| Document Title | Specification of Ethernet State Manager |
|---|---|
| **Document Owner** | AUTOSAR |
| **Document Responsibility** | AUTOSAR |
| **Document Identification No** | 415 |
| | |
| **Document Status** | published |
| **Part of AUTOSAR Standard** | Classic Platform |
| **Part of Standard Release** | R21-11 |

| Document Change History | | | |
|---|---|---|---|
| **Date** | **Release** | **Changed by** | **Change Description** |
| 2021-11-25 | R21-11 | AUTOSAR Release Management | • Update state machine behaviour in “ETHSM_STATE_ONLINE” and “ETHSM_STATE_WAIT_OFFLINE”<br>• Editorial changes |
| 2020-11-30 | R20-11 | AUTOSAR Release Management | • Reworked the statemachine specification of EthSM<br>• Added handling for request of “COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST”<br>• Cleaned up the document structure |
| 2019-11-28 | R19-11 | AUTOSAR Release Management | • No content changes<br>• Changed Document Status from Final to published |
| 2018-10-31 | 4.4.0 | AUTOSAR Release Management | • Error classification has been fixed<br>• Editorial changes |
| 2017-12-08 | 4.3.1 | AUTOSAR Release Management | • Default error is removed<br>• Editorial changes |
| 2016-11-30 | 4.3.0 | AUTOSAR Release Management | • Remove Set and Get Transceiver mode functionality<br>• Correct EthSM_TcpIpModeIndication callback return value<br>• Harmonize main function period with the other modules<br>• Remove Get current internal mode |

# Document Change History

| Date | Release | Changed by | Change Description |
|---|---|---|---|
| 2015-07-31 | 4.2.2 | AUTOSAR Release Management | • Harmonize Sequence diagrams, Network State Machine and Functional Description<br>• Debugging support marked as obsolete<br>• Report to DET if TcpIp state is not accepted<br>• Adaptations related to renaming of DET,<br>• Error Handling: tables for Runtime Errors and Transient Faults added |
| 2014-10-31 | 4.2.1 | AUTOSAR Release Management | • Change from Synchronous to Asynchronous API<br>• Additional callback functions added<br>• Existing behavior of functions changes<br>• Editorial changes |
| 2014-03-31 | 4.1.3 | AUTOSAR Release Management | • Corrective action after timeout<br>• Non mutually exclusive transitions from ETHSM_STATE_ONLINE<br>• Editorial changes |
| 2013-10-31 | 4.1.2 | AUTOSAR Release Management | • Optimization of full com request<br>• Standardization of internal state names<br>• Asynchronous behavior of several interfaces<br>• Several clarifications and corrections<br>• Editorial changes<br>• Removed chapter(s) on change documentation |
| 2013-03-15 | 4.1.1 | AUTOSAR Administration | • New State Machine (new sub states and new state conditions, new APIs)<br>• Update chapter 10<br>• Added Production Error if Transceiver Link is down<br>• General Update (corrections and formulations) |

| Document Change History | | | |
|---|---|---|---|
| **Date** | **Release** | **Changed by** | **Change Description** |
| 2011-12-22 | 4.0.3 | AUTOSAR Administration | • Update Chapter 10 (Parameter adjustment) |
| 2010-09-30 | 3.1.5 | AUTOSAR Administration | • Functional changes:<br>• Correction of the naming convention of SW modul version information<br>• Correction of chapter 10 - configuration parameter "EthSMNetworkIndex"<br>• Remove InstanceID from GetVersionId structure<br>• Additional callback function: Call of SoAd_BusSM_ModeIndication realized after the successful initialization of the EthTrcv and the EthController.<br>• Non functional changes:<br>• Adding a self loop with "No initialization" in the state diagramm |
| 2010-02-02 | 3.1.4 | AUTOSAR Administration | • Initial Release |

## Disclaimer

# Table of Contents

# 1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Ethernet State Manager.

In the AUTOSAR Layered Software Architecture, the Ethernet State Manager belongs to the ECU Abstraction Layer, or more precisely, to the Communication Hardware Abstraction.

The main task of the Ethernet State Manager can be summarized as follows:

[SWS_EthSM_00001]
⌈The Ethernet State Manager shall provide an abstract interface to the AUTOSAR Communication Manager to startup or shutdown the communication on an Ethernet cluster. ⌋()

[SWS_EthSM_00002]
⌈The Ethernet State Manager does not directly access the Ethernet hardware (Ethernet Communication Controller and Ethernet Transceiver), but by means of the Ethernet Interface. The Ethernet Interface redirects the request to the appropriate driver module.⌋()

This is an example of an Autosar architecture including an Ethernet network.

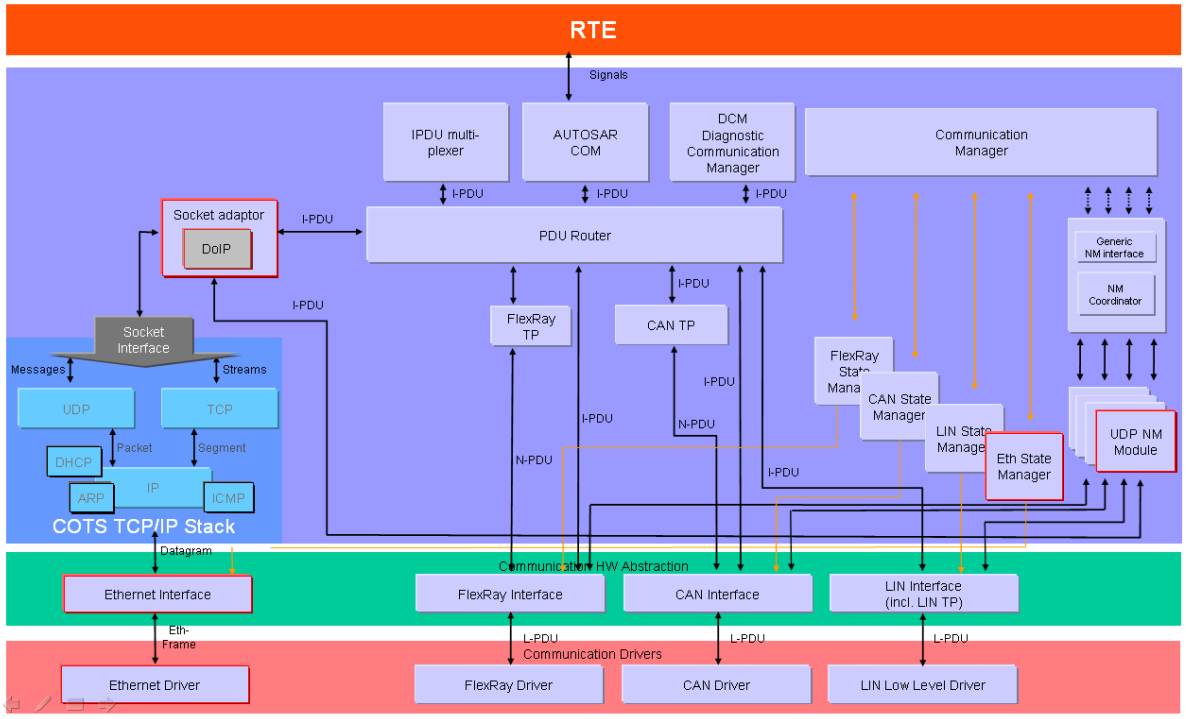



**Figure 1-1: Example of an Autosar architecture including an Ethernet network**

# 2 Acronyms and abbreviations

| *Abbreviation / Acronym:* | *Description:* |
|---|---|
| API | Application Program Interface |
| BSW | Basic Software |
| BswM | Basic Software Mode Manager |
| ComM | Communication Manager |
| DEM | Diagnostic Event Manager |
| DET | Default Error Tracer |
| EcuM | ECU State Manager |
| Eth | Ethernet Controller |
| EthTrcv | Ethernet Transceiver |
| EthSM | Ethernet State Manager |
| EthIf | Ethernet Interface |
| SchM | BSW Scheduler |
| SoAd | Socket Adapter |
| OA TC10 | Open Alliance TC10 specification (see [19]) |

# 3 Related documentation

## 3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of AUTOSAR COM
AUTOSAR_SWS_COM.pdf

[5] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[6] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[7] Specification of Communication Manager
AUTOSAR_SWS_ComManager.pdf

[8] Requirements on Mode Management
AUTOSAR_SRS_ModeManagement.pdf

[9] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[10] Specification of the Ethernet Interface
AUTOSAR_SWS_EthernetInterface.pdf

[11] Requirements on Ethernet in AUTOSAR
AUTOSAR_SRS_Ethernet.pdf

[12] Specification of Standard Types
AUTOSAR_SWS_StandardTypes

[13] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager.pdf

[14] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

[15] Specification of Basic Software Mode Manager
AUTOSAR_SWS_BSWModeManager.pdf

[16] Specification of Basic Software Mode Manager
AUTOSAR_SWS_SocketAdapter.pdf

[17] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

[18] Specification of TcpIp module
AUTOSAR_SWS_TcpIp.pdf

## 3.2 Standard and norms

[19] OPEN ALIANCE Sleep/Wake-up Specification Version 2.0 (Rel Feb 21, 2017), http://www.opensig.org/Automotive-Ethernet-Specifications/

## 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [17] (SWS BSW General), which is also valid for Ethernet State Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Ethernet State Manager.

# 4 Constraints and assumptions

## 4.1 Limitations

The EthSM can be used for Ethernet communication only. Its dedication is to operate with the EthIf to control one or multiple underlying Ethernet Controllers and Ethernet Transceiver Drivers. Other protocols than Ethernet (i.e. CAN, LIN or FlexRay) are not supported.

The following items are not supported by the current version of this specification:

- Wake on LAN. (Please note, wake-up and sleep on dataline according to the OA TC10 specification (see [19]) is supported)

The actual EthSM requires an IP-based communication stack. To get FULL_COMMUNICATION it is necessary to get an active IP communication. In further specifications, an alternative “low level” state machine will be introduced. This state machine only works on driver/transceiver level (without IP communication). This is necessary to realize other communication protocols (e.g. IEEE 1722).

## 4.2 Applicability to car domains

The Ethernet State Manager can be used for all domain applications always when the Ethernet protocol is used. The Ethernet BSW Stack can be used wherever high data rates are required.

# 5 Dependencies to other modules

**AUTOSAR BSW Scheduler**
The BSW Scheduler calls the main functions of the EthSM, which are necessary for the cyclic processes of the EthSM.

**AUTOSAR Communication Manager**
The ComM requests network communication modes and is notified by the EthSM when a communication mode is reached.

**AUTOSAR Ethernet Interface**
The EthSM uses the API of the EthIf to initialize the Ethernet Communication Hardware and to control the operating modes of the Ethernet Controllers and Ethernet Transceivers assigned to the Ethernet Networks.
The Ethernet Interface uses the API of the EthSM to provide the transceiver link state.

**AUTOSAR Default Error Tracer**
In order to be able to report development errors, the Ethernet State Manager has to have access to the error hook of the Default Error Tracer.

**AUTOSAR Diagnostic Event Manager**
In order to be able to report production errors the Ethernet State Manager has to have access to the Diagnostic Event Manager.

**ECU State Manager**
The EcuM initializes the EthSM.

**AUTOSAR Bsw Manager**
The BswM is notified by the EthSM when an internal state is reached.

**AUTOSAR TcpIp**
TcpIp is called to request the TCPIP state (e.g. Online, Offline, On Hold, …).
TcpIP uses the API of the EthSM to provide the TCPIP state.

## 5.1 File structure

### 5.1.1 Code file structure

For details refer to the chapter 5.1.6 “Code File Structure” in SWS_BSWGeneral.

Remark:
Actually the module EthSM doesn´t provide link time configuration and post-build time configuration.

