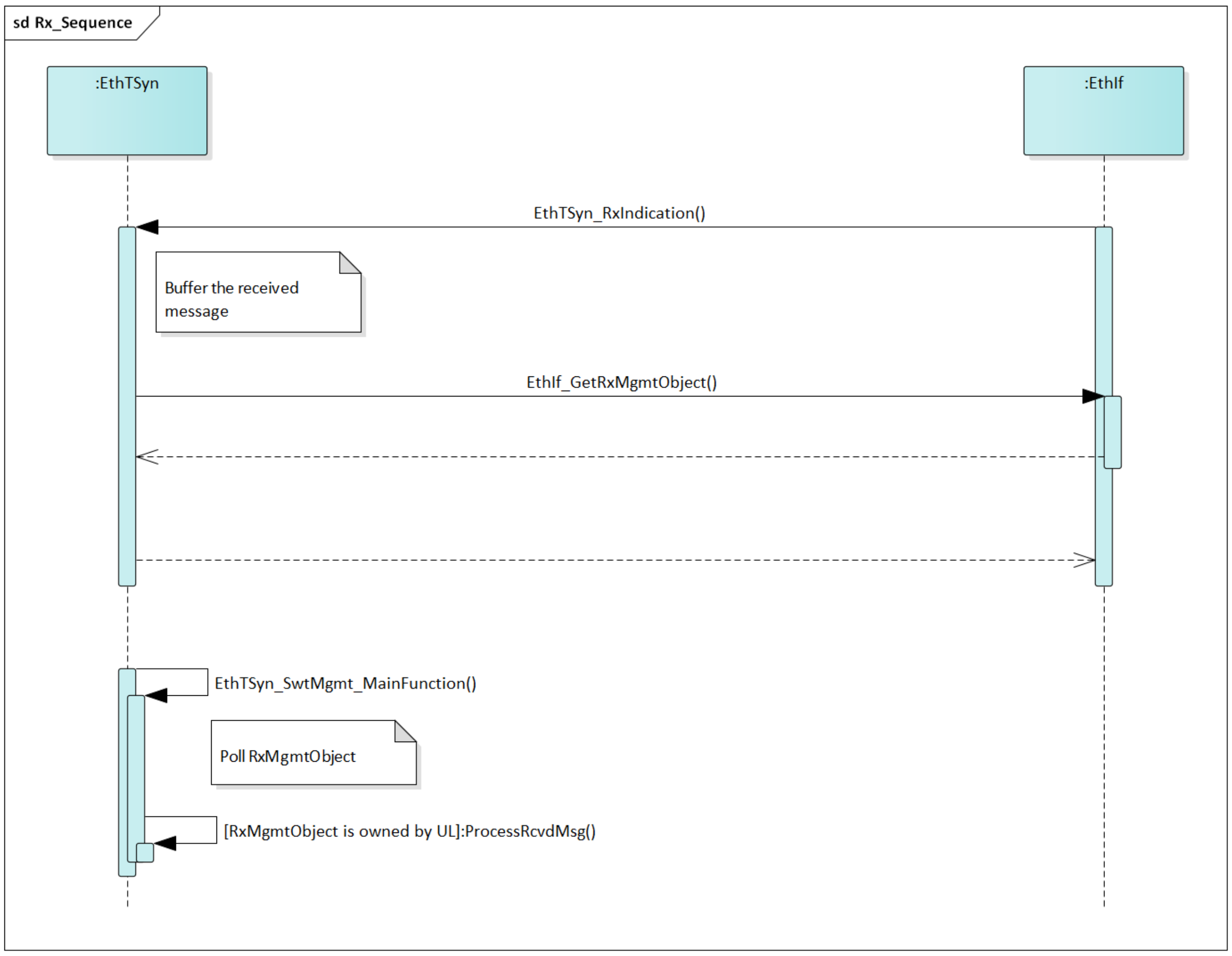

sd Rx_Sequence
:EthTSyn
:EthIf
EthTSyn_RxIndication()
Buffer the received message
EthIf_GetRxMgmtObject()
EthTSyn_SwtMgmt_MainFunction()
Poll RxMgmtObject
[RxMgmtObject is owned by UL]:ProcessRcvdMsg()

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

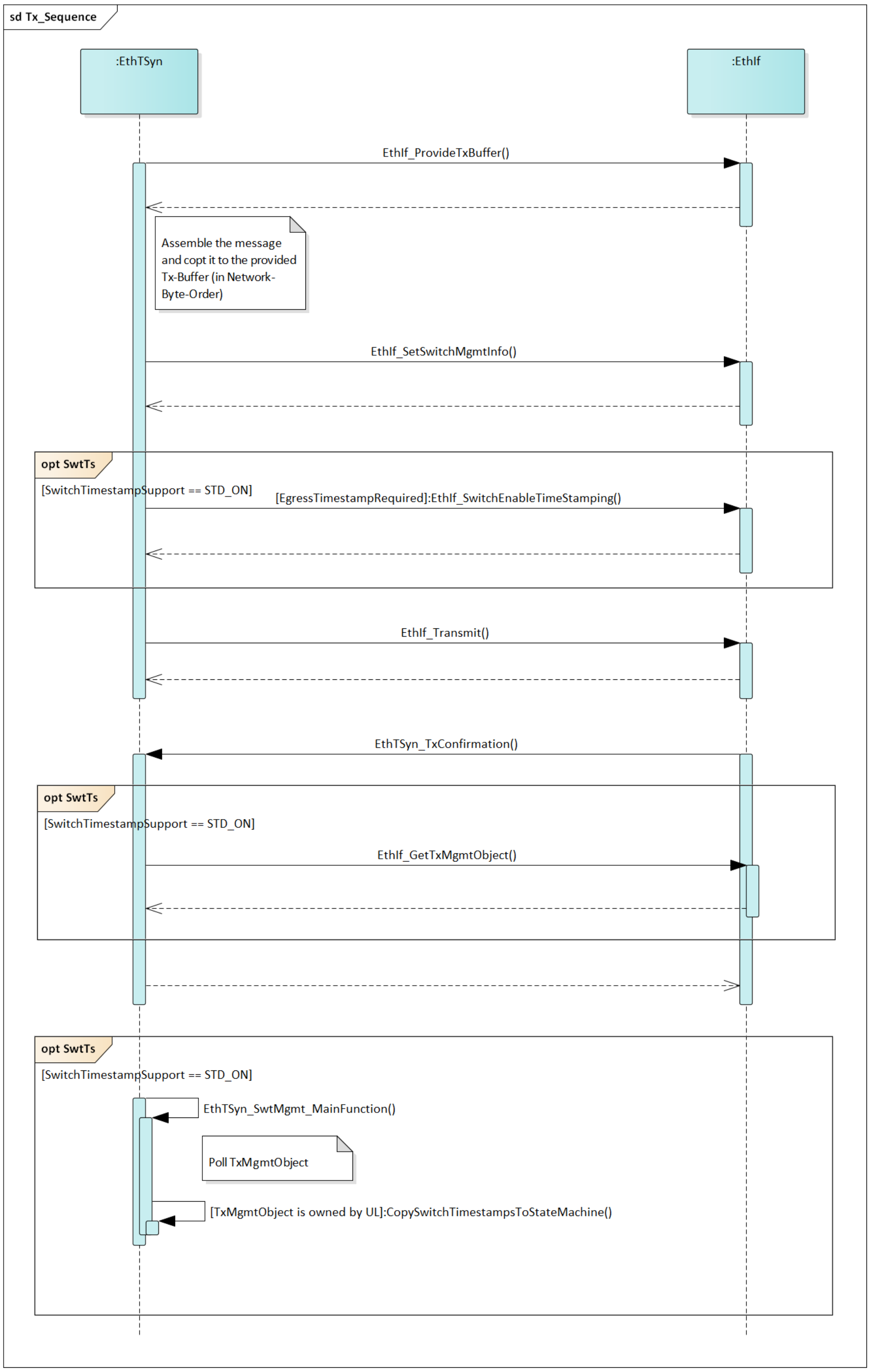

sd Tx_Sequence
:EthTSyn
:EthIf
EthIf_ProvideTxBuffer()
Assemble the message and copt it to the provided Tx-Buffer (in Network-Byte-Order)
EthIf_SetSwitchMgmtInfo()
opt SwtTs
[SwitchTimestampSupport == STD_ON]
[EgressTimestampRequired]:EthIf_SwitchEnableTimeStamping()
EthIf_Transmit()
EthTSyn_TxConfirmation()
opt SwtTs
[SwitchTimestampSupport == STD_ON]
EthIf_GetTxMgmtObject()
opt SwtTs
[SwitchTimestampSupport == STD_ON]
EthTSyn_SwtMgmt_MainFunction()
Poll TxMgmtObject
[TxMgmtObject is owned by UL]:CopySwitchTimestampsToStateMachine()

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

2.1.4.1 Switch and Host as one Time-Aware System

When the Ethernet Switch and the Host controller are considered as one time-aware-system (`EthTSynSwitchMgmtSwitchAndHostAsOneTimeAwareSystem`), the EthTSyn expects that the time of the Ethernet Switch and Host are equal. Therefore, the timestamps provided by the Ethernet Switch can be used as local time.

Caution
The synchronization of the Ethernet Switch and the Host (i.e. Ethernet Controller of the Host) must be ensured. This is not done by the EthTSyn.

2.1.4.2 Time-Aware-Bridge behavior

The behavior of the Time-Aware-Bridge depends on the selected bridge operation mode (`EthTSynSwitchMgmtBridgeMode`). The modes Transparent Clock (2.1.4.3) and Boundary Clock (2.1.4.4.6) are supported. Both modes are described in more details in the corresponding chapters. Table 2-6 provides a quick comparison of the two operation modes. In addition, chapter 2.1.4.6 describes the behavior when only master ports are configured.

