Elektrobit

EB tresos® AutoCore Generic 8 Mode Management documentation

product release 8.8.0

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview of EB tresos AutoCore Generic 8 Mode Management documentation

Welcome to the EB tresos AutoCore Generic 8 Mode Management (ACG8 Mode Management) product documentation.

This document provides:

- Chapter 3, “ACG8 Mode Management release notes“: release notes for the ACG8 Mode Management modules
- Chapter 4, “ACG8 Mode Management user guide“: containing background information and instructions
- Chapter 5, “ACG8 Mode Management module references“: information about configuration parameters and the application programming interface

2. Supported features

2.1. Overview

This chapter provides an overview of the ACG8 Mode Management and the features that are currently supported.

Section 2.2, “Supported BswM features” contains an overview of `BswM` features.

Section 2.3, “Supported ComM features” contains an overview of `ComM` features.

Section 2.4, “Supported EcuM features” contains an overview of `EcuM` features.

Section 2.5, “Supported Nm features” contains an overview of `Nm` features.

2.2. Supported BswM features

- **Support for BSW distribution:** Support for distribution of multiple BswM instances over multiple partitions.
- **Support for mode arbitration:** Support for mode arbitration based on configurable rules.
- **Support for immediate and deferred operation:** Support for processing operations immediately, once they are requested or in the main function cycle.
- **Support for mode control:** Support for processing configured action lists based on the mode arbitration results.
- **Support for triggered and conditional action lists:** Support for executing actions every time a rule is evaluated or only when the evaluation result changes.
- **Support for mode request ports:** Support for ports to be used by SWCs in order to request modes.
- **Support for mode switch ports:** Support for ports to be used by SWCs in order to be notified about mode switches.
- **Support for notifications of mode switches:** Support for ports to be used by BswM in order to be notified about mode switches (BSW or SWC).
- **Support for post-build:** Support for handling post-build loadable and selectable configuration.
- **Support for link time:** Support for handling configuration at link time.
- **Support for BswMTimer:** Support for timer handling to delay actions or wait for further mode controls according to AUTOSAR 4.3.0.

2.3. Supported ComM features

- **Support for three communication modes**: Support for three communication modes according to AUTOSAR specifications (COMM_FULL_COMMUNICATION, COMM_SILENT_COMMUNICATION, COMM_NO_COMMUNICATION).
- **Support for three communication states**: Support for three main communication states with communication capabilities according to AUTOSAR specifications (COMM_FULL_COMMUNICATION, COMM_SILENT_COMMUNICATION, COMM_NO_COMMUNICATION).
- **Support for communication sub-states**: Support for communication sub-states COMM_FULL_COM_NETWORK_REQUESTED, COMM_FULL_COM_READY_SLEEP and COMM_NO_COM_NO_PENDING_REQUEST, COMM_NO_COM_REQUEST_PENDING with communication capabilities according to AUTOSAR specifications.
- **Support for communication start-up**: Support for starting the communication due to user request of COMM_FULL_COMMUNICATION mode.
- **Support for communication shutdown**: Support for stopping the communication due to user request of COMM_NO_COMMUNICATION mode.
- **Support for communication passive wakeup**: Support for starting the communication due to wakeup event, network start or network restart indication.
- **Support for shutdown synchronization variants**: Support for shutdown synchronization variants (NONE, LIGHT, PASSIVE, FULL) according to AUTOSAR specifications.
- **Support for bus wakeup inhibition**: Support for inhibition of communication channels based on user requests.
- **Support for reading inhibition status**: Support for an interface to read the inhibition status of a channel.
- **Support for inhibition counter**: Support for a counter to store the number of rejected COMM_FULL_COMMUNICATION user requests, including interfaces to read and reset the value of the counter.
- **Support for limitation to NoCommunication**: Support for forcing the communication mode to NoCommunication even if users still request FullCommunication.
- **Support for providing the communication mode**: Support for a user interface to read the requested communication mode and current communication mode for a user.
- **Support for diagnostic communication request**: Support for processing diagnostic communication request and switching communication mode based on the diagnostic request.
- **Support for state change notification**: Support for notifications to ComM users and mode manager on mode changes.
- **Support for partial network cluster (PNC)**: Support for PNC functionality and state machine according to AUTOSAR specifications.
- **Support for the Ethernet switch**: Support communication control by an active or passive start-up (PNC wakeup) when Ethernet switch is used.

- **Support for post-build:** Support for handling post-build loadable and selectable configuration.
- **Support for BSW distribution**: Support for inter-core communication when different bus types are processed on different cores.

2.4. Supported EcuM features

- **Support for BSW distribution:** Support for distributing EcuM over multiple cores (master-satellite approach with inter-core communication).
- **Support configurable master core:** Support for configurable core designation of the master EcuM instance.
- **Support for startup phase:** Support for handling activities before EcuM_Init, activities in StartPreOs and StartPostOs sequences and support for driver initialization.
- **Support for shutdown phase:** Support for handling activities in the OffPreOS and OffPostOS sequences.
- **Support for sleep phase:** Support for handling activities in GoSleep, Halt, Poll and WakeupRestart sequences.
- **Support for UP phase:** Support for handling and validating wakeup sources and reset reasons.
- **Support for shutdown targets:** Support for sleep and reset states as shutdown targets.
- **Support for service interface according to AUTOSAR 3.2 and AUTOSAR 4.0.3:** Support interfaces for requesting and releasing of ECU states, notifying current mode, selecting and accessing the shutdown target and selecting the boot target.
- **Support for run request protocol:** Support for coordinating if the ECU must be kept alive or is ready to shut down.
- **Support for post-build:** Support for handling post-build selectable configuration.

2.5. Supported Nm features

- **Support for abstraction from bus-specific NM(s)**: Support for interfaces that do not require information about the communication buses used.
- **Support for NM coordinator**: Support for NM coordinator algorithm to keep coordinated buses alive and shut down coordinated buses.
- **Support for state change notification**: Support for updating a Com signal with information regarding changes of the states in bus-specific network management module.
- **Support for car wakeup**: Support for calling a user-defined callback function or giving an indication to BswM when a lower bus-specific NM module indicates the reception of a network management PDU with the car wakeup bit set.

- **Support for channel-based configurable RepeateMsgInd|NodeDetection|NodeIdEnabled**: Support for a per-channel configuration of parameters NmRepeatMsgIndEnabled, NmNodeDetectionEnabled, NmNodeIdEnabled.
- **Support for BSW distribution**: Support for inter-core communication when different bus types are processed on different cores.

3. ACG8 Mode Management release notes

3.1. Overview

This chapter provides the ACG8 Mode Management product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

3.2. Scope of the release

3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 27.1.0 b200625-0900

3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 Mode Management release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
BswM	4.0.3 []	1.2.0 [0000]	1.15.3	Elektrobit Automotive GmbH
ComM	4.1.3 []	4.1.1 [0]	5.19.4	Elektrobit Automotive GmbH
EcuM	4.0.3 []	3.0.0 [0000]	5.15.4	Elektrobit Automotive GmbH
Nm	4.0.3 []	3.0.0 [0000]	5.12.5	Elektrobit Automotive GmbH

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
BswMAs	1.21.1	Elektrobit Automotive GmbH

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

3.3. Module release notes

3.3.1. BswM module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 1.2.0
- Module version: 1.15.3.B337087
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 1.15.3
2020-06-19

[1] `$TRESOS_BASE` is the location at which you installed EB tresos Studio.

- ASCBSWM-1201 Fixed known issue: Generation fails if a BswMSwcModeRequest port references a VariableDataPrototype that uses an ApplicationPrimitiveDataType
- Added support for J1939 mode request ports and actions

Module version 1.15.2
2020-02-21

- Internal module improvement. This module version update does not affect module functionality

Module version 1.15.1
2019-10-11

- ASCBSWM-1160 Fixed known issue: Unused BswMSchMSwitch actions cause an invalid configuration if IMPLEMENTATION_CONFIG_VARIANT is VariantPreCompile

Module version 1.15.0
2019-06-14

- ASCBSWM-1142 Fixed known issue: Deadline monitoring cannot be disabled when BswMDeadlineMonitoringControl actions are executed before BswMPduGroupSwitch actions
- Added postbuild selectable support
- ASCBSWM-1133 Fixed known issue: Changing rule expressions at post-build time might lead to unexpected behavior
- ASCBSWM-1134 Fixed known issue: Referencing previously unused actions at post-build time leads to unexpected behavior
- Added support for queued mode arbitration
- BswM_Dcm_RequestCommunicationMode now uses service identifier 0xFE instead of 0x06

Module version 1.14.5
2019-02-15

- ASCBSWM-1123 Fixed known issue: Initialization of multiple cores can lead to data corruption and unexpected behaviour

Module version 1.14.4
2019-01-31

- ASCBSWM-1113 Fixed known issue: Incorrect allocation of the internal IpduGroupVector leads to an out-of-bounds access in Com
- ASCBSWM-1110 Fixed known issue: Invalid basic software module description if an immediate BswM-Timer port triggers the execution of a SchMSwitch action

Module version 1.14.3
2018-10-26

- Internal module improvement. This module version update does not affect module functionality
- ASCBSWM-1107 Fixed known issue: Infinite loop if more than 2040 I-PDU groups are configured

Module version 1.14.2
2018-06-22

- Added support for Rte Switch Acknowledgements
- Added memory sections for the BswM software component description
- Added constraint that BswMPduGroupSwitch and BswMDeadlineMonitoringControl actions can only be configured in one BswM instance
- Added support for BswM timers
- ASCBSWM-1074 Fixed known issue: BswM_PostBuildConfig uses different memory sections for declaration and definition

Module version 1.14.1
2018-02-16

- ASCBSWM-967 Fixed known issue: BswM generates invalid code if a mode condition referencing a BswM-NmIfCarWakeUpIndication port doesn't have 0 as a condition value
- ASCBSWM-981 Fixed known issue: Wrong memory section used for BswM_PtrModeRequestPortsTable
- ASCBSWM-995 Fixed known issue: BswM triggers the wrong PDUs if multiple BswMTriggerIPduSend actions exist
- ASCBSWM-1004 Fixed known issue: Generation fails if BswMGenericRequest ports are used only in some BswM configurations
- ASCBSWM-997 Fixed known issue: BswM triggers the wrong PDUs if ComIPduHandleId and PduRSourcePduHandleId have different values
- Enhanced the BSWM_E_WRONG_CONTEXT Det error when BswM API's are called from an invalid partition