[SWS_EthSM_00008] ⌈
The EthSM needs to report development errors if development errors are enabled by configuration. Therefore, it includes the header file Det.h.⌋()

[SWS_EthSM_00010] ⌈
The EthSM implementation (EthSM.c) references the API of the EthIf. Therefore, it includes the header file EthIf.h.⌋()

[SWS_EthSM_00013] ⌈
The EthSM module shall include the ComM_Bus_SM.h header file. ⌋()

### 5.1.2 Version Check

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

# 6 Requirements traceability

| Requirement | Description | Satisfied by |
|---|---|---|
| SRS_BSW_00003 | All software modules shall provide version and identification information | SWS_EthSM_00046, SWS_EthSM_00060 |
| SRS_BSW_00005 | Modules of the ÂµC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces | SWS_EthSM_00999 |
| SRS_BSW_00010 | The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms. | SWS_EthSM_00999 |
| SRS_BSW_00101 | The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function | SWS_EthSM_00043 |
| SRS_BSW_00159 | All modules of the AUTOSAR Basic Software shall support a tool based configuration | SWS_EthSM_00081 |
| SRS_BSW_00160 | Configuration files of AUTOSAR Basic SW module shall be readable for human beings | SWS_EthSM_00999 |
| SRS_BSW_00161 | The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers | SWS_EthSM_00999 |
| SRS_BSW_00162 | The AUTOSAR Basic Software shall provide a hardware abstraction layer | SWS_EthSM_00999 |
| SRS_BSW_00164 | The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules | SWS_EthSM_00999 |
| SRS_BSW_00168 | SW components shall be tested by a function defined in a common API in the Basis-SW | SWS_EthSM_00999 |
| SRS_BSW_00170 | The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands | SWS_EthSM_00999 |
| SRS_BSW_00306 | AUTOSAR Basic Software Modules shall be compiler and platform independent | SWS_EthSM_00999 |
| SRS_BSW_00308 | AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file | SWS_EthSM_00999 |
| SRS_BSW_00309 | All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword | SWS_EthSM_00999 |
| SRS_BSW_00314 | All internal driver modules shall separate the interrupt frame definition from the service routine | SWS_EthSM_00999 |
| SRS_BSW_00318 | Each AUTOSAR Basic Software Module file shall provide version numbers in the header file | SWS_EthSM_00060 |

| | | |
|---|---|---|
| SRS_BSW_00321 | The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules | SWS_EthSM_00999 |
| SRS_BSW_00325 | The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short | SWS_EthSM_00999 |
| SRS_BSW_00328 | All AUTOSAR Basic Software Modules shall avoid the duplication of code | SWS_EthSM_00999 |
| SRS_BSW_00331 | All Basic Software Modules shall strictly separate error and status information | SWS_EthSM_00999 |
| SRS_BSW_00333 | For each callback function it shall be specified if it is called from interrupt context or not | SWS_EthSM_00999 |
| SRS_BSW_00334 | All Basic Software Modules shall provide an XML file that contains the meta data | SWS_EthSM_00999 |
| SRS_BSW_00336 | Basic SW module shall be able to shutdown | SWS_EthSM_00999 |
| SRS_BSW_00341 | Module documentation shall contains all needed informations | SWS_EthSM_00999 |
| SRS_BSW_00343 | The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit | SWS_EthSM_00999 |
| SRS_BSW_00344 | BSW Modules shall support link-time configuration | SWS_EthSM_00999 |
| SRS_BSW_00347 | A Naming seperation of different instances of BSW drivers shall be in place | SWS_EthSM_00999 |
| SRS_BSW_00353 | All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header | SWS_EthSM_00999 |
| SRS_BSW_00358 | The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void | SWS_EthSM_00043 |
| SRS_BSW_00359 | All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible | SWS_EthSM_00999 |
| SRS_BSW_00360 | AUTOSAR Basic Software Modules callback functions are allowed to have parameters | SWS_EthSM_00999 |
| SRS_BSW_00361 | All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header | SWS_EthSM_00999 |
| SRS_BSW_00369 | All AUTOSAR Basic Software Modules shall not return specific development error codes via the API | SWS_EthSM_00999 |
| SRS_BSW_00371 | - | SWS_EthSM_00999 |
| SRS_BSW_00373 | The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention | SWS_EthSM_00999 |
| SRS_BSW_00374 | All Basic Software Modules shall provide a readable module vendor identification | SWS_EthSM_00060 |
| SRS_BSW_00375 | Basic Software Modules shall report wake-up | SWS_EthSM_00999 |

| | | |
|---|---|---|
| | reasons | |
| SRS_BSW_00377 | A Basic Software Module can return a module specific types | SWS_EthSM_00999 |
| SRS_BSW_00395 | The Basic Software Module specifications shall list all configuration parameter dependencies | SWS_EthSM_00999 |
| SRS_BSW_00398 | The link-time configuration is achieved on object code basis in the stage after compiling and before linking | SWS_EthSM_00999 |
| SRS_BSW_00399 | Parameter-sets shall be located in a separate segment and shall be loaded after the code | SWS_EthSM_00999 |
| SRS_BSW_00400 | Parameter shall be selected from multiple sets of parameters after code has been loaded and started | SWS_EthSM_00999 |
| SRS_BSW_00404 | BSW Modules shall support post-build configuration | SWS_EthSM_00999 |
| SRS_BSW_00405 | BSW Modules shall support multiple configuration sets | SWS_EthSM_00043 |
| SRS_BSW_00406 | A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called | SWS_EthSM_00054, SWS_EthSM_00060, SWS_EthSM_00115, SWS_EthSM_00120 |
| SRS_BSW_00407 | Each BSW module shall provide a function to read out the version information of a dedicated module implementation | SWS_EthSM_00046 |
| SRS_BSW_00413 | An index-based accessing of the instances of BSW modules shall be done | SWS_EthSM_00999 |
| SRS_BSW_00414 | Init functions shall have a pointer to a configuration structure as single parameter | SWS_EthSM_00043 |
| SRS_BSW_00416 | The sequence of modules to be initialized shall be configurable | SWS_EthSM_00999 |
| SRS_BSW_00417 | Software which is not part of the SW-C shall report error events only after the DEM is fully operational. | SWS_EthSM_00999 |
| SRS_BSW_00423 | BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template | SWS_EthSM_00999 |
| SRS_BSW_00424 | BSW module main processing functions shall not be allowed to enter a wait state | SWS_EthSM_00081 |
| SRS_BSW_00425 | The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects | SWS_EthSM_00081 |
| SRS_BSW_00426 | BSW Modules shall ensure data consistency of data which is shared between BSW modules | SWS_EthSM_00999 |
| SRS_BSW_00427 | ISR functions shall be defined and documented in the BSW module description template | SWS_EthSM_00999 |
| SRS_BSW_00428 | A BSW module shall state if its main | SWS_EthSM_00999 |

| | | |
|---|---|---|
| | processing function(s) has to be executed in a specific order or sequence | |
| SRS_BSW_00429 | Access to OS is restricted | SWS_EthSM_00999 |
| SRS_BSW_00432 | Modules should have separate main processing functions for read/receive and write/transmit data path | SWS_EthSM_00999 |
| SRS_BSW_00433 | Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler | SWS_EthSM_00999 |
| SRS_BSW_00437 | Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup | SWS_EthSM_00999 |
| SRS_BSW_00438 | Configuration data shall be defined in a structure | SWS_EthSM_00999 |
| SRS_Eth_00158 | The Ethernet state manager shall trigger requested modes for Ethernet hardware with wake-up capability even if the requested mode has already been reached. | SWS_EthSM_00206, SWS_EthSM_00207, SWS_EthSM_00208, SWS_EthSM_00212 |
| SRS_Eth_00159 | The Ethernet state manager shall forward sleep requests indicated by the maintained Ethernet hardware that act as communication slaves to the responsible upper layer. | SWS_EthSM_00213, SWS_EthSM_00214, SWS_EthSM_00215, SWS_EthSM_91004 |

# 7 Functional specification

An ECU can have different communication networks. Each network has to be identified with a unique network handle. The ComM requests communication modes from the networks. It knows by its configuration, which handle is assigned to what kind of network. In case of Ethernet, it uses the Ethernet state manager, which is responsible for the control flow abstraction of Ethernet networks. The following sections describe this in detail.

## 7.1 Translation of network communication mode requests

[SWS_EthSM_00014] ⌈
The EthSM shall provide to the ComM an API, which can be used by the ComM to request communication modes of Ethernet networks.⌋()

[SWS_EthSM_00015]⌈
Depending on the parameters handed over by this API, the EthSM shall execute a state transition of the related network mode state machine (refer to section 7.6). ⌋()

[SWS_EthSM_00016]⌈
This transition shall translate the request into a respective API call to control the assigned Ethernet peripherals.⌋()

## 7.2 Output of current network communication modes

The current communication mode of a network can be different from the requested mode. The EthSM has to provide the information on the current communication mode to the ComM by the two following kind of interfaces:

[SWS_EthSM_00017]⌈
The EthSM shall provide an API, which can be polled by the ComM to get the current communication mode of an Ethernet network. ⌋()

[SWS_EthSM_00018]⌈
The EthSM shall use a call-back notification of ComM to notify ComM of a change in communication modes.⌋()

## 7.3 Control of peripherals

### 7.3.1 Ethernet Interface Controllers

One Ethernet Interface Controllers and one ComM channel belong to one certain Ethernet network (handle).

[SWS_EthSM_00022]⌈
Depending on the network mode state machine, the EthSM shall control the Ethernet Interface Controller modes of each Ethernet network. ⌋()

[SWS_EthSM_00023]⌈
The EthSM shall use the API of the EthIf to control the operating modes of the assigned Ethernet Interface Controllers.⌋()

## 7.4 Multiple networks

The Ethernet State Manager shall be able to handle separate networks. This concerns separate physical networks (see also chapter 7.3) and also separate VLAN´s on the same physical network.
In both cases, the separation is done by separate handles per physical or virtual network. VLANs appear on higher layers (ComM) as separate networks. E.g.: If there is one physical Ethernet Interface Controller and two VLANs assigned to it, two ComM channels exists.

## 7.5 Background and Rationale

Explanation:
The application is responsible to recognize if the Ethernet network is needed or not.

One possible use case could be the usage of the Ethernet network in a tester connection (see description below).

Use Case: Use Ethernet in a tester connection
For example, the detection could takes place over a separate hardware pin of the ECU. In this case, the activation of the hardware pin and therefore the activation of the Ethernet network can only realized through the offboard-diagnostic tester. Reasons for the deactivation of the Ethernet network could be:

- The tester deactivate via the separate hardware pin the network
- The application deactivate the network
- The application recognize a timeout
- The link status of the network failed

Note:

The Ethernet network is requested by the application to wake-up or shutdown. The applicaction either need to communicate or do not need to communicate on the Ethernet network. Therefore, the EthSM statemashine has to provide the corresponding main states ETHSM_FULL_COMMUNICATION and ETHSM_NO_COMMUNICATION, but no other states, e.g. ETHSM_SILENT_COMMUNICATION.

