Document Title	Specification of Ethernet State Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	415
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2018-10-31	4.4.0	AUTOSAR Release Management	• Error classification has been fixed • Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Default error is removed • Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Remove Set and Get Transceiver mode functionality • Correct EthSM_TcpIpModeIndication callback return value • Harmonize main function period with the other modules • Remove Get current internal mode
2015-07-31	4.2.2	AUTOSAR Release Management	• Harmonize Sequence diagrams, Network State Machine and Functional Description • Debugging support marked as obsolete • Report to DET if TcpIp state is not accepted • Adaptations related to renaming of DET, • Error Handling: tables for Runtime Errors and Transient Faults added

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2014-10-31	4.2.1	AUTOSAR Release Management	• Change from Synchronous to Asynchronous API • Additional callback functions added • Existing behavior of functions changes • Editorial changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Corrective action after timeout • Non mutually exclusive transitions from ETHSM_STATE_ONLINE • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Optimization of full com request • Standardization of internal state names • Asynchronous behavior of several interfaces • Several clarifications and corrections • Editorial changes • 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) • Update chapter 10 • Added Production Error if Transceiver Link is down • General Update (corrections and formulations)
2011-12-22	4.0.3	AUTOSAR Administration	• Update Chapter 10 (Parameter adjustment)

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2010-09-30	3.1.5	AUTOSAR Administration	• Functional changes: • Correction of the naming convention of SW modul version information • Correction of chapter 10 - configuration parameter "EthSMNetworkIndex" • Remove InstanceID from GetVersionId structure • Additional callback function: Call of SoAd_BusSM_ModeIndication realized after the successful initialization of the EthTrcv and the EthController. • Non functional changes: • 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.

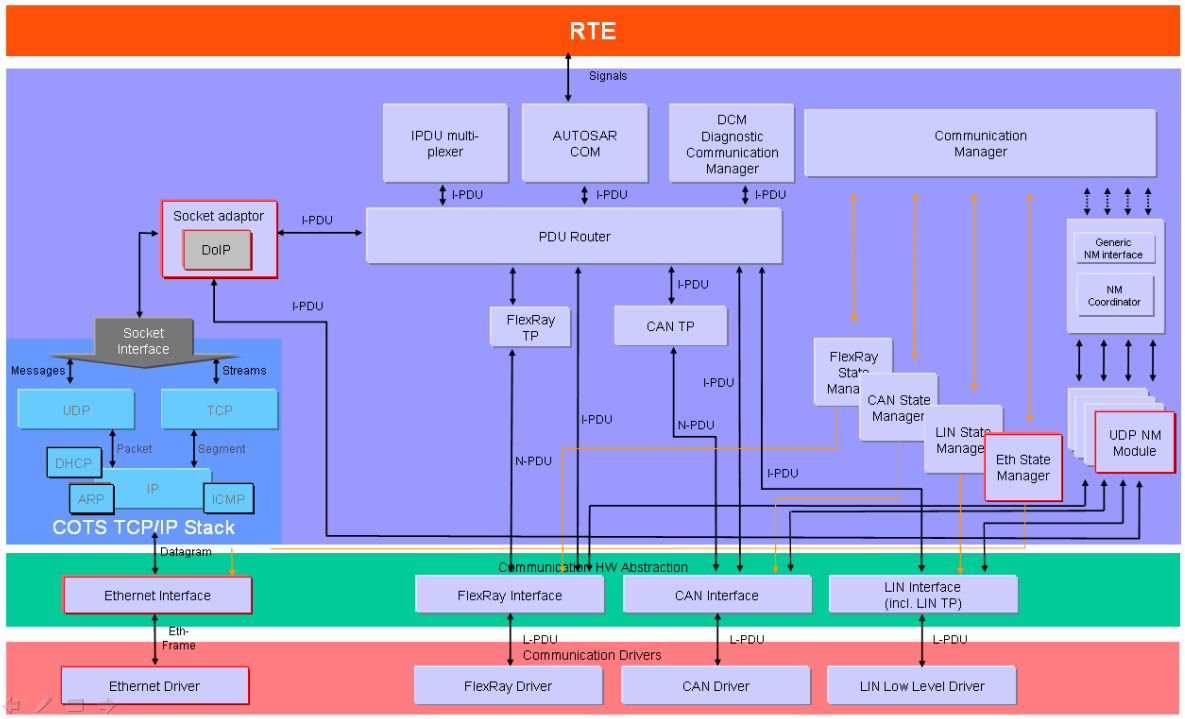


Figure 1-1: Example of an Autosar architecture including an Ethernet network

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

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 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

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 µC 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	SWS_EthSM_00060

	file	
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	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_EthSM_00999
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be	SWS_EthSM_00999

	named according the defined convention	
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 reasons	SWS_EthSM_00999
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	SWS_EthSM_00999

	modules	
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 processing function(s) has to be executed in a specific order or sequence	SWS_EthSM_00999
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

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 or more Ethernet Interface Controllers belong to a 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

[SWS_EthSM_00038] ⌈
The ComM calls the EthSM to request a certain communication mode. The Ethernet network only needs the communication modes FULL_COMMUNICATION and NO_COMMUNICATION. ⌋ ()

[SWS_EthSM_00085] ⌈
If FULL_COMMUNICATION is requested the Ethernet Interface Controller is set to the state ACTIVE. To reach FULL_COMMUNICATION it is also necessary to get an ACTIVE link state (Ethernet cable is connected) and an ONLINE TcpIP state (IP communication is available). The link state will be detected by the Ethernet

Transceiver module and will be communicated by the Ethernet Interface. The TcpIP state will be detected and communicated by the TcpIp module.⌋ ()

[SWS_EthSM_00086]⌈
If the ComM request NO_COMMUNICATION the Ethernet Interface Controller is set to the state DOWN.⌋ ()

Remark:
For the de-initialization no separate interface is necessary, the de-initialization is automatically realized in the EthIf.

It is also necessary to set the TcpIp state to OFFLINE.

[SWS_EthSM_00087]⌈
The Ethernet network has to be wake up by the application and it´s either on (FULL_COMMUNICATION) or off (NO_COMMUNICATION). So there is no need for other states e.g. like 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. The internal states are described in the Fig. 7-1 [SWS_EthSM_00041].⌋ ()