Module version 1.14.0
2017-09-21

- Major code refactoring for improving RAM and ROM usage
- Added support for Ethernet Switch Port Groups
- ASCBSWM-926 Fixed known issue: BswM generator produces invalid code or crashes if cross-partition references are used
- ASCBSWM-931 Fixed known issue: Inconsistency between generated code and BSWMD when unused BswMSchMSwitch actions exist
- Added support for Car Wakeup

Module version 1.13.0
2017-03-31

- ASCBSWM-880 Fixed known issue: Single instance BswM fails to compile in a multicore ECU

Module version 1.12.2
2017-03-13

- Symbolic values from user defined header files can be used as initial values for mode request ports or as condition values for mode conditions
- Mode request port tables generation order is now preserved
- ASCBSWM-853 Fixed known issue: Possible data corruption if processing nested rules with triggered action list execution is interrupted
- ASCBSWM-838 Fixed known issue: Premature handling of I-PDU groups when mode control is interrupted
- ASCBSWM-859 Fixed known issue: Uncompilable code when a mode request port is initialized with a symbolic name value
- ASCBSWM-870 Fixed known issue: Disabling deadline monitoring control might be ineffective
- ASCBSWM-346 Fixed known issue: The BswM may reference a data type that is not specified in ASR-4.-0.3 Dcm SWS

Module version 1.12.1
2016-12-15

- ASCBSWM-813 Fixed known issue: Uncompilable code when variant PostBuild and multiple instances are configured

- Headers for other BSW modules are generated only when the corresponding parameter is enabled
- Added support for DEM error reporting
- Simplified the configuration of generic request ports

Module version 1.12.0
2016-11-04

- Added a warning message in case a BswMSwitchPort is not used by any SchMSwitch or RteSwitch action
- Added validation for the TARGET parameter of a BswMRteModeRequestPortInterfaceRef
- Added a warning message if a BswMNestedExecutionOnly rule is not referenced by any action list.
- Improve validation of the BswMRteUsage parameter
- Enhanced the basic software module description
- ASCBSWM-795 Fixed known issue: Incorrect behaviour when multiple BswM partitions are on the same core
- ASCBSWM-800 Fixed known issue: Incorrect mode arbitration if more than 2 BswM configurations exist
- Fixed issue with exporting unused configuration parameters

Module version 1.11.0
2016-05-25

- ASCBSWM-705 Fixed known issue: DET errors are always reported from the first partition
- ASCBSWM-709 Fixed known issue: Missing semicolon in generated SchMSwitch action functions
- R-PORT Prototype definition corresponding to a SwcModeRequest port uses its configured initial value
- Improved configuration validations for logical expressions and rules
- ASCBSWM-715 Fixed known issue: Uncompilable code is produced when BswMBswRequestedMode is empty
- ASCBSWM-716 Fixed known issue: Invalid memory access when accessing initial values
- Optimization when no BswMPduGroupSwitch actions with BswMPduGroupSwitchReinit - TRUE exist
- ASCBSWM-736 Fixed known issue: Memory access violation if multiple BswM instances are used
- ASCBSWM-775 Fixed known issue: Invalid BSWMD and SWCD if multiple BswM configurations exist
- Optimizations for the BswM_MainFunction execution time
- DET error logged when BswM API functions are called from an invalid context
- Improved validation of BswMUserCalloutFunction

Module version 1.10.0

2016-02-10

- ASCBSWM-698 Fixed known issue: Invalid handling of duplicate I-PDU groups in case of PduGroupSwitch or DeadlineMonitoringControl actions

Module version 1.9.1

2016-01-15

- ASCBSWM-654 Fixed known issue: Wrong definition of BswMPduGroupSwitchActionType and BswMDeadLineMonitoringControlActionType produces invalid code
- Renamed short names of SERVICE-SW-COMPONENT-TYPE instances
- ASCBSWM-656 Fixed known issue: BswM generates uncompilable code when using Rte mode request ports
- ASCBSWM-659 Fixed known issue: Incorrect SchM and BswM headers included in a multi-core setup
- The symbols of runnable entities on different partitions need to be unique
- Configuration errors added in case actions using disabled external interfaces are configured
- Performance optimization when performing mode control
- Added support for adding a new empty BswM configuration container
- Removed unused header files BswM_RTE.h, BswM_COM.h and User_Callout.h
- Added configuration check that all referenced mode declaration groups have corresponding data type mapping sets
- ASCBSWM-667 Fixed known issue: BswM disables deadline monitoring for I-PDU groups
- Fixed compiler abstraction mismatch between Rte declarations and BswM definitions
- ASCBSWM-669 Fixed known issue: Generation error when using Com, PduRControl or LinScheduleSwitch actions in a multicore configuration
- Added better configuration checks for the parameters inside the EcuMSelectShutdownTarget action
- ASCBSWM-682 Fixed known issue: No DET error check for request id in case 255 GenericRequest ports are configured
- Removed restriction that the BswMModeRequesterIds need to be 0 based consecutive

Module version 1.9.0

2015-11-10

- Improved error handling for mode request ports during generation
- Added support for multiple instantiation

- Renamed BswM_LinIf.h to BswM_LinTp.h

Module version 1.8.0

2015-06-24

- Limited the number of artifacts displayed in the Rule-Port Dependencies window
- ASCBSWM-581 Fixed known issue: Ports of type BswMDcmApplicationUpdated are reported to be unused
- Limited the options for configuring the BswM_Dcm_ApplicationUpdated()
- Improved generated code of logical expression states
- Implemented RfC #54616: BswM_WdgM_RequestPartitionReset cannot be treated like the rest of indication services
- Improved error messages related to mode request ports in the BswM Editor
- Improved error handling inside the BswM Editor
- Removed BswM.h inclusion from BswM_ComM.h
- Moved the BswM Editor plugin to a separate repository
- ASCBSWM-617 Fixed known issue: Configuring the same BswMBswModeInitValueMode value in different ports lead to errors during generation
- Added support for using a variable data prototype reference in a BswMSwcModeRequest port

Module version 1.7.0

2015-02-20

- Removed SwcBswMapping constraint related to the existence of a SwitchPort
- Removed generated mode conditions macros
- Improved the names of the containers generated by the BswM Editor

Module version 1.6.0

2015-01-07

- Added support for CanSM enhanced bus off reporting
- Changed the default value of BswMRelocatablePbcfgEnable to true
- Moved generated header type definitions into BswM_Generated_Types.h
- Removed redundant code related to Pb relocatable in BswM.h

- ASCBSWM-513 Fixed known issue: The rule parameter BswMNestedExecutionOnly is ignored for mode request ports with immediate evaluation
- Introduced the BswM Editor
- Modified parameter of BswM_ExecuteModeControl function to const
- Added BswM Editor user's guide
- Added support for BswM_Dcm_ApplicationUpdated() API
- Updated the module version strings within the BswM Editor source code

Module version 1.5.0
2014-10-02

- ASCBSWM-453 Fixed known issue: A mode declaration cannot be used to define the initial value of a generic mode request port
- The EcuM can now initialize the BswM with a NULL_PTR
- ASCBSWM-449 Fixed known issue: The inclusion of the BswM post-build header file may lead to a compiler error
- ASCBSWM-478 Fixed known issue: The configuration parameter BswMSchMEnabled contains invalid XPath expressions
- ASCBSWM-481 Fixed known issue: Invalid warning caused by the BswM configuration parameter BswMEcuMShutdownTargetRef
- Reduced the number of statements within the function BswM_LT_Init
- Added INVALID checks for all REF parameters
- Adapted the use of EcuM wakeup source symbolic name constants
- Added support for the Ecu configuration container BswMCompuScaleModeValue
- The BswM action BswMRteModeRequest is now supported
- Added support for symbolic constants

Module version 1.4.0
2014-06-27

- ASCBSWM-458 Fixed known issue: The BswM module is uninitialized in case of a software reset

Module version 1.3.15
2014-06-13

- ASCBSWM-426 Fixed known issue: Compiler error due to the multiple inclusion of application header files
- The macro BSWM_PB_CONFIG_NAME is now mapped to the macro BSWM_CONFIG_NAME
- ASCBSWM-444 Fixed known issue: The Det check for requesting_user in BswM_RequestMode() is wrong
- Use of the macro BSWM_COM_ACTION_USED has been replaced where the macros BSWM_DEADLINE_MONITORING_CONTROL_ACTION_USED, BSWM_PDU_GROUP_SWITCH_ACTION_USED, BSWM_SWITCH_IPDU_MODE_ACTION_USED or BSWM_TRIGGER_IPDU_SEND_ACTION_USED can be used
- The macro BSWM_PDUR_ACTION_USED has been replaced with the macro BSWM_PDU_ROUTER_-CONTROL_ACTION_USED
- The function BswM_GetVersionInfo now ignores a request when a NULL_PTR is passed to the function
- The function BswM_Dcm_CommunicationMode_CurrentState now uses the new data type BswMDcmComModeCurrentStatePortType

Module version 1.3.14
2014-03-21

- Changed passing of corresponding numerical values to enumeration constants from the BswM to the Service Discovery module
- ASCBSWM-397 Fixed known issue: The generation of calls to Rte switch functions is not conform to AUTOSAR
- Implemented post-build and link-time support
- Delayed execution of mode arbitration requests is now supported
- Delayed execution of the Com_IpduGroupControl command is now supported
- ASCBSWM-421 Fixed known issue: The BswM generator may produce an invalid warning regarding unused action lists
- ASCBSWM-422 Fixed known issue: A triggered action list may be executed when the rule's result has not changed
- ASCBSWM-423 Fixed known issue: BswMLinTpModeRequest mode request ports may be ignored or incorrectly updated

