Document Title	Specification of Ethernet State Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	415
Document Classification	Standard

Document Version	1.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
25.10.2011	1.2.0	AUTOSAR Administration	Update Chapter 10 (Parameter adjustment)
15.11.2010	1.1.0	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
07.12.2009	1.0.0	AUTOSAR Administration	Initial Release

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:

[ETHSM0001]

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

[ETHSM0002]

⌈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.

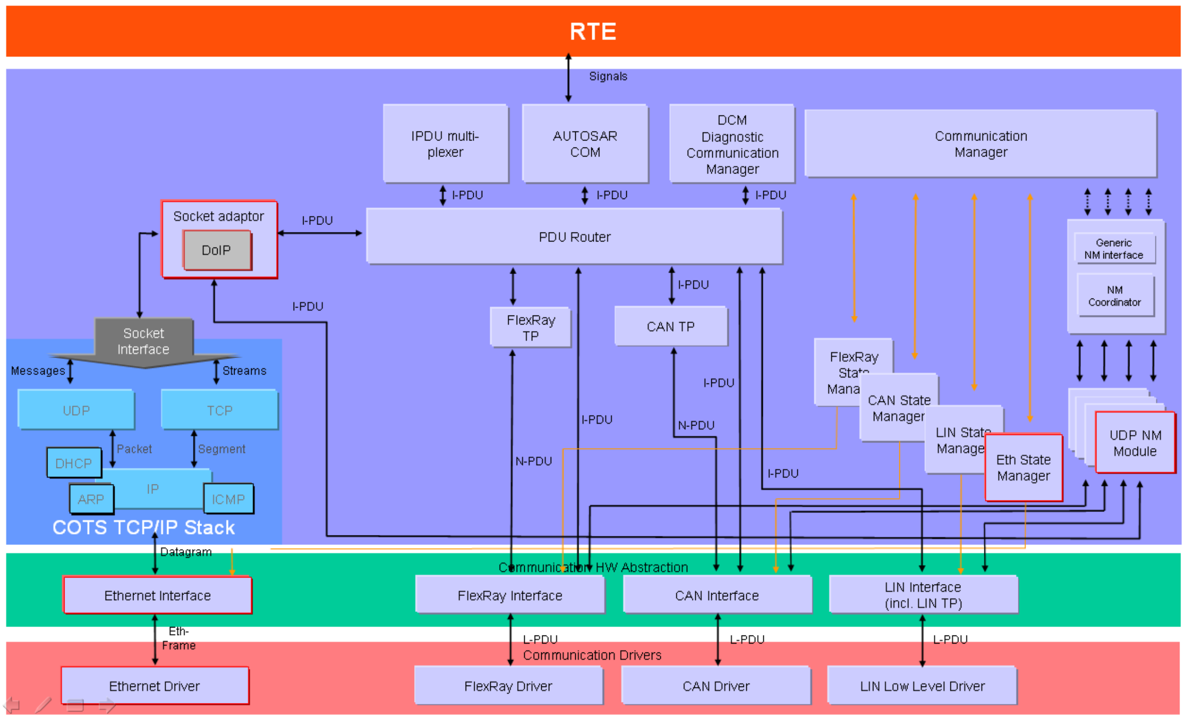


Figure 1-1: Example of an Autosar architecture including an Ethernet network

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	Development 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 Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

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

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

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 case of multiple Ethernet Controller (and/or multiple Ethernet Transceiver) of one ECU assigned to one Ethernet network is not fully defined. Thus, the current version can only be used for the case of one Ethernet Controller and one Ethernet transceiver per Ethernet network.

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.

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.

AUTOSAR Development 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 Development 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.

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

SocketAdapter
The SoAd is notified by the EthSM when an internal state is reached.

5.1 File structure

5.1.1 Code file structure

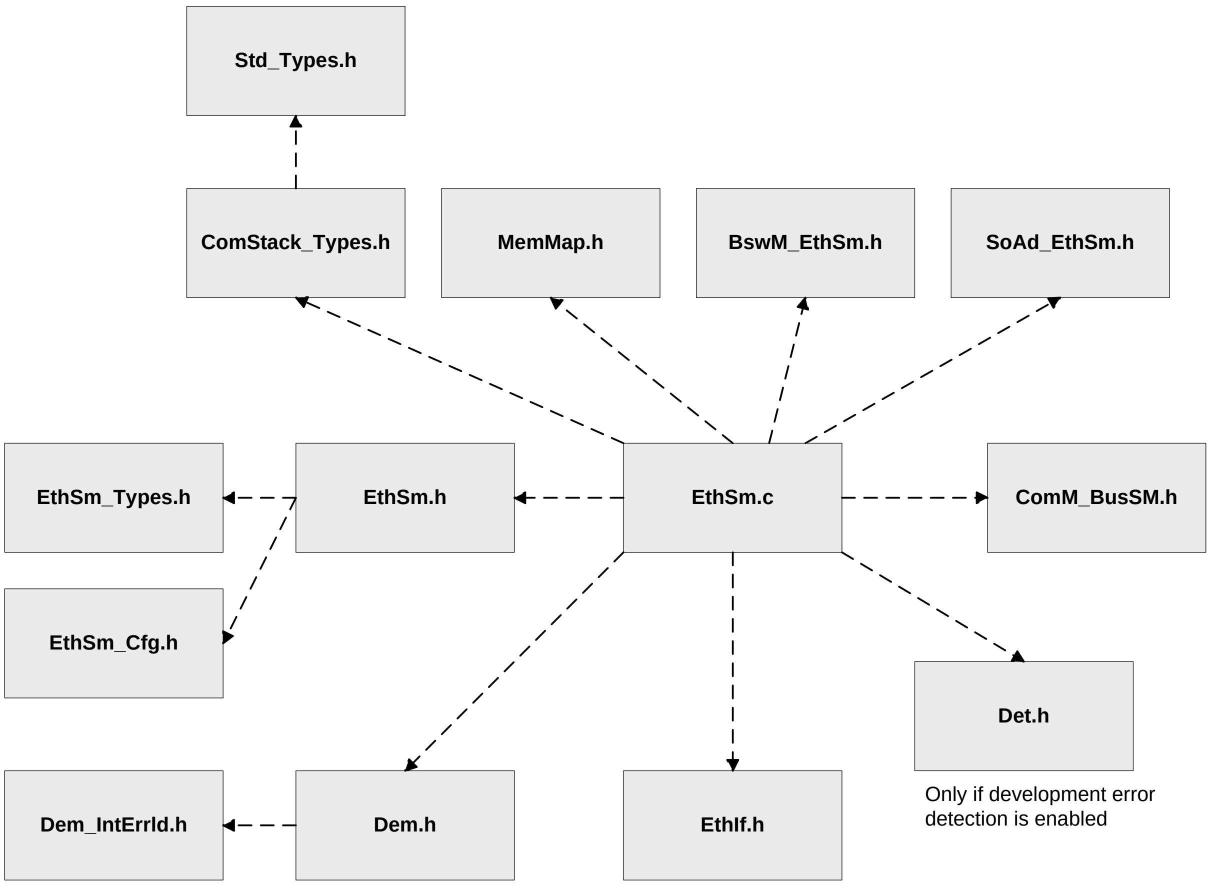
[ETHSM0003]⌈
The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:
- EthSM_Lcfg.c – for link time configurable parameters and
- EthSM_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.⌋ (BSW00380, BSW00419)

Actually the module EthSM doesn´t provide link time configuration and post-build time configuration and therefore the files EthSM_Lcfg.c and EthSM_PBcfg.c are currently not required.