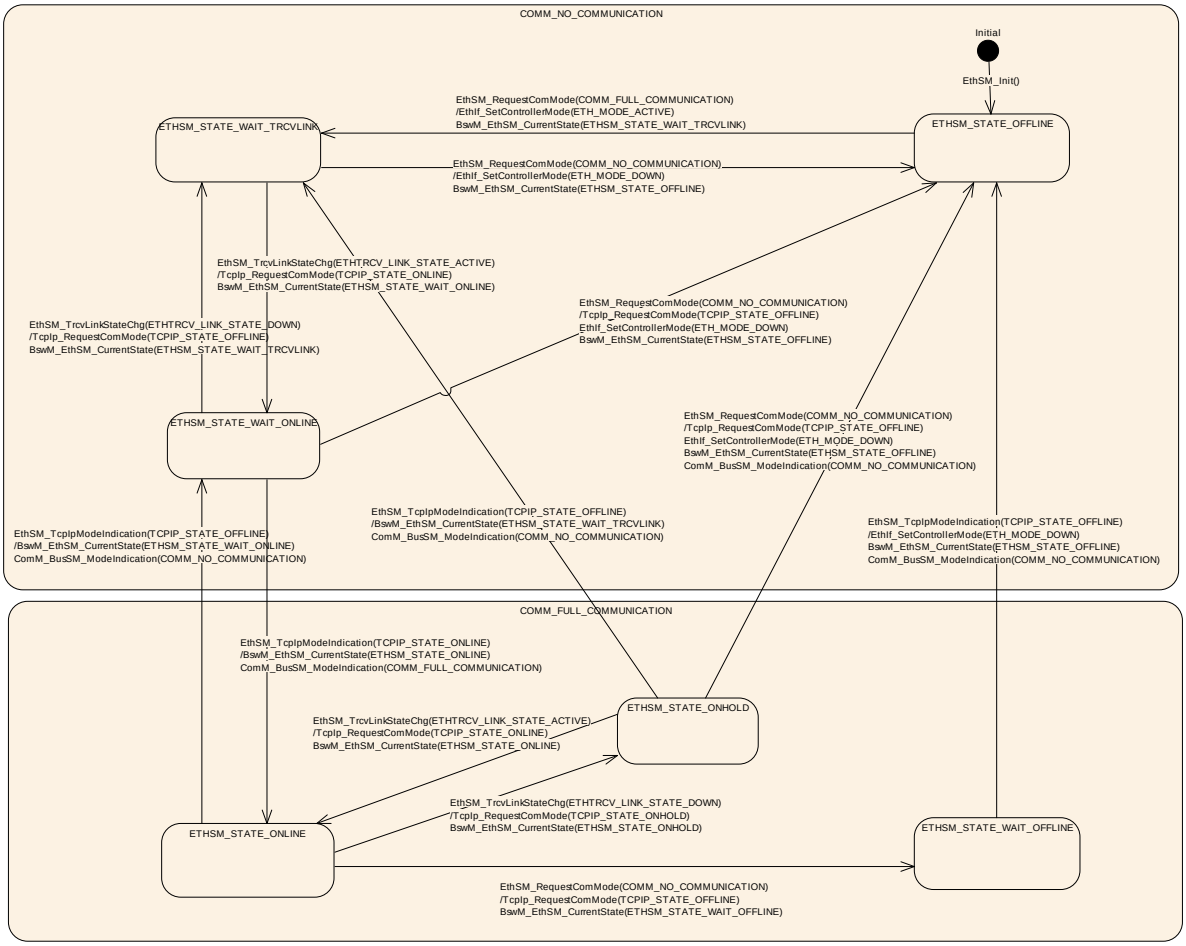


Figure 7-1: Network mode state machine of the EthSM

The Ethernet State Manager network mode state machine includes six sub states:

NO_COMMUNICATION	ETHSM_STATE_OFFLINE
	ETHSM_STATE_WAIT_TRCVLINK
	ETHSM_STATE_WAIT_ONLINE
FULL_COMMUNICATION	ETHSM_STATE_ONLINE
	ETHSM_STATE_ONHOLD
	ETHSM_STATE_WAIT_OFFLINE

The table below shows the detailed conditions of the sub states:

State	Controller Mode	Transceiver Mode	Transceiver Link	TcpIP Address	ComM Mode
ETHSM_STATE_OFFLINE	DOWN	DOWN	DOWN	OFFLINE	No Communication
ETHSM_STATE_WAIT_TRCVLINK	ACTIVE	ACTIVE	DOWN	OFFLINE	No Communication
ETHSM_STATE_WAIT_ONLINE	ACTIVE	ACTIVE	ACTIVE	OFFLINE	No Communication
ETHSM_STATE_ONLINE	ACTIVE	ACTIVE	ACTIVE	ONLINE	Full Communication
ETHSM_STATE_ONHOLD	ACTIVE	ACTIVE	DOWN	ONLINE	Full Communication
ETHSM_STATE_WAIT_OFFLINE	ACTIVE	ACTIVE	ACTIVE	ONLINE	Full Communication

To reach COMM_FULL_COMMUNICATION following conditions are necessary:

- 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 controller to 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 transceiver. After the link state is set to ACTIVE, the Ethernet State Manager is in the sub state ETHSM_STATE_WAIT_ONLINE.
In this sub state the state manager has to wait for the monitored TcpIp state information of the TcpIp module. After the TcpIP state is set to ACTIVE (= IP communication is available), the Ethernet State Manager is in the sub state ETHSM_STATE_ONLINE.
→ Now FULL_COMMUNICATION is reached.

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.
The initialization of the EthSM causes no further transactions in other modules. So no separate sequence diagram is needed.⌋ ()

7.6.2 Transition between substate WAIT_TRCVLINK and OFFLINE

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

transition the EthSM shall interact like specified in the sequence diagram Figure 9-1. ⌋ ()

[SWS_EthSM_00088] ⌈
The transition from ETHSM_STATE_OFFLINE to ETHSM_STATE_WAIT_TRCVLINK set the controller mode to ETH_MODE_ACTIVE. ⌋ ()

[SWS_EthSM_00097]⌈
After the successful transition from ETHSM_STATE_OFFLINE to ETHSM_STATE_WAIT_TRCVLINK the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_WAIT_TRCVLINK.⌋ ()

[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 sets the controller mode to ETH_MODE_DOWN. ⌋ ()

[SWS_EthSM_00130]⌈
After the successful transition from ETHSM_STATE_WAIT_TRCVLINK to ETHSM_STATE_OFFLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_OFFLINE.⌋ ()

7.6.3 Transition between substate WAIT_TRCVLINK and 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. In this transition the EthSM shall interact like specified in the sequence diagram 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.⌋ ()

[SWS_EthSM_00134]⌈
After the successful transition from ETHSM_STATE_WAIT_TRCVLINK to ETHSM_STATE_WAIT_ONLINE the Ethernet State Manager shall call the callback

function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_WAIT_ONLINE.⌋ ()

[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.⌋ ()

[SWS_EthSM_00138]⌈
After the successful transition from ETHSM_STATE_WAIT_ONLINE to ETHSM_STATE_WAIT_TRCVLINK the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_WAIT_TRCVLINK.⌋ ()

7.6.4 Transition from substate 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.⌋()

[SWS_EthSM_00144]⌈
After the successful transition from ETHSM_STATE_WAIT_ONLINE to ETHSM_STATE_OFFLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_OFFLINE.⌋ ()

7.6.5 Transition between substate 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. In this transition the EthSM shall interact like specified in the sequence diagram Figure 9-1.⌋ ()

[SWS_EthSM_00148]⌈
After the successful transition from ETHSM_STATE_WAIT_ONLINE to ETHSM_STATE_ONLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_ONLINE.⌋ ()

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

[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_00152] ⌈
After the successful transition from ETHSM_STATE_ONLINE to ETHSM_STATE_WAIT_ONLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_WAIT_ONLINE.⌋ ()

[SWS_EthSM_00154] ⌈
After the successful transition from ETHSM_STATE_ONLINE to ETHSM_STATE_WAIT_ONLINE the Ethernet State Manager shall call the callback function ComM_BusSM_ModeIndication of the ComM and transmit the communication mode (COMM_NO_COMMUNICATION). ⌋()

7.6.6 Transition from substate ONLINE to WAIT_OFFLINE

[SWS_EthSM_00155] ⌈
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. In this transition the EthSM shall interact like specified in the sequence diagram