Module version 1.3.13
2013-09-17

- Removed Det error checks for BSWM_E_REQ_USER_OUT_OF_RANGE for all BswM mode request sources except BswM_RequestMode

Module version 1.3.12

2013-08-28

- ASCBSWM-373 Fixed known issue: A compiler error occurs because of missing MemMap macro

Module version 1.3.11

2013-08-09

- Added a unique short name for required mode declaration group prototypes

Module version 1.3.10

2013-07-30

- Changed to correct the implementation configuration class of several BswM Ecu configuration parameters

Module version 1.3.9

2013-07-23

- Added an initial value definition of SwcModeRequest ports for the BswM service component description
- ASCBSWM-341 Fixed known issue: The basic software module description contains an invalid reference to its SwcBswMapping
- Added Intra Module Checks to avoid integration of incompatible files
- ASCBSWM-345 Fixed known issue: If only one mode request port is defined for a mode request source that accepts multiple channels, the BswM may assign the status of different channels to the mode request port
- ASCBSWM-348 Fixed known issue: A Det parameter check within the function BswM_EcuM_CurrentWakeup does not recognize an out of range parameter
- Added configuration parameter for queue length of mode switch ports

Module version 1.3.8

2013-06-18

- ASCBSWM-283 Fixed known issue: Only the first IPdu will be sent for a BswMTriggerIPduSend action
- Added Debug and Trace support for the BswM

Module version 1.3.7

2013-05-14

- ASCBSWM-224 Fixed known issue: The file BswM_Cfg.h may not conform to the ISO C90 standard

- Removed the validation of constants passed to the BswM in separate functions
- Added support for the AUTOSAR Service Discovery module

Module version 1.3.6
2013-04-15

- ASCBSWM-209 Fixed known issue: The BswM generator may produce duplicate mode switch point declarations
- ASCBSWM-212 Fixed known issue: Multiple wakeup sources passed to the BswM API function BswM_EcuM_CurrentWakeup will not be evaluated
- ASCBSWM-214 Fixed known issue: The BswM does not generate mode declaration initial values correctly
- ASCBSWM-198 Fixed known issue: The BswM may reference undefined mode declaration identifiers

Module version 1.3.5
2013-03-15

- ASCBSWM-96 Fixed known issue: The parameter CurrentState of the function BswM_FrSM_CurrentState may be validated incorrectly when compiler options are enabled
- ASCBSWM-202 Fixed known issue: The generation order of certain BswM code fragments is not constant
- Implemented BswMActions as functions
- Removed the definition of a default value if no initial value has been defined for a mode request port
- Changed the validation of constants passed to the BswM into separate functions
- Changed the data types of BswM variables which store boolean values to be explicitly defined as the standard type "boolean"
- Removed unnecessary calls to the macro TS_PARAM_UNUSED
- Changed the BswM service component description to define a queue length for each BswMSwitchPort

Module version 1.3.4
2013-02-11

- The BswM actions BswMEcuMSelectShutdownTarget, BswMEcuMGoDown and BswMPduRouterControl are now supported
- The BswM now supports BswMSchMSwitch actions
- The BswM now supports BswMBswModeNotification mode request ports
- Added XPath validation checks for the Ecu parameters BswMArgumentRef and BswMRuleExpressionRef

- ASCBSWM-171 Fixed known issue: The generated BswM will not compile if a triggered action list is executed from a conditional action list
- Updated config structure (reference paths) of BswM-ComM channel reference for the introduction of ComMConfigSet container in ComM
- Removed SvcAs request for SchM exclusive areas since they are no longer used

Module version 1.3.3
2013-01-11

- Mode arbitration is now disabled after BswM_Deinit has been executed

Module version 1.3.2
2012-12-14

- ASCBSWM-86 Fixed known issue: If one or more unused RteSwitchActions exist a null pointer exception may be thrown during the generation of the BswM

Module version 1.3.1
2012-11-16

- ASCBSWM-79 Fixed known issue: The function BswM_WdgM_RequestPartitionReset incorrectly validates the range of the parameter "Application"
- ASCBSWM-107 Fixed known issue: If multiple BswMComMIndication mode request ports reference the same ComMChannel only one mode request port will be evaluated
- ASCBSWM-88 Fixed known issue: The function BswM_LinSM_CurrentSchedule references the undefined macro BSWM_LINIF_SCHEDULE_TABLE_MAX
- ASCBSWM-83 Fixed known issue: If a BswMSwcModeNotification is incorrectly configured the BswM generation will terminate with a null pointer exception
- ASCBSWM-80 Fixed known issue: The BswM basic software module description contains an invalid reference to the to BswM EcuC module definition
- ASCBSWM-145 Fixed known issue: The function BswM_LinSM_CurrentSchedule incorrectly evaluates Lin network schedule indication requests
- ASCBSWM-148 Fixed known issue: BswMFrSMIndication mode request sources are not recognized by the BswM generator

Module version 1.3.0
2012-10-24

- The top-level structure of the software-component description in the ARXML files changed from `/AUTOSAR/BswM` to `/AUTOSAR_BswM`
- ASCBSWM-109 Fixed known issue: BswM generation fails with a Java null pointer exception when a Mode-Type-Request-Map is not complete
- ASCBSWM-84 Fixed known issue: The BswM does not support a EthSM Channel selection for mode switch ports
- ASCBSWM-76 Fixed known issue: The verification of EcuM wakeup sources is carried out with incorrect identifiers
- ASCBSWM-112 Fixed known issue: The generated code for BswMCanSMIndication request port is wrong if more than one CanSMIndication is used

Module version 1.2.1
2012-06-27

- Internal module improvement. This module version update does not affect module functionality

Module version 1.2.0
2012-03-23

- Updating configuration and functionality to 4.0.3, preserving AR 4.0.2 paths to other module configurations

Module version 1.1.1
2011-10-14

- Integrated SWS enhancements and clarifications; other minor enhancements

Module version 1.1.0
2011-09-09

- AUTOSAR 4.0 version (full)

3.3.1.2. New features

- Support for J1939 mode request ports and actions

  BswM now supports mode request ports of type BswMJ1939DcmBroadcastStatus and BswMJ1939NmIndication. Also, actions of type BswMJ1939DcmStateSwitch and BswMJ1939RmStateSwitch are supported as well.

Configuration parameters BswMJ1939DcmEnabled and BswMJ1939NmEnabled need to be enabled if BswMJ1939DcmBroadcastStatus and BswMJ1939NmIndication ports are used.

The newly added API's BswM_J1939DcmBroadcastStatus and BswM_J1939Nm_StateChangeNotification, corresponding to BswMJ1939DcmBroadcastStatus and BswMJ1939NmIndication ports, are provided via the new header files BswM_J1939Dcm.h and BswM_J1939Nm.h.

3.3.1.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- This module provides no EB-specific enhancements.

3.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- BswMMainFunctionPeriod can be configured up to 255 seconds

  In order to optimize the BswMTimer implementation, the BswMMainFunctionPeriod parameter has been restricted to a maximum value of 255 seconds.

  This should not affect any project as this value is sufficiently high.

- Unsupported actions

  Description:

  The following actions are not supported.

  - `BswMTriggerSlaveRTEStop`
  - `BswMTriggerStartUpPhase2`

  Requirements:

  BswM0919_Conf, BswM0921_Conf, BswM0920_Conf,

- The short-names of the BswM service component description artifacts

  Description:

  The short-names of the following BswM service component description artifacts are not generated according to the requirement BswM0137: - RequirePort - ProvidePort - RunnableEntity - DataReceiveEvent

  Requirements:

BswM0137

- BswMDcmComMChannelRef in place of BswMDcmComMNetwork

  Description:

  The configuration parameter BswMDcmComMNetwork has been replaced with a reference-parameter BswMDcmComMChannelRef which references a ComMChannel.

  Rationale:

  References to ComM channel is much less error-prone than allowing arbitrary strings to be entered as ComM-channel-identifiers. This is the topic of AUTOSAR bugzilla issue http://www.autosar.org/bugzilla/show_bug.cgi?id=53057

  Requirements:

  BswM0876_Conf

- The parameter BswMEcuMSleepModeRef in place of BswMEcuMShutdownTargetRef

  Description:

  The configuration parameter BswMEcuMShutdownTargetRef has been replaced with a reference-parameter BswMEcuMSleepModeRef.

  Rationale:

  The Ecu configuration parameters for the shutdown target selection were not complete. BswMEcuMShutdownTargetRef could only reference a sleep target and not a reset target. The shutdown target selection has been adapted here: http://www.autosar.org/bugzilla/show_bug.cgi?id=54134

  Requirements:

  BswM0962_Conf

- Unsupported APIs BswM_TriggerStartUpPhase2 and BswM_TriggerSlaveRTEStop

  Description:

  The APIs BswM_TriggerStartUpPhase2() and BswM_TriggerSlaveRTEStop() are related to Multicore feature whose functionality is to trigger (start or stop) the RTE/SchM on slave core is not supported.

  Requirements:

  BswM0140, BswM0142, BswM0145, BswM0141, BswM0143, BswM0144

- Removed Det checks for BSWM_E_REQ_USER_OUT_OF_RANGE

  Description:

The Development Error Tracer checks associated to `BSWM_E_REQ_USER_OUT_OF_RANGE` have been removed since the BswM does not know the valid ranges of the user parameter.

Requirements:

BswM0092, BswM0094, BswM0096, BswM0100, BswM0102, BswM0108, BswM0111, BswM0151, BswM0155, BswM0114, BswM0098

- Optional element `BswMConditionValue`

  Description:

  The BswM requires that the ECU configuration parameter `BswMConditionValue` be always defined. It is not optional.

  Requirements:

  BswM0816_Conf

- Disabled parameter `BswMWdgMRequestPartitionResetRef`

  Description:

  The BswMWdgMRequestPartitionResetRef parameter cannot be used to indicate the partition for which the reset request is made for. This is due to the fact the Os module does not have the OsAppEcucPartitionRef parameter enabled. As a result, this functionality is broken. In order to correctly provide the partition for which the reset request is made for, the new EB specific parameter BswMWdgMOsApplicationRef must be used.