Further more following files shall be used for implementation
- EthSM.c – for implementation of the provided functionality

5.1.2 Header file structure



[ETHSM0004] ⌈
The header file EthSM.h exports the API of the EthSM file includes further header files and declares the function prototypes, which are supposed to be referenced by user modules.⌋()

[ETHSM0005] ⌈
The header file EthSM_Cfg.h shall contain the pre-compile parameters of the module.⌋(BSW00381, BSW00412)

[ETHSM0006] ⌈
The header file EthSM_Types.h exports the EthSM specific types.⌋()

[ETHSM0007] ⌈
The EthSM implementation (EthSM.c) references its header file EthSM.h to get access to its own API declaration and to its configuration parameters.⌋()

[ETHSM0008] ⌈
The EthSM needs to report development errors if development errors are enabled by configuration. Therefore, it includes the header file Det.h.⌋()

[ETHSM0009] ⌈
The EthSM includes the header file MemMap.h in order to map its code and data into specific memory sections.⌋()

[ETHSM0010] ⌈
The EthSM implementation (EthSM.c) references the API of the EthIf. Therefore, it includes the header file EthIf.h.⌋()

[ETHSM0011] ⌈
The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.⌋()

[ETHSM0013] ⌈
The header file ComM_BusSM.h shall export the part of the ComM API required by EthSM.⌋()

[ETHSM0080] ⌈
The header file BswM_EthSM.h shall export the part of the BswM API required by EthSM.⌋()

[ETHSM0106]⌈
The header file SoAd_EthSM.h shall export the part of the SoAd API required by EthSM.⌋()

5.1.3 Version Check

[ETHSM0082] ⌈
The implementer of the EthSM module should avoid the integration of incompatible files and implement therefore a version check according to BSW004.

The EthSM module shall check the version defines ETHSM_AR_RELEASE_MAJOR_VERSION and ETHSM_AR_RELEASE_MINOR_VERSION of the included header files to be identical to the C file of the EthSM module.
The EthSM shall also check, that the version defines ETHSM_SW_MAJOR_VERSION, ETHSM_SW_MINOR_VERSION, ETHSM_AR_RELEASE_MAJOR_VERSION, ETHSM_AR_RELEASE_MINOR_VERSION and ETHSM_AR_RELEASE_REVISION_VERSION are identical to the C file of the EthSM module.⌋(BSW167, BSW004)

[ETHSM0107]⌈
The EthSM module shall perform Inter Module Checks to avoid integration of incompatible files. The imported included files shall be checked by preprocessing directives. ⌋()

The following version numbers shall be verified:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION
Where <MODULENAME> is the module short name of the other (external) modules which provide header files included by the EthSM module.

If the values are not identical to the expected values, an error shall be reported.

6 Requirements traceability

Requirement	Satisfied by
-	ETHSM0099
-	ETHSM0032
-	ETHSM0021
-	ETHSM0041
-	ETHSM0072
-	ETHSM0086
-	ETHSM0020
-	ETHSM0106
-	ETHSM0102
-	ETHSM0050
-	ETHSM0078
-	ETHSM0105
-	ETHSM0100
-	ETHSM0013
-	ETHSM0055
-	ETHSM0035
-	ETHSM0009
-	ETHSM0090
-	ETHSM0022
-	ETHSM0007
-	ETHSM0010
-	ETHSM0083
-	ETHSM0038
-	ETHSM0075
-	ETHSM0015
-	ETHSM0027
-	ETHSM0107
-	ETHSM0001
-	ETHSM0071
-	ETHSM0002
-	ETHSM0052
-	ETHSM0093
-	ETHSM0057
-	ETHSM0085
-	ETHSM0073
-	ETHSM0077

-	ETHSM0014
-	ETHSM0011
-	ETHSM0006
-	ETHSM0045
-	ETHSM0023
-	ETHSM0008
-	ETHSM0018
-	ETHSM0084
-	ETHSM0076
-	ETHSM0019
-	ETHSM0096
-	ETHSM0004
-	ETHSM0095
-	ETHSM0089
-	ETHSM0103
-	ETHSM0088
-	ETHSM0059
-	ETHSM0029
-	ETHSM0016
-	ETHSM0024
-	ETHSM0017
-	ETHSM0058
-	ETHSM0097
-	ETHSM0026
-	ETHSM0049
-	ETHSM0091
-	ETHSM0053
-	ETHSM0028
-	ETHSM0047
-	ETHSM0098
-	ETHSM0025
-	ETHSM0087
-	ETHSM0051
-	ETHSM0101
-	ETHSM0074
-	ETHSM0080
BSW003	ETHSM0046, ETHSM0060
BSW00306	ETHSM999
BSW00308	ETHSM999
BSW00309	ETHSM999
BSW00314	ETHSM999

BSW00318	ETHSM0060
BSW00321	ETHSM999
BSW00323	ETHSM0034
BSW00325	ETHSM999
BSW00326	ETHSM999
BSW00328	ETHSM999
BSW00331	ETHSM999
BSW00333	ETHSM999
BSW00334	ETHSM999
BSW00335	ETHSM0039
BSW00336	ETHSM999
BSW00341	ETHSM999
BSW00343	ETHSM999
BSW00344	ETHSM999
BSW00347	ETHSM999
BSW00353	ETHSM999
BSW00355	ETHSM999
BSW00358	ETHSM0043
BSW00359	ETHSM999
BSW00360	ETHSM999
BSW00361	ETHSM999
BSW00369	ETHSM999
BSW00371	ETHSM999
BSW00373	ETHSM999
BSW00374	ETHSM0060
BSW00375	ETHSM999
BSW00377	ETHSM999
BSW00379	ETHSM0060
BSW00380	ETHSM0003
BSW00381	ETHSM0005
BSW00386	ETHSM0033, ETHSM0031, ETHSM0030
BSW00387	ETHSM999
BSW00395	ETHSM999
BSW00398	ETHSM999
BSW00399	ETHSM999
BSW004	ETHSM0082
BSW00400	ETHSM999
BSW00406	ETHSM0054, ETHSM0060
BSW00407	ETHSM0046
BSW00411	ETHSM0048
BSW00412	ETHSM0005

BSW00413	ETHSM999
BSW00414	ETHSM0043, ETHSM0044, ETHSM0039
BSW00416	ETHSM999
BSW00417	ETHSM999
BSW00419	ETHSM0003
BSW00423	ETHSM999
BSW00426	ETHSM999
BSW00427	ETHSM999
BSW00428	ETHSM999
BSW00429	ETHSM999
BSW00431	ETHSM999
BSW00432	ETHSM999
BSW00433	ETHSM999
BSW00434	ETHSM999
BSW00437	ETHSM999
BSW00438	ETHSM999
BSW005	ETHSM999
BSW010	ETHSM999
BSW0404	ETHSM999
BSW0405	ETHSM0043, ETHSM0039
BSW101	ETHSM0043
BSW160	ETHSM999
BSW161	ETHSM999
BSW162	ETHSM999
BSW164	ETHSM999
BSW167	ETHSM0082
BSW168	ETHSM999
BSW170	ETHSM999

According to [3] AUTOSAR General Requirements on Basic Software Modules (General BSW Requirements):

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	Not applicable
[BSW0404] Reference to post build time configuration	Not applicable
[BSW0405] Reference to multiple configuration sets	[ETHSM0039, [ETHSM0043
[BSW00345] Pre-compile time configuration	[ETHSM0061
[BSW159] Tool based configuration	[ETHSM0081

[BSW167] Static configuration checking	[ETHSM0082
[BSW171] Configurability of optional functionality	Chapter 10.2, 7.7
[BSW170] Data for reconfiguration of SW-components	Not applicable (requirement on SWC-module)
[BSW00380] Separate C-Files for configuration parameters	[ETHSM0003
[BSW00419] Separate C-Files for pre-compile time configuration parameters	[ETHSM0003
[BSW00381] Separate configuration header file for pre-compile time parameters	[ETHSM0005
[BSW00412] Separate configuration header file for configuration parameters	[ETHSM0005
[BSW00383] List dependencies of configuration files	Chapter 10.2
[BSW00384] List dependencies to other modules	Chapter 5
[BSW00387] Specify the configuration class of callback function	Not applicable
[BSW00388] Introduce containers	Chapter 10.2
[BSW00389] Containers shall have names	Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2
[BSW00391] Parameter shall have unique names	Chapter 10.2
[BSW00392] Parameters shall have a type	Chapter 10.2
[BSW00393] Parameters shall have a range	Chapter 10.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2
[BSW00395] List the required parameters (per parameter)	Not applicable
[BSW00396] Configuration classes	Chapter 10.2
[BSW00397] Pre--compile--time parameters	Chapter 10.2
[BSW00398] Link--time parameters	Not applicable
[BSW00399] Loadable Post--build time parameters	Not applicable
[BSW00400] Selectable Post--build time parameters	Not applicable
[BSW00438] Post Build Configuration Data Structure	Not applicable
[BSW00402] Published information	Chapter 10.3
[BSW00375] Notification of wake-up reason	Not applicable (no wake up interrupt)
[BSW101] Initialization interface	[ETHSM0043