## 7.6 Network mode state machine

[SWS_EthSM_00024]⌈
The EthSM shall implement for each configured network handle one network mode state machine (EthSM statemachine).⌋()

[SWS_EthSM_00203] ⌈
The EthSM statemachine shall consist of 2 main states: ETHSM_FULL_COMMUNICATION and ETHSM_NO_COMMUNICATION.⌋()

[SWS_EthSM_00204] ⌈
The ETHSM_FULL_COMMUNICATION shall have 3 sub-states: ETHSM_STATE_OFFLINE, ETHSM_STATE_WAIT_TRCVLINK and ETHSM_STATE_WAIT_ONLINE.⌋()

[SWS_EthSM_00205] ⌈
The ETHSM_NO_COMMUNICATION shall have 3 sub-states: ETHSM_STATE_ONLINE, ETHSM_STATE_ONHOLD and ETHSM_STATE_WAIT_OFFLINE.⌋()

The following points give an overview, on how the EthSM statemachine generally reacts on requested communication modes:

1. If communication mode COMM_FULL_COMMUNICATION is requested for a certain Ethernet network and the corresponding EthSM statemachine is in state ETHSM_NO_COMMUNICATION, then at least the following preconditions have to be fulfilled to transit to ETHSM_FULL_COMMUNICATION:
    - Request ETH_MODE_ACTIVE for the corresponding Ethernet Interface Controller.
    - The corresponding Ethernet hardware has indicated ETHTRCV_LINK_STATE_ACTIVE (active link state, thus, Ethernet cable is connected) via EthSM_TrcvLinkStateChg
    - The corresponding TcpIpCtrl has indicated TCPIP_STATE_ONLINE (IP communication is available) via EthSM_TcpIpModeIndication
2. If communication mode COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST is requested for a certain Ethernet network, EthSMWakeupSleepOnDatalineEnabled is set to TRUE and the corresponding EthSM statemachine is in state ETHSM_NO_COMMUNICATION, then at least the following preconditions have to be fulfilled to transit to ETHSM_FULL_COMMUNICATION:

- Request ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST for the corresponding Ethernet Interface Controller.
- The corresponding Ethernet hardware has indicated ETHTRCV_LINK_STATE_ACTIVE (active link state, thus, Ethernet cable is connected) via EthSM_TrcvLinkStateChg
- The corresponding TcpIpCtrl has indicated TCPIP_STATE_ONLINE (IP communication is available) via EthSM_TcpIpModeIndication

3. If ComM request COMM_SILENT_COMMUNICATION and the corresponding EthSM state machine is in state ETHSM_FULL_COMMUNICATION, then the following actions will be performed:
   - EthSM request ETH_MODE_ACTIVE_TX_OFFLINE for the corresponding Ethernet Interface Controller, if forwarding of silent communication is enabled (see ECUC_EthSM_00110)
   - EthSM state machine transit to ETHSM_STATE_WAIT_OFFLINE
4. If the ComM requests COMM_NO_COMMUNICATION and the corresponding EthSM state machine is in state ETHSM_STATE_WAIT_OFFLINE, then the following preconditions have to be fulfilled to transist to ETHSM_NO_COMMUNICATION:
   - EthSM request ETH_MODE_DOWN for the corresponding Ethernet Interface Controller
   - The corresponding TcpIpCtrl has indicated TCPIP_STATE_OFFLINE
5. If communication mode COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST is requested for a certain Ethernet network, EthSMWakeupSleepOnDatalineEnabled is set to TRUE and the corresponding EthSM statemachine is in state ETHSM_FULL_COMMUNICATION, then the EthSM request ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST for the corresponding Ethernet Interface Controller, even though the current state is ETHSM_FULL_COMMUNICATION

The call of silent communication in point 3 will disable the transmission path of the corresponding Ethernet Interface Controller, if the forwarding of the request is configured (see ECUC_EthSM_00110)

The re-trigger of the Ethernet Interface Controller described in point 5. in the previous section is used to request the lower layer to trigger a wake-up on the network. This could be used for example for Ethernet hardware which is compliant with OA TC10 (see [19])

The internal states are shown in Figure 7-1. Details are described in the following sub chapters. An overview of the internal states are described in [SWS_EthSM_00041].

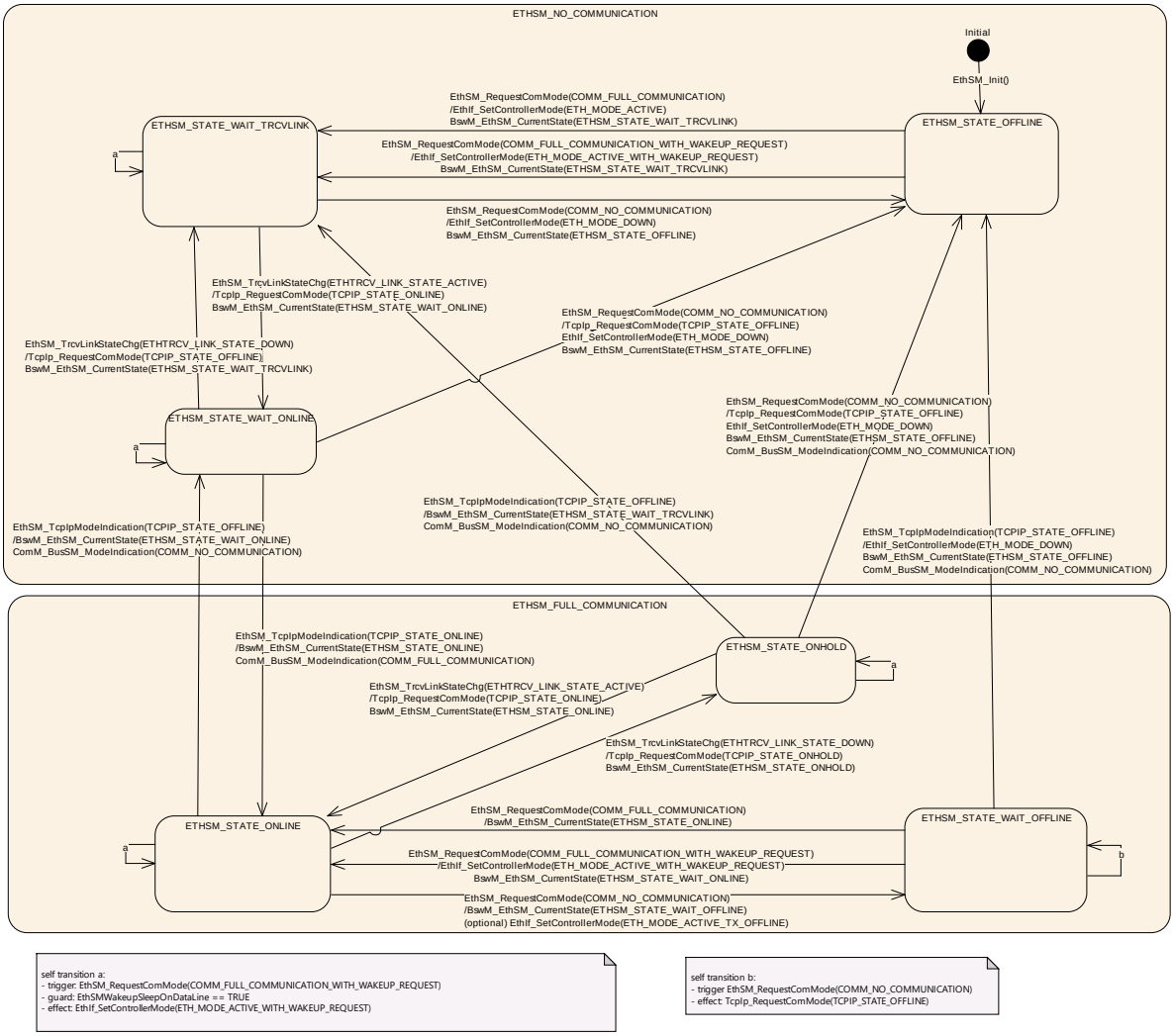



**Figure 7-1: Network mode state machine of the EthSM**

Table 1 shows the expected state of the EthSM sub state, the state of the affected Ethernet hardware and the IP assignment, according to the requested communication mode of the ComM.

| Requested ComM Mode | EthSM sub state | Controller state | Transceiver state | Transceiver link state | TcpIpCtrl state |
|---|---|---|---|---|---|
| | | | | | |
| No Com | ETHSM_STATE_OFFLINE | DOWN | DOWN | DOWN | OFFLINE |
| | | | | | |
| No Com | ETHSM_STATE_WAIT_TRCVLINK | ACTIVE | ACTIVE | DOWN | OFFLINE |
| | | | | | |
| No Com | ETHSM_STATE_WAIT_ONLINE | ACTIVE | ACTIVE | ACTIVE | OFFLINE |
| | | | | | |
| Full Com | ETHSM_STATE_ONLINE | ACTIVE | ACTIVE | ACTIVE | ONLINE |
| | | | | | |
| Full Com | ETHSM_STATE_ONHOLD | ACTIVE | ACTIVE | DOWN | ONLINE |
| | | | | | |
| Silent Com | ETHSM_STATE_WAIT_OFFLINE | ACTIVE or ACTIVE_ TX_OFFLINE *) | ACTIVE | ACTIVE | ONLINE |
| No Com | ETHSM_STATE_WAIT_OFFLINE | ACTIVE or ACTIVE_ TX_OFFLINE *) | ACTIVE | ACTIVE | ONLINE |

**Table 1 EthSM sub states and the excepted states of the according Ethernet hardware and the IP assignment. *) Please note, requested mode (either ACTIVE or ACTIVE_TX_OFFLINE) is configurable (see ECUC_EthSM_00110)**

The following preconditions have to be fulfilled to transit to ETHSM_FULL_COMMUNICATION:

- Ethernet Interface Controller and transceiver are active
- The transceiver link state is active
- An active IP communication is available

The first step is set the Ethernet controller to ETH_MODE_ACTIVE. After this is done, the Ethernet State Manager is in the sub state ETHSM_STATE_WAIT_TRCVLINK. In this sub state the state manager has to wait for the monitored link state information of the Ethernet transceiver. After the link state is set to ETHTRCV_MODE_ACTIVE, the Ethernet State Manager transit to sub state ETHSM_STATE_WAIT_ONLINE. In this sub state the Ethernet state manager has to wait for the monitored TcpIp stateinformation of the TcpIp module. After the TcpIp module indicate TCPIP_STATE_ONLINE (= IP communication is available), the Ethernet State Manager transit to main state ETHSM_ FULL_COMMUNICATION and sub state ETHSM_STATE_ONLINE.

### 7.6.1 Initial transition

[SWS_EthSM_00025] ⌜
After the initialization of the EthSM the state machine shall have a transition to ETHSM_STATE_OFFLINE.⌟()

Note: The initialization of the EthSM causes no further transactions in other modules. So no separate sequence diagram is needed.

### 7.6.2 Behaviour in sub-state ETHSM_STATE_OFFLINE

#### 7.6.2.1 Transition from sub-state OFFLINE to WAIT_TRCVLINK

[SWS_EthSM_00026] ⌈
In the state ETHSM_STATE_OFFLINE the state machine shall have a transition to ETHSM_STATE_WAIT_TRCVLINK, if ComM requests COMM_FULL_COMMUNICATION or COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST for the corresponding network handle.⌋()

Note: The transition is shown as sequence diagram in Figure 9-1.

[SWS_EthSM_00088] ⌜
The transition from ETHSM_STATE_OFFLINE to ETHSM_STATE_WAIT_TRCVLINK shall set the controller mode according to the requested ComM communication mode:

- If ComM requested COMM_FULL_COMMUNICATION, then the controller mode of the corresponding EthIfCtrl shall be requested with ETH_MODE_ACTIVE
- If ComM requested COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST, then the controller mode of corresponding EthIfCtrl shall be requested with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST

⌟()

### 7.6.3 Behavior in sub-state ETHSM_STATE_WAIT_TRCVLINK

[SWS_EthSM_00206] ⌈
If EthSMWakeupSleepOnDatalineEnabled is set to TRUE and the EthSM statmachine is in state ETHSM_STATE_WAIT_TRCVLINK, and ComM request now COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST, then EthSM shall request the corresponding EthIfCtrl with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST.⌋(SRS_Eth_00158)

#### 7.6.3.1 Transition from sub-state WAIT_TRCVLINK to WAIT_OFFLINE

[SWS_EthSM_00127] ⌈

In the state ETHSM_STATE_WAIT_TRCVLINK the state machine shall have a transition to ETHSM_STATE_OFFLINE, if the ComM requests COMM_NO_COMMUNICATION for the corresponding network handle.⌋()

[SWS_EthSM_00128]⌈
The transition from ETHSM_STATE_WAIT_TRCVLINK to ETHSM_STATE_OFFLINE shall set the controller mode to ETH_MODE_DOWN. ⌋()

#### 7.6.3.2 Transition from sub-state WAIT_TRCVLINK to WAIT_ONLINE

[SWS_EthSM_00132] ⌈
In the state ETHSM_STATE_WAIT_TRCVLINK the state machine shall have a transition to ETHSM_STATE_WAIT_ONLINE, if the Ethernet Interface reports ETHTRCV_LINK_STATE_ACTIVE for the corresponding network handle.⌋()

Note: The transition is shown as sequence diagram in Figure 9-1.