  Requirements:

  BswM0917_Conf

- BswM doesn't provide the declaration for `BswM_MainFunction`

  Description:

  The BswM will not provide the declaration of the BswM_MainFunction anymore. In case the BswM has only one instance the declaration of BswM_MainFunction will be provided by the Schm. In case of multiple instantiation, each partition will have its own BswM_1_{Partition}_MainFunction.

  Requirements:

  BswM0053, BswM0075, BswM.Impl.Api.BswM_MainFunction_Partition

- `BswMPartitionRef` references an `OsApplication` instead of a `EcucPartition`

  Description:

The BswMPartitionRef parameter doesn't reference an EcucPartition but an OsApplication due to Os restrictions.

Requirements:

ECUC_BswM_00984

- BswM always requires the SchM interface

  Description:

  The `BswMSchMEnabled` parameter will be disabled and it's default value shall be 'true'. The rationale is that the SchM interface is always needed by the BswM.

  Requirements:

  BswM00947_Conf

- BswM provides the header files for other modules only if BswM<mod> is set to TRUE

  Description:

  The newer version of this requirement (from Autosar 4.2) is preffered because it reduces the number of header files which are provided by the BswM.

  Requirements:

  BswM0026

- BswM_UserType and BswM_ModeType types are not configuration dependent.

  Description:

  According to BswM0041, the range of the BswM_UserType and BswM_ModeType types should be configuration dependent. Instead, the current implementation defines these types as uint16 to simplify the implementation. BswM0041 also mentions BswM_ConfigType but this is already covered by BswM0042.

  Requirements:

  BswM0041

- BswMRequestedModeMax is not used by the BswM.

  Description:

  As the BswM_ModeType is now fixed at uint16, it doesn't make any sense for the user to configure the BswMRequestedModeMax anymore.

  Requirements:

BswM0875_Conf

3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- For this module no limitations are known.

3.3.1.6. Open-source software

BswM does not use open-source software.

3.3.2. BswMAs module release notes

- Module version: 1.21.1.B337087
- Supplier: Elektrobit Automotive GmbH

3.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 1.21.1
2020-06-19

- Added optional inline name to Action element in the grammar.
- Added names conflicts resolving dialog.
- Added optional inline name to Expression and ModeCondition element in the grammar.
- Added optional inline name to ActionList and ActionListItem element in the grammar.

Module version 1.20.1
2020-02-21

- Fixed loading of clean configuration containing one freshly created port.

Module version 1.20.0

2019-10-11

- Added support for ModeCondition with invalid path reference.
- Added support for BswMNvMJobModeIndication ports with invalid NvMService value.
- Added support for ActionLists with invalid ActionListItems
- Added support for invalid logical operators in complex expressions.
- Added support for invalid references in logical expressions.
- Added support for invalid timer references in timer controls.
- Only nested rule is suggested for the NestedRule element.
- When non nested rule is added as NestedRule element a "Rule must be nested" validation error is shown.
- Fixed issue with ports not being proposed by content assist when they were defined under the rule.

Module version 1.19.0

2019-06-14

- Added suggestions for timer values in logical expressions.
- Updated grammar for rule's initState value
- For GenericPort parameter BswMRequestModeMax is not mandatory any more.
- Added suggestion for logical expression rules. Added suggestion for Timer Value in Timer Control action. Added dependency view to rule which runs TimerControl. Remove suggestion of InitialValue for BswMTimer.
- Updated grammar for RteSwitchPortRef field of BswMRteSwitchAckNotification
- Added invalid DataMapping handling
- Added possibility to save invalid reference paths
- Updated grammar for DataMapping field of BswMBswModeNotification, BswMSwcModeNotification and BswMSwcModeRequest
- Added support for invalid values of queue length inside SwitchPort
- Added support for invalid values of BswMModeRequesterId and BswMRequestedModeMax in GenericRequestPort
- Added support for invalid values of SwitchIPduModeValue in SwitchIPduModeAction
- Added possibility to save switch ports with empty reference path.
- Added support for rules with invalid expression path.
- Added support for rules with invalid path to TrueActionList or FalseActionList.
- Added support for invalid BswMTimerControl actions.

- Added support for empty path references.
- Added support for invalid enum values.
- Added support for invalid SendPort and SwitchPort references.
- ASCBSWMAS-117 Upon serialization (i.e. saving the BswM configuration back to the Ecu data model when the BswM Editor is closed and saved), if the user had deleted any artifacts from the BswM configuration, then only identifiable elements (ports and rules) will be deleted from the BswM configuration model. For elements which are not identifiable (actions, action lists, conditions, expressions), user will be asked for confirmation whether those object also should be removed.

Module version 1.18.0
2019-02-15

- Internal module improvement. This module version update does not affect module functionality

Module version 1.17.0
2018-10-26

- Added support for Timer according to ASR4.3.
- Added support for the BswMLegacyDcmEnabled parameter.

Module version 1.16.0
2018-06-22

- Added support for RTE mode switch acknowledgments: added a new type of mode request port (BswMRteSwitchAckNotification).
- Added QF-Tests for UI.

Module version 1.15.0
2017-12-15

- Added design documentation.
- Added buttons in the BswM Editor toolbar for adding and removing editor tabs/BswM configurations.
- Added requirements for the BswM Assistant.
- Added support for ethernet switch port groups: added a new type of mode request port (BswMEthIfPortGroupLinkStateChg) and a new type of action (BswMEthIfSwitchPortGroupRequestMode).

- ASCBSWMAS-103 Fixed known issue: Switch Ports which are referenced elsewhere in the configuration, in actions of type BswMRteSwitch or BswMSchMSwitch, will end up duplicated in the Ecu data model upon closing the BswM Editor (there will be 2 containers with the same name, defining the same switch port).
- Added Unit Tests for supported BswM artifacts, loading and saving BswM configurations.
- Added support for ReportFailedToDemRef.
- ASCBSWMAS-135 Fixed known issue: If there are 2 rules defined in a BswM configuration, where rule #1 is defined before rule #2 and rule #1 contains a reference to rule #2 in one of its action lists (rule #2 is referenced as a nested rule in one of the action lists of rule #1), then the BswM configuration will fail to load in the BswM Editor (the corresponding BswM Editor tab is empty after opening the BswM Editor). Also fixed: upon saving a BswM configuration which fails to load, the identifiable artifacts from that configuration will be removed.
- Added support for Car Wakeup: added a new type of mode request port (BswMNmIfCarWakeUpIndication) and a new boolean parameter under BswMGeneral (NM Enabled).
- Added remaining Unit Tests that were still missing.

Module version 1.14.0
2017-05-05

Module version 1.13.0
2017-03-31

- For configuration artifacts from the original configuration which have not been changed in the BswM editor, their shortnames and other attributes are preserved when the BswM Editor is closed and saved.
- When there are configuration artifacts which are inconsistent and cannot be loaded into the BswM Editor, then if the BswM Editor is saved those artifacts are no longer removed from the BswM configuration.

Module version 1.12.0
2016-11-04

- Improved validation in case of PduGroupSwitch and DeadlineMonitoringControl actions
- Improved validation in case of rules without any true or false action lists set

Module version 1.11.0
2016-05-25

- ASCBSWMAS-61 Improved validations for data type mapping sets
- ASCBSWMAS-60 Fixed known issue: Data type mapping sets are not correctly serialized if BswM-DataTypeMappingSetRef is not already enabled

Module version 1.10.0
2016-02-08

Module version 1.9.1
2016-01-15

- Removed the QuickOutline warning showing up in Tresos at plugin load
- ASCBSWMAS-38 Fixed known issue: Invalid configuration in case Rte actions are used in configuration which isn't named 'BswMConfig'
- ASCBSWMAS-23 Fixed known issue: Editor crashes randomly when using content assist
- ASCBSWMAS-41 Fixed known issue: the BswM Editor now enables the BswMUserIncludeFiles container if user includes are configured and automatically removes duplicate entries
- ASCBSWMAS-44 Fixed known issue: Editor doesn't display an error message when a referenced mode request port is removed
- ASCBSWMAS-45 Improved validation in case of EcuMSelectShutdownTarget action
- ASCBSWMAS-47 Improved validation in case of LinScheduleSwitch action

Module version 1.9.0
2015-11-10

- Added support for multiple instantiation

Module version 1.8.0
2015-06-24

- Added support for the BswMSwcModeRequestVariableDataPrototypeRef

Module version 1.7.0
2015-05-28

- Moved the BswM Editor to the new BswMAs repository

3.3.2.2. New features

- No new features have been added since the last release.

3.3.2.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.2.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- For this module no limitations are known.

3.3.2.6. Open-source software

Open-source software information is not available for this module.

3.3.3. ComM module release notes

- AUTOSAR R4.1 Rev 3
- AUTOSAR SWS document version: 4.1.1
- Module version: 5.19.4.B337087
- Supplier: Elektrobit Automotive GmbH

3.3.3.1. Change log

This chapter lists the changes between different versions.