[BSW00416] Sequence of Initialization	Not applicable
[BSW00406] Check module initialization	[ETHSM0054, [ETHSM0060
[BSW00437] NoInit--Area in RAM	Not applicable (not in scope of this spec)
[BSW168] Diagnostic interface	Not applicable (requirement on SWC-module)
[BSW00407] Function to read out published parameters	[ETHSM0046
[BSW00423] Usage of SW--C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (not in scope of this spec)
[BSW00424] BSW main processing function task allocation	[ETHSM0081
[BSW00425] Trigger conditions for schedulable objects	[ETHSM0081
[BSW00426] Exclusive areas in BSW modules	Not applicable (not in scope of this spec)
[BSW00427] ISR description for BSW modules	Not applicable (not in scope of this spec)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (not in scope of this spec)
[BSW00429] Restricted BSW OS functionality access	Not applicable (not in scope of this spec)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (not in scope of this spec)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (not in scope of this spec)
[BSW00433] Calling of main processing functions	Not applicable (not in scope of this spec)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (not in scope of this spec)
[BSW00336] Shutdown interface	Not applicable (no deinitialisation function)
[BSW00337] Classification of errors	Chapter 7.6
[BSW00338] Detection and Reporting of development errors	Chapter 7.7
[BSW00369] Do not return development error codes via API	Not applicable
[BSW00339] Reporting of production relevant errors and exceptions	[ETHSM0034
[BSW00422] Pre--de--bouncing of production relevant error status	Chapter 7.6
[BSW00417] Reporting of Error Events by Non--Basic Software	Not applicable (not in scope of this spec)
[BSW00323] API parameter checking	Chapter 8.3, [ETHSM0031
[BSW004] Version check	[ETHSM0082
[BSW00409] Header files for production	Chapter 7.6

code error IDs	
[BSW00385] List possible error notifications	Chapter 7.6, 8.3
[BSW00386] Configuration for detecting an error	[ETHSM0030, [ETHSM0031, [ETHSM0033
[BSW161] Microcontroller abstraction	Not applicable (not in scope of this spec)
[BSW162] ECU layout abstraction	Not applicable (not in scope of this spec)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (not in scope of this spec)
[BSW00415] User dependent include files	Chapter 5.1.2
[BSW164] Implementation of interrupt service routines	Not applicable (not in scope of this spec)
[BSW00325] Runtime of interrupt service routines	Not applicable (not in scope of this spec)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (not in scope of this spec)
[BSW00342] Usage of source code and object code	Chapter 10.2
[BSW00343] Specification and configuration of time	Not applicable (not in scope of this spec)
[BSW160] Human--readable configuration data	Not applicable (not in scope of this spec)
[BSW007] HIS MISRA C	This is mostly a requirement on the construction and not the design (i.e. SWS). The API chapter 8 is following MISRA C
[BSW00300] Module naming convention	Chapter 5.1
[BSW00413] Accessing instances of BSW modules	Not applicable (not in scope of this spec)
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (not in scope of this spec)
[BSW00305] Self--defined data types naming convention	Chapter 8.2
[BSW00307] Global variables naming convention	Chapter 10.2
[BSW00310] API naming convention	Chapter 8.3
[BSW00373] Main processing function naming convention	Not applicable
[BSW00327] Error values naming convention	Chapter 7.6
[BSW00335] Status values naming convention	[ETHSM0039
[BSW00350] Development error detection keyword	Chapter 7.7
[BSW00408] Configuration parameter naming convention	Chapter 10.2
[BSW00410] Compiler switches shall	Chapter 10.2

have defined values	
[BSW00411] Get version info keyword	[ETHSM0048, Chapter 10.2
[BSW00346] Basic set of module files	Chapter 5.1
[BSW158] Separation of configuration from implementation	Chapter 5.1
[BSW00314] Separation of interrupt frames and service routines	Not applicable (not in scope of this spec)
[BSW00370] Separation of callback interface from API	Chapter 5.1
[BSW00435] Header File Structure for the Basic Software Scheduler	Chapter 5.1
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Chapter 5.1
[BSW00348] Standard type header	Chapter 5.1
[BSW00353] Platform specific type header	Not applicable (not in scope of this spec)
[BSW00361] Compiler specific language extension header	Not applicable (not in scope of this spec)
[BSW00301] Limit imported information	Chapter 5.1
[BSW00302] Limit exported information	Chapter 5.1
[BSW00328] Avoid duplication of code	Not applicable (not in scope of this spec)
[BSW00312] Shared code shall be reentrant	Chapter 8.3
[BSW006] Platform independency	Figure 1-1
[BSW00357] Standard API return type [	Chapter 8.3
[BSW00377] Module specific API return types	Not applicable (not used)
[BSW00304] AUTOSAR integer data types	Chapter 10.2
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (not used)
[BSW00378] AUTOSAR boolean type	Chapter 10.2
[BSW00306] Avoid direct use of compiler and platform specific keywords [	Not applicable (not used)
[BSW00308] Definition of global data	Not applicable (not used)
[BSW00309] Global data with read--only constraint	Not applicable (not used)
[BSW00371] Do not pass function pointers via API	Not applicable
[BSW00358] Return type of init() functions	[ETHSM0043
[BSW00414] Parameter of init function	[ETHSM0039, [ETHSM0043, [ETHSM0044
[BSW00376] Return type and parameters of main processing functions	Chapter 8.3
[BSW00359] Return type of callback functions	Not applicable

[BSW00360] Parameters of callback functions	Not applicable
[BSW00329] Avoidance of generic interfaces	Generic interfaces are not used
[BSW00330] Usage of macros / inline functions instead of functions	No restriction
[BSW00331] Separation of error and status values	Not applicable
[BSW009] Module User Documentation	usage of SWS Template
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10.2
[BSW172] Compatibility and documentation of scheduling strategy	Chapter 8
[BSW010] Memory resource documentation	Not applicable (not in scope of this spec)
[BSW00333] Documentation of callback function context	Not applicable
[BSW00374] Module vendor identification	[ETHSM0060
[BSW00379] Module identification	[ETHSM0060
[BSW003] Version identification	[ETHSM0060, [ETHSM0046
[BSW00318] Format of module version numbers	[ETHSM0060
[BSW00321] Enumeration of module version numbers	Not applicable
[BSW00341] Microcontroller compatibility documentation	Not applicable (not in scope of this spec)
[BSW00334] Provision of XML file	Not applicable