Bridge Mode / Functionality	Transparent Clock	Boundary Clock
Sync message reception	Received and forwarded	Received and terminated
FollowUp message reception	Received and forwarded	Received and terminated
Sync/FollowUp message transmission	Triggered by reception of the corresponding message or cyclically in case of a sync timeout	Triggered cyclically when GLOBAL_TIME_BASE_BIT of the corresponding time domain is set
Switch residence time	Added to the FollowUpCorretionField	Added to the FollowUpPreciseOriginTimestamp
Sync-Timeout	Optional. When enabled, a sync receive timeout is observed. Upon occurrence of such a timeout, cyclic transmission of Sync/FollowUp messages is started	-
SourcePortIdentity modification	Optional	Mandatory
Pdelay measurement	Configurable for each switch port individually Pdelay measured on the switch slave port is used for the optional synchronization of the Host and added to the FollowUpCorrectionField of a forwarded FollowUp message	Configurable for each switch port individually Pdelay measured on the switch slave port is used for synchronization of the Host only

Acting as Time Master (only master port configured)	Sync/FollowUp messages are cyclically transmitted when the GLOBAL_TIME_BASE_BIT of the corresponding time domain is set. The residence time of the Switch is added to the FollowUpCorrectionField.	Sync/FollowUp messages are cyclically transmitted when the GLOBAL_TIME_BASE_BIT of the corresponding time domain is set. The residence time of the Switch is added to the FollowUpPreciseOriginTimestamp while the FollowUpCorrectionField is set to zero.

Table 2-6 Comparison of Transparent Clock and Boundary Clock

Note
The GLOBAL_TIME_BASE_BIT is maintained by the StbM. For more details, please refer to [7] Specification of Synchronized Time-Base Manager

2.1.4.3 Switch residence time

The time it takes a Sync message to pass the Ethernet switch is called *switch residence time*. It can either be calculated by using timestamps provided by the Ethernet switch or configured statically. There are three different situations for the residence time which are described in the following with the help of Figure 2-4:

1. Management CPU is configured as time master. A Sync message is e.g., transmitted on Port 2 of the switch. The residence time of the switch is calculated as **`T2 – T1`** or the static value configured via `EthTSynGlobalTimeUplinkToTxSwitchResidenceTime` is used (when the value is > 0)
2. Management CPU is configured as time slave. A Sync message is e.g., received on Port 2 of the switch. The residence time of the switch is calculated as **`T1 – T2`** or the static value configured via `EthTSynGlobalTimeRxToUplinkSwitchResidenceTime` is used (when the value is > 0)
3. Management CPU is configured as time slave without a PTP role, but bridge is configured as transparent clock. A Sync message is e.g., received on Port 2 of the switch and forwarded on Port 3 of the switch. The residence time of the switch is calculated as **`T3 – T2`** or the static value `EthTSynGlobalTimeRxToUplinkSwitchResidenceTime` + `EthTSynGlobalTimeUplinkToTxSwitchResidenceTime` is used (when both values are > 0). In case a static value is used here, the SW residence time (i.e., the time required by the Management CPU to handle the message forwarding) is not considered/added by the EthTSyn.

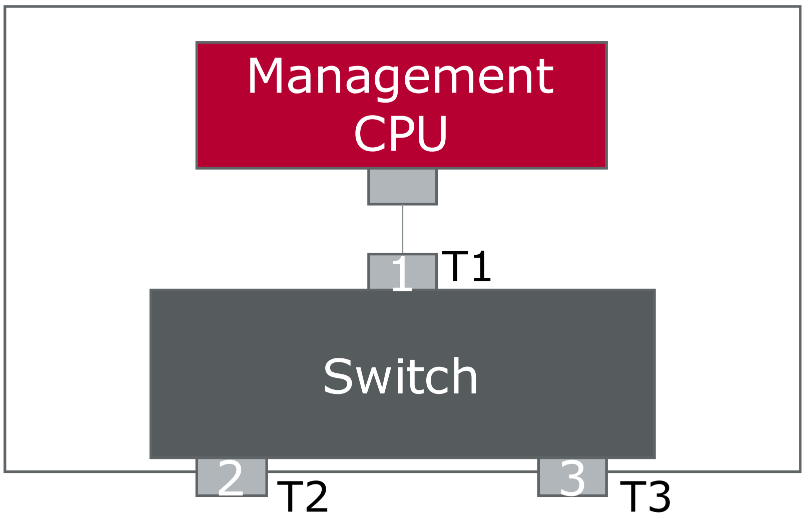


Figure 2-4 Switch timestamps

2.1.4.4 Time-Aware-Bridge (Transparent Clock)

In this section, the behavior of a ‘real’ bridge (i.e. one slave port and at least one master port are configured) as transparent clock is described. For more detailed information about a master only configuration, please refer to 2.1.4.6.

2.1.4.4.1 Sync and FollowUp message reception

When a valid Sync message is received on the switch slave port, it is forwarded on each master port of the bridge without waiting for the corresponding FollowUp message. In addition, the Sync message is forwarded to the end station port, if configured.

When a valid FollowUp message is received on the bridge slave port, it is forwarded on each master port if the corresponding Sync message was already transmitted on the port. In addition, the FollowUp message is forwarded to the end station port, if configured.

2.1.4.4.2 Sync and FollowUp message transmission

The Sync and FollowUp message transmission on a bridge master port is either triggered by the reception of the corresponding message (i.e. message is forwarded, see 2.1.4.4.1) or cyclically in case of a sync timeout (see 2.1.4.4.3).

Upon forwarding of a Sync or FollowUp message, the source port identity might be modified (see 2.1.4.4.4). In addition, the current valid Pdelay of the bridge slave port and the switch residence time of the forwarded Sync message are added to the correction field of the forwarded FollowUp message.

In case of cyclic message transmission due to a sync timeout, the precise origin timestamp carried in the FollowUp message will remain the same as received with the last valid FollowUp message. The correction field of the FollowUp message will be updated such that the current valid Pdelay of the bridge slave port and the time passed between reception and transmission of the corresponding Sync message are included.

2.1.4.4.3 Sync timeout handling

A transparent clock can observe a sync timeout, if the corresponding `EthTSynSwitchMgmtGlobalTimeSyncTimout` of the time domain is configured. In case a sync timeout is detected, the transparent clock will start cyclic Sync and FollowUp message transmission (see 2.1.4.4.2). This cyclic message transmission is terminated with the reception of the next valid Sync message. In case a matching valid FollowUp message is received as well, the regular forwarding is applied again starting with reception of a valid Sync afterwards.

In case another sync timeout is detected before the corresponding FollowUp or the valid next Sync/FollowUp message pair is received, the transparent clock will re-start cyclic Sync and FollowUp message transmission again.

For the cyclic transmission, the last valid received Sync/FollowUp message pair is used as basis. A Sync/FollowUp message pair is considered as valid when it was forwarded by the transparent clock. This means, that the Sequence Id of the message is incremented with each cyclic transmission and starts with the sequence id of the last valid received Sync/FollowUp message pair + 1.

Note

- The sync timeout observation is started after reception of the first valid Sync and FollowUp message pair. In case a valid Sync is received but no corresponding valid FollowUp, no sync timeout is detected.
- The sync timeout can only be used when the switch residence time is calculated and not configured to a static value (see also 2.1.4.3)

2.1.4.4.4 Source port identity modification

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

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

2.1.4.4.5 AUTOSAR TLV and CRC handling (with disabled message compliance)

A transparent clock will neither add nor remove any AUTOSAR FollowUp TLV in a forwarded FollowUp message. Therefore, the configuration of a bridge master port with respect to the AUTOSAR TLVs and CRC does not influence the behavior. However, since the time secured TLV uses information of the FollowUp message for CRC calculation which might be modified by the transparent clock, its CRC is re-calculated upon forwarding of a FollowUp message. For re-calculation of the TimeSecured TLV CRC, the used flags are taken from the received FollowUp message.

The CRC validation on reception of a FollowUp message on a slave port can be configured individually for the switch port and the optional end station port. In case CRC validation is performed on the switch slave port, a FollowUp message containing any AUTOSAR TLV with an invalid CRC will be dropped and is hence neither forwarded on any switch master port nor to the optional end station slave port.

Hint: When the CRC is already validated on the switch port, there is no need to validate it on the end station port as well since the message is forwarded to the end station without any modification.

Note
In case a FollowUp message with TimeSecured TLV and invalid CRC is received and CRC validation of the switch slave port is set to `IGNORE` (i.e. message is received and forwarded anyway), the CRC of the TimeSecured TLV of a forwarded FollowUp message on any switch master port will be invalidated as well.

Note
In case message compliance is enabled, all AUTOSAR TLVs (including the CRC) are ignored on reception and forwarded without any modification.

2.1.4.4.6 Asymmetric Transparent Clock

The asymmetric transparent clock is a special type of the transparent clock. It is called asymmetric because the EthTSyn only handles forwarding of FollowUp messages but not for Sync messages. Forwarding of the Sync messages needs to be done by the Ethernet switch hardware (but details are out of scope of this document). The EthTSyn still needs to receive the Sync message together with its ingress and all related egress timestamps. The timestamp information are required for calculation of the switch residence time and thus updating of the `FollowUpCorrectionField` upon forwarding of the corresponding FollowUp message.

Note
With an asymmetric transparent clock, it is not possible to use the following features:

- Sync timeout (see also 2.1.4.4.3)
- Source port identity modification (see also 2.1.4.4.4)
- Static residence time (see also 2.1.4.3)
- Time validation (see also 2.1.12)

2.1.4.5 Time-Aware-Bridge (Boundary Clock)

In this section, the behavior of a ‘real’ bridge (i.e. one slave port and at least one master port are configured) as boundary clock is described. For more detailed information about a master only configuration, please refer to 2.1.4.6. For a boundary clock, the existence of an end station port is mandatory.

2.1.4.5.1 Sync and FollowUp message reception

When a valid Sync message is received on the switch slave port, it is forwarded to the end station port and terminated.

When a valid FollowUp message is received on the bridge slave port, it is forwarded to the end station port and terminated.

2.1.4.5.2 Sync and FollowUp message transmission

The Sync and FollowUp message transmission on a bridge master port is triggered cyclically when the `GLOBAL_TIME_BASE_BIT` of the corresponding time domain s set. A new precise origin timestamp is generated on the Host (which includes the switch residence time) for each transmitted FollowUp message while the correction field is set to zero.

2.1.4.5.3 Sync timeout handling

A boundary clock always transmits Sync and FollowUp messages cyclically. Therefore, no sync timeout observation is required.

2.1.4.5.4 Source port identity modification

A boundary clock does not forward received Sync and FollowUp messages. Upon generation of Sync and FollowUp messages, the source port identity will always be set according to the bridge master port the message is transmitted on. The configuration option `EthTSynSwitchMgmtKeepSourcePortIdentity` does not influence this behavior.