Module version 5.19.4

2020-06-19

- ASCCOMM-1233 Fixed known issue: Compilation error is reported when multicore is enabled, number of configured PNCs is less than two and only one ComM channel is used

Module version 5.19.3

2020-02-21

- ASCCOMM-1198 Fixed known issue: A ComM channel configured with ComMNmVariant set to NONE and ComMBusType different than COMM_BUS_TYPE_INTERNAL always goes to state COMM_NO_-COMMUNICATION

Module version 5.19.2

2019-10-11

- Removed nested compiler abstraction macro
- Changed Max PncId from 63 to 511
- ASCCOMM-1181 Fixed known issue: ComM generates a non-compliant ARXML file for AUTOSAR 4
- ASCCOMM-1182 Fixed known issue: Inhibition status is not accessible via ComM_ChannelWakeup interface
- ASCCOMM-1187 Fixed known issue: Rte is not correctly informed about full communication requesters when wake-up inhibition or mode limitation is used

Module version 5.19.1

2019-06-14

- ASCCOMM-1138 Fixed known issue: A passive wakeup can be ignored for channels that have entered COMM_FULL_COM_NETWORK_REQUESTED and mode limitation is enabled
- ASCCOMM-1147 Fixed known issue: ComM could enter COMM_FULL_COM_READY_SLEEP while a user request is set
- ASCCOMM-1148 Fixed known issue: ComM could enter COMM_NO_COMMUNICATION while a user request is set
- ASCCOMM-1163 Fixed known issue: ComM fails to compile if multi-core is enabled and all ComM channels use the ComMNmVariant NONE or LIGHT
- Added Support for ComM0PncVectorAvoidance
- ASCCOMM-1166 Fixed known issue: ComM channel may fail to shut down when ComMDcmUsage and ComMModeLimitationEnabled are enabled

Module version 5.19.0
2019-02-15

- Changed structure of generated templates for post-build selectable support

Module version 5.18.1
2019-01-31

- Improved robustness check for references, optional parameters property and enable parameters property
- ASCCOMM-1068 Fixed known issue: ComM PNC identifiers are not correctly imported from system description

Module version 5.18.0
2018-10-26

- Implemented Multi-core support
- ASCCOMM-1087 Fixed known issue: Zero size array is generated for a gateway without ERA signals received on active channels
- ASCCOMM-1090 Fixed known issue: ComM generates an invalid basic software module description if no configuration set is provided
- ASCCOMM-1072 Fixed known issue: Wrong Pn information is sent on the bus if <Bus>Nm configures PnInfoOffset greater than 1
- Support for VLANs by means of managed and managing ComM channels
- ASCCOMM-1101 Fixed known issue: ComM_RteNotificationDoActFkp is not defined in correct memory section

Module version 5.17.5
2018-06-22

- ComMMode-ModeDeclarationGroup add support for ALPHABETIC_ORDER
- Added memory sections for the ComM software component description
- Updated ComM ComMPncId to be according to Asr4.3 Standard
- ASCCOMM-1004 Fixed known issue: Gateway ECU transmits incorrect PN information

Module version 5.17.4
2018-02-16

- ASCCOMM-992 Fixed known issue: The network management variant NONE is not available for LIN channels
- ASCCOMM-995 Fixed known issue: Root configuration alias always generated with ComMConfigSet_0

Module version 5.17.3
2017-12-15

- Improved Limit to COMM_NO_COMMUNICATION mode
- Inverted logic for AUTOSAR 4.0.2 and AUTOSAR 3.x symbolic names
- Post-build selectable support
- EcuM PNC wakeup indication don't wakeup all PNC if ComMSynchronousWakeUp is set TRUE
- ASCCOMM-985 Fixed known issue: Variable ComMPncMainfunctionRunning is not assigned to a memory section

Module version 5.17.2
2017-09-22

- Implemented ComM PNC wake-up indication extension
- ASCCOMM-935 Fixed known issue: ComM PNC state machine can change the state based on outdated PN information

Module version 5.17.1
2017-06-30

- Added support for Ethernet Switch Port Groups

Module version 5.17.0
2017-03-31

- ASCCOMM-864 Fixed known issue: Code generation error for specific configuration
- ASCCOMM-893 Fixed known issue: ComM state machine may get locked if its interfaces are called from different contexts
- Improved ComM could reject user requests even if the requests should be accepted in the current state
- ASCCOMM-897 Fixed known issue: The value of ComMPncPrepareSleepTimer could be inconsistent
- Removed duplicated REFINED_MODULE_DEF

Module version 5.16.0
2016-10-31

- Generate RTE_MODE symbols using explicit values instead of lexicographical order
- ASCCOMM-872 Fixed known issue: ComM_DCM_ActiveDiagnostic and ComM_DCM_InactiveDiagnostic can produce out of bounds memory access

Module version 5.15.0
2016-05-24

- ASCCOMM-838 Fixed known issue: Inconsistent state between ComM and BusNm
- ASCCOMM-840 Fixed known issue: ServiceNeedsWizard generates NvM block NVM_BLOCK_COMM independant of ComMGlobalNvMBlockDescriptor
- ASCCOMM-842 Fixed known issue: Code does not compile in case a single PNC is configuured and ComMPncGatewayEnabled is set to true

Module version 5.14.0
2016-02-10

- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`

Module version 5.13.0
2015-11-06

- Internal module improvement. This module version update does not affect module functionality

Module version 5.12.0
2015-07-28

- ASCCOMM-795 Fixed known issue: Invalid xdm checks regarding the need of ComMPncComSignal elements

Module version 5.11.0
2015-06-24

- Improved initialization of RxEiraMerged

Module version 5.10.0

2015-02-20

- Limit the ComMNmVariant values according to NmPassiveModeEnabled
- ASCCOMM-761 Fixed known issue: Invalid configuration in case of a single passive channel
- Fixed compiler warning regarding assignment of a variable

Module version 5.9.0

2015-01-07

- Recompute the block-size calculation of the ComM ServiceNeeds
- Replace enums with specific defines

Module version 5.8.0

2014-10-02

- ASCCOMM-681 Fixed known issue: Invalid filtering of EIRA Tx Signals
- ASCCOMM-694 Fixed known issue: PNC, which was not requested by a ComM user, keeps the bus awake
- ASCCOMM-697 Fixed known issue: Invalid handling of multiple EIRA Signals
- Fixed an inconsistency in the AUTOSAR specifications 4.0.3 and 4.1 regarding partial networking that can lead to a livelock situation that prevents buses going to sleep according to Rfc #64979
- ASCCOMM-726 Fixed known issue: Wrong description of `ComMPncReleaseChannel`
- DCC compiler warnings
- Redesign channel state machine and PNC state machine for ComM
- Protect variables using Exclusive Areas where necessary
- Partial Network Gateway Coordinator
- ASCCOMM-746 Fixed known issue: Compile error occurs in case a single PNC is configured

Module version 5.7.0

2014-04-25

- ASCCOMM-671 Fixed known issue: PNC, which was not requested by a ComM user, keeps the bus awake
- ASCCOMM-663 Fixed known issue: `PreventWakeUp` inhibition cannot be set while channel is on `FULL_COM`
- ASCCOMM-692 Fixed known issue: Fix typo in label of `ComMPncComSignalDirection` parameter

Module version 5.6.1

2014-01-27

- ASCCOMM-652 Fixed known issue: Enabling parameter `ComMRelocatablePbcfgEnable` leads to a compiler error if BswM is also configured for `PostBuild`

Module version 5.5.2

2014-01-14

- ASCCOMM-625 Fixed known issue: `ComM_GetMaxComMode()` returns an incorrect mode when wake-up inhibition is active
- ASCCOMM-626 Fixed known issue: ComM channel main function gets stuck in an endless loop if the total number of Rx EIRA Com signals is greater than 255
- ASCCOMM-628 Fixed known issue: Concurrent access to internal variables leads to unexpected behavior
- Implemented support of `VARIANT-POST-BUILD` for ComM
- ASCCOMM-638 Fixed known issue: Wrong type definition of `ComM_ASR40_UserHandleType` in AUTOSAR 3.2 projects

Module version 5.5.1

2013-10-11

- ASCCOMM-581 Fixed known issue: Parameters `ComMPncComSignalRef` and `ComMBusSMRequestRetryCount` throw errors when *Config time feature* is enabled
- ASCCOMM-577 Fixed known issue: Possible deadlock of a PNC in `PNC_READY_SLEEP` state
- ASCCOMM-514 Fixed known issue: A channel which is woken-up passively may shut down unexpectedly
- ASCCOMM-602 Fixed known issue: Mode Switch Points are not generated for users connected to a channel via PNCs

Module version 5.5.0

2013-06-26

- Added AUTOSAR 3.2 support of Rte Interface and SWCD
- ASCCOMM-441 Fixed known issue: Multiple Com signals cannot be used to communicate PNC status information
- Improved the robustness of the channel/PNC state machine designs by revising the event handling; removed configuration parameter `ComMEventQueueSize`
- ASCCOMM-467 Fixed known issue: Invalid error during configuration: The length of the referenced Rx `ComSignal` does not match the size of EIRA stored in ComM

- ASCCOMM-499 Fixed known issue: A channel with `ComMNmVariant = PASSIVE` cannot be mapped to a PNC
- ASCCOMM-516 Fixed known issue: If the values of `ComMPncIds` are not configured in ascending order, requesting or releasing a PNC may cause a state change in a different PNC
- ASCCOMM-521 Fixed known issue: The same channel cannot be referenced by a user via different PNCs
- ASCCOMM-492 Fixed known issue: ComM does not leave `SILENT COMMUNICATION` mode
- ASCCOMM-428 Fixed known issue: If a user requests full communication after a network channel is released, ComM and Nm may get out of synchronization
- ASCCOMM-510 Fixed known issue: `ComM_GetRequestedComMode()` returns `COMM_NO_COMMUNICATION` even though the last requested mode is `COMM_FULL_COMMUNICATION`
- ASCCOMM-548 Fixed known issue: Unexpected module behavior if `ComM_Init()` is preempted by another ComM API
- ASCCOMM-543 Fixed known issue: A faulty user request that is incorrectly stored may keep the bus awake after mode limitation is disabled
- ASCCOMM-550 Fixed known issue: Compiler warnings due to too large shift counts on 16-bit architecture

Module version 5.4.0
2013-02-19

- ASCCOMM-456 Fixed known issue: ComM never requests `NO_COMMUNICATION` at BusSM modules in case `FULL_COMMUNICATION` mode cannot be reached
- ASCCOMM-445 Fixed known issue: User request of `COMM_FULL_COMMUNICATION` is not granted if inhibition gets disabled
- Added sanity checks to ensure that the value of configuration parameter `ComMDcmUsage` matches existence of the Dcm module in the project
- ASCCOMM-463 Fixed known issue: Compiler warnings due to undefined symbol `COMM_DCM_ENABLED` in `ComM_Dcm.h`
- Corrected the schema structure by adding mandatory container `ComMConfigSet`
- Memory allocation keywords were implemented in compliance to AUTOSAR 4.0.3
- ASCCOMM-465 Fixed known issue: Compiler errors when symbolic names according to AUTOSAR 4.-0.3 are used
- ASCCOMM-390 Fixed known issue: Channel with Nm Variant set to `LIGHT` does not go to `COMM_NO_COMMUNICATION` mode if Nm Light Timeout is configured as `0`