Document: AUTOSAR requirements on Basic Software, cluster Ethernet

Requirement	***Satisfied by***
[BSW41900001] SoAd TCP connection setup	Not relevant
[BSW41900002] SoAd IP address configuration	Not relevant
[BSW41900004] SoAd Multi-homed hosts	Not relevant
[BSW41900005] SoAd Use of UDP and TCP	Not relevant
[BSW41900006] SoAd No Protocol overhead	Not relevant
[BSW41900007] SoAd COTS Compatibility	Not relevant
[BSW41900008] SoAd immediate retry	Not relevant
[BSW41900009] SoAd Connection shutdown	Not relevant
[BSW41900010] SoAd Resource management	Not relevant
[BSW41900011] SoAd Resource	Not relevant

predictability	
[BSW41900012] SoAd No buffer memory	Not relevant
[BSW41900013] SoAd Reduced Copy operation	Not relevant
[BSW41900014] TCPIP IPv4 implementation	Not relevant
[BSW41900015] TCPIP ARP implementation	Not relevant
[BSW41900016] TCPIP ICMP implementation	Not relevant
[BSW41900017] TCPIP TCP implementation	Not relevant
[BSW41900018] TCPIP UDP implementation	Not relevant
[BSW41900019] TCPIP TCP+UDP implementation	Not relevant
[BSW41900020] TCPIP DHCP implementation	Not relevant
[BSW41900021] TCPIP DHCP “host name option” implementation	Not relevant
[BSW41900022] TCPIP link local IP implementation	Not relevant
[BSW41900024] DoIP routing	Not relevant
[BSW41900025] DoIP message recognition	Not relevant
[BSW41900026] DoIP Vehicle Identification	Not relevant
[BSW41900027] DoIP diagnostic message	Not relevant
[BSW41900028] DoIP Socket handling	Not relevant
[BSW41900029] EthIf: Interface of the module	Not relevant
[BSW41900030] EthIf: Hardware abstraction	Not relevant
[BSW41900031] EthIf: Interrupt / Polling mode	Not relevant
[BSW41900032] EthIf: Hardware configuration and initialization	Not relevant
[BSW41900033] EthIf: Link state change indication	Not relevant
[BSW41900034] Eth: Hardware abstraction	Not relevant
[BSW41900035] Eth: Interrupt / Polling mode	Not relevant
[BSW41900036] Eth: Hardware configuration and initialization	Not relevant
[BSW41900037] UdpNm: Network management information	Not relevant

[BSW41900038] EthTrcv: Hardware abstraction	Not relevant
[BSW41900039] EthTrcv: Hardware configuration and initialization	Not relevant
[BSW41900040] EthTrcv: Link state change indication	Not relevant
[BSW41900041] EthSM: Network abstraction	Chapter 7.3, 7.4, 7.5, 8.3.3, 9, 10.2
[BSW41900042] UdpNm: Network abstraction	Not relevant
[BSW41900043] EthSM: Network configuration and initialization	Chapter 8 and 10.2
[BSW41900044] The TCP/IP stack is not an AUTOSAR module	Not relevant
[BSW41900045] TCPIP automatic IP address assignment	Not relevant
[BSW41900046] SoAd DoIP implementation	Not relevant
[BSW41900047] DoIp DHCP "host name option" access	Not relevant
[BSW41900048] SoAd PDU routing	Not relevant

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

[ETHSM0014] ⌈
The EthSM shall provide to the ComM an API, which can be used by the ComM to request communication modes of Ethernet networks.⌋()

[ETHSM0015]⌈
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 7.5). ⌋()

[ETHSM0016]⌈
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:

[ETHSM0017]⌈
The EthSM shall provide an API, which can be polled by the ComM to get the current communication mode of an Ethernet network. ⌋()

[ETHSM0018]⌈
The EthSM shall use a call out function of ComM to notify ComM of a change in communication modes.⌋()

7.3 Control of peripherals

7.3.1 Ethernet Transceivers

One or more Ethernet transceivers belong to a certain Ethernet network (handle).

[ETHSM0019]⌈
The assignment between network handles and transceivers shall be part of the EthSM configuration (see chapter 10.2).⌋()

[ETHSM0020]⌈
The EthSM shall control the Ethernet transceivers depending on the state transitions of its network mode state machines.⌋()

[ETHSM0021]⌈
The EthSM shall use the API of the EthIf for the control of the Ethernet transceiver modes.⌋()

7.3.2 Ethernet Controllers

One or more Ethernet controllers belong to a certain Ethernet network (handle).

[ETHSM0022]⌈
Depending on the network mode state machine, the EthSM shall control the Ethernet controller modes of each Ethernet network. ⌋()

[ETHSM0023]⌈
The EthSM shall use the API of the EthIf to control the operating modes of the assigned Ethernet controllers.⌋()

7.4 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

[ETHSM0038] ⌈

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

[ETHSM0085] ⌈
If FULL_COMMUNICATION is requested the Ethernet State Manager first has to initialize the Ethernet controller and the Ethernet transceiver. After the correct initialization, the Ethernet controller and the Ethernet transceiver are set to the state ACTIVE. ⌋()

[ETHSM0086]⌈
If the ComM request NO_COMMUNICATION the Ethernet controller and the Ethernet transceiver are set to the state DOWN.⌋()

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

[ETHSM0087]⌈
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.⌋()

7.5 Network mode state machine

[ETHSM0024]⌈
The EthSM shall implement for each configured network handle one network mode state machine, which is specified in Figure 7-1. The internal states are described in [ETHSM0041.⌋()

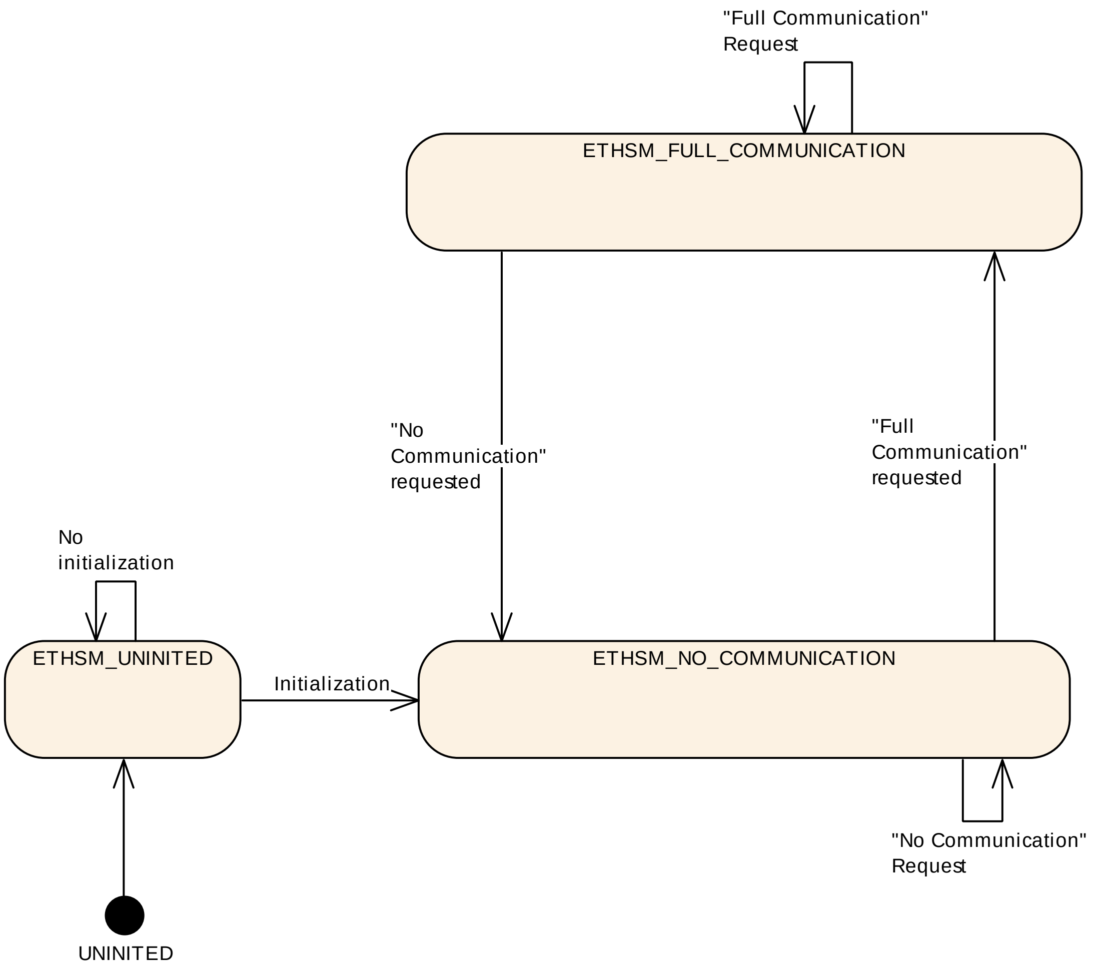

"Full Communication" Request
ETHSM_FULL_COMMUNICATION
"No Communication" requested
"Full Communication" requested
No initialization
ETHSM_UNINITED
Initialization
ETHSM_NO_COMMUNICATION
"No Communication" Request
UNINITED