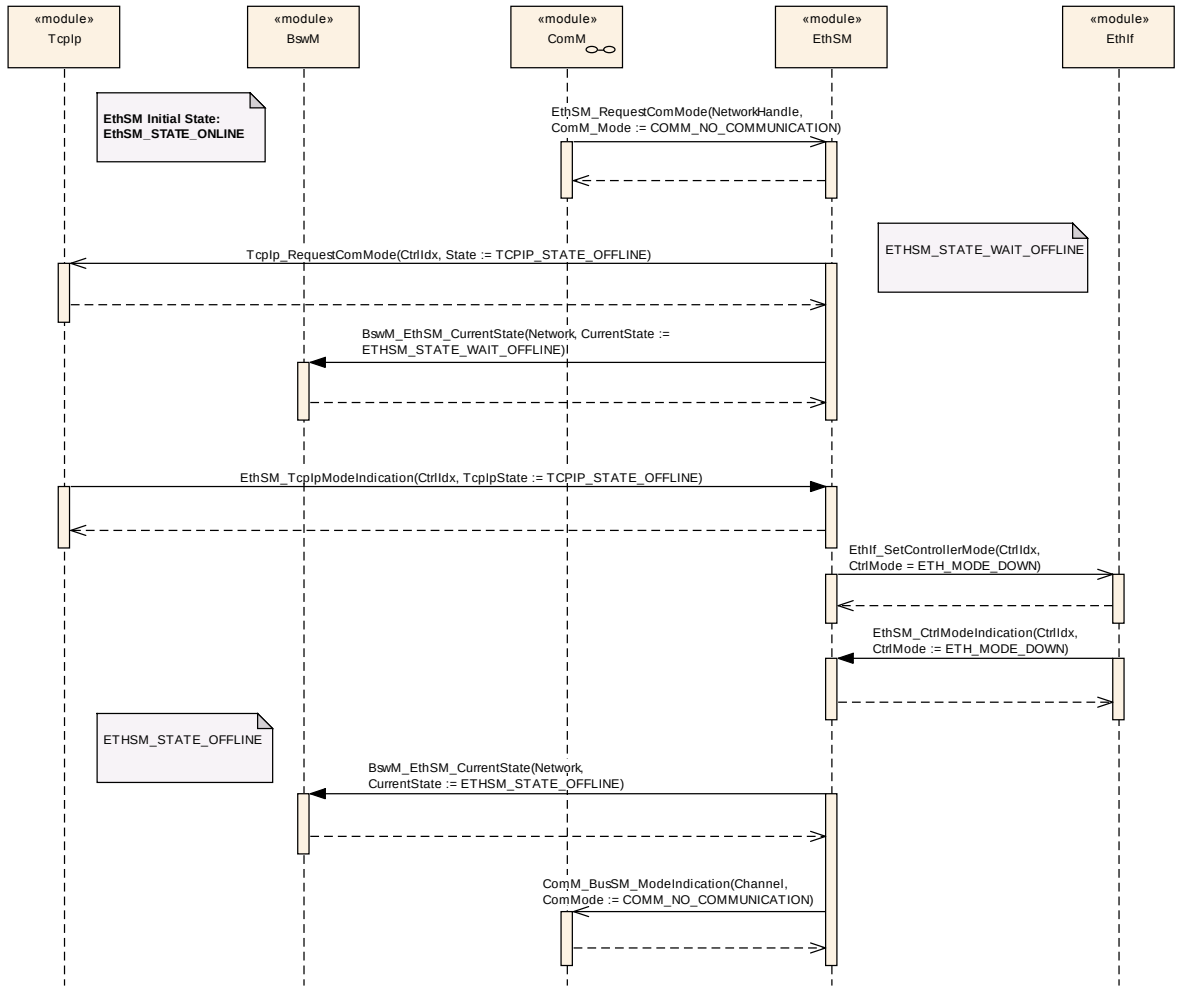


Figure 9-2.] ()

[SWS_EthSM_00157] [
After entering the state ETHSM_STATE_WAIT_OFFLINE, the API TcpIp_RequestComMode shall be called with TCPIP_STATE_OFFLINE.]()

[SWS_EthSM_00158][
After the successful transition from ETHSM_STATE_ONLINE to ETHSM_STATE_WAIT_OFFLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_WAIT_OFFLINE.] ()

7.6.7 Transition from substate 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. In this transition the EthSM shall interact like specified in the sequence diagram

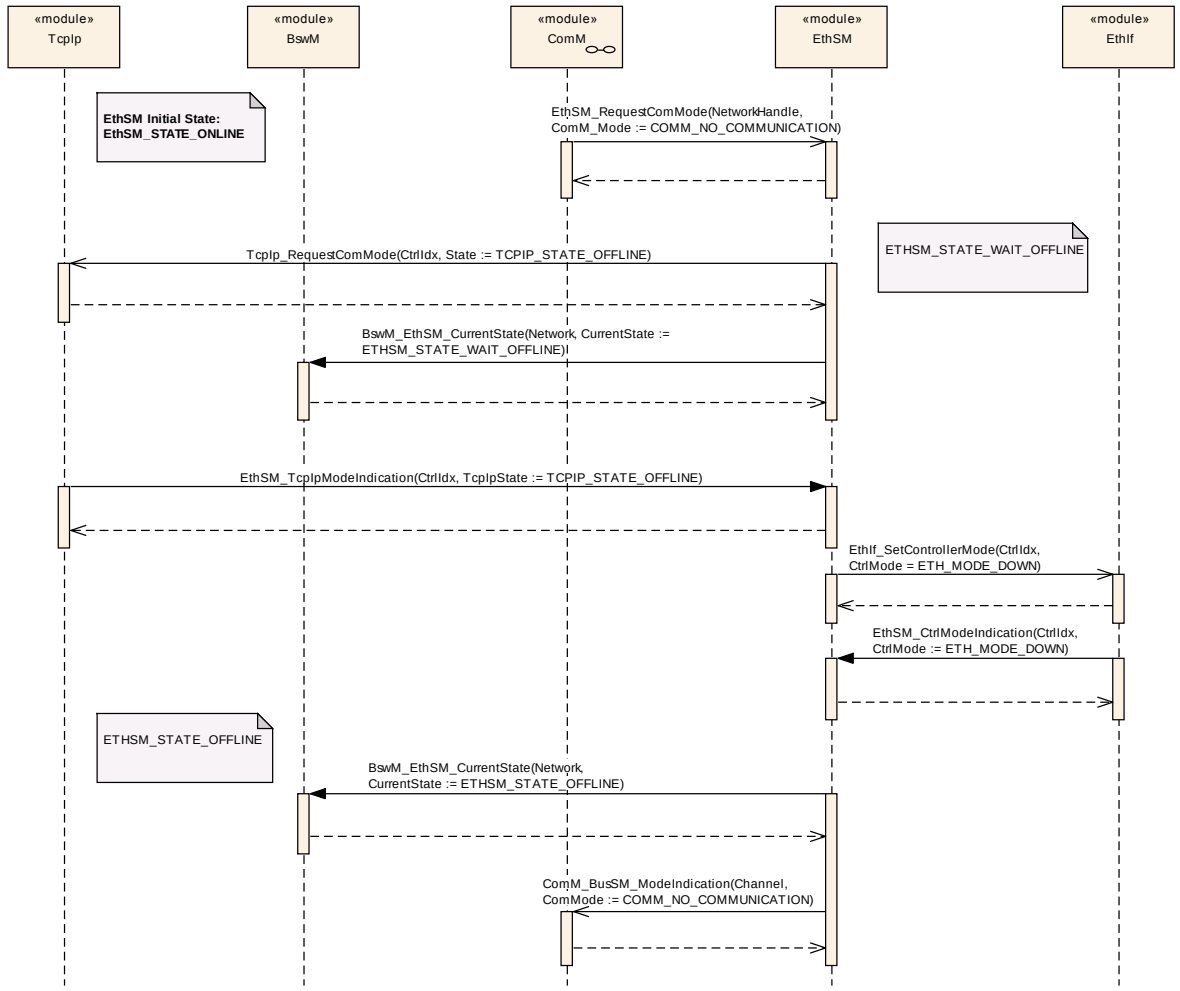


Figure 9-2.] ()