2.1.4.5.5 AUTOSAR TLV and CRC handling

For a boundary clock, the usage of AUTOSAR FollowUp TLVs together with its type (i.e. secured or not secured) can be configured for each master port individually.

The CRC validation on reception of a FollowUp message on a slave port can be configured individually for the switch port and the end station port.

Hint: When the CRC is already validated on the switch port, there is no need to validate it on the end station port as well since the message is forwarded to the end station without any modification.

Note
In case of enabled message compliance all AUTOSAR TLVs (including the CRC) are ignored on reception.

2.1.4.6 Bridge as Grand Master

In a bridge configuration, where only master ports are configured for a time domain, the bridge is acting as grand master. In this case, Sync and FollowUp messages will be transmitted on each Master-Port in the configured cycle. Depending on the selected bridge mode (`EthTSynSwitchMgmtBridgeMode`), the residence time of the Ethernet Switch is either used as *FollowUpCorrectionField* (Transparent Clock) or added to the *FollowUpPreciseOriginTimestamp* (Boundary Clock).

2.1.4.7 Pdelay

In a Bridge configuration, the Pdelay mechanism can be configured for each Bridge-Port individually. For a detailed description see chapter 2.1.7.

2.1.5 Clock Master Role

In clock master role the EthTSyn cyclically transmits Sync and FollowUp from the Ethernet controller (Eth) that is referenced by the EthTSyn GlobalTimeDomain, after the GLOBAL_TIME_BASE_BIT is set. 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] Specification of Time Synchronization over Ethernet, Chapter 9.2.

If enabled, Announce messages providing information about the local clock properties are transmitted cyclically with the same cycle time as Sync messages. This additional information is used for evaluation of BMCA (Best Master Clock Algorithm) for best master clock determination that is not supported by EthTSyn.

2.1.5.1 Immediate time synchronization

In addition to the regular cyclic time propagation a time master can be configured to propagate its time immediately on changes. This is called immediate time synchronization. An update of the time is detected via the StbM. After an immediate time synchronization event, the time master will wait for the configured cyclic message resume time before transmitting the first regular cyclic Sync and FollowUp messages again.

Note
In case the cyclic message resume time is too short (e.g. 0s), the time master might not transmit the FollowUp message belonging to the immediate Sync because a new Sync cycle is started before transmission of the FollowUp message took place.

2.1.6 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 2.1.7) is performed.

A more detailed description can be found in [1] Specification of Time Synchronization over Ethernet, Chapter 9.3.

2.1.6.1 Announce

According to [9] IEEE 1588-2008: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems 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 2.1.6.2 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.

2.1.6.2 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.

2.1.6.3 FollowUp correction field threshold

A correction field threshold can be optionally configured via the BSWMD parameter `EthTSynFollowUpCorrectionFieldThreshold`. The correction field value of a received FollowUp message is compared to this threshold. In case that the value exceeds the threshold, the FollowUp message is discarded. If the parameter is set to “0”, all FollowUp messages with non-zero correction field are discarded.

Note
Two separate values can be configured if switch management is used: one for the switch slave port and one for the endstation slave port. Since the threshold of the switch port is checked before the message is forwarded to the endstation port, it does not make sense to configure a higher value for the endstation than for the switch port.

2.1.7 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 2-5 shows the sequence diagram of the path delay measurement.

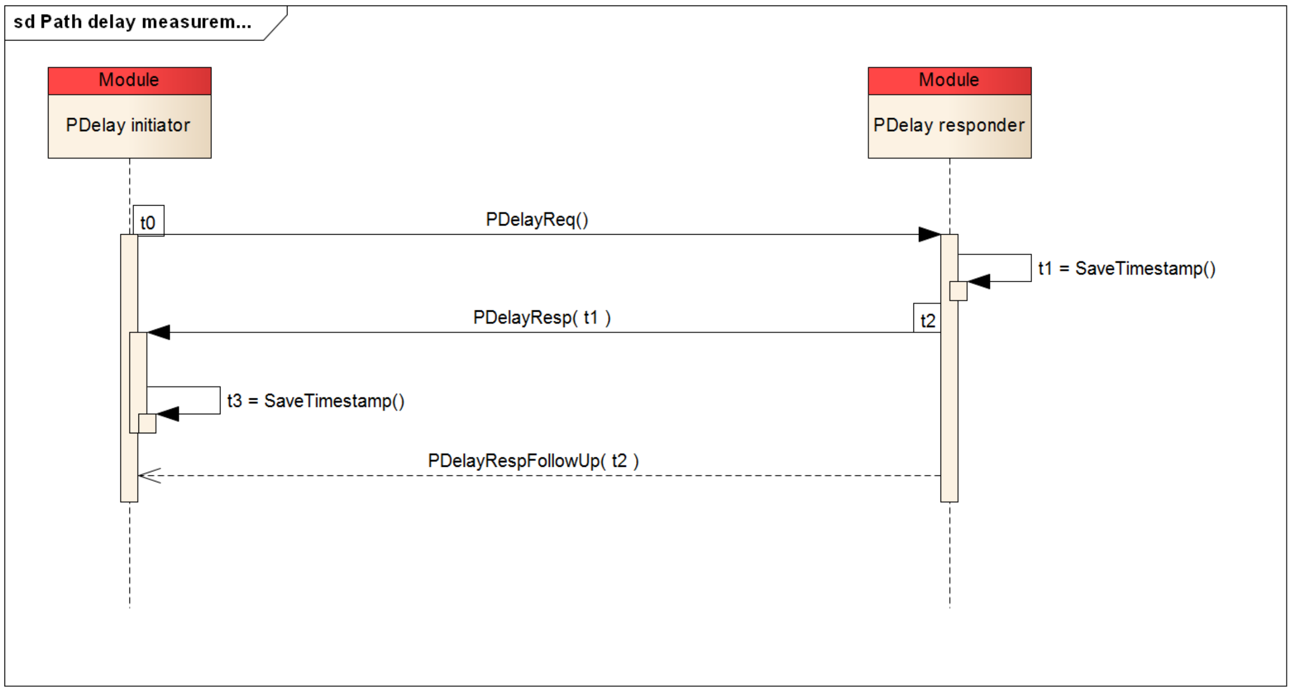


Figure 2-5 Sequence diagram of path delay measurement

2.1.7.1 Flexible Pdelay configuration

The Pdelay configuration options shown in Table 2-7 and Table 2-8 (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. If the Pdelay initiator functionality is also enabled, PdelayReq messages are transmitted and a new Pdelay value is calculated, but it is only used for determination of the AsCapable state and not for time synchronization.
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 2.1.8

Table 2-7 General Pdelay configuration options

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

Note
If the configuration options for both Pdelay Initiator and Use Static Pdelay are disabled, the value for Initial Pdelay is always used. Disabling Pdelay initiator therefore implies that a static Pdelay value is used even if the configuration option Use Static Pdelay is disabled.

Configuration Option	Description
Pdelay Average Weight	Represents the weight factor (w) for the Pdelay calculation: $PDelay_{New} = \frac{\left((w-1) * PDelay_{Old}\right) + PDelay_{Current}}{w}$
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 2-8 Pdelay initiator configuration options

Configuration Option	Description
Pdelay Resp Use Global Time	Enable or Disable usage of global times for the Pdelay responder. If enabled, the RequestReceiptTimestamp and ResponseOriginTimestamp transmitted with a PdelayResp/PdelayRespFup are converted into the synchronized global time. Otherwise, the local time is used.

Table 2-9 Pdelay responder configuration options

Note
Usage of global times for the Pdelay responder is only supported when Message compliance is enabled, and switch management is disabled. Furthermore, time validation cannot be used for a time domain where usage of global times is enabled for any Pdelay responder.

Note
If several EthTSyn port configurations reference the same EthIf controller (i.e. they are mapped to the same physical port), only the Pdelay configuration of the lowest timedomain is used for Pdelay measurements.

2.1.8 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.

2.1.9 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.

2.1.10 Debounce time

In order to avoid traffic bursts on an Ethernet link, the EthTSyn is capable to use a debounce time mechanism. The debounce time describes the minimal time interval between the transmission of two EthTSyn messages on one Ethernet link. It is therefore independent of the TimeDomain. Although the debounce time is configurable for each EthTSyn port individually, it must be ensured that the debounce time is equal for each EthTSyn port using the same underlying HW-Port. Usage of the debounce time mechanism can be de-/activated by configuration.

The following three message groups describe the transmission priority if more than one message is waiting for transmission when the debounce time expires:

1. Sync, Follow_Up, Announce
2. Pdelay_Req
3. Pdelay_Resp, Pdelay_Resp_Fup

The groups are listed in order of descending priority. Sync messages, for example, are transmitted with a higher priority than Pdelay_Req messages. The priority of messages within the same message group is determined by the TimeDomain Id where lower Id means higher priority. An Announce message with TimeDomain Id = 3, for example, has a higher priority than a Sync message with Id = 5.

Deviations from this priority can occur if the transmission of one message fails. In this case, the message with the second highest priority will be transmitted and the debounce timer is restarted. A new attempt for transmitting the first message is only possible after the debounce time is expired again.

As consequence of the prioritization, the debounce time must be chosen carefully with the number of different messages and their cyclic transmission times in mind. Otherwise, the cyclic transmission can be prolonged or lower priority messages will never be transmitted. The following examples illustrate possible problems:

> In case that Sync and Follow_Up messages of one time domain are transmitted with a cyclic transmission time of 100 ms, the debounce time must not be more than 50 ms. If it is for example 70 ms, two consecutive Sync messages are transmitted with a time difference of 140 ms which effectively increases the cyclic transmission time by 40 ms.

> In case that Sync and Follow_Up messages of five time domain are transmitted with a cyclic transmission time of 100 ms each, the debounce time must not be more than 10 ms. Otherwise, the messages of the time domain with highest Id will never be transmitted because there is always a message with higher priority ready for transmission.

> A similar problem can occur for messages of different groups. In case that Sync and Follow_Up messages of one time domain and Pdelay_Req messages are transmitted with a cyclic transmission time of 100 ms each, the debounce time must not be more than 33 ms. Otherwise, the Pdelay_Req message will never be transmitted.

> If switch management is enabled and the bridge mode is transparent clock, received Sync and Follow_Up messages are forwarded to all master port. Consequently, the maximum possible debounce time also depends on the cyclic transmission time of the global time grandmaster.