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

7.5.1 Initial transition

[ETHSM0025] ⌈
In the state ETHSM_UNINITED the state machine shall have a transition to ETHSM_NO_COMMUNNICATION, if the EthSM is initialized.
Remark:
The initialization of the EthSM causes no further transactions in other modules. So no separate sequence diagram is needed.⌋()

7.5.2 Transition from no to full communication

[ETHSM0026] ⌈
In the state ETHSM_NO_COMMUNICATION the state machine shall have a transition to ETHSM_FULL_COMMUNICATION, 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.⌋ ()

[ETHSM0088] ⌈
The transition from ETHSM_NO_COMMUNICATION to ETHSM_FULL_COMMUNICATION shall initialize the controller and set the controller mode to ETH_MODE_ACTIVE. ⌋()

[ETHSM0089] ⌈
The transition from ETHSM_NO_COMMUNICATION to ETHSM_FULL_COMMUNICATION shall initialize the transceiver and set the transceiver mode to ETHTRCV_MODE_ACTIVE.⌋()

[ETHSM0096] ⌈
After the successful transition from ETHSM_NO_COMMUNICATION to ETHSM_FULL_COMMUNICATION the Ethernet State Manager shall call the callback function ComM_BusSM_ModeIndication of the ComM and transmit the communication mode (COMM_FULL_COMMUNICATION). ⌋()

[ETHSM0097]⌈
After the successful transition from ETHSM_NO_COMMUNICATION to ETHSM_FULL_COMMUNICATION the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_FULL_COMMUNICATION.⌋()

[ETHSM0098]⌈
After a failed transition from ETHSM_NO_COMMUNICATION to ETHSM_FULL_COMMUNICATION the Ethernet State Manager shall call the callback function ComM_BusSM_ModeIndication of the ComM and transmit the furthermore valid communication mode (COMM_NO_COMMUNICATION).⌋()

[ETHSM0099]⌈
After a failed transition from ETHSM_NO_COMMUNICATION to ETHSM_FULL_COMMUNICATION the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the furthermore valid internal state ETHSM_NO_COMMUNICATION.⌋()

7.5.3 Transition from full to no communication

[ETHSM0027] ⌈
In the state ETHSM_FULL_COMMUNICATION the state machine shall have a transition to ETHSM_NO_COMMUNICATION, 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. ⌋()

[ETHSM0090]⌈
The transition from ETHSM_FULL_COMMUNICATION to ETHSM_NO_COMMUNICATION sets the controller mode to ETH_MODE_DOWN. ⌋ ()

[ETHSM0091]⌈
The transition from ETHSM_FULL_COMMUNICATION to ETHSM_NO_COMMUNICATION sets the transceiver mode to ETHTRCV_MODE_DOWN.⌋()

[ETHSM0100]⌈
After the successful transition from ETHSM_FULL_COMMUNICATION to ETHSM_NO_COMMUNICATION the Ethernet State Manager shall call the callback function ComM_BusSM_ModeIndication of the ComM and transmit the communication mode (COMM_NO_COMMUNICATION). ⌋()

[ETHSM0101] ⌈
After the successful transition from ETHSM_FULL_COMMUNICATION to ETHSM_NO_COMMUNICATION the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the internal state ETHSM_NO_COMMUNICATION.⌋()

[ETHSM0102] ⌈
After a failed transition from ETHSM_FULL_COMMUNICATION to ETHSM_NO_COMMUNICATION the Ethernet State Manager shall call the callback function ComM_BusSM_ModeIndication of the ComM and transmit the furthermore valid communication mode (COMM_FULL_COMMUNICATION).⌋()

[ETHSM0103] ⌈
After a failed transition from ETHSM_FULL_COMMUNICATION to ETHSM_NO_COMMUNICATION the Ethernet State Manager shall call the callback function BswM_EthSM_CurrentState of the BswM and transmit the furthermore valid internal state ETHSM_FULL_COMMUNICATION.⌋()

7.5.4 Information about state transitions

[ETHSM0083] ⌈

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, e.g. ETHSM_NO_COMMUNICATION or ETHSM_FULL_COMMUNICATION. ⌋()

7.6 Error classification

[ETHSM0028] ⌈
Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.⌋()

[ETHSM0029] ⌈
Development error values are of type uint8.⌋()

Type or error	***Relevance***	***Related error code***	***Value [hex]***
Invalid communication mode requested	Development	`ETHSM_E_INVALID_NETWORK_MODE`	`0x01`
EthSM module was not initialized	Development	`ETHSM_E_UNINIT`	`0x02`
Invalid pointer in parameter list	Development	`ETHSM_E_PARAM_POINTER`	`0x03`
Invalid parameter in parameter list	Development	`ETHSM_E_INVALID_NETWORK_HANDLE`	`0x04`

7.7 Error detection

[ETHSM0030] ⌈
The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch EthSmDevErrorDetect (see chapter 10) shall activate or deactivate the detection of all development errors.⌋(BSW00386)

[ETHSM0031] ⌈
If the EthSmDevErrorDetect switch is enabled API parameter checking is enabled.
The detailed description of the detected errors can be found in chapter 7.6 and chapter 8.⌋(BSW00386)

[ETHSM0032] ⌈
The detection of production code errors cannot be switched off.⌋()

7.8 Error notification

[ETHSM0033] ⌈
Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch EthSmDevErrorDetect is set (see chapter 10). ⌋(BSW00386)

[ETHSM0034] ⌈
Production errors shall be reported to Diagnostic Event Manager.⌋(BSW00323)

7.9 Debugging

[ETHSM0071] ⌈
Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.⌋()

[ETHSM0072] ⌈
All type definitions of variables which shall be debugged, shall be accessible by the header file EthSM.h.⌋()

[ETHSM0073] ⌈
The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-"sizeof".⌋()

[ETHSM0074] ⌈
Variables available for debugging shall be described in the respective Basic Software Module Description.⌋()

[ETHSM0076] ⌈
The state ETHSM_FULL_COMMUNICATION shall be available for debugging.⌋()

[ETHSM0077] ⌈
The state ETHSM_NO_COMMUNICATION shall be available for debugging.⌋()

[ETHSM0075] ⌈
The state ETHSM_UNINITED shall be available for debugging.⌋()

Additional recommendation:

For all defined states, it shall be possible to identify them during debugging. In this case, it should be recommended, that these states are available for debugging.

7.10 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 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 controller and Ethernet transceiver (Variant 2 in Figure 7-2).

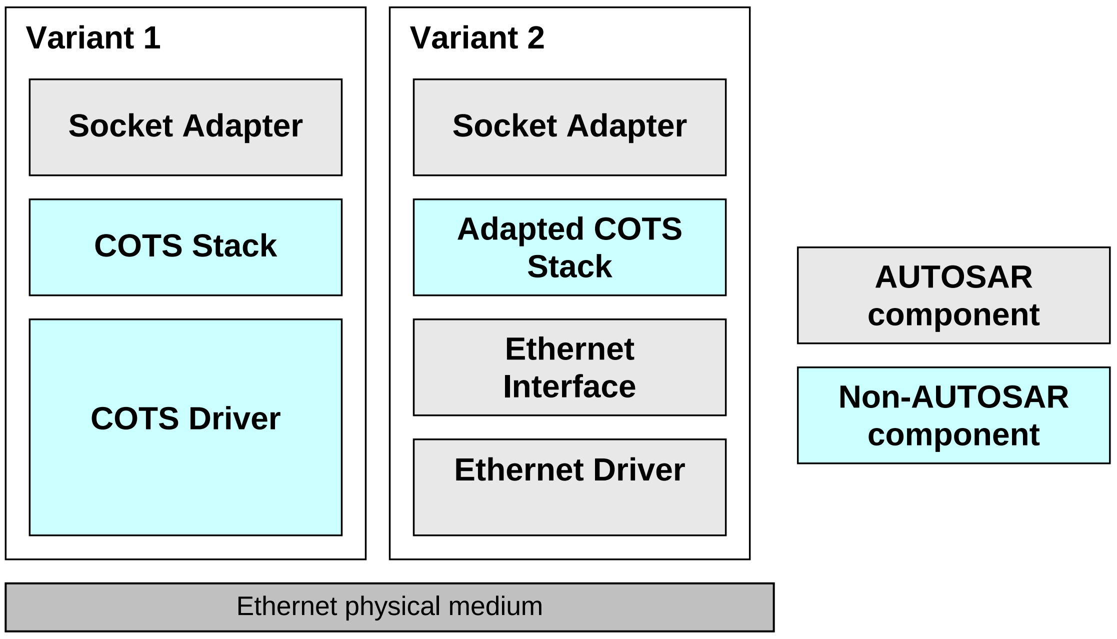


Figure 7-2: BSW stack architecture variants

[ETHSM0078] ⌈
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

Module	***Imported Type***
ComM	ComM_ModeType
ComStack_Types	NetworkHandleType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

8.2.1 EthSM_ConfigType

[ETHSM0039] ⌈

Name:	`EthSM_ConfigType`
Type:	`Structure`
Range:	Implementation specific.
Description:	This type defines a data structure for the post build parameters of the EthSM. At initialization the EthSM gets a pointer to a structure of this type to get access to its configuration data, which is necessary for initialization.