[SWS_EthSM_00161] [
The transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_OFFLINE sets the controller mode to ETH_MODE_DOWN.] ()

[SWS_EthSM_00163] [
After the successful transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_OFFLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_OFFLINE.] ()

[SWS_EthSM_00165] [
After the successful transition from ETHSM_STATE_WAIT_OFFLINE to ETHSM_STATE_OFFLINE the Ethernet State Manager shall call the callback function ComM_BusSM_ModeIndication of the ComM and transmit the communication mode (COMM_NO_COMMUNICATION).]()

7.6.8 Transition between substate ONLINE and 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. In this transition the EthSM shall interact like specified in the sequence diagram

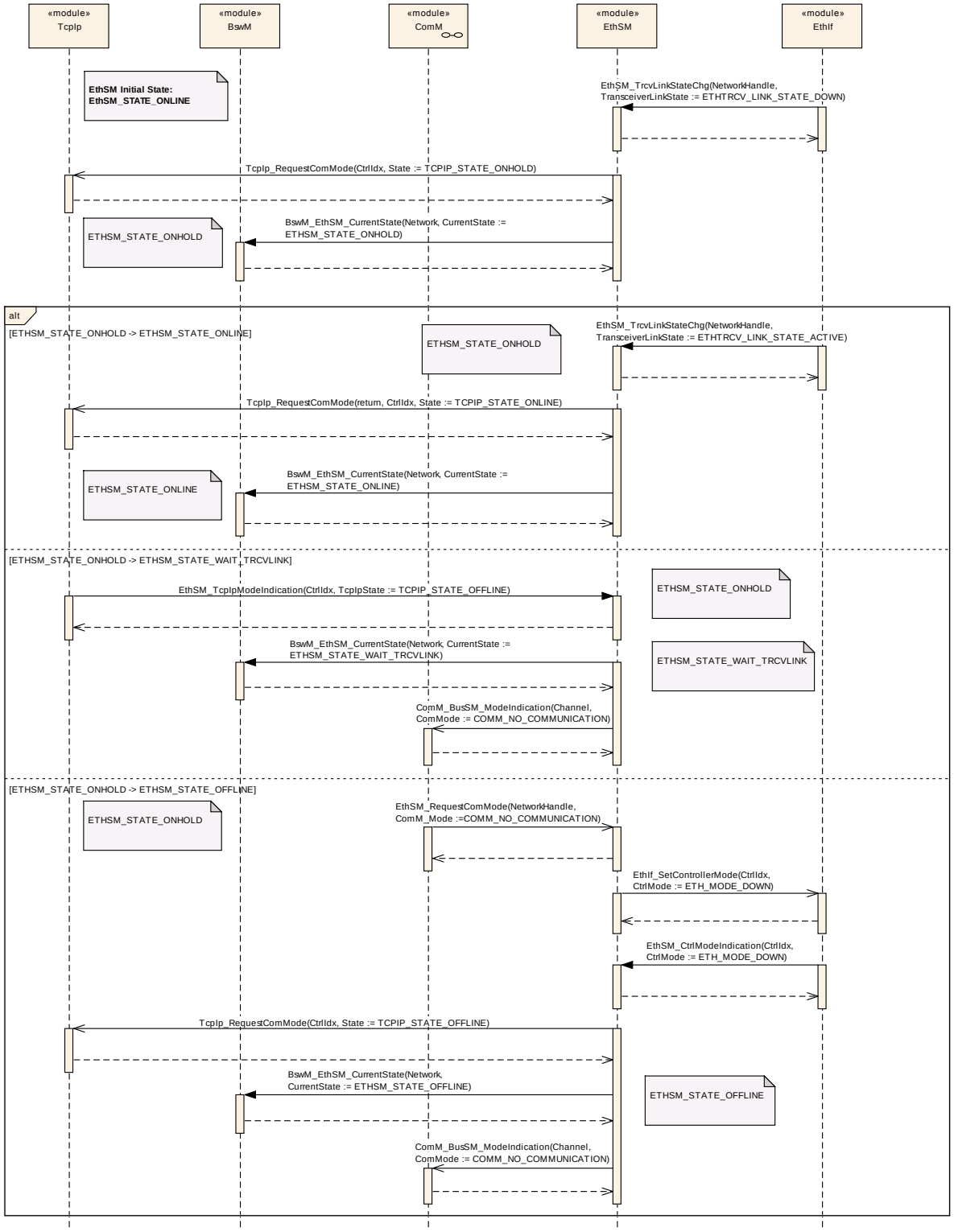


Figure 9-3.] ()

[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_00168]⌈
After the successful transition from ETHSM_STATE_ONLINE to ETHSM_STATE_ONHOLD the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_ONHOLD.⌋ ()

[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_00172]⌈
After the successful transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_ONLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_ONLINE.⌋ ()

[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.⌋ ()

[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.⌋ ()

7.6.9 Transition from substate 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. In this transition the EthSM shall interact like specified in the sequence diagram