2.1.11 Time synchronization of ethernet switches

For some use cases (e.g. TSN) it might be necessary to synchronize the time of an ethernet switch. The EthTSyn offers the possibility to handle the time synchronization of one or multiple ethernet switches. For information about the configuration for switch time synchronization refer to chapter 5.2.

Caution
A version of 11.00.00 or greater of the MICROSAR Classic EthIf is required to use this Feature.

2.1.11.1 Cascaded switches

For the synchronization of cascaded switches, a Master-Slave concept is used. The switch which is directly connected to the grand master must be the Master-Switch for the cascade. All performed corrections are only calculated for the Master-Switch and applied to the Master-Switch and all Slave-Switches of the cascade.

A separate synchronization of all cascaded switches is not supported.

Caution
Limitations for cascaded switches:

For the synchronization of cascaded switches, it is mandatory that the PTP Clock of all involved switches is driven by the same oscillator or that the Master-Switch is providing the PTP Clock for all Slave-Switches.

2.1.11.2 Notification about changes in the switch synchronization state

For each switch cascade which is synchronized by the EthTSyn, one user can register a notification for changes in the synchronization state of the switch cascade. An overview about the reported synchronization states and their meaning can be found in Table 2-10.

Reported switch sync state	Description
ETHTSYN_SYNC	The time of the switch cascade is synchronized to the time master, i.e. the computed switch time offset with the last sync event does not exceed the configured sync precision limit.
ETHTSYN_UNSYNC	The time of the switch cascade is not synchronized to the time master, i.e. the computed switch time offset exceeded the configured sync precision limit for at least ‘max out of sync count’ consecutive sync events.
ETHTSYN_UNCERTAIN	It is unclear whether the switch time is synchronized to the time master or not. This happens when:

	▶ Computation of the time offset failed ▶ Applying the time correction to at least one of the switches failed ▶ Synchronization event timeout was detected

Table 2-10 Reported switch sync states

Caution
If the synchronization state UNCERTAIN is reached because applying the time correction to a slave switch failed, but was successful for the master switch, this slave switch will never be synchronized again without a reset because the clocks of slave and master switch are differently set now. However, the master switch can reach and report the state SYNC during the next synchronization events.

2.1.11.3 Rate regulator time synchronization mechanism

The rate regulator mechanism is used for time synchronization of Ethernet switches. The basic idea is to synchronize without performing time leaps for offset correction. Hence, only the rate ratio (i.e. clock frequency) is adapted in such a way that the offset is zero after a certain number of synchronization events.

The adapted rate ratio is determined by a configured number (*N*) of parallel rate measurements which all start and end with a synchronization event. This process is illustrated in Figure 2-6 for *N* = 3 rate measurements. The horizontal axis describes the progression of time. Blue arrows represent points of time when synchronization events occur. Red, yellow and green lines are the three rate measurements. Dots on these lines mark the end and beginning of the respective measurement. The measurements begin at successive synchronization events and the first measurement does not end until the last measurement started. Consequently, *N* is also the duration of each rate measurement when this duration is measured in number of synchronization cycles. The red, yellow and green vertical arrows indicate which measurement was used to calculate the new rate ratio (RR) at a given synchronization event.

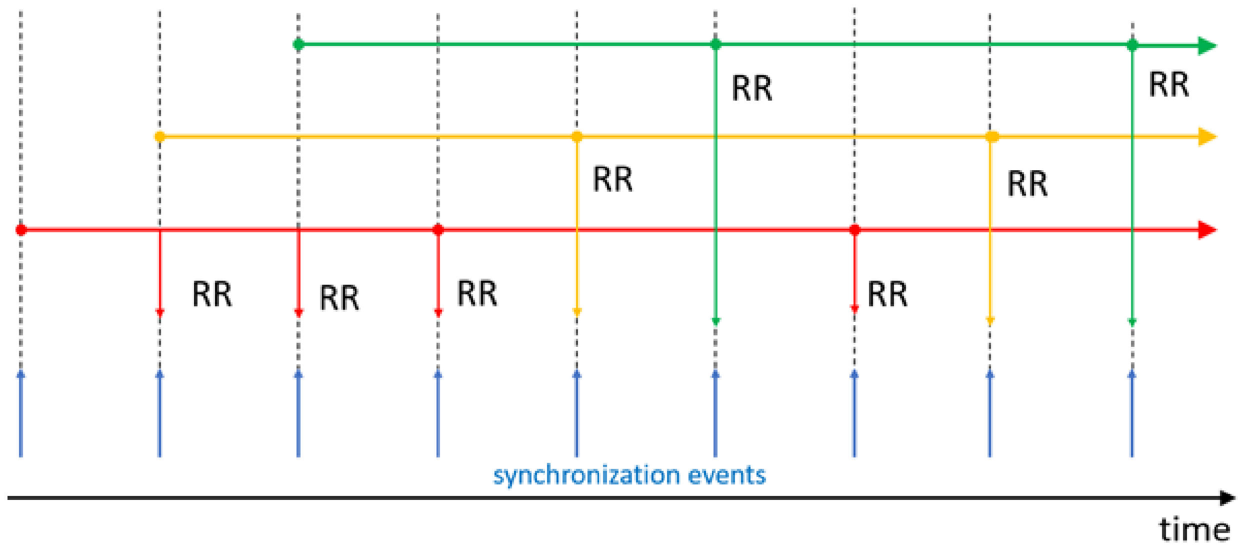


Figure 2-6 Schematic representation of the measurement sequence used by the rate regulator time synchronization mechanism

The rate ratio itself consists of two components: the grand master rate ratio (RR^{GM}) and the offset correction rate ratio (RR^{Off}) which are calculated according to the following equation:

$$\mathrm{RR} = \mathrm{RR}^{\mathrm{GM}} + \mathrm{RR}^{\mathrm{Off}} = \frac{\mathrm{MTD}}{\mathrm{STD}} + \frac{N}{M}\frac{\mathrm{Off}}{\mathrm{STD}}$$

MTD and STD are the differences of the master time and switch time, respectively, between two successive synchronization events. If the duration of the rate measurements is longer than one synchronization cycle ($N > 1$), the accumulated value of MTD and STD over all synchronization cycles of the current measurement is used for the calculation.

The grand master rate ratio corrects the difference of the clock frequencies between master and slave whereas the offset correction rate ratio adjusts the frequency in a way that the offset vanishes after a configured number (M) of further synchronization cycles. If the mechanism is configured to correct the offset during one synchronization cycle ($M = 1$), it can lead to an overshoot of the synchronization mechanism and induce a further offset. A higher value of M might increase the duration until synchronization is established but stabilizes the regulation.

Another possible source for errors is a random fluctuation of the duration of each synchronization cycle (e.g. due to jitter). This leads to an error in the calculation of the offset correction rate ratio which can in turn induce a further offset and prevent the system from synchronizing. This effect can be reduced with a higher value of N because the increased duration of each rate measurement decreases the impact of random errors.

There are two configurable options in which the rate regulator uses time leap to correct the offset. They are described in the following subchapters.

2.1.11.3.1 Initial offset correction

After initialization, the clocks of different devices involved in the time synchronization can have significantly different time values. For example, the global time master might receive its clock from an external source (e.g. GPS) while the slave just starts with a time of zero.

The time offset between master and slave can therefore be so high that the correction via rate ratio might not be possible or take too much time until synchronization is established.

In order to prevent this, it is possible to enable an offset correction via time leap and grand master rate ratio before synchronization was established the first time.

2.1.11.3.2 Offset jump correction threshold

If the clock of the time master exhibits a time leap during operation, the resulting offset between master and slave can be so high that the correction via rate ratio might not be possible or take too much time until synchronization is established.

In order to prevent this, it is possible to configure an offset jump correction threshold. If the offset exceeds the value of this threshold, it will be corrected by time leap instead of rate regulation.

Caution

It is assumed that the rate ratio (i.e. clock speed ratio) between master and switch is relatively small (~200ppm, depending on the clock source).

In case the offset jump correction is used and the configured threshold is smaller than the offset caused by the rate ratio within one sync cycle, no synchronization can be established.

2.1.12 Time validation

When the time validation feature is enabled for a time domain, the EthTSyn reports the data used for time synchronization (Master/Slave) and Pdelay calculation (PdelayInitiator/PdelayResponder) of the time domain to the StbM by using the corresponding Interfaces of the StbM. For a more detailed description of the feature please refer to [2] Specification of Time Synchronization over Ethernet and [8] Specification of Synchronized Time-Base Manager

Note
The EthTSyn mainly follows AUTOSAR version 4.4. However, the time validation was introduced with Version 19-11 and is implemented according to this version with the following deviations:

> The parameter `EthTSynTimeValidationSupport` in the `EthTSynGeneral` container is not available. The time validation feature is automatically activated depending on the setting within the referenced `StbMSynchronizedTimeBase`

> It is expected that a `StbMSynchronizedTimeBase` includes an optional container `StbMTimeValidation`. The existence of this container enables the time validation feature.

> It is expected that the structure `StbM_EthTimeMasterMeasurementType` has a member `correctionField (sint64)` (like the `StbM_EthTimeSlaveMeasurementType`)

2.1.13 Destination MAC address

The destination MAC address used by the EthTSyn for transmitted frames can be configured via the BSWMD parameter `EthTSynDestPhyAddr` in the `EthTSynGeneral` container. Besides the general selection, it is also possible to overwrite the destination MAC address for specific time domain(s) via the BSWMD parameter `EthTSynGlobalTimeDomainDestPhyAddr` in the `EthTSynGlobalTimeDomain` container.

Note
It is assumed that the configured destination MAC addresses are also relevant for frame reception. So, in case of a multicast MAC address, the EthTSyn will update the physical address filter accordingly to allow frame reception. However, it is also possible to configure a unicast address. In this case, the EthTSyn will **not** update the physical address filter. Therefore, the user must ensure that frame reception is possible if it is required.

2.2 Interaction with EthIf and StbM BSW module

The interaction of the EthTSyn with the EthIf and StbM BSW modules is described in [1] Specification of Time Synchronization over Ethernet.

Caution
It is assumed that the times retrieved via `EthIf_GetCurrentTime()`, `EthIf_GetIngressTimeStamp()`, `EthIf_GetEgressTimeStamp()` and `StbM_GetCurrentVirtualLocalTime()` are monotonously increasing. In addition, times retrieved from the HW via EthIf are transformed into a virtual local time. Therefore, it is expected that these times do not exceed the value range of (2^{64} – 1) nanoseconds (~584 years).