⌋(BSW0405, BSW00335, BSW00414)

8.2.2 EthSM_NetworkModeStateType

[ETHSM0041] ⌈

Name:	`EthSM_NetworkModeStateType`	
Type:	`Enumeration`	
Range:	`ETHSM_UNINITED`	EthSM is uninitialized in this state.
	`ETHSM_NO_COMMUNICATION`	ComM requests COMM_NO_COMMUNICATION in this state.
	`ETHSM_FULL_COMMUNICATION`	ComM requests COMM_FULL_COMMUNICATION in this state.
Description:	This type shall define the states of the network mode state machine.	

⌋()

8.3 Function definitions

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

8.3.1 EthSM_Init

[ETHSM0043] ⌈

Service name:	EthSM_Init	
Syntax:	`Std_ReturnType EthSM_Init( const EthSM_ConfigType* ConfigPtr )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	--
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	--
Description:	This function initialize the EthSM.	

⌋(BSW0405, BSW101, BSW00358, BSW00414)

[ETHSM0044] ⌈
Instead of the prototype specified in [ETHSM0043: the EthSM shall declare following prototype for the API `EthSM_Init` and use a void parameter instead of the `ConfigPtr`: `void EthSM_Init(void)` ⌋(BSW00414)

[ETHSM0045] ⌈
The function `EthSM_Init` shall report the development error `ETHSM_E_PARAM_POINTER` to the DET, if the user of this function hands over a NULL-pointer as `ConfigPtr`. ⌋()

8.3.2 EthSM_GetVersionInfo

[ETHSM0046] ⌈

Service name:	EthSM_GetVersionInfo	
Syntax:	`void EthSM_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0x00	
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.	

⌋(BSW00407, BSW003)

[ETHSM0047] ⌈

The function EthSM_GetVersionInfo shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).⌋()

[ETHSM0048] ⌈
The function EthSM_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: ETHSM_VERSION_INFO_API⌋(BSW00411)

[ETHSM0049] ⌈
If source code for caller and callee of EthSM_GetVersionInfo is available, the EthSM should realize EthSM_GetVersionInfo as a macro, defined in the module's header file.⌋()

8.3.3 EthSM_RequestComMode

[ETHSM0050] ⌈

Service name:	EthSM_RequestComMode	

Syntax:	`Std_ReturnType EthSM_RequestComMode(`
	`    NetworkHandleType NetworkHandle,`
	`    ComM_ModeType ComM_Mode`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
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.	

⌋()

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

[ETHSM0051] ⌈
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.⌋()

[ETHSM0052] ⌈
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.⌋()

[ETHSM0095] ⌈
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.⌋()

[ETHSM0053] ⌈
If the function `EthSM_RequestComMode` accepts the request, it shall store the requested communication mode for the network handle and shall execute the corresponding network mode state machine.⌋()

[ETHSM0054] ⌈
The function `EthSM_RequestComMode` shall report `ETHSM_E_UNINIT` to the DET, if the EthSM is not initialized yet.⌋(BSW00406)

[ETHSM0105] ⌈
The function `EthSM_RequestComMode` calls functions of the EthIf (see chapter 8.6.1 and 9). If these functions returns E_NOT_OK it is also necessary to set the return value of `EthSM_RequestComMode` to E_NOT_OK. In this case no state transitions will be realized.⌋()

8.3.4 EthSM_GetCurrentComMode

[ETHSM0055] ⌈

Service name:	EthSM_GetCurrentComMode	

Syntax:	`Std_ReturnType EthSM_GetCurrentComMode(`
	`    NetworkHandleType NetworkHandle,`
	`    ComM_ModeType* ComM_ModePtr`
	`)`	

Service ID[hex]:	0x00	
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.	

⌋()

[ETHSM0057] ⌈
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.⌋()

[ETHSM0058] ⌈
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.⌋()

[ETHSM0059] ⌈
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.

[ETHSM0060] ⌈
The function `EthSM_GetCurrentComMode` shall report `ETHSM_E_UNINIT` to the DET, if the EthSM is not initialized yet.⌋(BSW00406, BSW00374, BSW00379, BSW003, BSW00318)

[ETHSM0084] ⌈
The function `EthSM_GetCurrentComMode` shall report `ETHSM_E_PARAM_POINTER` to the DET, if the pointer of the parameter list is invalid.⌋()

8.4 Call-back notifications

This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the file <Module Prefix>_Cbk.h

Actual no callback functions are provided.

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

[ETHSM0035] ⌈

Service name:	EthSM_MainFunction

Syntax:	`void EthSM_MainFunction(`
	`    void`
	`)`

Service ID[hex]:	0x00
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.

⌋()

[ETHSM0093] ⌈
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. ⌋()

Terms and definitions:
Fixed cyclic: Fixed cyclic means that one cycle time is defined at configuration and shall not be changed because functionality is requiring that fixed timing (e.g. filters).
Variable cyclic: Variable cyclic means that the cycle times are defined at configuration, but might be mode dependent and therefore vary during runtime.
On pre condition: On pre condition means that no cycle time can be defined. The function will be called when conditions are fulfilled. Alternatively, the function may be called cyclically however the cycle time will be assigned dynamically during runtime by other modules.

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.

API function	***Module***	***Description***
EthIf_SetControllerMode	EthIf	To control operating states of Ethernet controllers
EthIf_SetTransceiverMode	EthIf	To control operating states of Ethernet Transceivers

EthIf_ControllerInit	EthIf	To initialize the Ethernet controllers
EthIf_TransceiverInit	EthIf	To initialize the Ethernet transceivers
EthIf_GetControllerMode	EthIf	To gets the actual operating states of the Ethernet Controllers
EthIf_GetTransceiverMode	EthIf	To gets the actual operating states of the Ethernet Transceivers
Dem_ReportErrorStatus	DEM	To report production errors to DEM
ComM_BusSM_ModeIndication	ComM	The ComM provides this function to be notified about communication mode changes of the Ethernet networks.
BswM_EthSM_CurrentState	BswM	The BswM provides this function to be notified about communication mode changes of the Ethernet networks. The EthSM reports through this interface the internal states (see Figure 7-1)
SoAd_BusSM_ModeIndication	SoAd	The SoAd provides this function to be notified about communication mode changes of the Ethernet networks.