[SWS_EthSM_00133]⌈
The transition from ETHSM_STATE_WAIT_TRCVLINK to ETHSM_STATE_WAIT_ONLINE shall request the TcpIp state TCPIP_STATE_ONLINE from the TcpIp module.⌋()

### 7.6.4 Behaviour in sub-state ETHSM_STATE_WAIT_ONLINE

[SWS_EthSM_00207] ⌈
If EthSMWakeupSleepOnDatalineEnabled is set to TRUE and EthSM is in state ETHSM_STATE_WAIT_ONLINE and ComM requests now COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST, then EthSM shall request the corresponding EthIfCtrl with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST.⌋(SRS_Eth_00158)

#### 7.6.4.1 Transition from sub-state WAIT_ONLINE to WAIT_TRCVLINK

[SWS_EthSM_00136] ⌈
In the state ETHSM_STATE_WAIT_ONLINE the state machine shall have a transition to ETHSM_STATE_WAIT_TRCVLINK, if the Ethernet interface reports ETHTRCV_LINK_STATE_DOWN for the corresponding network handle.⌋()

[SWS_EthSM_00137]⌈
The transition from ETHSM_STATE_WAIT_ONLINE to ETHSM_STATE_WAIT_TRCVLINK shall request the TcpIp state TCPIP_STATE_OFFLINE from the TcpIp module.⌋()

#### 7.6.4.2 Transition from sub-state WAIT_ONLINE to OFFLINE

[SWS_EthSM_00140] ⌈

In the state ETHSM_STATE_WAIT_ONLINE the state machine shall have a transition to ETHSM_STATE_OFFLINE, if the ComM requests COMM_NO_COMMUNICATION for the corresponding network handle.⌋()

[SWS_EthSM_00141] ⌈
The transition from ETHSM_STATE_WAIT_ONLINE to ETHSM_STATE_OFFLINE sets the controller mode to ETH_MODE_DOWN. ⌋()

[SWS_EthSM_00143] ⌈
The transition from ETHSM_STATE_WAIT_ONLINE to ETHSM_STATE_OFFLINE shall request the TcpIp state TCPIP_STATE_OFFLINE from the TcpIp module.⌋()

#### 7.6.4.3 Transition from sub-state WAIT_ONLINE and ONLINE

[SWS_EthSM_00146] ⌈
In the state ETHSM_STATE_WAIT_ONLINE the state machine shall have a transition to ETHSM_STATE_ONLINE, if the TcpIp modul reports TCPIP_STATE_ONLINE for the corresponding network handle.⌋()

Note: The transition is shown as sequence diagram in Figure 9-1.

[SWS_EthSM_00150]⌈
After the successful transition from ETHSM_STATE_WAIT_ONLINE to ETHSM_STATE_ONLINE the Ethernet State Manager shall set the main state to ETHSM_FULL_COMMUNICATION and call the callback function ComM_BusSM_ModeIndication of the ComM with communication mode COMM_FULL_COMMUNICATION.⌋()

### 7.6.5 Behaviour in sub-state ETHSM_STATE_ONLINE

[SWS_EthSM_00208] ⌈
⌈If the EthSM statmachine is in state ETHSM_STATE_ONLINE, EthSMWakeupSleepOnDatalineEnabled is set to TRUE and ComM request COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST, then the controller mode of the corresponding EthIfCtrl shall be requested with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST.⌋(SRS_Eth_00158)

#### 7.6.5.1 Transition from sub-state ONLINE to WAIT_ONLINE

[SWS_EthSM_00151] ⌈
In the state ETHSM_STATE_ONLINE the state machine shall have a transition to ETHSM_STATE_WAIT_ONLINE, if the TcpIp modul reports TCPIP_STATE_OFFLINE for the corresponding network handle.⌋()

[SWS_EthSM_00154] ⌈
After the successful transition from ETHSM_STATE_ONLINE to ETHSM_STATE_WAIT_ONLINE the Ethernet State Manager shall set the main state to ETHSM_NO_COMMUNICATION and call the callback function

ComM_BusSM_ModeIndication of the ComM with communication mode COMM_NO_COMMUNICATION.⌋()

#### 7.6.5.2 Transition from sub-state ONLINE to WAIT_OFFLINE

[SWS_EthSM_00155] {OBSOLETE replaced by [SWS_EthSM_00216]}⌈
In the state ETHSM_STATE_ONLINE the state machine shall have a transition to ETHSM_STATE_WAIT_OFFLINE, if the ComM requests COMM_NO_COMMUNICATION for the corresponding network handle.⌋()

[SWS_EthSM_00216] {DRAFT} ⌈
In the state ETHSM_STATE_ONLINE the state machine shall do a transition to ETHSM_STATE_WAIT_OFFLINE, if the ComM requests COMM_SILENT_COMMUNICATION for the corresponding network handle.⌋()

Note: The transition is shown as sequence diagram in Figure 9-2.

[SWS_EthSM_00157] {OBSOLETE replaced by [SWS_EthSM_00217]} ⌈
After entering the state ETHSM_STATE_WAIT_OFFLINE, the API TcpIp_RequestComMode shall be called with TCPIP_STATE_OFFLINE. ⌋()

[SWS_EthSM_00217] {DRAFT} ⌈
After entering the state ETHSM_STATE_WAIT_OFFLINE, the API EthIf_SetControllerMode shall be called with ETH_MODE_ACTIVE_TX_OFFLINE, if EthSMForwardSilentCommunicationEnabled is set to TRUE.⌋()

#### 7.6.5.3 Transition from sub-state ONLINE to ONHOLD

[SWS_EthSM_00166] ⌈
In the state ETHSM_STATE_ONLINE the state machine shall have a transition to ETHSM_STATE_ONHOLD, if the Ethernet Interface reports ETHTRCV_LINK_STATE_DOWN for the corresponding network handle. ⌋()

Note: The transition is shown as sequence diagram in Figure 9-3.

[SWS_EthSM_00167] ⌈
The transition from ETHSM_STATE_ONLINE to ETHSM_STATE_ONHOLD shall request the TcpIp state TCPIP_STATE_ONHOLD from the TcpIp module.⌋()

[SWS_EthSM_00188]⌈
If the optional configuration parameter ETHSM_E_LINK_DOWN exists, ETHSM_E_LINK_DOWN with EventStatus DEM_EVENT_STATUS_FAILED shall be reported to the DEM module when switching from ETHSM_STATE_ONLINE to ETHSM_STATE_ONHOLD.⌋()

### 7.6.6 Behaviour in sub-state ETHSM_STATE_WAIT_OFFLINE

[SWS_EthSM_00218] ⌈

In the state ETHSM_STATE_WAIT_OFFLINE, if ComM requests COMM_NO_COMMUNICATION for the corresponding network handle, EthSM shall call the API TcpIp_RequestComMode with TCPIP_STATE_OFFLINE.⌋()

#### 7.6.6.1 Transition from sub-state WAIT_OFFLINE to OFFLINE

[SWS_EthSM_00160] ⌈
In the state ETHSM_STATE_WAIT_OFFLINE the state machine shall have a transition to ETHSM_STATE_OFFLINE, if the TcpIp modul reports TCPIP_STATE_OFFLINE for the corresponding network handle.⌋()

[SWS_EthSM_00161] ⌈
The transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_OFFLINE shall set the controller mode to ETH_MODE_DOWN.⌋()

[SWS_EthSM_00165] ⌈
After the successful transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_OFFLINE the Ethernet State Manager shall set the main state to ETHSM_NO_COMMUNICATION and call the callback function ComM_BusSM_ModeIndication of the ComM with communication mode COMM_NO_COMMUNICATION.⌋()

#### 7.6.6.2 Transistion from sub-state WAIT_OFFLINE to ONLINE

[SWS_EthSM_00209] ⌈
In the state ETHSM_STATE_WAIT_OFFLINE the state machine shall have a transition to ETHSM_STATE_ONLINE, if ComM requests COMM_FULL_COMMUNICATION or COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST for the corresponding network handle.⌋()

Note: The transition is shown as sequence diagram in Figure 9-2.

[SWS_EthSM_00210] ⌈
The transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_ONLINE shall set the controller mode to ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST, if ComM requested the EthSM statemachine with COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST.⌋()

[SWS_EthSM_00219] DRAFT ⌈
The transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_ONLINE shall set the controller mode to ETH_MODE_ACTIVE, if ComM requested the EthSM statemachine with COMM_FULL_COMMUNICATION.⌋()

[SWS_EthSM_00211] ⌈
The transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_WAIT_ONLINE shall request the TcpIp state TCPIP_STATE_ONLINE from the TcpIp module.⌋()

### 7.6.7 Behaviour in sub-state ETHSM_STATE_ONHOLD

[SWS_EthSM_00212] ⌈
If the EthSM statmachine is in state ETHSM_STATE_ONHOLD, EthSMWakeupSleepOnDatalineEnabled is set to TRUE and ComM request COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST, then the controller mode of the corresponding EthIfCtrl shall be requested with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST.⌋(SRS_Eth_00158)

#### 7.6.7.1 Transistion from ONHOLD to ONLINE

[SWS_EthSM_00170] ⌈
In the state ETHSM_STATE_ONHOLD the state machine shall have a transition to ETHSM_STATE_ONLINE, if the Ethernet interface reports ETHTRCV_LINK_STATE_ACTIVE for the corresponding network handle.⌋()

[SWS_EthSM_00171] ⌈
The transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_ONLINE shall request the TcpIp state TCPIP_STATE_ONLINE from the TcpIp module.⌋()

[SWS_EthSM_00196]⌈
If the optional configuration parameter ETHSM_E_LINK_DOWN exists, ETHSM_E_LINK_DOWN with EventStatus DEM_EVENT_STATUS_PASSED shall be reported to the DEM module when switching from ETHSM_STATE_ONHOLD to ETHSM_STATE_ONLINE.⌋()

#### 7.6.7.2 Transition from sub-state ONHOLD to WAIT_TRCVLINK

[SWS_EthSM_00174] ⌈
In the state ETHSM_STATE_ONHOLD the state machine shall have a transition to ETHSM_STATE_WAIT_TRCVLINK, if the TcpIp modul reports TCPIP_STATE_OFFLINE for the corresponding network handle.⌋()

Note: The transition is shown as sequence diagram in Figure 9-3.