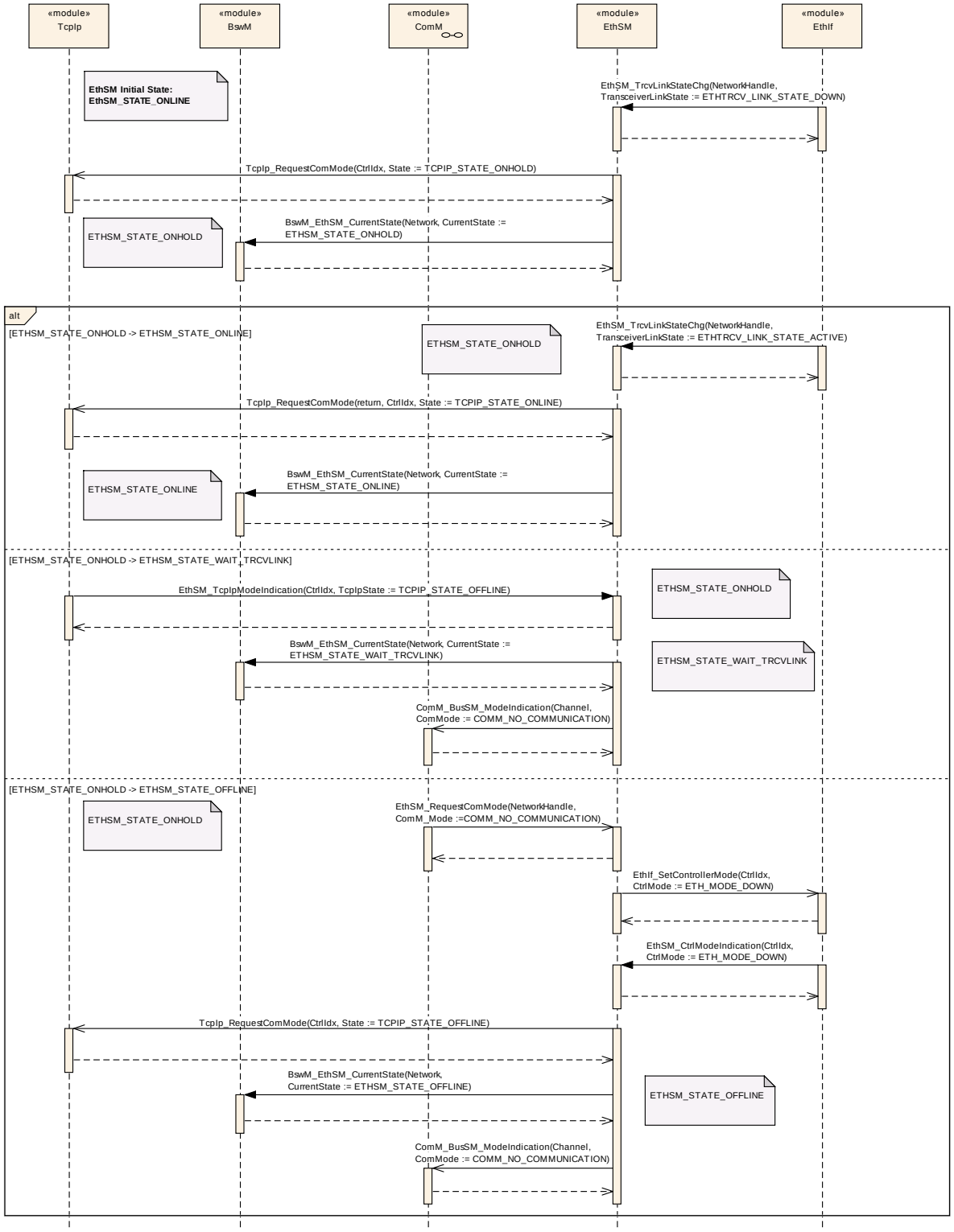


Figure 9-3.] ()

[SWS_EthSM_00175] [
After the successful transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_WAIT_TRCVLINK the Ethernet State Manager shall call the

callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_WAIT_TRCVLINK.] ()

[SWS_EthSM_00177] [
After the successful transition from ETHSM_STATE_ONHOLD to E ETHSM_STATE_WAIT_TRCVLINK the Ethernet State Manager shall call the callback function ComM_BusSM_ModeIndication of the ComM and transmit the communication mode (COMM_NO_COMMUNICATION).] ()

7.6.10 Transition from substate 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. In this transition the EthSM shall interact like specified in the sequence diagram

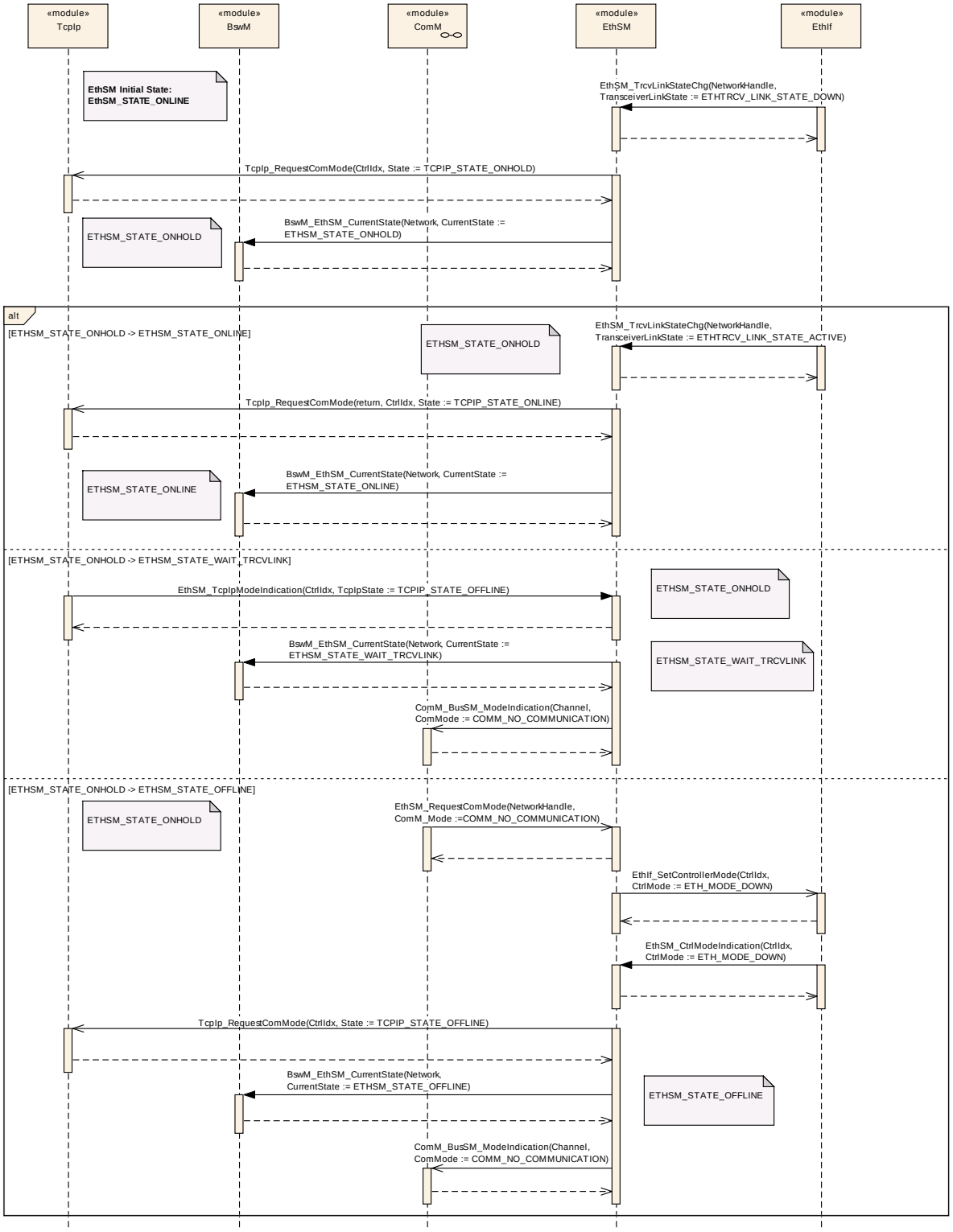


Figure 9-3.] ()

[SWS_EthSM_00179] [
The transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_OFFLINE sets 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_00182]⌈
After the successful transition from ETHSM_STATE_ONHOLD to ETHSM_STATE_OFFLINE the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_STATE_OFFLINE.⌋ ()

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

7.6.11 Information about state transitions

[SWS_EthSM_00083] ⌈
After the state machine has finished a state transition, the Ethernet State Manager has to inform the ComM and the BswM about the actual state of the Ethernet State Manager (see Figure 9-1 and