Module version 5.3.0
2012-10-12

- ASCCOMM-361 Fixed known issue: ComM state machine never leaves the substate COMM_NO_COM_REQUEST_PENDING if BusSM module does not indicate COMM_FULL_COMMUNICATION or a user requests COMM_NO_COMMUNICATION
- ASCCOMM-384 Fixed known issue: Communication channel is shut down even if *Full Communication* is requested when PNC support is enabled and only one channel is configured
- ASCCOMM-351 Fixed known issue: ComM prohibits inhibiting a channel when active diagnostic session is indicated for another channel
- ASCCOMM-371 Fixed known issue: Memory access violation when length of the ComSignal to receive EIRA is greater than the size of EIRA or ComMPncId is greater than the number of PNCs configured
- Removed obsolete DET error reporting of error code COMM_E_ERROR_IN_PROV_SERVICE
- ComM-Dcm interface made optional
- ASCCOMM-300 Fixed known issuee: Software component might not get notified about mode switch
- ASCCOMM-356 Fixed known issue: Undefined symbol names for ComM_ModeType
- The top-level structure of the software-component description in the ARXML files changed from /AUTOSAR/ComM to /AUTOSAR_ComM
- ASCCOMM-342 Fixed known issue: ComM state machine gets stuck in FULL_COMMUNICATION mode
- ASCCOMM-398 Fixed known issue: ComM may fall back to state *No communication* during processing of a *Full communication* request due to pending bus sleep mode event

Module version 5.2.1
2012-06-27

- ASCCOMM-350 Fixed known issue: Wake up inhibition gets active for a channel although it is in state COMM_FULL_COMMUNICATION in contrast to requirement [ComM219]
- ASCCOMM-364 Fixed known issue: Memory section not closed (correctly for the variable ComM_PncTimer)
- Implementation of support for bus type Eth
- ASCCOMM-362 Fixed known issue: Declaration of ComM Com call back function is unavailable if RTE usage is ON

Module version 5.2.0
2012-03-23

- ASCCOMM-268 Fixed known issue: Timings ComMTMinFullComModeDuration and ComMNmLightTimeout might vary between different channels and also be shorter than expected time
- ASCCOMM-267 Fixed known issue: Implement the return feature COMM_E_UNINIT in all ComM APIs

- ASCCOMM-289 Fixed known issue: Function `ComM_RequestComMode()` incorrectly returns `E_OK` during an active mode limitation
- ASCCOMM-297 Fixed known issue: ComM state machine gets stuck and therefore may never release the run request from EcuM
- ASCCOMM-308 Fixed known issue: Usage of Nm related inclusions/functions without using Nm
- ASCCOMM-314 Fixed known issue: ComM doesn't grant requested communication modes if inhibitions are deactivated

Module version 5.1.0
2011-08-29

- Initial AUTOSAR 4.0 version

3.3.3.2. New features

- No new features have been added since the last release.

3.3.3.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- Code size optimization

  To minimize the code size for simple module configurations the ComM makes use of preprocessor conditionals removing if-branches, loops, and array look-ups wherever possible.

  Rationale:

  This enhancement implements partly the HIS requirements concerning code size: HisComM0001, HisComM0002, HisComM0003, HisComM0004, HisComM0005.

- Configurable parameter `ComMPncReleaseChannel` used for immediately stopping transmission of Nm messages after all PNCs of a channel are released

  In case `ComMPncReleaseChannel` is `COMM_NO_INTERNAL_REQUESTS`, all Full Communication requests are released after all PNCs of a channel leave the `PNC_REQUESTED` state.

  In order to maintain compatibility with the previous version, the default value of the `ComMPncReleaseChannel` is `COMM_NO_INTERNAL_AND_EXTERNAL_REQUESTS`. In this case, all Full Communication requests are released when all PNCs of a channel enter the `PNC_NO_COMMUNICATION` state.

  Rationale:

AUTOSAR 4.0.3 does not specify the exact moment when the PNC Full Communication requests shall be released. In order to stay forward compatible, the previous implementation used the new requirements from AUTOSAR 4.1 which specified that all Full Communication requests would be released in the `PNC_NO_COMMUNICATION` state.

However, in case of external requests, this approach sometimes leads to livelock situation where the bus never goes to sleep and all ECU's stay awake. In this case, the best solution is to release the Full Communication requests after leaving the `PNC_REQUESTED` state. The `ComMPncReleaseChannel` parameter allows to switch between the two behaviors.

This issue is currently discussed on the AUTOSAR Bugzilla Rfc #63643.

- Extended the range for PNC ids

  Description:

  The range for PNC ids changed from 8..63 to 8..255

  Rationale:

  Since in BusNms the PnInfoOffset changed to 1..31 and becuse the PNC id is the absolute position in the PDU, to be able to use offsets greater than 8 bytes PNC id range has been extended.

3.3.3.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- Communication mode indication and BusSM mode

  Description:

  According to ComM083 and ComM084, a call of `ComM_GetCurrentComMode()` shall be propagated to the underlying Bus State Manager (`BusSM_GetCurrentComMode()`) and the mode retrieved from the BusSM shall be indicated to the upper layer as the current communication mode. In contrast to these requirements, ComM does not propagate a call of `ComM_GetCurrentComMode()` to the underlying Bus State Manager in case multicore is enabled. Instead, ComM indicates the communication mode based on the stored communication mode which is previously indicated by the underlying Bus State Manager.

  Rationale:

  This approach is chosen due to the fact that ComM and BusSm can be on different cores and a call to `BusSM_GetCurrentComMode()` would lock also the core on which the BusSM is.

  Requirements:

  ComM083, ComM084

- Types `ComM_InitStatusType` and `ComM_PncModeType`

  Description:

  The types `ComM_InitStatusType` and `ComM_PncModeType` are not enumerations as prescribed by the AUTOSAR SWS but of type `uint8`.

  Rationale:

  The use of the `uint8` type saves RAM and ROM because an enumeration value would take up 2 or 4 bytes per value depending on the CPU architecture.

  Requirements:

  ComM863 refinements: ComM.Types_Implicit1_Refine and ComM.Types_Implicit6_Refine

- No multiple network requests in state `COMM_FULL_COM_NETWORK_REQUESTED`

  Description:

  According to requirements ComM869 and ComM870, if a user requests `COMM_FULL_COMMUNICATION` or the DCM indicates *Active Diagnostic* in substate `COMM_FULL_COM_NETWORK_REQUESTED` and configuration parameter `ComMNmVariant` is set to `FULL`, the ComM module shall invoke `Nm_NetworkRequest()` for the corresponding NM channel(s). The deviation is present for AUTOSAR 4.0.3

  In contrast to this, `Nm_NetworkRequest()` is not called if a user requests `COMM_FULL_COMMUNICATION` or the DCM indicates *Active Diagnostic* and the channel is already in substate `COMM_FULL_COM_NETWORK_REQUESTED`. Instead, `Nm_NetworkRequest()` is called only once when the substate `COMM_FULL_COM_NETWORK_REQUESTED` is entered after a user requests `COMM_FULL_COMMUNICATION` or the DCM indicates *Active Diagnostic* in state `COMM_NO_COM_NO_PENDING_REQUEST`.

  Rationale:

  The network is already requested once when the substate `COMM_FULL_COM_NETWORK_REQUESTED` is entered after a user requests `COMM_FULL_COMMUNICATION` or the DCM indicates Active Diagnostic in state `COMM_NO_COM_NO_PENDING_REQUEST`. Therefore it does not make sense to request the network again each time a user requests `COMM_FULL_COMMUNICATION` unless due to ComM980. Besides, there is no textual requirement demanding network to be requested *on entering* substate `COMM_FULL_COM_NETWORK_REQUESTED`. This issue is reported to AUTOSAR in http://www.autosar.org/bugzilla/show_bug.cgi?id=59575.

  Requirements:

  ComM869, ComM870

- Inclusion of header files `CanSM.h` and `Dcm.h`

  Description:

According to requirement ComM506, ComM shall include the header files `CanSM.h` and `Dcm.h`. In contrary to this, ComM does not include the header files `CanSM.h` and `Dcm.h` but includes `CanSM_ComM.h` and `Dcm_Cbk.h` instead. The deviation is present for AUTOSAR 4.0.3.

Rationale:

Requirement CANSM008 in the AUTOSAR specification of the CAN State Manager specifies that `CanSM.h` does not provide the declarations of API functions `CanSM_GetCurrentComMode()` and `CanSM_RequestComMode()`. Additionally, requirement CANSM009 specifies that the header file `CanSM_ComM.h` exports the APIs dedicated to the ComM module.

Similarly, requirement [Dcm110] in the AUTOSAR specification of the Diagnostic Communication Manager specifies that `Dcm.h` contains only those types, functions and parameters that are visible to SWCs. Additionally, the callback APIs `Dcm_ComM_NoComModeEntered()`, `Dcm_ComM_SilentComModeEntered()` and `Dcm_ComM_FullComModeEntered()` are specified in section *8.4 Callback Notifications* of the same document with the description that the function prototypes of these callback APIs are provided in the file `Dcm_Cbk.h`.

This issue has been fixed in R4.1.1 of the Specification of Communication Manager where requirement SWS_ComM_00506 specifies that ComM shall include the header files `Dcm_Cbk.h` and `CanSM_ComM.h`.

Requirements:

ComM506

- Behavior of `ComM_Nm_BusSleepMode()`

Description:

A call of `ComM_Nm_BusSleepMode()` shall perform the transition of the hardware and transceiver to bus-sleep mode in a synchronous way according to the AUTOSAR ComM SWS. In contrast to that the EB ComM performs this operation asynchronously during the main function call. Deviation is present for AUTOSAR 4.0.3

Rationale:

The synchronous handling of these actions require locking of global resources, is error-prone, and inefficient with respect to run-time and code size.