[SWS_EthSM_00177] ⌈
After the successful transition from ETHSM_STATE_ONHOLD to E ETHSM_STATE_WAIT_TRCVLINK the Ethernet State Manager shall set the main state to ETHSM_NO_COMMUNICATION and call the callback function ComM_BusSM_ModeIndication of the ComM with communication mode COMM_NO_COMMUNICATION.⌋()

#### 7.6.7.3 Transition from sub-state ONHOLD to OFFLINE

[SWS_EthSM_00178] ⌈
In the state ETHSM_STATE_ONHOLD the state machine shall have a transition to ETHSM_STATE_OFFLINE, if the ComM requests COMM_NO_COMMUNICATION for the corresponding network handle.⌋()

[SWS_EthSM_00179] ⌈

The transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_OFFLINE shall set the controller mode to ETH_MODE_DOWN. ⌋()

[SWS_EthSM_00181]⌈
The transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_OFFLINE shall request the TcpIp state TCPIP_STATE_OFFLINE from the TcpIp module.⌋()

[SWS_EthSM_00184]⌈
After the successful transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_OFFLINE the Ethernet State Manager shall set the main state to ETHSM_NO_COMMUNICATION and call the callback function ComM_BusSM_ModeIndication of the ComM with communication mode COMM_NO_COMMUNICATION.⌋()

### 7.6.8 Information about state transitions

[SWS_EthSM_00083] ⌈
Evertime the EthSM statemachine enters a sub-state, the EthSM shall report the entered sub-state towards the BswM by calling BswM_EthSM_CurrentState(<entered sub-state>) (e.g. when entering sub-state ETHSM_STATE_ONLINE, EthSM shall call BswM_EthSM_CurrentState(ETHSM_STATE_ONLINE). The transistion to ETHSM_STATE_OFFLINE after initialization (see [SWS_EthSM_00025]) shall not be reported towards the BswM.⌋()
Note: The BswM need the information about the EthSM internal states (see [SWS_EthSM_00041])

Note: The ComM needs the information about the communication states, e.g. COMM_FULL_COMMUNICATION or COMM_NO_COMMUNICATION. The calls towards the ComM are specified explicitely within the sub-states.

## 7.7 Commercial Off The Shelf stack usage

A commercial off the shelf stack (COTS) shall be useable. The commercial stack is useable without adaptation (Variant 1 in Figure 7-2). However, the Ethernet State Manager is not able to control the Ethernet Interface Controller and Ethernet transceiver in this case. The commercial stack may be adapted for usage with the Ethernet Interface. In this case, the Ethernet State Manager is able to control both Ethernet Interface Controller and Ethernet transceiver (Variant 2 in Figure 7-2).

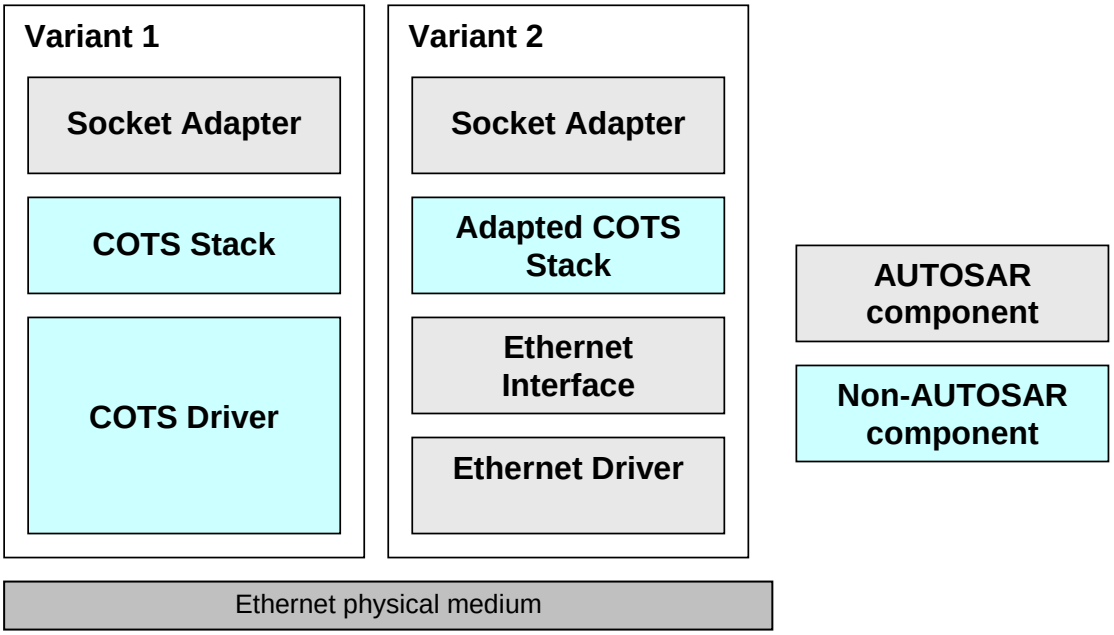



**Figure 7-2: BSW stack architecture variants**

[SWS_EthSM_00078]⌈
It is possible to set the Ethernet State Manager in a dummy mode (see chapter 10 configuration specification). In this mode, the Ethernet State Manager doesn't support the API to the Ethernet interface. The API to the ComM is available but the functionality is deactivated. The function calls from the ComM will be answered with the return value E_OK.⌋()

## 7.8 Error classification

Section 7.2 "Error Handling" of the document "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

### 7.8.1 Development Errors

**[SWS_EthSM_00200]⌈**

| *Type of error* | *Related error code* | *Error value* |
|---|---|---|
| Invalid communication mode requested | ETHSM_E_INVALID_NETWORK_MODE | 0x01 |
| EthSM module was not initialized | ETHSM_E_UNINIT | 0x02 |
| Invalid pointer in parameter list | ETHSM_E_PARAM_POINTER | 0x03 |

| | | |
|---|---|---|
| Invalid parameter in parameter list | ETHSM_E_INVALID_NETWORK_HANDLE | 0x04 |
| Invalid TcpIp state | ETHSM_E_INVALID_TCP_IP_MODE | 0x05 |
| Invalid parameter in parameter list | ETHSM_E_PARAM_CONTROLLER | 0x07 |

](

### 7.8.2 Runtime Errors

There are no runtime rrrors

### 7.8.3 Transient Faults

There are no transient faults.

### 7.8.4 Production Errors

[SWS_EthSM_00202]

| | | |
|---|---|---|
| ***Error Name:*** | ETHSM_E_LINK_DOWN | |
| ***Short Description:*** | Link down detection | |
| ***Long Description:*** | It shall be reported when the transceiver switches to “down” while communication has already been established and is requested because of communication request | |
| ***Recommended DTC:*** | N/A | |
| ***Detection Criteria:*** | FAIL | During transition from ETHSM_STATE_ONLINE to ETHSM_STATE_ONHOLD, which is triggered by EthSM_TrcvLinkStateChg(ETHTRCV_LINK_STATE_DOWN) |
| | PASS | During transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_ONLINE, which is triggered by EthSM_TrcvLinkStateChg(ETHTRCV_LINK_STATE_ACTIVE) |
| ***Secondary Parameters:*** | None | |
| ***Time Required:*** | PRE_FAIL: Immediately<br>PASS: Configuration dependent | |
| ***Monitor Frequency*** | Continuous | |
| ***MIL illumniation:*** | N/A | |

](

### 7.8.5 Extended Production Errors

There are no extended production errors.

# 8 API specification

## 8.1 Imported types

**[SWS_EthSM_91001]⌈**

| *Module* | *Header File* | *Imported Type* |
|---|---|---|
| ComM | Rte_ComM_Type.h | ComM_ModeType |
| ComStack_Types | ComStack_Types.h | NetworkHandleType |
| Dem | Rte_Dem_Type.h | Dem_EventIdType |
| | Rte_Dem_Type.h | Dem_EventStatusType |
| Eth | Eth_GeneralTypes.h | Eth_ModeType (draft) |
| EthTrcv | Eth_GeneralTypes.h | EthTrcv_LinkStateType |
| Std | Std_Types.h | Std_ReturnType |
| | Std_Types.h | Std_VersionInfoType |
| TcpIp | TcpIp.h | TcpIp_StateType |

⌋()

## 8.2 Type definitions

### 8.2.1 EthSM_NetworkModeStateType

**[SWS_EthSM_00041]⌈**

| | | | |
|---|---|---|---|
| ***Name*** | EthSM_NetworkModeStateType | | |
| ***Kind*** | Enumeration | | |
| ***Range*** | ETHSM_ STATE_ OFFLINE | -- | EthSM is initialized in this state. (TcpIp state is OFFLINE) |
| | ETHSM_ STATE_ WAIT_ TRCVLINK | -- | ComM requests COMM_FULL_COMMUNICATION in this state. Controller will be set to ACTIVE. EthSM waits for transceiver link state (ACTIVE). |
| | ETHSM_ STATE_ WAIT_ONLINE | -- | Transceiver link state is ACTIVE EthSM waits for IP communication (TcpIP state is ONLINE) |
| | ETHSM_ STATE_ | -- | IP communication is available ComM state COMM_FULL_ COMMUNICATION is reached |

| | | | |
|---|---|---|---|
| | ONLINE | | |
| | ETHSM_ STATE_ ONHOLD | -- | EthSM lost active transceiver link state, TcpIP state is still ONLINE) |
| | ETHSM_ STATE_ WAIT_ OFFLINE | -- | ComM requests COMM_SILENT_COMMUNICATION (TcpIp state is still ONLINE) and if configured (see ECUC_Eth SM_00110), to disable transmission of Ethernet frames. Afterwards ComM requests COMM_NO_COMMUNICATION in this state. (TcpIp state is requested with OFFLINE) |
| ***Description*** | This type shall define the states of the network mode state machine. | | |
| ***Available via*** | EthSM.h | | |

]()

## 8.3 Function definitions

This is a list of functions provided for upper layer modules.

Note:
Depending on the Ethernet hardware, it may become necessary that implementations deviate from API specifications in respect to the asynchronous/synchronous behaviour.

### 8.3.1 EthSM_Init

**[SWS_EthSM_00043]**[

| | |
|---|---|
| ***Service Name*** | EthSM_Init |
| ***Syntax*** | `void EthSM_Init (`<br>`  void`<br>`)` |
| ***Service ID [hex]*** | 0x07 |
| ***Sync/Async*** | Synchronous |
| ***Reentrancy*** | Non Reentrant |
| ***Parameters (in)*** | None |
| ***Parameters (inout)*** | None |
| ***Parameters (out)*** | None |
| ***Return value*** | None |
| ***Description*** | This function initialize the EthSM. |
| ***Available via*** | EthSM.h |

](SRS_BSW_00405, SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414)

### 8.3.2 EthSM_GetVersionInfo

**[SWS_EthSM_00046]**[

| | |
|---|---|
| ***Service Name*** | EthSM_GetVersionInfo |
| ***Syntax*** | `void EthSM_GetVersionInfo (`<br>`  Std_VersionInfoType* versioninfo`<br>`)` |
| ***Service ID [hex]*** | 0x02 |
| ***Sync/Async*** | Synchronous |
| ***Reentrancy*** | Reentrant |

| | | |
|---|---|---|
| ***Parameters (in)*** | None | |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | versioninfo | Pointer where to put out the version information. |
| ***Return value*** | None | |
| ***Description*** | This service puts out the version information of this module. | |
| ***Available via*** | EthSM.h | |

](SRS_BSW_00407, SRS_BSW_00003)

### 8.3.3 EthSM_RequestComMode

**[SWS_EthSM_00050]**[

| | | |
|---|---|---|
| ***Service Name*** | EthSM_RequestComMode | |
| ***Syntax*** | `Std_ReturnType EthSM_RequestComMode ( NetworkHandleType NetworkHandle, ComM_ModeType ComM_Mode )` | |
| ***Service ID [hex]*** | 0x05 | |
| ***Sync/Async*** | Asynchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | NetworkHandle | Handle of destinated communication network for request |
| | ComM_Mode | Requested communication mode |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | E_OK: Service accepted<br>E_NOT_OK: Service denied |
| ***Description*** | Handles the communication mode and sets the Ethernet network active or passive. | |
| ***Available via*** | EthSM.h | |

]()
Remark: The function reentrancy is limited to different network handles. Reentrancy for the same network is not to be regarded here.