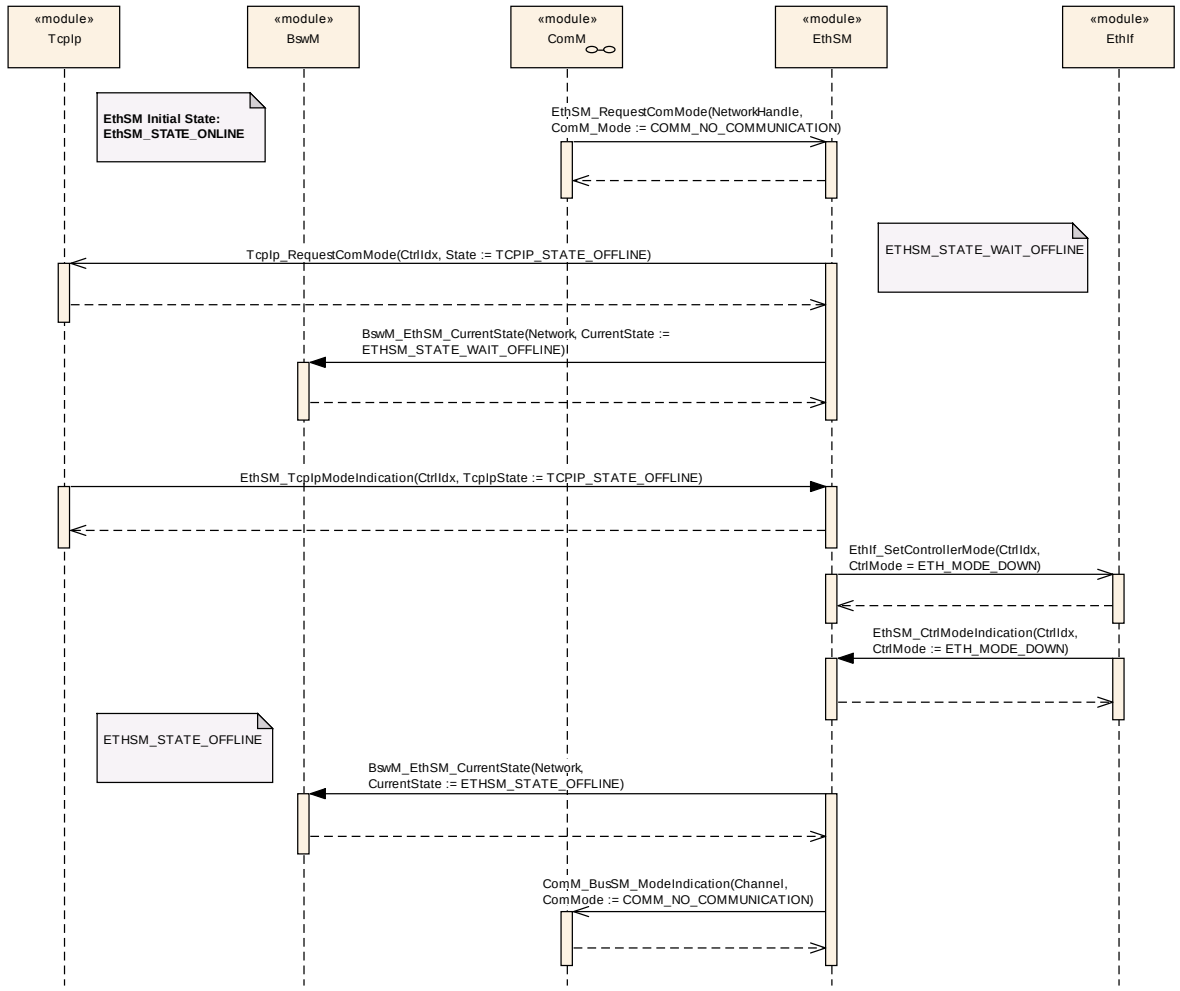


Figure 9-2).

The ComM needs the information about the communication states, e.g. COMM_FULL_COMMUNICATION or COMM_NO_COMMUNICATION.

The BswM needs the information about the EthSM internal states, see [SWS_EthSM_00041].⌋ ()

7.7 Error notification

For details refer to the chapters 7.2.1 “Error classification” & 7.2.2 “Development Errors” in *SWS_BSWGeneral.*

7.8 Error classification

7.8.1 Development Errors

[SWS_EthSM_00200]⌈

Type or error	*Related error code*	*Value [hex]*
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 Errors

7.8.3 Transcient 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 PASS: Configuration dependent	
Monitor Frequency	Continuous	
MIL illumniation:	N/A	

] ()

7.8.5 Extended Production Errors

There are no extended production errors.

7.9 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).

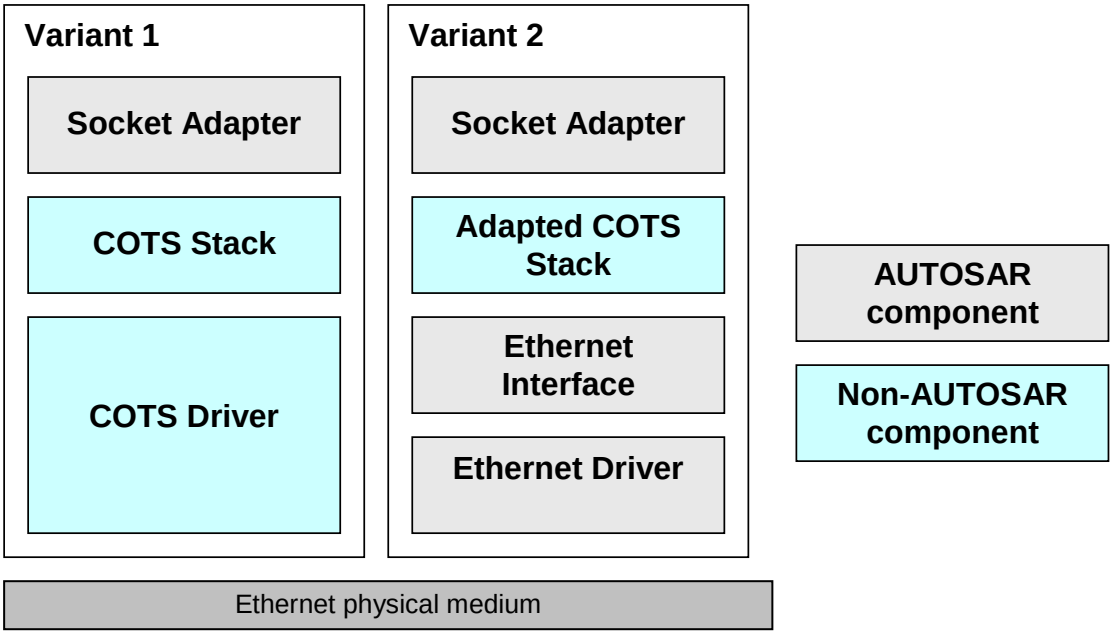


Figure 7-2: BSW stack architecture variants

[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.] ()

8 API specification

8.1 Imported types

[SWS_EthSM_91001] ⌈

Module	*Header File*	*Imported Type*
ComM	Rte_ComM_Type.h	ComM_ModeType
ComStack_Types	ComStackTypes.h	NetworkHandleType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Eth_GeneralTypes	Eth_GeneralTypes.h	EthTrcv_LinkStateType
	Eth_GeneralTypes.h	Eth_ModeType
Std_Types	StandardTypes.h	Std_ReturnType
	StandardTypes.h	Std_VersionInfoType
TcpIp	TcpIp.h	TcpIp_StateType

⌋ ()

8.2 Type definitions

8.2.1 EthSM_NetworkModeStateType

[SWS_EthSM_00041] ⌈

Name:	EthSM_NetworkModeStateType		
Type:	Enumeration		
Range:	ETHSM_STATE_OFFLINE	--	EthSM is initialized in this state.
	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 = ONLINE)
	ETHSM_STATE_ONLINE	--	IP communication is available ComM state COMM_FULL_COMMUNICATION is reached
	ETHSM_STATE_ONHOLD	--	EthSM lost active transceiver link state, TcpIP state is still ONLINE)
	ETHSM_STATE_WAIT_OFFLINE	--	ComM requests COMM_NO_-COMMUNICATION in this state.
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(`
	`  void`
	`)`

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(`
	`  Std_VersionInfoType* versioninfo`
	`)`	

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 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][

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_00087] ()

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_ModePtr	Pointer where to put out the current communication mode
Return value:	Std_ReturnType	E_OK: Service accepted 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
	TransceiverLinkState	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
	TcpIpState	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)

8.5 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(`
	`  void`
	`)`

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.
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*
Det_ReportError	Det.h	Service to report development errors.
EthIf_GetControllerMode	EthIf.h	Obtains the state of the indexed controller