2.3 Initialization

The EthTSyn is initialized by calling the `EthTSyn_InitMemory()` and `EthTSyn_Init()` services with the address of the pre-compile configuration data passed as 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
Since EthTSyn is using the EthIf and Eth BSW module they must be initialized before executing the EthTSyn initialization process.

2.4 States

The transmission mode is a state of the EthTSyn, which can be changed by external applications. It can either be `ETHTSYN_TX_ON` or `ETHTSYN_TX_OFF`. It is set to `ETHTSYN_TX_ON` during initialization, which means that EthTSyn can transmit messages afterwards and is operational. The transmission mode can be changed via the module API `EthTSyn_SetTransmissionMode()`. If it is changed to `ETHTSYN_TX_OFF`, EthTSyn will no longer transmit any messages on the given Ethernet interface.

2.5 Main Functions

The EthTSyn has multiple main functions. Depending on the configuration they exist or not. The following tables list all main functions with information about existence and their purpose.

EthTSyn_MainFunction()		
Existence	Timing	Description
Always	Scheduled according to the period provided by configuration parameter `EthTSynMainFunctionPeriod`.	Processes all state machines and timers of the EthTSyn.

Table 2-11 EthTSyn_MainFunction() description

EthTSyn_SwtMgmt_MainFunction(()		
Existence	Timing	Description
`EthTSynEnableSwitchMgmt` is enabled	Scheduled according to the period provided by configuration parameter `EthTSynSwitchMgmtMainFunctionPeriod`.	Processes the switch management Rx/Tx management objects as well as received messages.

Table 2-12 EthTSyn_SwtMgmt_MainFunction() description

2.6 Error Handling

2.6.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [5], if development error reporting is enabled (i.e. pre-compile parameter `ETHTSYN_DEV_ERROR_REPORT==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 4.2 and 4.4. 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()`
0x08	`EthTSyn_TrcvLinkStateChg()`
0x09	`EthTSyn_MainFunction()`
0x51	`EthTSyn_SwtMgmt_MainFunction()`
0x54	`EthTSyn_SwtMgmtQ_Push()`
0x55	`EthTSyn_GetAndResetPortParameterStatistics()`
0x56	`EthTSyn_Ts_IsTsValid()`
0x57	`EthTSyn_Ts_IsStbmTsValid()`
0x58	`EthTSyn_Ts_TsToIntVltChecked()`

Table 2-13 Service IDs: Development error reporting

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

Error Code		Description
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_MSG_LENGTH	Received message with an invalid Message Length (either in Header or total frame length)
0x31	ETHTSYN_E_PROTOCOL_VERSION	Received message with an invalid Protocol Version
0x32	ETHTSYN_E_MSG_TYPE	Received message with an invalid Message Type
0x33	ETHTSYN_E_FRAME_TYPE	Received message with an invalid Frame Type
0x34	ETHTSYN_E_DOMAIN_NUMBER	Received message with an invalid Domain Number
0x35	ETHTSYN_E_TRCV_DOWN	EthTSyn_RxIndication() called for inactive controller
0x36	ETHTSYN_E_NO_PORT_FOUND	Received message which does not belong to any configured EthTSyn port
0x37	ETHTSYN_E_FUP_TX_LENGTH	Generated FollowUp Tx length is invalid
0x38	ETHTSYN_E_PORT_IDX	API called with invalid port index
0x39	ETHTSYN_E_INV_TS_FORMAT	Detected a timestamp with invalid format
0x50	ETHTSYN_SWT_MGMT_E_MSG_BUFFER_OVERFLOW	Received message while there is no free message buffer available
0x51	ETHTSYN_SWT_MGMT_E_MSG_BUFFER_PAYLOAD_OVERFLOW	Payload too big for the message buffer
0x52	ETHTSYN_SWT_MGMT_E_GET_RX_MGMT_OBJ_FAILED	EthIf_GetRxMgmtObject() returned with error.
0x53	ETHTSYN_SWT_MGMT_E_SWT_MGMT_Q_INV_FREE_BUF_IDX	SwtMgmt-buffer was not created before storing the SwtMgmtObj pointer.

Table 2-14 Development errors reported to DET

2.6.2 Runtime Error Reporting

By default, runtime error are reported to the DET using the service Det_ReportRuntimeError() as specified in see [5], if master-slave conflict detection is enabled (i.e. `ETHTSYN_MASTER_SLAVE_CONFLICT_DETECTION==STD_ON`).

The reported EthTSyn ID is 164.

The following table presents the service IDs and the related services:

Service ID	Service
0x50	`EthTSyn_Rx_IntProcRcvdMsg()`
0x52	`EthTSyn_SlaveRx_ProcRcvdSync()`
0x53	`EthTSyn_SlaveRx_FwdRcvdSync()`

Table 2-15 Service IDs: Runtime error reporting

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

Error Code		Description
0x01	`ETHTSYN_E_TMCONFLICT`	Time Master Conflict – Time Master received Sync message from another Time Master
0x02	`ETHTSYN_E_TSCONFLICT`	Time Slave Conflict – Time Slave received Sync message from different Time Masters

Table 2-16 Runtime errors reported to DET

2.6.3 Production Code Error Reporting

No production error code reporting to the DEM (see [6]) is supported.

Error Code	Description
none	

Table 2-17 Errors reported to DEM

3 Integration

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

3.1 Scope of Delivery

The delivery of the EthTSyn contains the files which are described in the chapters 3.1.1 and 3.1.2.

3.1.1 Static Files

File Name	Description
`EthTSyn.c`	Implementation of Init, Scheduling and VersionInfo units
`EthTSyn.h`	API declarations for EthTSyn module
`EthTSyn_AnnounceRcvSm_Int.h`	Internal API declarations of AnnounceRcvSm unit
`EthTSyn_AnnounceSendSm_Int.h`	Internal API declarations of AnnounceSendSm unit
`EthTSyn_AsymSiteSyncSyncSm_Int.h`	Internal API declarations of AsymSiteSyncSyncSm unit
`EthTSyn_Cbk.h`	API call-back declarations for EthTSyn module
`EthTSyn_ComCtrl.c`	Implementation of ComCtrl unit
`EthTSyn_ComCtrl.h`	API declarations of ComCtrl unit
`EthTSyn_ComCtrl_Cbk.h`	API call-back declarations of ComCtrl unit
`EthTSyn_ComCtrl_Int.h`	Internal API declarations of ComCtrl unit
`EthTSyn_CrcHndl.c`	Implementation of CrcHndl unit
`EthTSyn_CrcHndl_Int.h`	Internal API declarations of CrcHndl unit
`EthTSyn_FupRxHandler_Int.h`	Internal API declarations of FupRxHandler unit
`EthTSyn_Init.h`	API declarations of Init unit
`EthTSyn_Init_Int.h`	Internal API declarations of Init unit
`EthTSyn_Master.c`	Implementation of AnnounceSendSm, SyncSendSm, SyncSendTx and PortSyncSendSm units
`EthTSyn_MsgDefs_Int.h`	Macros which are used for message serialization and deserialization
`EthTSyn_PdelayInitiator.c`	Implementation of PdReqSm and PdReqTrcv units
`EthTSyn_PdelayResponder.c`	Implementation of PdRespSm unit
`EthTSyn_PdReqSm_Int.h`	Internal API declarations of PdReqSm unit
`EthTSyn_PdReqTrcv_Int.h`	Internal API declarations of PdReqTrcv unit
`EthTSyn_PdRespSm_Int.h`	Internal API declarations of PdRespSm unit
`EthTSyn_PortParamStats.c`	Implementation of PortParamStats unit
`EthTSyn_PortParamStats.h`	API declaration of PortParamStats unit
`EthTSyn_PortParamStats_Int.h`	Internal API declarations of PortParamStats unit
`EthTSyn_PortSyncSendSm_Int.h`	Internal API declarations of PortSyncSendSm unit

File Name	Description
EthTSyn_RateMeas_Int.h	Internal API declarations of RateMeas unit
EthTSyn_Reception.c	Implementation of Rx and SwtMgmtRx units
EthTSyn_Rx_Cbk.h	API call-back declaration of Rx unit
EthTSyn_Rx_Int.h	Internal API declarations of Rx unit
EthTSyn_SiteSyncSync.c	Implementation of AsymSiteSyncSyncSm, SiteSyncSyncSm, SiteSyncSyncTx and SiteSyncSyncTlv units
EthTSyn_SiteSyncSyncSm_Int.h	Internal API declarations of SiteSyncSyncSm unit
EthTSyn_SiteSyncSyncTlv_Int.h	Internal API declarations of SiteSyncSyncTlv unit
EthTSyn_SiteSyncSyncTx_Int.h	Internal API declarations of SiteSyncSyncTx unit
EthTSyn_Slave.c	Implementation of AnnounceRcvSm, SyncRcvSm, SlaveRx and FupRxHandler units
EthTSyn_SlaveRx_Int.h	Internal API declarations of SlaveRx unit
EthTSyn_SwitchTimeSynchronization.c	Implementation of SwtTimeSyncSm and RateMeas units
EthTSyn_SwtMgmtQ.c	Implementation of SwtMgmtQ unit
EthTSyn_SwtMgmtQ_Int.h	Internal API declarations of SwtMgmtQ unit
EthTSyn_SwtMgmtRx_Int.h	Internal API declarations of SwtMgmtRx unit
EthTSyn_SwtMgmtTx_Int.h	Internal API declarations of SwtMgmtTx unit
EthTSyn_SwtTimeSyncSm_Int.h	Internal API declarations of SwtTimeSyncSm unit
EthTSyn_SyncRcvSm_Int.h	Internal API declarations of SyncRcvSm unit
EthTSyn_SyncSendSm_Int.h	Internal API declarations of SyncSendSm unit
EthTSyn_SyncSendTx_Int.h	Internal API declarations of SyncSendTx unit
EthTSyn_Timestamp.c	Implementation of Timestamp unit
EthTSyn_Timestamp_Int.h	Internal API declarations of Timestamp unit
EthTSyn_Transmission.c	Implementation of Tx, TxConf and SwtMgmtTx units
EthTSyn_Tx_Int.h	Internal API declarations of Tx unit
EthTSyn_TxConf_Cbk.h	API call-back declaration of Timestamp unit
EthTSyn_TxConf_Int.h	Internal API declarations of TxConf unit
EthTSyn_Types.h	Type definitions for EthTSyn module
EthTSyn_Types_Int.h	Internal type definitions for EthTSyn module
EthTSyn_VersionInfo.h	API declaration of VersionInfo unit

Table 3-1 Static files

3.1.2 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_GeneratedCfgAccess.h`	Generated header file with functions/macros for access of the generated data
`EthTSyn_GeneratedComplexTypes.h`	Generated header file with complex (i.e. structs, etc.) type definitions required for generated data
`EthTSyn_GeneratedSimpleTypes.h`	Generated header files with basic type definitions required for generated data
`EthTSyn_Lcfg.c`	Generated pre-compile and link time configuration data
`EthTSyn_Lcfg.h`	Generated header for pre-compile and link time configuration data

Table 3-2 Generated files

3.2 Critical Sections

The EthTSyn uses the Critical Section implementation of the SchM.