[SWS_EthSM_00051] 
The function `EthSM_RequestComMode` checks the network handle of the request. It only accepts the request, if the network handle of the request is a handle contained in

the EthSM configuration (configuration parameter `EthSMNetworkHandle`). In this case the return value is set to E_OK.
If it is not contained in the configuration, the function denies the request. In this case the return value is set to E_NOT_OK.⌋()

[SWS_EthSM_00052]⌈
The function `EthSM_RequestComMode` shall report `ETHSM_E_INVALID_NETWORK_HANDLE` to the DET, if it does not accept the network handle of the request.⌋()

[SWS_EthSM_00095]⌈
The function `EthSM_RequestComMode` shall report `ETHSM_E_INVALID_NETWORK_MODE` to the DET, if it does not accept the ComM_Mode of the request.⌋()

[SWS_EthSM_00053]⌈
If the function `EthSM_RequestComMode` accepts the function call, it shall store the communication mode for the network handle and the corresponding network mode switch of the state machine shall be initiated in the next main function cycle latest.⌋()

[SWS_EthSM_00054]⌈
The function `EthSM_RequestComMode` shall report `ETHSM_E_UNINIT` to the DET, if the EthSM is not initialized yet.⌋(SRS_BSW_00406)

[SWS_EthSM_00199] {OBSOLETE}⌈
The function `EthSM_RequestComMode` shall accept SilentCom request from ComM and will return `E_OK`. No error shall be reported to ComM in this case, though SilentCom is not available according to SWS_EthSM_00203 and SWS_EthSM_00038⌋()

### 8.3.4 EthSM_GetCurrentComMode

**[SWS_EthSM_00055]⌈**

| | | |
|---|---|---|
| ***Service Name*** | EthSM_GetCurrentComMode | |
| ***Syntax*** | `Std_ReturnType EthSM_GetCurrentComMode ( NetworkHandleType NetworkHandle, ComM_ModeType* ComM_ModePtr )` | |
| ***Service ID [hex]*** | 0x04 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | NetworkHandle | Network handle whose current communication mode shall be put out |

| | | |
|---|---|---|
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | ComM_Mode Ptr | Pointer where to put out the current communication mode |
| ***Return value*** | Std_Return-Type | E_OK: Service accepted<br>E_NOT_OK: Service denied |
| ***Description*** | This service shall put out the current communication mode of a Ethernet network. | |
| ***Available via*** | EthSM.h | |

⌋()

[SWS_EthSM_00057] ⌈

The function `EthSM_GetCurrentComMode` checks the network handle of the service request. It only accepts the service, if the network handle of the request is a handle contained in the EthSM configuration (configuration parameter `EthSMNetworkHandle`). In this case the return value is set to E_OK.
If it is not contained in the configuration, the function denies the request. In this case the return value is set to E_NOT_OK.⌋()

[SWS_EthSM_00058] ⌈

The function `EthSM_GetCurrentComMode` shall report `ETHSM_E_INVALID_NETWORK_HANDLE` to the DET, if it does not accept the network handle of the request.⌋()

[SWS_EthSM_00059] ⌈

The function `EthSM_GetCurrentComMode` puts out the current communication mode for the network handle to the designated pointer of type `ComM_ModeType`, if it accepts the request.⌋()

Remark: Because the Ethernet hardware needs a certain time to proceed with the request and there is currently no notification mechanism specified, the real hardware mode and the mode notified by the EthSM might be different until the hardware is ready.

[SWS_EthSM_00060] ⌈

The function `EthSM_GetCurrentComMode` shall report `ETHSM_E_UNINIT` to the DET, if the EthSM is not initialized yet. ⌋(SRS_BSW_00406, SRS_BSW_00374, SRS_BSW_00003, SRS_BSW_00318)

## 8.4 Call-back notifications

### 8.4.1 EthSM_CtrlModeIndication

**[SWS_EthSM_00190]⌈**

| | | |
|---|---|---|
| ***Service Name*** | EthSM_CtrlModeIndication | |
| ***Syntax*** | `void EthSM_CtrlModeIndication ( uint8 CtrlIdx, Eth_ModeType CtrlMode )` | |
| ***Service ID [hex]*** | 0x09 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant (only for different Ethernet controllers) | |
| ***Parameters (in)*** | CtrlIdx | Ethernet Interface Controller whose mode has changed |
| | CtrlMode | Notified Ethernet Interface Controller mode |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | Called when mode has been read out. Either triggered by previous EthIf_GetControllerMode or by EthIf_SetControllerMode call. Can directly be called within the trigger functions. | |
| ***Available via*** | EthSM.h | |

⌋()

[SWS_EthSM_00191]⌈

If the function `EthSM_CtrlModeIndication` gets a `CtrlIdx`, which is not configured in the configuration of the EthSM module, it shall call the function `Det_ReportError` with `ErrorId` parameter `ETHSM_E_PARAM_CONTROLLER.`⌋()

[SWS_EthSM_00192]⌈

If the EthSM module is not initialized, when the function `EthSM_CtrlModeIndication` is called, then the function `EthSM_CtrlModeIndication` shall call the function `Det_ReportError` with `ErrorId` parameter `ETHSM_E_UNINIT.`⌋()

### 8.4.2 EthSM_TrcvLinkStateChg

**[SWS_EthSM_00109]⌈**

| | | |
|---|---|---|
| ***Service Name*** | EthSM_TrcvLinkStateChg | |
| ***Syntax*** | `void EthSM_TrcvLinkStateChg ( uint8 CtrlIdx, EthTrcv_LinkStateType TransceiverLinkState )` | |
| ***Service ID [hex]*** | 0x06 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the Ethernet controller within the context of the Ethernet Interface |
| | TransceiverLink State | Actual transceiver link state of the specific network handle |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | This service is called by the Ethernet Interface to report a transceiver link state change. | |
| ***Available via*** | EthSM.h | |

⌋()

[SWS_EthSM_00112]⌈
The function `EthSM_TrcvLinkStateChg` shall report `ETHSM_E_PARAM_CONTROLLER` to the DET, if it does not accept the `CtrlIdx` of the function call.⌋()

[SWS_EthSM_00114]⌈
If the function `EthSM_TrcvLinkStateChg` does not report a DET error, it shall store the transceiver link state for the affected network handle and the corresponding network mode switch of the state machine shall be initiated in the next main function cycle latest.⌋
()

[SWS_EthSM_00115]⌈
The function `EthSM_TrcvLinkStateChg` shall report `ETHSM_E_UNINIT` to the DET, if the EthSM is not initialized yet.⌋(SRS_BSW_00406)

### 8.4.3 EthSM_TcpIpModeIndication

**[SWS_EthSM_00110]⌈**

| | | |
|---|---|---|
| ***Service Name*** | EthSM_TcpIpModeIndication | |
| ***Syntax*** | `void EthSM_TcpIpModeIndication ( uint8 CtrlIdx, TcpIp_StateType TcpIpState )` | |
| ***Service ID [hex]*** | 0x08 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | EthIf controller index to identify the communication network where the TcpIp state is changed |
| | TcpIp State | Actual TcpIp state of the specific network handle |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | This service is called by the TcpIp to report the actual TcpIp state (e.g. online, offline). | |
| ***Available via*** | EthSM_TcpIp.h | |

⌋()

[SWS_EthSM_00116]⌈
If the function `EthSM_TcpIpModeIndication` gets a CtrlIdx, which is not configured in the configuration of the EthSM module, it shall call the function Det_ReportError with ErrorId parameter ETHSM_E_PARAM_CONTROLLER.⌋ ()

[SWS_EthSM_00118]⌈
If development error detection is enabled, the parameter TcpIpState shall be checked for being in the allowed range.
In case it is outside of the allowed range, the function `EthSM_TcpIpModeIndication` shall ignore the state indication and report development error `ETHSM_E_INVALID_TCP_IP_MODE` to the DET.⌋()

[SWS_EthSM_00119]⌈
If the function `EthSM_TcpIpModeIndication` accepts the function call, it shall store the TcpIp state for the affected network handle and the corresponding network mode switch of the state machine shall be initiated in the next main function cycle latest.⌋()

[SWS_EthSM_00120]⌈
The function `EthSM_TcpIpModeIndication` shall report `ETHSM_E_UNINIT` to the DET, if the EthSM is not initialized yet.⌋(SRS_BSW_00406)

### 8.4.4 EthSM_SleepIndication

**[SWS_EthSM_91004]**{DRAFT} ⌈

| | | |
|---|---|---|
| ***Service Name*** | EthSM_SleepIndication (draft) | |
| ***Syntax*** | `void EthSM_SleepIndication ( uint8 Ctrl_Idx )` | |
| ***Service ID [hex]*** | 0xa | |
| ***Sync/Async*** | Asynchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | Ctrl_Idx | Ethernet Interface Controller where the corresponding EthTrcv detect a sleep indication on the network.Sleep. |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | This API is called by the EthIf and indicate that a sleep indication was detected on the network. This API is only called if the ECU is acting as a passive communication slave on the corresponding communication channel (the referenced EthTrcv of the affected EthIfTransceiver has set EthTrcvActAsSlavePassiveEnabled to TRUE). This could be used e.g. for Ethernet hardware which is compliant to the OA TC10. In this case the Ethernet hardware detect an Sleep.Indication which was triggered by a Sleep.Request of the connected link partner.<br>**Tags:** atp.Status=draft | |
| ***Available via*** | EthSM.h | |

⌋(SRS_Eth_00159)

[SWS_EthSM_00213] DRAFT ⌈
If the function EthSM_SleepIndication gets a CtrlIdx, which is not configured in the configuration of the EthSM module, it shall call the function Det_ReportError with ErrorId parameter ETHSM_E_PARAM_CONTROLLER, if development error reporting is enabled (see EthSMDevErrorDetect).
⌋(SRS_Eth_00159)

[SWS_EthSM_00214] DRAFT ⌈

If the EthSM module is not initialized, when the function EthSM_SleepIndication is called, then the function EthSM_SleepIndication shall call the function Det_ReportError with ErrorId parameter ETHSM_E_UNINIT, if development error reporting is enabled (see EthSMDevErrorDetect).⌋(SRS_Eth_00159)

[SWS_EthSM_00215] DRAFT ⌈
If EthSM_SleepIndication is called, EthSM shall forward this call to ComM by calling ComM_BusSM_BusSleepMode with the corresponding communication channel.⌋(SRS_Eth_00159)

## 8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

### 8.5.1 EthSM_MainFunction

**[SWS_EthSM_00035]⌈**

| | |
|---|---|
| ***Service Name*** | EthSM_MainFunction |
| ***Syntax*** | `void EthSM_MainFunction (`<br>`  void`<br>`)` |
| ***Service ID [hex]*** | 0x01 |
| ***Sync/Async*** | Synchronous |
| ***Reentrancy*** | Non Reentrant |
| ***Parameters (in)*** | None |
| ***Parameters (inout)*** | None |
| ***Parameters (out)*** | None |
| ***Return value*** | None |
| ***Description*** | Cyclic Main Function which is called from the Scheduler. |
| ***Available via*** | EthSM_SchM.h |

⌋()
[SWS_EthSM_00093] ⌈
The function EthSM_MainFunction shall be called cyclically with a fixed cycle time. The cycle time could be defined via the configuration parameter ETHSM_MAIN_FUNCTION_PERIOD. ⌋()

[SWS_EthSM_00197] ⌈

The main function of the EthSM module shall operate the effects of the EthSM state machine, which the EthSM module shall implement for each configured network. ]()

[SWS_EthSM_00198] [
The EthSM shall monitor the requested and current state of the Ethernet Interface Controller. If the EthSM detects a mismatch, it shall bring the hardware back to the corresponding state. (i.e. FullCOM requires the state ETH_MODE_ACTIVE; NoCom requires ETH_MODE_DOWN) ]()

## 8.6 Expected Interfaces

In this chapter all interfaces required from other modules are listed.