⌋ ()

9 Sequence diagrams

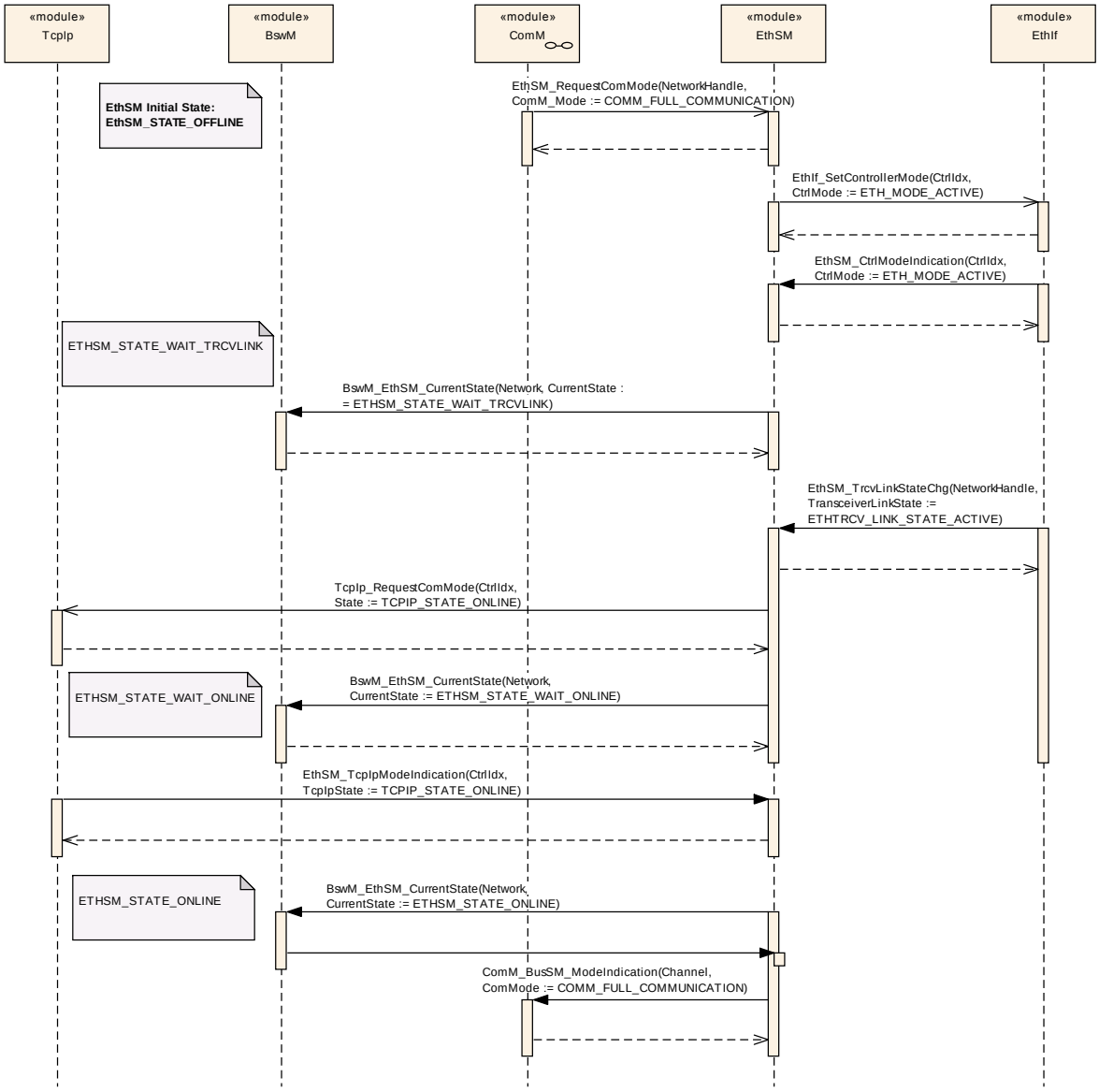


Figure 9-1: Network mode state machine – transition from no to full communication

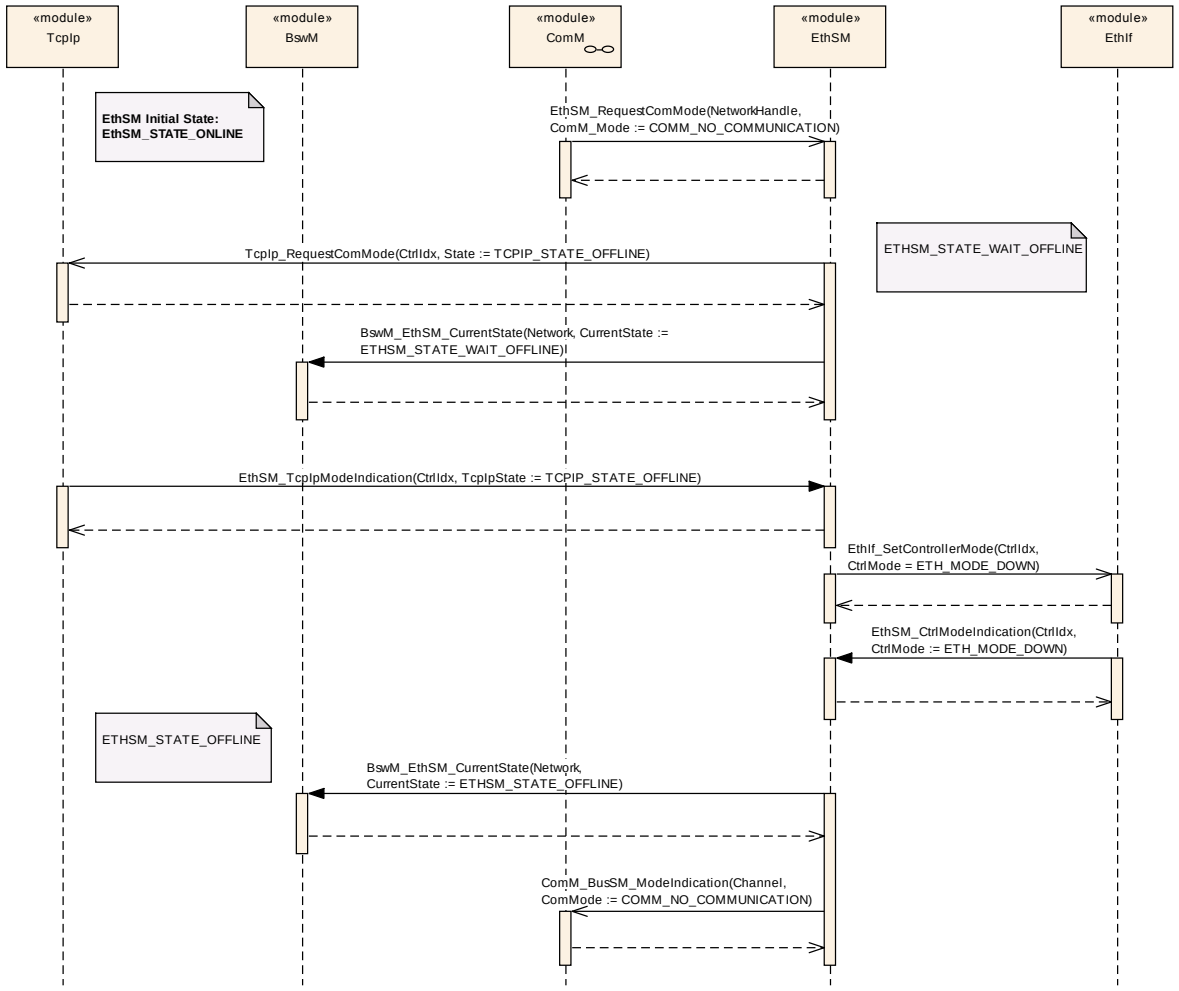


Figure 9-2: Network mode state machine – transition from full to no communication