3.2.1 Exclusive Area 0

ETHTSYN_EXCLUSIVE_AREA_0

Purpose:
Ensures consistency of global RAM variables.

Interfaces:

- `SchM_Enter_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_0`
- `SchM_Exit_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_0`

Runtime:
Long: deep nested exclusive areas.

Dependency:

- `EthTSyn_MainFunction()`
- `EthTSyn_SwtMgmt_MainFunction()`
- `EthTSyn_TrcvLinkStateChg()`
- `EthTSyn_SetTransmissionMode()`
- `EthTSyn_RxIndication()`
- `EthTSyn_TxConfirmation()`

> `EthTSyn_GetAndResetPortParameterStatistics()`

Recommendation:
It is recommended to use global interrupt locks for this critical section.

Table 3-3 Exclusive Area 0

3.2.2 Exclusive Area Rx

ETHTSYN_EXCLUSIVE_AREA_RX

Purpose:
Ensures consistency of global RAM variables.

Interfaces:
> `SchM_Enter_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_RX`
> `SchM_Exit_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_RX`

Runtime:
Long: deep nested exclusive areas.

Dependency:
> `EthTSyn_MainFunction()`
> `EthTSyn_SwtMgmt_MainFunction()`
> `EthTSyn_TrcvLinkStateChg()`
> `EthTSyn_RxIndication()`
> `EthTSyn_TxConfirmation()`

Recommendation:
It is recommended to disable Ethernet reception interrupts for this critical section.

Table 3-4 Exclusive Area Rx

3.2.3 Exclusive Area Tx

ETHTSYN_EXCLUSIVE_AREA_TX

Purpose:
Ensures consistency of global RAM variables.

Interfaces:
> `SchM_Enter_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_TX`
> `SchM_Exit_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_TX`

Runtime:
Long: deep nested exclusive areas.

Dependency:
> `EthTSyn_MainFunction()`
> `EthTSyn_SwtMgmt_MainFunction()`
> `EthTSyn_TrcvLinkStateChg()`
> `EthTSyn_RxIndication()`
> `EthTSyn_TxConfirmation()`

Recommendation:
It is recommended to disable Ethernet transmission interrupts for this critical section.

Table 3-5 Exclusive Area Tx

3.2.4 Exclusive Area Link State

ETHTSYN_EXCLUSIVE_AREA_LINK_STATE

Purpose:
Ensures consistency of global RAM variables.

Interfaces:
> `SchM_Enter_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_LINK_STATE`
> `SchM_Exit_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_LINK_STATE`

Runtime:
Medium: one nesting level.

Dependency:
> `EthTSyn_MainFunction()`
> `EthTSyn_SwtMgmt_MainFunction()`
> `EthTSyn_TrcvLinkStateChg()`
> `EthTSyn_RxIndication()`
> `EthTSyn_TxConfirmation()`

Recommendation:
It is recommended to disable transceiver link state reporting for this critical section.

Table 3-6 Exclusive Area Link State

3.2.5 Exclusive Area Get Ts

ETHTSYN_EXCLUSIVE_AREA_GET_TS
Purpose: > Ensures consistency of global RAM variables related to ingress and egress timestamping > Prevents interrupts between invocation of EthTSyn_TxConfirmation respectively EthTSyn_RxIndication and getting the egress respectively ingress timestamp to minimize the delay > Prevents interrupts between getting the current Ethernet time via EthIf_GetCurrentTime and getting the current virtual local time via StbM_GetCurrentVirtualLocalTime to minimize the delay **Interfaces:** > `SchM_Enter_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_GET_TS` > `SchM_Exit_EthTSyn_ETHTSYN_EXCLUSIVE_AREA_GET_TS` **Runtime:** Long: This exclusive area covers calls to several sub-functions and to foreign modules (EthIf and StbM). **Dependency:** > `EthTSyn_MainFunction()` > `EthTSyn_SwtMgmt_MainFunction()` > `EthTSyn_RxIndication()` > `EthTSyn_TxConfirmation()` **Recommendation:** It is recommended to use global interrupt locks for this critical section, unless the StbM uses the Os as time source.

Table 3-7 Exclusive Area Get Ts

Caution
If the StbM is configured to use OS APIs and the implementation method of the exclusive area `ETHTSYN_EXCLUSIVE_AREA_GET_TS` 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 `ETHTSYN_EXCLUSIVE_AREA_GET_TS` inside the RTE / SchM configuration needs to be set to `OS_RESOURCE` or `CUSTOM`.

If `OS_RESOURCE` is selected the ETH 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 whithout 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.

Caution
Be aware that the EthTSyn calls all these critical sections nested (with themselves and other critical sections) but exits the critical sections in the reverse order they were entered. Please assert that this is supported by the implementation of the critical sections.

3.3 Main Functions

The EthTSyn expects that its own main functions do not interrupt each other. Furthermore, it expects that its own main functions do not interrupt the main functions of related modules and are not interrupted by main functions of related modules.

The EthTSyn related modules are the EthIf and StbM modules.

Caution
Consider main function expectations when using preemptive tasks.

4 API Description

For an interface overview please see Figure 1-2.

4.1 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_OFF`
`EthTSyn_SyncStateType`	enum	Synchronization state	`ETHTSYN_SYNC` `ETHTSYN_UNSYNC` `ETHTSYN_UNCERTAIN` `ETHTSYN_NEVERSYNC`

Table 4-1 Type definitions

[EthTSyn_PortParamStatsType]

This structure contains the port parameter statistics as defined in chapter 14.7 of [9].

Type Name	C-Type	Description	Value Range
rxSyncCount	uint32	Number of Sync messages received by SyncRcvSm	`0..2^32-1`
rxFollowUpCount	uint32	Number of FollowUp messages received by SyncRcvSm	`0..2^32-1`
rxPdelayRequestCount	uint32	Number of Pdelay_Req messages received by PdRespSm	`0..2^32-1`
rxPdelayResponseCount	uint32	Number of Pdelay_Resp messages received by PdReqSm	`0..2^32-1`
rxPdelayResponseFollowUpCount	uint32	Number of Pdelay_Resp_Fup messages received by PdReqSm	`0..2^32-1`
rxAnnounceCount	uint32	Number of Announce messages received by AnnounceRcvSm	`0..2^32-1`
rxPTPPacketDiscardCount	uint32	Number of PTP messages discarded for any of the following reasons: ▶ Announce message is discarded by AnnounceRcvSm	`0..2^32-1`

		▶ FollowUp message not received within timeout ▶ Pdelay_Resp/Pdelay_Resp_Fup message not received within timeout	
syncReceiptTimeoutCount	uint32	Not supported. Always reported as 0	`0..2^32-1`
announceReceiptTimeoutCount	uint32	Not supported. Always reported as 0	`0..2^32-1`
pdelayAllowedLostResponsesExceededCount	uint32	Number of times the Pdelay allowed lost response counter was exceeded	`0..2^32-1`
txSyncCount	uint32	Number of transmitted Sync messages	`0..2^32-1`
txFollowUpCount	uint32	Number of transmitted FollowUp messages	`0..2^32-1`
txPdelayRequestCount	uint32	Number of transmitted Pdelay_Req messages	`0..2^32-1`
txPdelayResponseCount	uint32	Number of transmitted Pdelay_Resp messages	`0..2^32-1`
txPdelayResponseFollowUpCount	uint32	Number of transmitted Pdelay_Resp_Fup messages	`0..2^32-1`
txAnnounceCount	uint32	Number of transmitted Announce messages	`0..2^32-1`

Table 4-2 EthTSyn_PortParamStatsType

4.2 Services provided by EthTSyn

4.2.1 EthTSyn_GetVersionInfo

Prototype	
`void **EthTSyn_GetVersionInfo** (ETHTSYN_P2VAR(Std_VersionInfoType) VersionInfoPtr)`	
Parameter	
VersionInfoPtr [out]	Pointer to where to store the version information. Parameter must not be NULL.
Return Code	
void	none
Functional Description	
Returns the version information. EthTSyn_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component.	
Particularities and Limitations	
-	
Call Context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 4-3 EthTSyn_GetVersionInfo

4.2.2 EthTSyn_Init

Prototype	
`void **EthTSyn_Init** (EthTSyn_ConfigType* CfgPtr)`	
Parameter	
CfgPtr [in]	Pointer to configuration
Return Code	
void	none
Functional Description	
Initialization of the EthTSyn module.	
Particularities and Limitations	
The function EthTSyn_InitMemory() must be called first	
Call Context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 4-4 EthTSyn_Init

4.2.3 EthTSyn_InitMemory

Prototype	
`void` **`EthTSyn_InitMemory`** `(void)`	
Parameter	
void	none
Return Code	
void	none
Functional Description	
Memory initialization of the EthTSyn module. Service to initialize module global variables at power up. This function initializes the variables in *_INIT_* sections. Used in case they are not initialized by the startup code.	
Particularities and Limitations	
-	
Call Context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 4-5 EthTSyn_InitMemory

4.2.4 EthTSyn_MainFunction

Prototype	
`void` **`EthTSyn_MainFunction`** `(void)`	
Parameter	
void	none
Return Code	
void	none
Functional Description	
Handles cyclic tasks of the EthTSyn module.	
Particularities and Limitations	
-	
Call Context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 4-6 EthTSyn_MainFunction

4.2.5 EthTSyn_SwtMgmt_MainFunction

Prototype	
`void` **`EthTSyn_SwtMgmt_MainFunction`** `(void)`	
Parameter	
void	none
Return Code	
void	none
Functional Description	
Handles the switch management Rx/Tx management objects used by the EthTSyn and processes received messages.	
Particularities and Limitations	
-	
Call Context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 4-7 EthTSyn_SwtMgmt_MainFunction