### 8.6.1 Mandatory Interfaces

This chapter defines all interfaces which are required to fulfill the core functionality of the module.
**[SWS_EthSM_91002]**[

| *API Function* | *Header File* | *Description* |
|---|---|---|
| BswM_EthSM_CurrentState | BswM_EthSM.h | Function called by EthSM to indicate its current state. |
| ComM_BusSM_ModeIndication | ComM.h | Indication of the actual bus mode by the corresponding Bus State Manager. ComM shall propagate the indicated state to the users with means of the RTE and BswM. |
| Dem_SetEventStatus | Dem.h | Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEventReportingType} == STANDARD_REPORTING) |
| EthIf_SetControllerMode | EthIf.h | Enables / disables the indexed controller |
| TcpIp_RequestComMode | TcpIp.h | By this API service the TCP/IP stack is requested to change the TcpIp state of the communication network identified by EthIf controller index. |

]()

### 8.6.2 Optional Interfaces

This chapter defines all interfaces which are required to fulfill an optional functionality of the module.
**[SWS_EthSM_91003]**[

| *API Function* | *Header File* | *Description* |
|---|---|---|
| ComM_BusS-M_BusSleep-Mode | ComM.h | Notification of the corresponding Bus State Manager that the actual bus mode is Bus-Sleep.<br>Only applicable for ComM channels with ComMNmVariant set to SLAVE_ACTIVE or SLAVE_PASSIVE.<br>E.g. LIN slaves (ComMNMVariant = SLAVE_ACTIVE) or Ethernet channels with OA TC10 compliant Ethernet hardware which act as passive communication slave (ComMNMVariant = SLAVE_PASSIVE and EthTrcvActAsSlavePassiveEnabled set to TRUE) |
| Det_Report-Error | Det.h | Service to report development errors. |
| EthIf_Get-ControllerMode | EthIf.h | Obtains the state of the indexed controller |

]()

# 9 Sequence diagrams

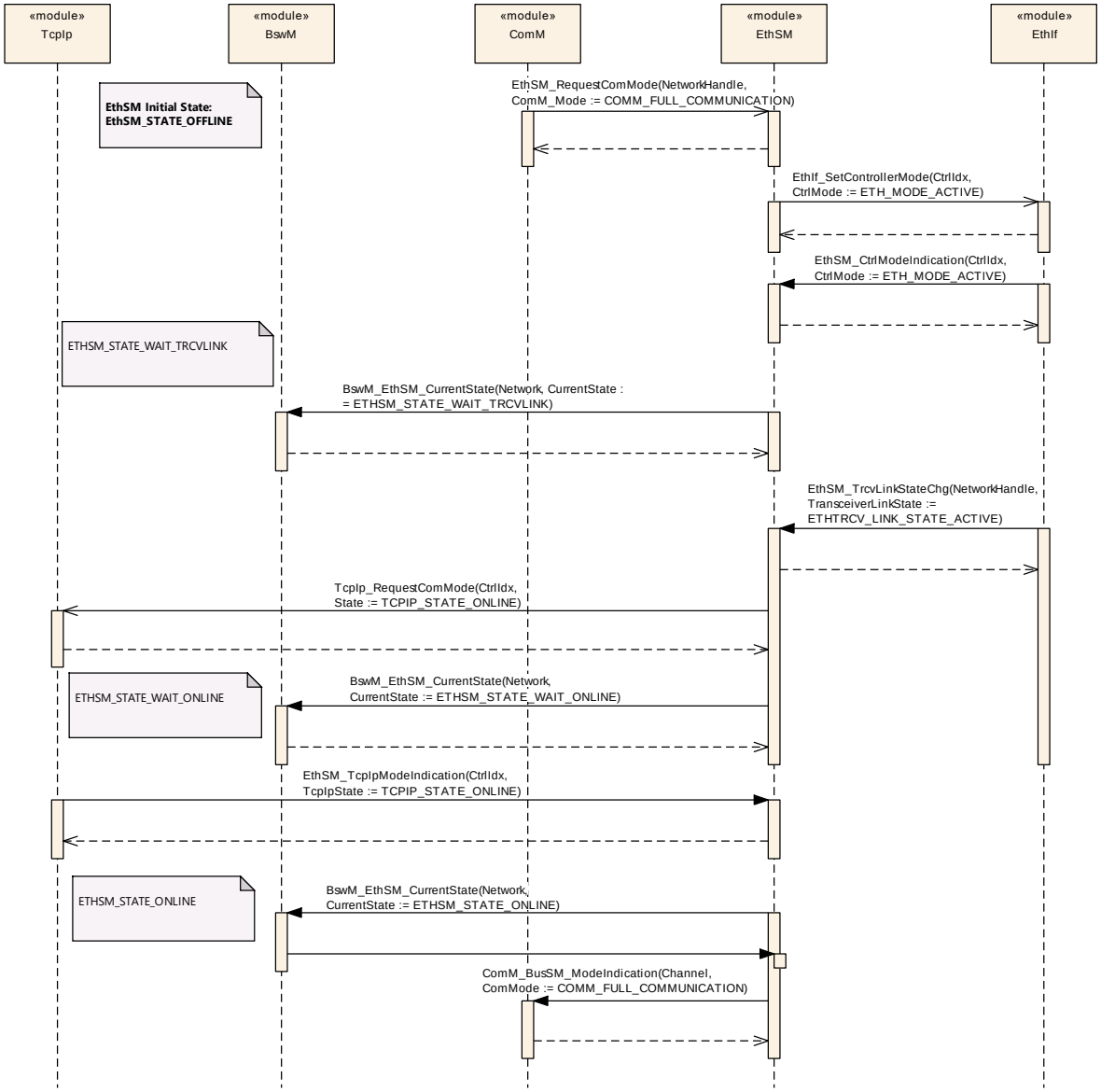



**Figure 9-1: Network mode state machine – transition from no to full communication**

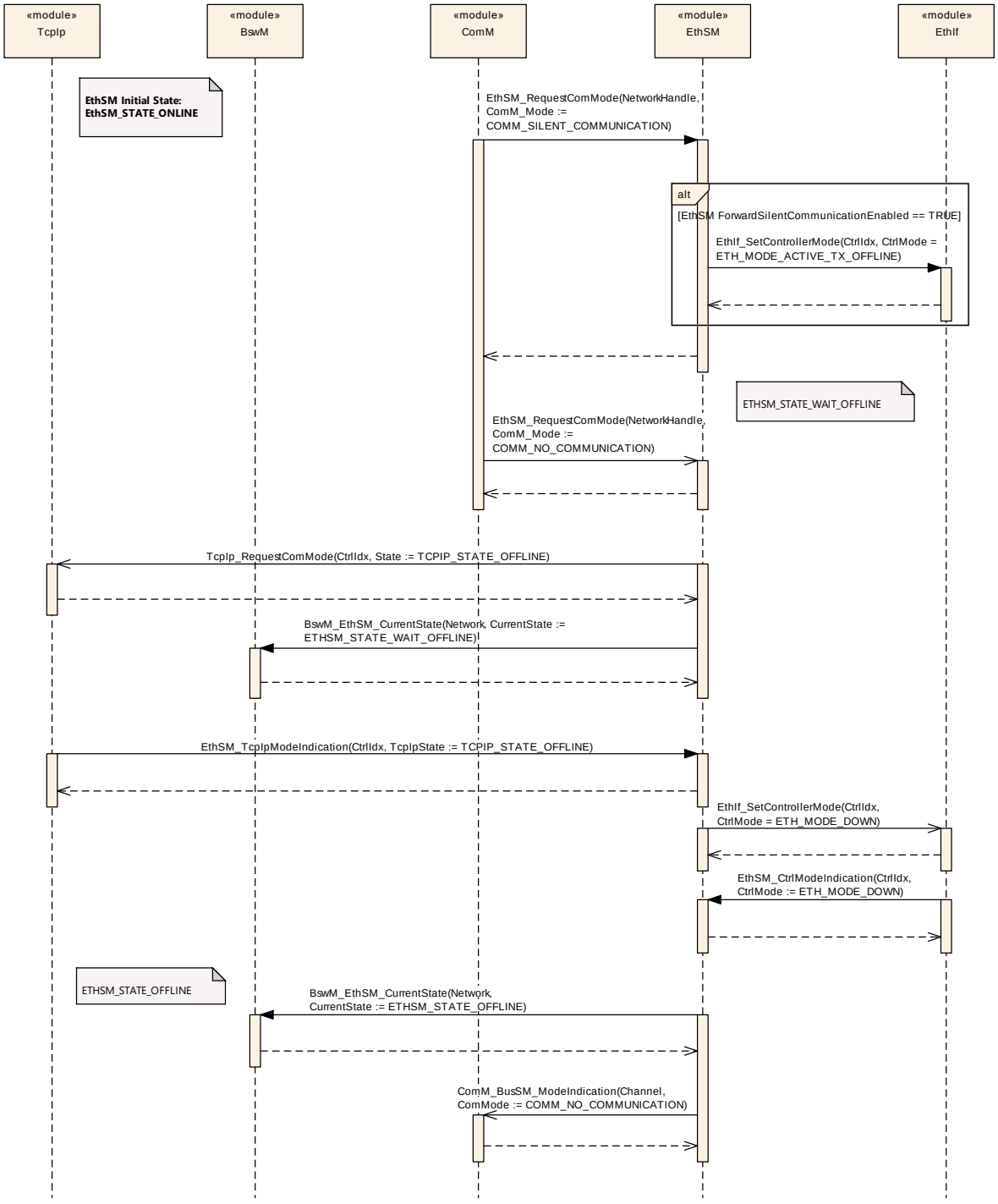



**Figure 9-2: Network mode state machine – transition from full to no communication**

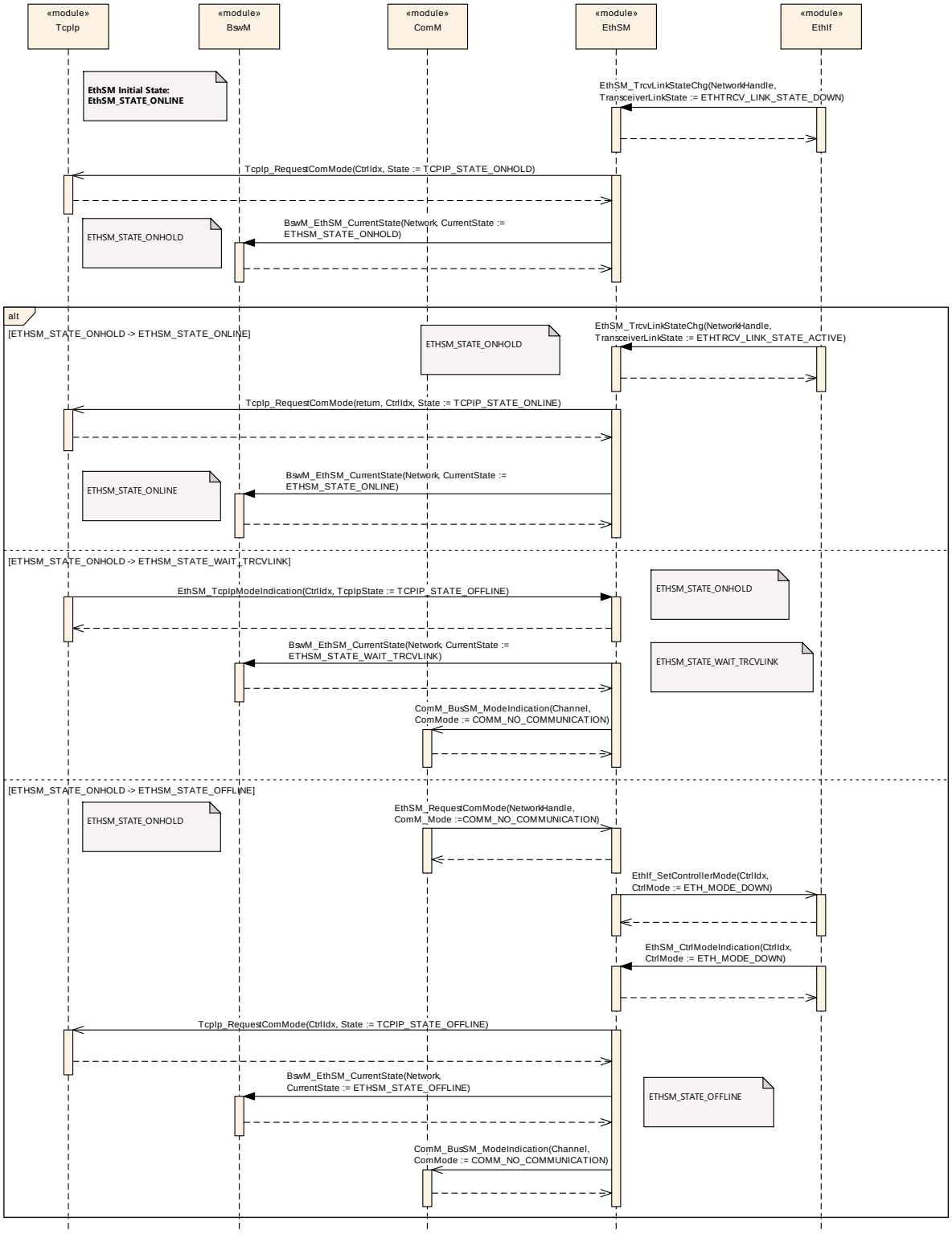



**Figure 9-3: Network mode state machine – sub state ONHOLD**

# 10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module EthSM.

Chapter 10.3 specifies published information of the module EthSM.