Requirements:

ComM392

- Availability of `ComM_SetECUGroupClassification()`

Description:

The API function `ComM_SetECUGroupClassification()` is only available if either the configuration parameter `ComMWakeupInhibitionEnabled` or `ComMModeLimitationEnabled` is set to `true`. This contradicts requirement ComM552 in which no availability condition is defined.

Rationale:

The API function `ComM_SetECUGroupClassification()` will never be needed for module configurations with `ComMWakeupInhibitionEnabled = false` and `ComMModeLimitationEnabled = false`.

Requirements:

ComM552

- Correction: Diagnostic requests ignored for *PASSIVE* channels

  Description:

  A call of `ComM_DCM_ActiveDiagnostic()` in state `COMM_NO_COM_NO_PENDING_REQUEST` or `COMM_SILENT_COMMUNICATION` or `COMM_FULL_COM_READY_SLEEP` shall trigger the necessary transitions towards `COMM_FULL_COM_NETWORK_REQUESTED` state according to the requirements ComM876, ComM878, and ComM883 respectively. In contrast to these requirements, the call of `ComM_DCM_ActiveDiagnostic()` for a channel with `ComMNmVariant = PASSIVE` is simply ignored and hence does not trigger any expected transitions. Deviation is present for AUTOSAR 4.0.3

  Rationale:

  Active Diagnostic is not possible on a *PASSIVE* channel since it is not possible to keep the channel awake. This problem is reported to AUTOSAR in http://www.autosar.org/bugzilla/show_bug.cgi?id=56661.

  Requirements:

  ComM876, ComM878, ComM883

- No AUTOSAR Debugging support

  Description:

  ComM is not instrumented for the usage with AUTOSAR Debugging. Deviation is present for AUTOSAR 4.0.3

  Rationale:

  Debugging support according to the AUTOSAR standard is not planned.

  Requirements:

  ComM850, ComM851

- Symbolic port name support

  Description:

  The port names provided by the ComM are not named by their numeric index of the configured ComM user or ComM channel as suggested by the AUTOSAR ComM SWS in section 7.14. Instead the ports are postfixed by the symbolic name of the configured ComM user or ComM channel.

  Rationale:

  With symbolic names port names do not change when ports are deleted or inserted and renumbered. Therefore ports must not be reconnected.

  Requirements:

  ComM662

- Limited HIS Optimizations

  Description:

  According to HIS requirements (HisComM0001, HisComM0002, HisComM0003, HisComM0004 and HisComM0005), if only one single `ComMChannel` is configured, the module code should be optimized. That is, in the above condition the ComM module should do the following:

  - Make use of the `ComMChannel` configuration directly in the code.
  - Not make branches at runtime depending on this configuration data.
  - Avoid to separately store this configuration data.
  - Not contain any code for the user to channel mapping like switch blocks, mapping tables, callback lists.
  - Not use the indirection to access data indexed by the user ID.

  In contrast to these requirements, the above optimizations are not done for the parameters `ComMNoWakeup`, `ComMNoFullCom`, and `ComMNoCom`.

  Requirements:

  HisComM0001, HisComM0002, HisComM0003, HisComM0004 and HisComM0005

- Variant `VARIANT-PRE-COMPILE` is not supported.

  Description

  The ComM supports the variant `VARIANT-POST-BUILD` only. The implementation of variant `VARIANT-PRE-COMPILE` is not required since all functionality of `VARIANT-PRE-COMPILE` is implemented by `VARIANT-POST-BUILD`.

3.3.3.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- For this module no limitations are known.

3.3.3.6. Open-source software

ComM does not use open-source software.

3.3.4. EcuM module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 3.0.0
- Module version: 5.15.4.B337087
- Supplier: Elektrobit Automotive GmbH

3.3.4.1. Change log

This chapter lists the changes between different versions.

Module version 5.15.4
2020-06-19

- Internal module improvement. This module version update does not affect module functionality.

Module version 5.15.3
2020-02-21

- Internal module improvement. This module version update does not affect module functionality.

Module version 5.15.2
2019-10-11

- ASCECUM-1169 Fixed known issue: Linker error may occur because of incomplete memory mapping of EcuM_DriverInitListOneCall() and EcuM_DriverRestartListCall().
- ASCECUM-1181 Fixed known issue: EcuM may fail to start the OS because an invalid OsAppMode is used.
- Removed AUTOSAR 3.x symbolic name value macros and inverted logic to only provide AUTOSAR 4.0.-2 compliant macros if ECUM_PROVIDE_LEGACY_SYMBOLIC_NAMES is defined.

Module version 5.15.1
2019-06-14

- ASCECUM-1160 Fixed known issue: Linker error may occur because of incomplete memory mapping of EcuM_SchM_Init() and EcuM_SchM_Deinit().
- ASCECUM-1161 Fixed known issue: Wrong MemMap INIT_POLICY for EcuM global variables may cause undefined behavior because of unexpected return value.

Module version 5.15.0
2019-03-28

- Added support for Post-build variant handling.

Module version 5.14.4
2019-02-15

- Internal module improvement. This module version update does not affect module functionality.

Module version 5.14.3
2018-10-26

- Internal module improvement. This module version update does not affect module functionality.
- Added configuration parameter to call `StartCore()` also for the core that boots up the system

Module version 5.14.2
2018-06-22

- Added memory sections for the EcuM software component description

- ASCECUM-590 Fixed known issue: Validated wakeup events will be ignored when set before entering the SLEEP state
- ASCECUM-1088 Fixed known issue: Wrong EcuM instance ID reported to Det if defensive programming is enabled
- ASCECUM-1011 Fixed known issue: Uncompilable code if EcuM wakeup sources are configured only with a ComM PNC reference

Module version 5.14.1
2018-02-16

- Internal module improvement. This module version update does not affect module functionality.

Module version 5.14.0
2017-09-22

- Added support for Ethernet Switch Port Groups
- Added support for BswM notifications of each EcuM state change.

Module version 5.13.0
2017-03-31

- ASCECUM-923 Fixed known issue: EcuM does not inform BswM about wakeup state changes if the StartupTwo sequence is interrupted

Module version 5.12.1
2017-03-10

- ASCECUM-908 Fixed known issue: The single-core EcuM never invokes EcuM_AL_DriverRestart
- ASCECUM-899 Fixed known issue: EcuM wrongfully enters the Sleep state if the GoSleep sequence is interrupted

Module version 5.12.0
2016-11-04

- Enhanced the basic software module description

- ASCECUM-876 Fixed known issue: The slave-core EcuM may not perform all of its activities during ECU shutdown
- StartCore will not be called for the already started initial core
- The multicore EcuM supports now a core synchronization safety mechanism
- Fixed issue with exporting unused configuration parameters

Module version 5.11.0
2016-05-25

- ASCECUM-852 Fixed known issue: The single-core EcuM does not report the RamHash result to Dem
- ASCECUM-855 Fixed known issue: DET errors are always reported from the first active core

Module version 5.10.1
2016-04-15

- ASCECUM-848 Fixed known issue: The multicore EcuM does not complete the ECU shutdown and sleep operations
- The multicore EcuM supports now a configurable designation of its master core

Module version 5.10.0
2016-02-10

- ASCECUM-821 Fixed known issue: The definition of the ECUM_SLAVE_SYNC_MASK may contain invalid characters
- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`

Module version 5.9.1
2016-01-15

- Adapted the usage of the OsResource RES_SCHEDULER.
- ASCECUM-806 Fixed known issue: Compilation fails when the OS module doesn't provide the OS_CORE_ID_MASTER macro.
- ASCECUM-803 Fixed known issue: EcuM master-slave synchronization is missing during EcuM_GoPoll()
- Handled return value of SchM Sender-Receiver APIs used in EcuM APIs

Module version 5.9.0
2015-11-10

- Added support for master-satellite EcuM multi-core

Module version 5.8.0
2015-06-24

- EcuM.c has been changed from a generated file to a static file.
- Implemented the EcuM multi-core startup and shutdown sequences.
- ASCECUM-764 Fixed known issue: EcuM Fixed Modes are available even if EcuM Fixed Support is disabled.

Module version 5.7.0
2015-02-20

- Fixed warnings present during SystemDescriptionImport within EcuM files

Module version 5.6.0
2015-01-07

- Fixed MISRA violations

Module version 5.5.0
2014-10-02

- ASCECUM-687 Fixed known issue: The EcuM API function `EcuM_SelectShutdownTarget` may not be defined
- The attribute postBuildChangeable has been defined for all EcuC containers that can have multiple entries
- Created service needs requests for Dem events
- Defined RunnableEntityArgument elements for functions related to the EcuM Run Request API.
- ASCECUM-685 Fixed known issue: Symbolic name constants for EcuM wakeup sources are incorrectly generated by the EcuM.