4.2.6 EthTSyn_SetTransmissionMode

Prototype	
`void` **`EthTSyn_SetTransmissionMode`** `(uint8 CtrlIdx, EthTSyn_TransmissionModeType Mode)`	
Parameter	
CtrlIdx [in]	EthIf controller index
Mode [in]	Mode that shall be applied: ETHTSYN_TX_OFF - turn off EthTSyn Tx capability ETHTSYN_TX_ON - turn on EthTSyn Tx capability
Return Code	
void	none
Functional Description	
Set the EthTSyn transmission mode. This API is used to turn on and off the TX capabilities of the EthTSyn.	
Particularities and Limitations	
-	
Call Context	
> TASK|ISR2 > This function is Synchronous > This function is Non-Reentrant	

Table 4-8 EthTSyn_SetTransmissionMode

4.2.7 EthTSyn_GetAndResetPortParameterStatistics

Prototype	
`Std_ReturnType` **`EthTSyn_GetAndResetPortParameterStatistics`** `(uint8 PortIdx, boolean StatisticsResetNeeded, ETHTSYN_P2VAR(EthTSyn_PortParamStatsType) PortParamStatsPtr)`	
Parameter	
`PortIdx [in]`	Index of the EthTSyn port
`StatisticsResetNeeded [in]`	Indicates if the port parameter counter shall be reset
`PortParamStatsPtr [out]`	Pointer to an object where to store the port parameter statistics. Parameter must not be NULL.
Return code	
Std_ReturnType	E_OK - Successfully retrieved port parameter statistics
Std_ReturnType	E_NOT_OK - Retrieving port parameter statistics failed
Functional Description	
Retrieves and optionally resets the port parameter statistics of an EthTSyn port.	

Particularities and Limitations
Configuration Variant(s): ETHTSYN_PORT_PARAM_STATS_SUPPORT
Call context
> TASK|ISR2 > This function is Synchronous > This function is Non-Reentrant

Table 4-9 EthTSyn_GetAndResetPortParameterStatistics

4.3 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_GetRxMgmtObject()`
EthIf	`EthIf_GetTxMgmtObject()`
EthIf	`EthIf_ProvideTxBuffer()`
EthIf	`EthIf_SetSwitchMgmtInfo()`
EthIf	`EthIf_SwitchEnableTimeStamping()`
EthIf	`EthIf_SwitchSetCorrectionTime()`
EthIf	`EthIf_Transmit()`
StbM	`StbM_BusGetCurrentTime()`
StbM	`StbM_BusSetGlobalTime()`
StbM	`StbM_EthSetMasterTimingData()`
StbM	`StbM_EthSetPdelayInitiatorData()`
StbM	`StbM_EthSetPdelayResponderData()`
StbM	`StbM_EthSetSlaveTimingData()`
StbM	`StbM_GetCurrentTime()`

Component	API
StbM	`StbM_GetCurrentVirtualLocalTime()`
StbM	`StbM_GetOffset()`
StbM	`StbM_GetTimeBaseStatus()`
StbM	`StbM_GetTimeBaseUpdateCounter()`
VStdLib	`VStdLib_MemClr()`
VStdLib	`VStdLib_MemCpy_s()`

Table 4-10 Services used by the EthTSyn

4.4 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.

4.4.1 EthTSyn_RxIndication

Prototype	
`void` **`EthTSyn_RxIndication`** `(uint8 CtrlIdx, Eth_FrameType FrameType, boolean IsBroadcast, ETHTSYN_P2VAR(uint8) PhysAddrPtr, ETHTSYN_P2VAR(uint8) DataPtr, uint16 LenByte)`	
Parameter	
CtrlIdx [in]	EthIf controller index
FrameType [in]	Frame type of received Ethernet frame
IsBroadcast [in]	Broadcast indication: FALSE - frame is not a broadcast frame TRUE - frame is a broadcast frame
PhysAddrPtr [in]	Pointer to the 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	
Processing of received EthTSyn frames. By this API service the EthTSyn gets an indication and the data of a received frame.	
Particularities and Limitations	
-	
Call Context	
> TASK|ISR2 > This function is Synchronous > This function is Non-Reentrant	

Table 4-11 EthTSyn_RxIndication

4.4.2 EthTSyn_TxConfirmation

Prototype	
`void EthTSyn_TxConfirmation (uint8 CtrlIdx, uint8 BufIdx)`	
Parameter	
CtrlIdx [in]	EthIf controller index
BufIdx [in]	Index of the buffer resource
Return Code	
void	none
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	
-	
Call Context	
> TASK|ISR2 > This function is Synchronous > This function is Non-Reentrant	

Table 4-12 EthTSyn_TxConfirmation

4.4.3 EthTSyn_TrcvLinkStateChg

Prototype	
`void EthTSyn_TrcvLinkStateChg (uint8 CtrlIdx, EthTrcv_LinkStateType TrcvLinkState)`	
Parameter	
CtrlIdx [in]	Index of the Ethernet controller
TrcvLinkState [in]	New link state of the transceiver ETHTRCV_LINK_STATE_DOWN - Link Down ETHTRCV_LINK_STATE_ACTIVE - Link Up
Return Code	
void	none
Functional Description	
Callback function that notifies a changed state of the transceiver link. Allows resetting state machine in case of unexpected Link loss to avoid inconsistent Sync and Follow_Up sequences	
Particularities and Limitations	
-	
Call Context	
> TASK|ISR2 > This function is Synchronous > This function is Non-Reentrant	

Table 4-13 EthTSyn_TrcvLinkStateChg

4.5 Configurable Interfaces

4.5.1 Notifications

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

4.5.1.1 SyncSentCbk

Prototype	
`void EthTSyn_SyncSentCbk (EthTSyn_IntPortIdxType PortIdx, const uint8* TxBufPtr)`	
Parameter	
PortIdx[in]	Index of the EthTSyn port the sync message was transmitted on
TxBufPtr[in]	Pointer to the Ethernet transmission buffer of the transmitted sync message (Network-Byte-Order)
Return Code	
void	none

Functional Description
This callback function can be used for optional notification of a transmitted sync message.
Particularities and Limitations
-
Call Context
> Task

Table 4-14 SyncSentCbk

4.5.1.2 FollowUpSentCbk

Prototype	
`void` **`EthTSyn_FollowUpSentCbk`** `(EthTSyn_IntPortIdxType PortIdx, const uint8* TxBufPtr)`	
Parameter	
PortIdx[in]	Index of the EthTSyn port the follow up message was transmitted on
TxBufPtr[in]	Pointer to the Ethernet transmission buffer of the transmitted follow up message (Network-Byte-Order)
Return Code	
void	none
Functional Description	
This callback function can be used for optional notification of a transmitted follow up message	
Particularities and Limitations	
-	
Call Context	
> Task	

Table 4-15 FollowUpSentCbk

4.5.1.3 SwitchSyncStateChangeCbk

Prototype	
`void` **`EthTSyn_SwitchSyncStateChgCbk`** `(uint8 EthTSynSwitchIdx, EthTSyn_SyncStateType SyncState)`	
Parameter	
EthTSynSwitchIdx[in]	Index of the ethernet switch (cascade) in the context of the EthTSyn
SyncState[in]	The synchronization state of the passed switch (cascade)
Return Code	
void	none

Functional Description
This callback function can be used for optional notification about changes in the synchronization state of an ethernet switch (cascade).
Particularities and Limitations
-
Call Context
> Task

Table 4-16 SwitchSyncStateChangeCbk

5 Configuration

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

5.1 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.

5.2 Hints for configuration of switch management functionality

5.2.1 EthTSynSwitchMgmtNumMsgForwardRetries

When the EthTSyn is configured to manage an Ethernet switch and the `EthTSynSwitchMgmtBridgeMode` is set either to `TRANSPARENT_CLOCK` or `ASYMMETRIC_TRANSPARENT_CLOCK`, Sync and/or FollowUp messages are forwarded by the EthTSyn. This parameter determines how often the forwarding of a Sync or FollowUp message is retried by the EthTSyn when transmission fails for any reason. The following points should be considered for choosing the amount of retries:

- The following things lead to failing transmission and thus consume a retry:
  - No Ethernet transmission buffer is available
  - Egress timestamps cannot be enabled
  - The debounce timer is not yet expired
  - For a FollowUp: Egress timestamp of the corresponding Sync is not available
- The retry time is calculated via EthTSynSwitchMgmtNumMsgForwardRetries * EthTSynMainFunctionPeriod
- When EthTSynSwitchMgmtGlobalTimeSyncTimeout is enabled:
  - 2 times the retry time (because retry applies to both Sync and FollowUp forwarding) should not exceed the configured sync timeout (a sync timeout would be detected before the worst-case amount of retries where done)
- The retry time should not be greater than the expected Sync cycle time (because a new Sync would be received before the retry counter expired)

5.2.2 EthTSynSwitchMgmtGlobalTimeSyncTimout

When the EthTSyn is configured to manage an Ethernet switch and the `EthTSynSwitchMgmtBridgeMode` is set to `TRANSPARENT_CLOCK`, the sync timeout handling as described in chapter 2.1.4.4.3 can be used. The timeout value can be configured via the parameter `EthTSynSwitchMgmtGlobalTimeSyncTimout`. When it is used, the value should be a greater than the expected Sync cycle. Otherwise, a timeout is detected even when the Sync and FollowUp messages are received as expected.

5.2.3 Asymmetric transparent clock

For correct operation of the asymmetric transparent clock, proper configuration of the Ethernet switch driver is important. The message forwarding rules created for Sync messages need to match the port configuration within the EthTSyn. This means that Sync messages need to be forwarded to every switch master port of the EthTSyn and to the host itself. For details on how this configuration can be done please see [13].

5.3 Configuration of switch time synchronization

Note
Configuration of switch time synchronization is only required for some special use-cases (e.g., TSN). It is recommended to not use the switch time synchronization unless it is necessary. For more details about the feature see also chapter 2.1.11.

Figure 5-1 shows an overview of the switch time synchronization configuration for one switch (cascade). For each switch cascade such a configuration is required. A short explanation for the highlighted configuration options is provided in the following paragraphs:

1 Choose whether rate ratio correction should be applied to slave switches or not.

2 Reference to the (Master-)Switch which should be synchronized (in context of the EthIf).

3 Automatically calculated index of the EthTSyn switch time sync cascade. This index is provided as symbolic name value and is e.g. used in the optional switch sync state change notification.