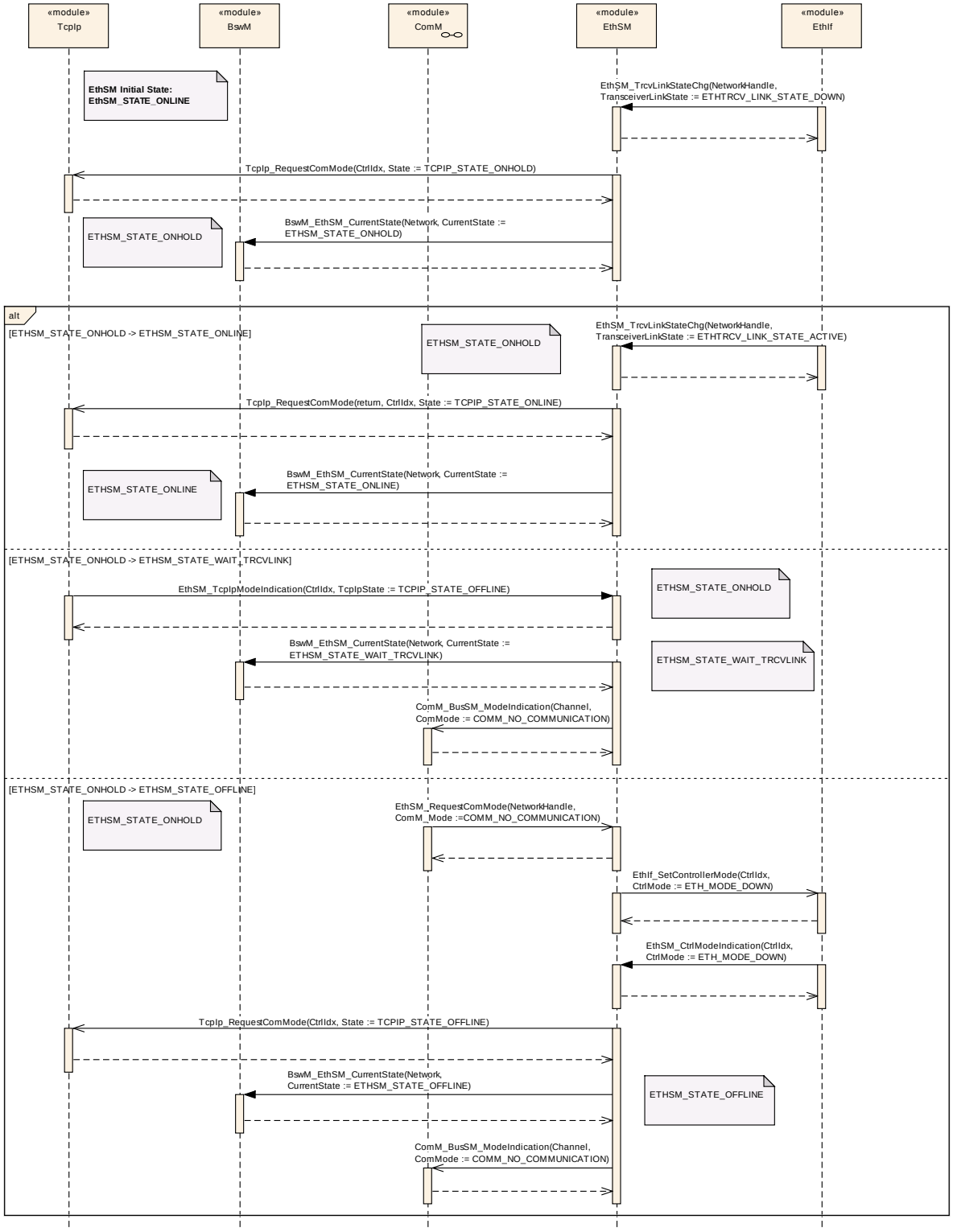


Figure 9-3: Network mode state machine – sub state ONHOLD

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.

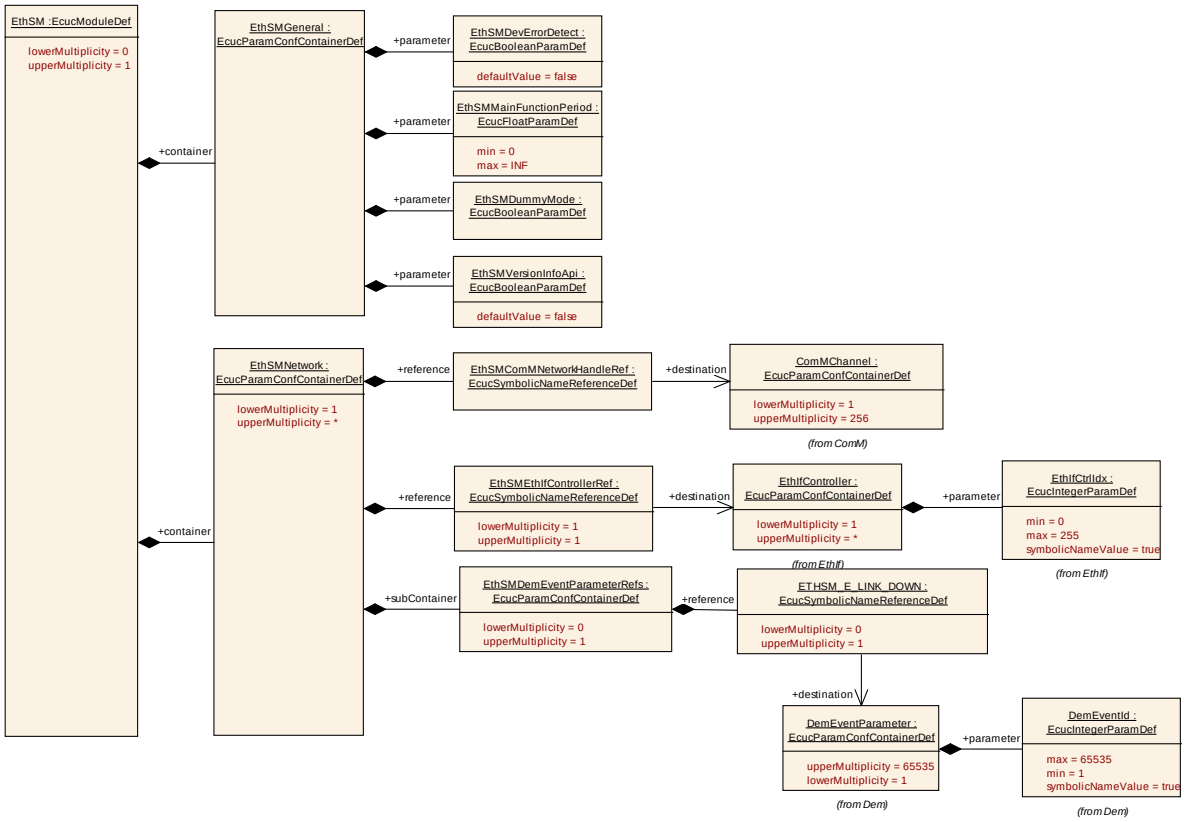
10.2.1 Configuration Tool

[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
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. • true: detection and notification is enabled. • 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
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_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
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)