## 10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

## 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

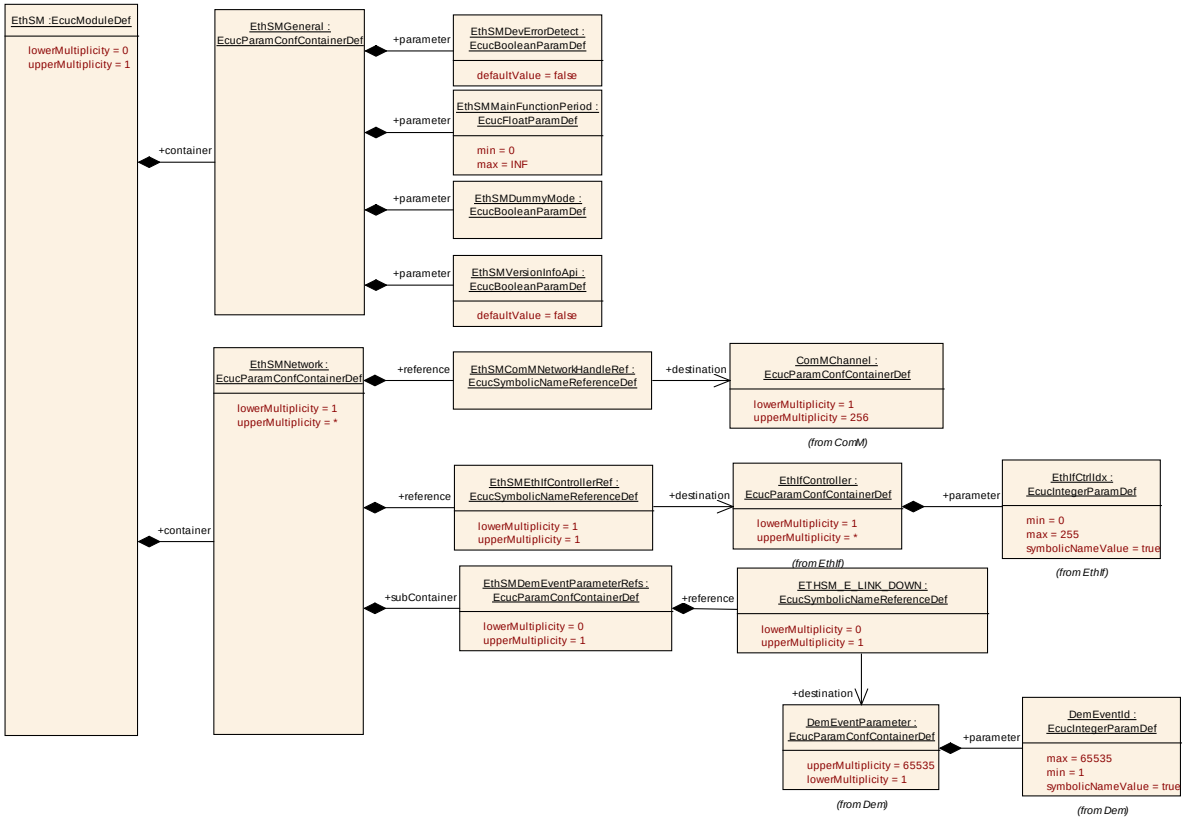
### 10.2.1 Configuration Tool

[SWS_EthSM_00081]⌈
A configuration tool will create a configuration structure that is understood by the EthSM.⌋(SRS_BSW_00159, SRS_BSW_00424, SRS_BSW_00425)

### 10.2.2 EthSM

| ***SWS Item*** | **ECUC_EthSM_00108 :** |
|---|---|
| ***Module Name*** | *EthSM* |
| ***Module Description*** | Configuration of the Ethernet State Manager |
| ***Post-Build Variant Support*** | false |
| ***Supported Config Variants*** | VARIANT-PRE-COMPILE |

| ***Included Containers*** | | |
|---|---|---|
| ***Container Name*** | ***Multiplicity*** | ***Scope / Dependency*** |
| EthSMGeneral | 1 | This container contains the global parameter of the Ethernet State Manager. |
| EthSMNetwork | 1..* | This container contains the Ethernet network-specific parameters of each Ethernet network. It also contains the reference to combination of controller and transceiver assigned to an Ethernet network. |




## 10.2.3 EthSMGeneral

| | |
|---|---|
| ***SWS Item*** | **ECUC_EthSM_00063 :** |
| ***Container Name*** | EthSMGeneral |
| ***Parent Container*** | EthSM |
| ***Description*** | This container contains the global parameter of the Ethernet State Manager. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00065 :** | | |
| ***Name*** | EthSMDevErrorDetect | | |
| ***Parent Container*** | EthSMGeneral | | |
| ***Description*** | Switches the development error detection and notification on or off.<br>• true: detection and notification is enabled.<br>• false: detection and notification is disabled. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | false | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | |
|---|---|
| ***SWS Item*** | **ECUC_EthSM_00079 :** |

| | | | |
|---|---|---|---|
| ***Name*** | EthSMDummyMode | | |
| ***Parent Container*** | EthSMGeneral | | |
| ***Description*** | Disables the API to the EthIf. The API to the ComM is available but the functionality is deactivated. The function calls from the ComM will be answered with the return value E_OK. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | -- | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00066 :** | | |
| ***Name*** | EthSMMainFunctionPeriod | | |
| ***Parent Container*** | EthSMGeneral | | |
| ***Description*** | Specifies the period in seconds that the MainFunction has to be triggered with. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucFloatParamDef | | |
| ***Range*** | ]0 .. INF[ | | |
| ***Default value*** | -- | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00092 :** | | |
| ***Name*** | EthSMVersionInfoApi | | |
| ***Parent Container*** | EthSMGeneral | | |
| ***Description*** | Enables and disables the version info API. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | false | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| |
|---|
| ***No Included Containers*** |

## 10.2.4 EthSMNetwork

| | |
|---|---|
| ***SWS Item*** | **ECUC_EthSM_00067 :** |
| ***Container Name*** | EthSMNetwork |
| ***Parent Container*** | EthSM |
| ***Description*** | This container contains the Ethernet network-specific parameters of each Ethernet network. It also contains the reference to combination of controller and transceiver assigned to an Ethernet network. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00110 :** | | |
| ***Name*** | EthSMForwardSilentCommunicationEnabled | | |
| ***Parent Container*** | EthSMNetwork | | |
| ***Description*** | Specifies if a request COMM_SILENT_COMMUNICATION shall be forwarded to the corresponding EthIfController.<br>If the parameter is set to TRUE and EthSM in called with COMM_SILENT_COMMUNICATION, then EthSM call the corresponding EthIfController with ETH_MODE_ACTIVE_TX_OFFLINE to reject any transmission request. Therefore, only reception handling of Ethernet frames is performed.<br><br>If the parameter is set to FALSE and EthSM in called with COMM_SILENT_COMMUNICATION, then EthSM will NOT forward the call to EthIf. Therefore, transmission AND reception handling of Ethernet frames is performed.<br>**Tags:**<br>atp.Status=draft | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | false | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00109 :** | | |
| ***Name*** | EthSMWakeupSleepOnDatalineEnabled | | |
| ***Parent Container*** | EthSMNetwork | | |
| ***Description*** | Specifies if an active communication request shall be forwarded to the lower layer to trigger a wake-up on the Ethernet network, e.g. trigger a wake-up on dataline if OA TC10 compliant Ethernet hardware is used.<br>If the parameter is set to TRUE and EthSM in called with COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST, then EthSM call the corresponding EthIfController everytime with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST to trigger an wake-up request in the lower layer. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | EcucBooleanParamDef | | |
| ***Default value*** | false | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00068 :** | | |
| ***Name*** | EthSMComMNetworkHandleRef | | |
| ***Parent Container*** | EthSMNetwork | | |
| ***Description*** | Unique handle to identify one certain Ethernet network. Reference to one of the network handles configured for the ComM. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | Symbolic name reference to [ ComMChannel ] | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |

| | | | |
|---|---|---|---|
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00105 :** | | |
| ***Name*** | EthSMEthIfControllerRef | | |
| ***Parent Container*** | EthSMNetwork | | |
| ***Description*** | Reference to EthIfCtrl container where a ETH controller and transceiver (optional) combination is configured. | | |
| ***Multiplicity*** | 1 | | |
| ***Type*** | Symbolic name reference to [ EthIfController ] | | |
| ***Post-Build Variant Value*** | false | | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| ***Included Containers*** | | |
|---|---|---|
| ***Container Name*** | ***Multiplicity*** | ***Scope / Dependency*** |
| EthSMDemEventParameterRefs | 0..1 | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. |

## 10.2.5 EthSMDemEventParameterRefs

| | |
|---|---|
| ***SWS Item*** | **ECUC_EthSM_00106 :** |
| ***Container Name*** | EthSMDemEventParameterRefs |
| ***Parent Container*** | EthSMNetwork |
| ***Description*** | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. |
| ***Configuration Parameters*** | |

| | | | |
|---|---|---|---|
| ***SWS Item*** | **ECUC_EthSM_00107 :** | | |
| ***Name*** | ETHSM_E_LINK_DOWN | | |
| ***Parent Container*** | EthSMDemEventParameterRefs | | |
| ***Description*** | Reference to configured DEM event to report bus off errors for this Eth network. | | |
| ***Multiplicity*** | 0..1 | | |
| ***Type*** | Symbolic name reference to [ DemEventParameter ] | | |
| ***Post-Build Variant Multiplicity*** | false | | |
| ***Post-Build Variant Value*** | false | | |
| ***Multiplicity Configuration Class*** | ***Pre-compile time*** | X | All Variants |
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Value Configuration Class*** | ***Pre-compile time*** | X | All Variants |

| | | | |
|---|---|---|---|
| | ***Link time*** | -- | |
| | ***Post-build time*** | -- | |
| ***Scope / Dependency*** | scope: local | | |

| ***No Included Containers*** |
|---|

## 10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

# 11 Not applicable requirements

**[SWS_EthSM_00999]**「 These requirements are not applicable to this specification. 」
(SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00170, SRS_BSW_00395, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00438, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00336, SRS_BSW_00369, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00343, SRS_BSW_00160, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00373, SRS_BSW_00314, SRS_BSW_00353, SRS_BSW_00361, SRS_BSW_00328, SRS_BSW_00377, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00371, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00331, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00321, SRS_BSW_00341, SRS_BSW_00334)