4 Configuration of the time synchronization parameters, e.g. the time domain used for synchronization and the optional switch sync state change notification callback.

5 Optional reference(s) to the Slave-Switch(s) of the cascade (in context of the EthIf)

6 The used synchronization mechanism:

6.1 RateRegulator: Optionally offset is corrected by time leap before synchronization was established the first time. Afterwards only rate ratio is used to compensate frequency differences and to reduce the offset. For a detailed description of this mechanism and its configuration parameters see chapter 2.1.11.3. An overview of the rate regulator configuration is shown in Figure 5-2.

6.1.1 Number of parallel rate measurements. This is also the duration of each rate measurement when this duration is measured in number of synchronization cycles.

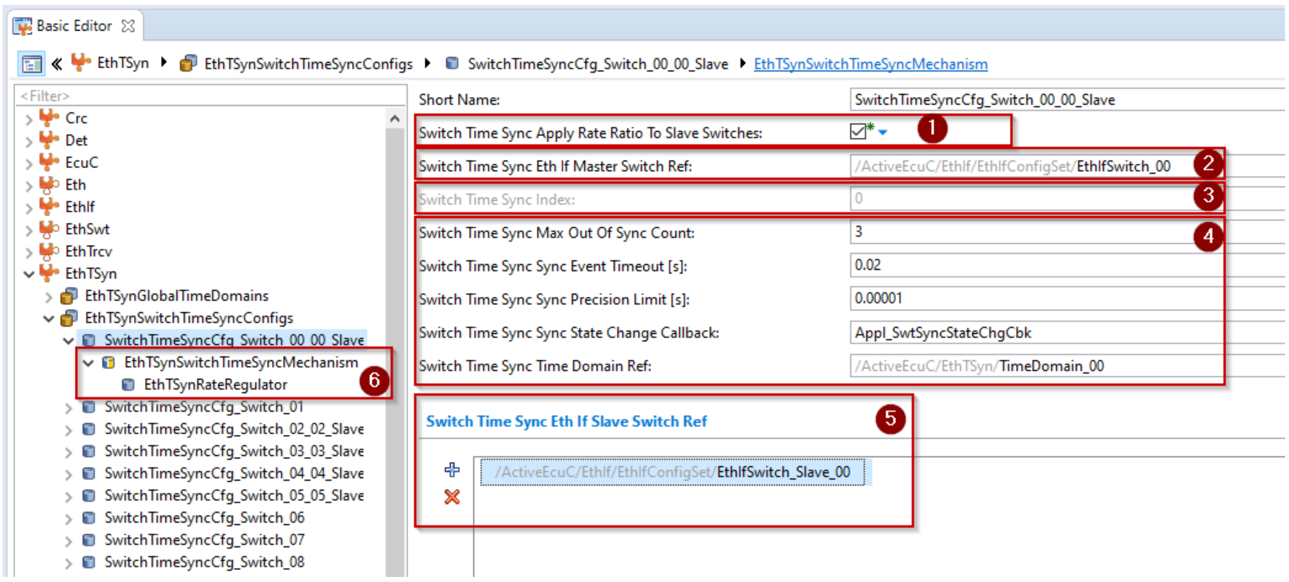
6.1.2 Duration over which the offset shall be reduced to zero by rate regulation. This duration is measured in number of synchronization cycles.

6.1.3 If the current offset exceeds this optional threshold, it will be corrected by time leap instead of rate regulation.

6.1.4 Configuration of the maximum and minimum allowed rate ratio correction. These values are always relative to the last rate ratio.

6.1.5 Choose whether the initial offset correction by time leap should be used or not.

Note
For a more detailed explanation of the different configuration parameters please refer to their description provided in EthTSyn_bswmd.arxml.


Basic Editor

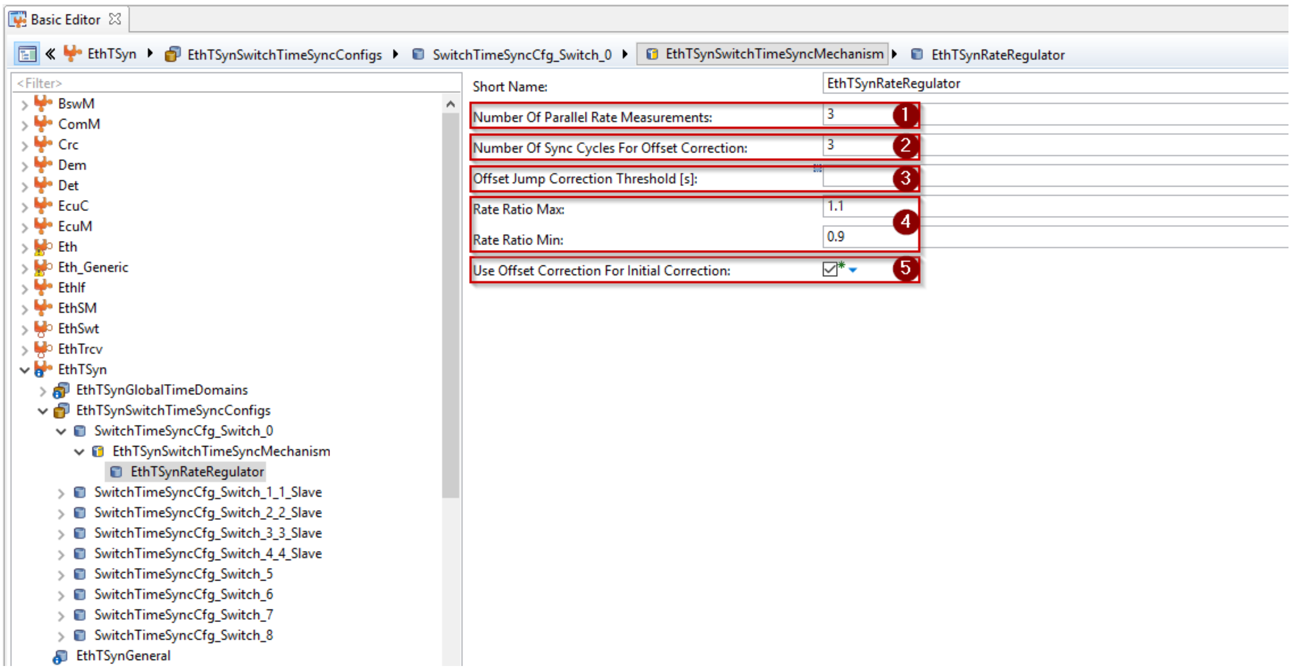
« EthTSyn ▸ EthTSynSwitchTimeSyncConfigs ▸ SwitchTimeSyncCfg_Switch_00_00_Slave ▸ EthTSynSwitchTimeSyncMechanism

<Filter>
- Crc
- Det
- EcuC
- Eth
- EthIf
- EthSwt
- EthTrcv
- EthTSyn
  - EthTSynGlobalTimeDomains
  - EthTSynSwitchTimeSyncConfigs
    - SwitchTimeSyncCfg_Switch_00_00_Slave
      - EthTSynSwitchTimeSyncMechanism
        - EthTSynRateRegulator (6)
    - SwitchTimeSyncCfg_Switch_01
    - SwitchTimeSyncCfg_Switch_02_02_Slave
    - SwitchTimeSyncCfg_Switch_03_03_Slave
    - SwitchTimeSyncCfg_Switch_04_04_Slave
    - SwitchTimeSyncCfg_Switch_05_05_Slave
    - SwitchTimeSyncCfg_Switch_06
    - SwitchTimeSyncCfg_Switch_07
    - SwitchTimeSyncCfg_Switch_08

Short Name:	SwitchTimeSyncCfg_Switch_00_00_Slave	
Switch Time Sync Apply Rate Ratio To Slave Switches:	✓	1
Switch Time Sync Eth If Master Switch Ref:	/ActiveEcuC/EthIf/EthIfConfigSet/EthIfSwitch_00	2
Switch Time Sync Index:	0	3
Switch Time Sync Max Out Of Sync Count:	3	4
Switch Time Sync Sync Event Timeout [s]:	0.02	4
Switch Time Sync Sync Precision Limit [s]:	0.00001	4
Switch Time Sync Sync State Change Callback:	Appl_SwtSyncStateChgCbk	4
Switch Time Sync Time Domain Ref:	/ActiveEcuC/EthTSyn/TimeDomain_00	4

Switch Time Sync Eth If Slave Switch Ref (5)
/ActiveEcuC/EthIf/EthIfConfigSet/EthIfSwitch_Slave_00

Figure 5-1 Configuration of switch time synchronization


Basic Editor

« EthTSyn ▸ EthTSynSwitchTimeSyncConfigs ▸ SwitchTimeSyncCfg_Switch_0 ▸ EthTSynSwitchTimeSyncMechanism ▸ EthTSynRateRegulator

<Filter>
- BswM
- ComM
- Crc
- Dem
- Det
- EcuC
- EcuM
- Eth
- Eth_Generic
- EthIf
- EthSM
- EthSwt
- EthTrcv
- EthTSyn
  - EthTSynGlobalTimeDomains
  - EthTSynSwitchTimeSyncConfigs
    - SwitchTimeSyncCfg_Switch_0
      - EthTSynSwitchTimeSyncMechanism
        - EthTSynRateRegulator
    - SwitchTimeSyncCfg_Switch_1_1_Slave
    - SwitchTimeSyncCfg_Switch_2_2_Slave
    - SwitchTimeSyncCfg_Switch_3_3_Slave
    - SwitchTimeSyncCfg_Switch_4_4_Slave
    - SwitchTimeSyncCfg_Switch_5
    - SwitchTimeSyncCfg_Switch_6
    - SwitchTimeSyncCfg_Switch_7
    - SwitchTimeSyncCfg_Switch_8
  - EthTSynGeneral

Short Name:	EthTSynRateRegulator	
Number Of Parallel Rate Measurements:	3	1
Number Of Sync Cycles For Offset Correction:	3	2
Offset Jump Correction Threshold [s]:		3
Rate Ratio Max:	1.1	4
Rate Ratio Min:	0.9	4
Use Offset Correction For Initial Correction:	✓	5

Figure 5-2 Configuration of rate regulator

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
Configurator Pro	DaVinci Configurator Pro 5 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
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
TSN	Time-Sensitive Networking

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

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

www.vector.com