8.6.2 Optional Interfaces

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

API function	***Module***	***Description***	***Configuration parameter (description see chapter 10)***
Det_ReportError	Det	Development error notification	`ETHSM_DEV_ERROR_DETECT`

9 Sequence diagrams

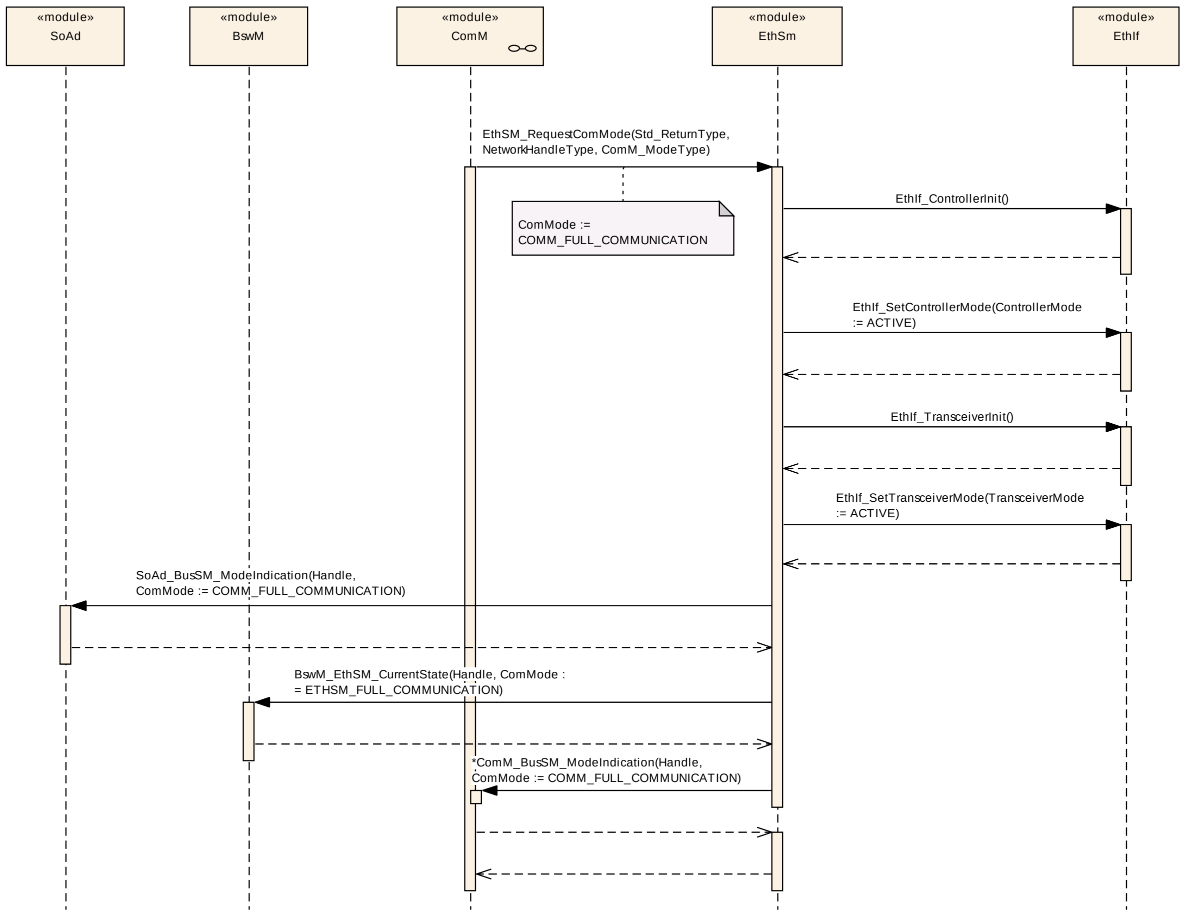

«module» SoAd
«module» BswM
«module» ComM
«module» EthSm
«module» EthIf
EthSM_RequestComMode(Std_ReturnType, NetworkHandleType, ComM_ModeType)
ComMode := COMM_FULL_COMMUNICATION
EthIf_ControllerInit()
EthIf_SetControllerMode(ControllerMode := ACTIVE)
EthIf_TransceiverInit()
EthIf_SetTransceiverMode(TransceiverMode := ACTIVE)
SoAd_BusSM_ModeIndication(Handle, ComMode := COMM_FULL_COMMUNICATION)
BswM_EthSM_CurrentState(Handle, ComMode := ETHSM_FULL_COMMUNICATION)
*ComM_BusSM_ModeIndication(Handle, ComMode := COMM_FULL_COMMUNICATION)

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

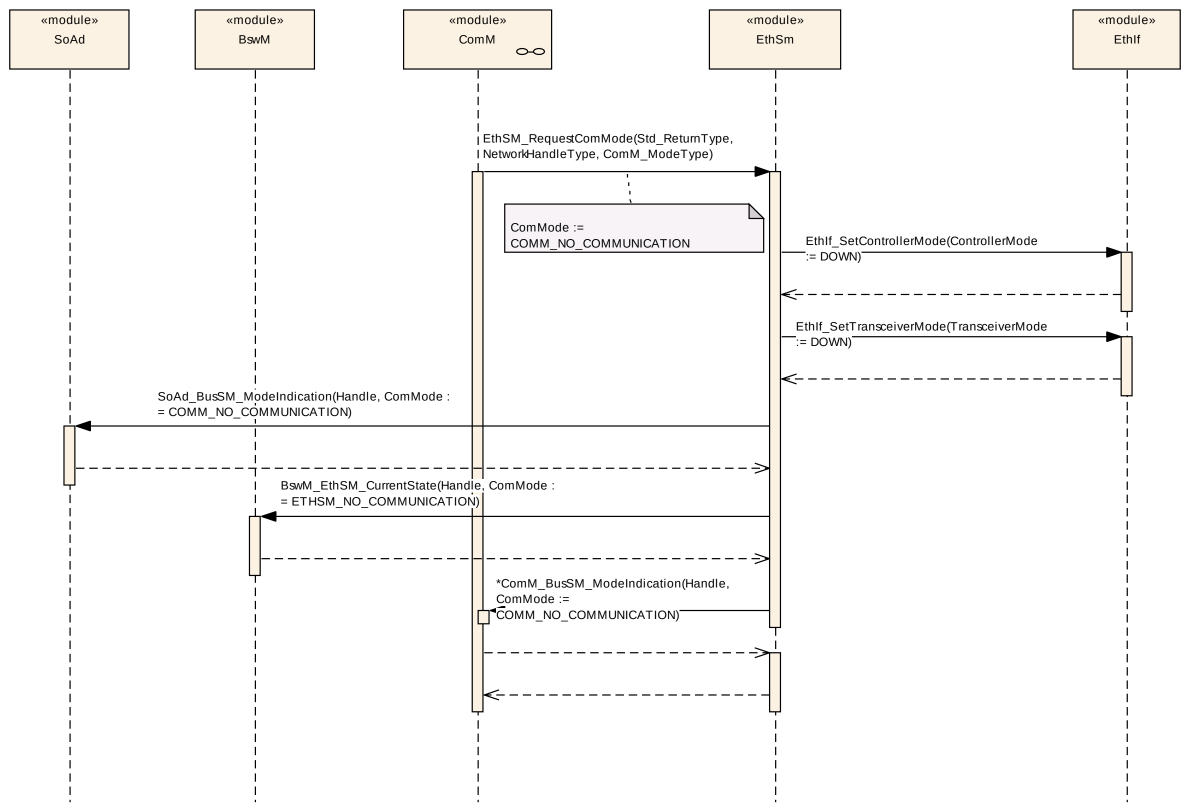

«module» SoAd
«module» BswM
«module» ComM
«module» EthSm
«module» EthIf
EthSM_RequestComMode(Std_ReturnType, NetworkHandleType, ComM_ModeType)
ComMode := COMM_NO_COMMUNICATION
EthIf_SetControllerMode(ControllerMode := DOWN)
EthIf_SetTransceiverMode(TransceiverMode := DOWN)
SoAd_BusSM_ModeIndication(Handle, ComMode : = COMM_NO_COMMUNICATION)
BswM_EthSM_CurrentState(Handle, ComMode : = ETHSM_NO_COMMUNICATION)
*ComM_BusSM_ModeIndication(Handle, ComMode := COMM_NO_COMMUNICATION)

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

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

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [5]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

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

[ETHSM0081]⌈
A configuration tool will create a configuration structure that is understood by the EthSM.⌋(BSW159, BSW00424, BSW00425)

10.2.2 Variants

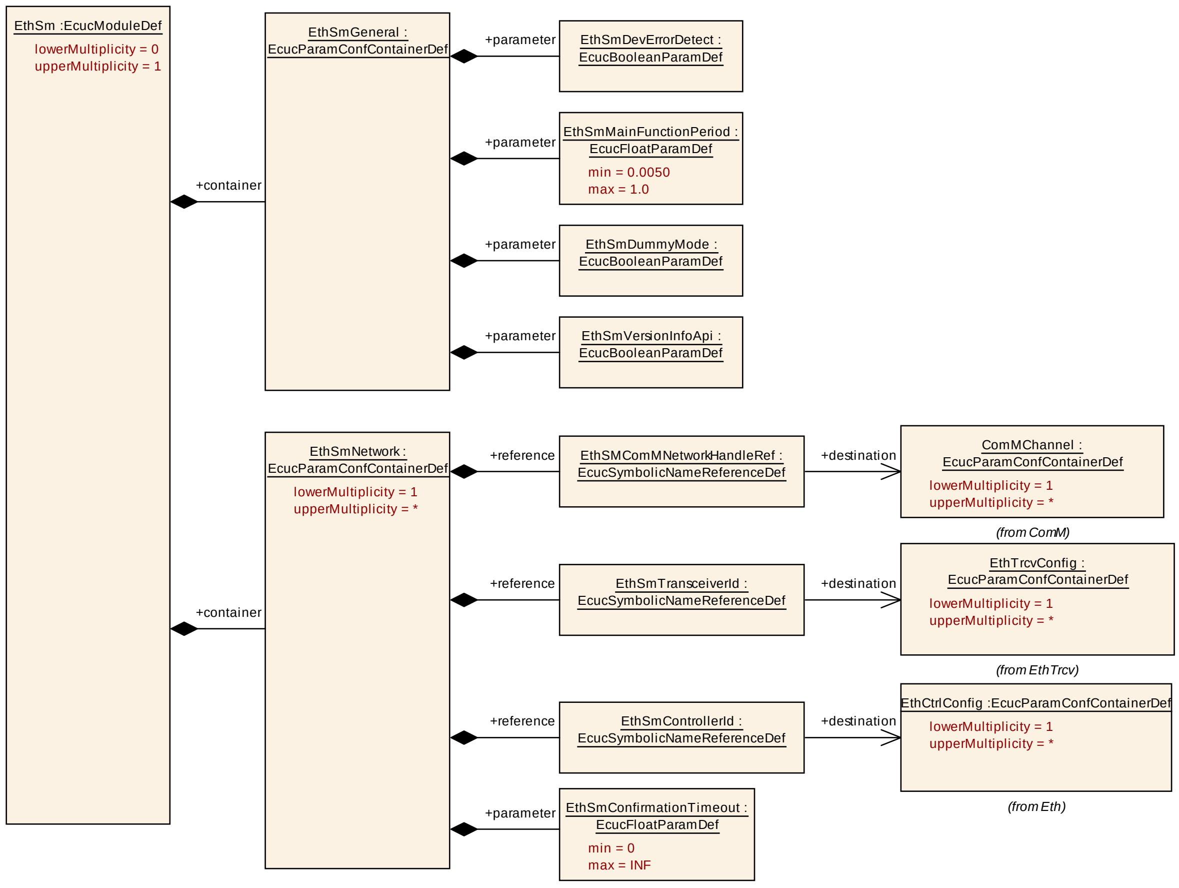
[ETHSM0061] ⌈
Actual the only provided configuration variant is the use of pre-compile parameters. Not provided are link time parameters, post build time parameters or mixes of them.

Variant 1: Only pre-compile parameters⌋(BSW00345)

10.2.3 EthSm

Module Name	*EthSm*
Module Description	Configuration of the Ethernet State Manager

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 controller and transceiver IDs assigned to a Ethernet network.


EthSm :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
EthSmGeneral : EcucParamConfContainerDef
+parameter
EthSmDevErrorDetect : EcucBooleanParamDef
+parameter
EthSmMainFunctionPeriod : EcucFloatParamDef
min = 0.0050
max = 1.0
+parameter
EthSmDummyMode : EcucBooleanParamDef
+parameter
EthSmVersionInfoApi : EcucBooleanParamDef
+container
EthSmNetwork : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+reference
EthSMComMNetworkHandleRef : EcucSymbolicNameReferenceDef
+destination
ComMChannel : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from ComM)
+reference
EthSmTransceiverId : EcucSymbolicNameReferenceDef
+destination
EthTrcvConfig : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from EthTrcv)
+reference
EthSmControllerId : EcucSymbolicNameReferenceDef
+destination
EthCtrlConfig :EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from Eth)
+parameter
EthSmConfirmationTimeout : EcucFloatParamDef
min = 0
max = INF

10.2.4 EthSmGeneral

SWS Item	**ETHSM0063_Conf :**
Container Name	EthSmGeneral
Description	This container contains the global parameter of the Ethernet State Manager.
Configuration Parameters	

SWS Item	**ETHSM0065_Conf :**		
Name	EthSmDevErrorDetect {ETHSM_DEV_ERROR_DETECT}		
Description	Enables and disables the development errer detection and notification mechanism.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ETHSM0079_Conf :**
Name	EthSmDummyMode {ETHSM_DUMMY_MODE}
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ETHSM0066_Conf :**		
Name	EthSmMainFunctionPeriod {ETHSM_MAIN_FUNCTION_PERIOD}		
Description	Specifies the period in seconds that the MainFunction has to be triggered with.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0.005 .. 1`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ETHSM0092_Conf :**		
Name	EthSmVersionInfoApi {ETHSM_VERSION_INFO_API}		
Description	Enables and disables the version info API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.5 EthSmNetwork

SWS Item	**ETHSM0067_Conf :**
Container Name	EthSmNetwork
Description	This container contains the Ethernet network-specific parameters of each Ethernet network. It also contains the controller and transceiver IDs assigned to a Ethernet network.
Configuration Parameters	

SWS Item	**ETHSM0104_Conf :**	
Name	EthSmConfirmationTimeout {ETHSM_CONFIRMATION_TIMEOUT}	
Description	Timeout in seconds for the calls to EthIf: EthIf_ControllerInit EthIf_TransceiverInit EthIf_SetControllerMode EthIf_SetTransceiverMode	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	`0 .. INF`	

Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ETHSM0068_Conf :**		
Name	EthSMComMNetworkHandleRef {ETHSM_NETWORK_HANDLE}		
Description	Unique handle to identify one certain Ethernet network. Reference to one of the network handles configured for the ComM.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ETHSM0070_Conf :**		
Name	EthSmControllerId {ETHSM_CONTROLLER_ID}		
Description	ID of the Ethernet controller assigned to the configured network handle. Reference to one of the controllers of Eth configuration.		
Multiplicity	1		
Type	Reference to [EthCtrlConfig]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ETHSM0069_Conf :**		
Name	EthSmTransceiverId {ETHSM_TRANSCEIVER_ID}		
Description	ID of the ethernet transceiver assigned to the configured network handle. Reference to one of the transceivers of EthTrcv configuration.		
Multiplicity	1		
Type	Reference to [EthTrcvConfig]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

Note:
The Ethernet State Manager could handle several Ethernet networks. But actual there are limitations concern the assignment of transceiver and controller to each network (see chapter 4.1).

10.3 Published Information

[ETHSM0064] ⌈
The standardized common published parameters as required by BSW00402 in the SRS General on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].⌋()

Additional module-specific published parameters are listed below if applicable.

11 Not applicable requirements

[ETHSM999] 「 These requirements are not applicable to this specification. 」(BSW00344, BSW0404, BSW170, BSW00387, BSW00395, BSW00398, BSW00399, BSW00400, BSW00438, BSW00375, BSW00416, BSW00437, BSW168, BSW00423, BSW00426, BSW00427, BSW00428, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW00336, BSW00369, BSW00417, BSW161, BSW162, BSW005, BSW164, BSW00325, BSW00326, BSW00343, BSW160, BSW00413, BSW00347, BSW00373, BSW00314, BSW00353, BSW00361, BSW00328, BSW00377, BSW00355, BSW00306, BSW00308, BSW00309, BSW00371, BSW00359, BSW00360, BSW00331, BSW010, BSW00333, BSW00321, BSW00341, BSW00334)