Module version 5.4.9
2014-06-13

- ASCECUM-664 Fixed known issue: Link-time errors for EcuM data types may occur if compiler optimizations are used

Module version 5.4.8
2014-04-25

- ASCECUM-676 Fixed known issue: Wakeup sources in the `PENDING` state at the time of an ECU reset cannot be restarted
- ASCECUM-688 Fixed known issue: The header file `EcuM_Cbk.h` is not included by the `EcuM.h`

Module version 5.4.7
2014-03-21

- ASCECUM-653 Fixed known issue: EcuM may generate the wrong definition for the constant `ECUM_NORMALMCUMODE`
- Updated the basic software module description for Debug & Trace support
- ASCECUM-661 Fixed known issue: Port-defined argument values are not defined for the `EcuM_StateRequest` API
- ASCECUM-668 Fixed known issue: The EcuM notifies the wrong mode when the state `PREP_SHUTDOWN` is entered
- ASCECUM-670 Fixed known issue: Invalid warning about the parameter `EcuMModuleInitConfigStr` when the `PreInit` service is configured
- ASCECUM-672 Fixed known issue: The event `ECUM_E_ALL_RUN_REQUESTS_KILLED` is reported to DEM even if it is not enabled in the EcuM configuration
- ASCECUM-673 Fixed known issue: The EcuM may not start wakeup sources that have been previously cleared

Module version 5.4.6
2013-09-17

- ASCECUM-640 Fixed know issue: The EcuM source code may not compile when EcuM fixed support is enabled
- Added description of production errors to the module reference documentation

Module version 5.4.5
2013-08-20

- Added a range check for the configuration parameter `EcuMValidationTimeout`
- ASCECUM-632 Fixed know issue: The EcuM passes the wrong Rte modes when the ASR32 and ASR40 service APIs are enabled
- ASCECUM-635 Fixed know issue: A preprocessor error may be generated if an application header file is included before the `EcuM.h`

Module version 5.4.4
2013-06-21

- EcuM wakeup source status changes are now implemented within an internal state machine
- Refactored the function `EcuM_GoHaltOrPoll` into the functions `EcuM_GoHalt`, `EcuM_GoPoll`, `EcuM_GoSleep`, and `EcuM_WakeupRestart`
- Refactored the handling of `RUN` and `POST_RUN` requests within the `EcuM_MainFunction` into the function `EcuM_HandleRunRequest`
- ASCECUM-601 Fixed known issue: When the EcuM is in a substate of `RUN`, events for non-communication channels will not be ignored
- Added missing Det checks for `ECUM_E_UNINIT`
- Removed references of symbolic name parameters by the ECU configuration short name
- Added AUTOSAR 3.2 support of Rte Interface and SWCD
- Added AUTOSAR 3.2 support of Rte Interface and SWCD
- Added Debug & Trace support for the EcuM

Module version 5.4.3
2013-05-15

- ASCECUM-580 Fixed known issue: A validated wakeup event will not be returned by `EcuM_GetValidatedWakeupEvent` after the `EcuM_MainFunction` has been executed
- The EcuM mode switch port `currentMode` now defines a queue length
- ASCECUM-582 Fixed known issue: The function `EcuM_SelectShutdownTarget` does not ignore invalid parameters when development error detection is disabled
- ASCECUM-581 Fixed known issue: The function `EcuM_ClearWakeupEvent` does not clear validated and expired wakeup events
- ASCECUM-569 Fixed known issue: Wakeup events that are set while transitioning to the `SLEEP` state are incorrectly set to expired
- ASCECUM-583 Fixed known issue: Wakeup event handling during the initialization of the EcuM is not supported

- ASCECUM-578 Fixed known issue: The EcuM may incorrectly signal the BswM that a pending wakeup event has been cleared
- ASCECUM-571 Fixed known issue: When the `SLEEP` phase is entered, the EcuM will signal the BswM that all wakeup sources are in the state `ECUM_WKSTATUS_NONE`
- Merged the files `EcuM_Callouts.c` and `EcuM_CalloutList.c` into the file `EcuM_Callout_Stubs.c`
- Default implementations of `EcuMDriverInitListZero`, `EcuMDriverInitListOne`, and `EcuMDriverRestartList` are now defined within the file `EcuM_Cfg.c`

Module version 5.4.2
2013-04-16

- ASCECUM-528 Fixed known issue: ECU cannot be woken up by peripherals other than communication channels
- ASCECUM-566 Fixed known issue: The EcuM may not compile as it may reference undefined constants within the file `EcuM_Cfg.c`
- Reworked the inclusion of the header file `Rte_EcuM_Types.h`
- ASCECUM-567 Fixed known issue: The BswM is not notified when the validation of a wakeup source fails

Module version 5.4.1
2013-03-18

- ASCECUM-546 Fixed known issue: The EcuM service component description may contain invalid reference values
- ASCECUM-552 Fixed known issue: The EcuM may report compiler warnings as it may not include `Rte_Main.h` when Rte support is enabled

Module version 5.4.0
2013-02-21

- Changed the reference path of `ComMChannel` in parameter `EcuMComMChannelRef` to `/AUTOSAR/EcucDefs/ComM/ComMConfigSet/ComMChannel`
- ASCECUM-512 Fixed known issue: `EcuM_StartWakeupSource` called multiple times
- ASCECUM-526 Fixed known issue: Compilation error due to undefined symbol for `McuModeSettingConf`
- ASCECUM-529 Fixed known issue: The EcuM generates the identifiers of symbolic name parameters incorrectly

Module version 5.3.1

2013-01-18

- ASCECUM-519 Fixed known issue: If the configured Wake-up Source IDs are not consecutive some wake-up events may not be correctly processed

Module version 5.3.0

2012-10-12

- ASCECUM-454 Fixed known issue: `SwcBswMapping` is located at the wrong location in BSWMD
- ASCECUM-458 Fixed known issue: Validity check of parameter `EcuMComMChannelRef` causes exception if reference is invalid
- ASCECUM-481 Fixed known issue: `EcuM_Bswmd.arxml` and `EcuM_swc_internal.arxml` are inconsistent to each other
- ASCECUM-476 Fixed known issue: Wakeup is not notified to ComM when only a single channel is configured in ComM
- The top-level structure of the software-component description in the ARXML files changed from `/AUTOSAR/EcuM` to `/AUTOSAR_EcuM`
- Removed support for multiple `EcuMConfiguration` containers
- Added a compatibilty interface for basic EcuM Fixed support
- EcuM updated to AUTOSAR 4.0. Rev 3

Module version 5.2.0

2012-03-16

- EBAECUM-269 Fixed known issue: In order to adhere to the ecuc_sws_6007 in AUTOSAR 4.0 Rev 2 EcuM Fixed parameter shall be added to VSMD
- Preconfigured the mapping of Mcu reset reasons to EcuM wakeup sources
- Changed the interpretation of the configuration parameter `EcuMWakeupSourceId` from a bit array value to a bit position value

Module version 5.1.0

2011-08-26

- Initial AUTOSAR 4.0 version

3.3.4.2. New features

- No new features have been added since the last release.

3.3.4.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- `EcuM_Init()` performs DET checks

  Description:

  The implementation throws the DET error `ECUM_E_NULL_POINTER` and cancels the execution of `EcuM_Init()` if `EcuM_DeterminePbConfiguration()` returns `NULL`.

  Rationale:

  By this check a missing post-build time configuration will not lead to dereferencing a `NULL` pointer.

- Enhanced production error reporting

  Description:

  An enhanced production error reporting mechanism has been introduced. This allows to configure the following options independently for each Dem event:

  - Report production errors to the Diagnostics Event Manager (Dem).
  - Report production errors to the Development Error Tracer (Det) as development errors.
  - Do not report production errors at all.

  If a production error is redirected towards the `Det`, you may configure the reported `Det` error-ID.

  Rationale:

  This enhancement implements the HIS requirements concerning fault operation and error detection: HisGen0007, HisGen0008 and HisGen0009.

- Support for non-default header file inclusion for initialization lists

  Description:

  The vendor-specific configuration parameter `EcuMModuleHeaderFile` allows to specify dedicated include files for each entry in the initialization lists.

  Rationale:

  Per default the AUTOSAR configuration allows the EcuM only to include the regular API headers of the AUTOSAR modules given by the parameter `EcuMModuleId`. To call `LinTp_Init()` which is declared in LinIf.h or to be able to integrate non-AUTOSAR modules it is necessary to specify header files with arbitrary names.

- Support for initializations of non-AUTOSAR modules in initialization lists

  Description:

  The vendor-specific configuration parameter `EcuMModuleInitConfigStr` allows to specify to give a configuration pointer to the initialization function of non-AUTOSAR modules.

  Rationale:

  Per default the AUTOSAR configuration allows only to pass pointers to module configuration referenced by `EcuMModuleConfigurationRefs` to initialization functions. With `EcuMModuleInitConfigStr` the user is able to overwrite the default behavior for any module.

- Provision of `ShutdownHook()` function

  Description:

  EcuM provides a minimalistic implementation of the Os callback function `ShutdownHook()`. If the EcuM shall provide this implementation it can be activated with the EB-specific configuration parameter `EcuMProvideShutdownHook`.

  Rationale:

  If the Os is integrated with the EcuM in a standard way, then the implementation of the `ShutdownHook()` is always the same. Using this feature will save time at integration as it is no longer necessary to provide this standard implementation explicitly.

- Support for initialization of modules with upper multiplicity greater than 1

  Description:

  EcuM supports initialization of modules which have an upper multiplicity greater than one. The initialization service shall be generated in the following format: `EcuMModuleID_VendorId_VendorApiInfix_EcuMModuleService>().`

  Rationale:

  Modules with an upper multiplicity greater than 1 are usually some driver modules, e.g. Fr. For these modules initialization APIs are available in the following format: `EcuMModuleID_VendorId_VendorApiInfix_EcuMModuleService>().`

3.3.4.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- Time Tiggered Increased Inoperation

  Description:

  The feature *Time Tiggered Increased Inoperation* (TTII) is not supported.

- Configuration Consistency check (reference to product description: ASCPD-145)

  Description:

  Configuration consistency check is not implemented and the related DEM error `ECUM_E_CONFIGURATION_DATA_INCONSISTENT` is not supported.

  Requirements:

  ECUM163_Conf

- No support for alarm clock functionality (reference to product description: ASCPD-143)

  Description:

  All API functions related to wakeup alarm handling are not supported. These are `EcuM_SetRelWakeupAlarm()`, `EcuM_SetAbsWakeupAlarm()`, `EcuM_AbortWakeupAlarm()`, `EcuM_GetCurrentTime()`, `EcuM_GetWakeupTime()`, `EcuM_SetClock()`.

  Requirements:

  EcuM4002, EcuM4069, EcuM4086, EcuM4087, EcuM4088, EcuM4070, EcuM4071, EcuM4072, EcuM4073, EcuM4089, EcuM4009, EcuM4010, EcuM3013, EcuM4054, EcuM4055, EcuM4056, EcuM4057, EcuM4058, EcuM4059, EcuM4060, EcuM3019, EcuM4061, EcuM4062, EcuM4063, EcuM4064, ECUM199_Conf, ECUM200_Conf, ECUM184_Conf, ECUM186_Conf, ECUM188_Conf, ECUM195_Conf, ECUM201_Conf, ECUM146_Conf, ECUM197_Conf, ECUM198_Conf

- No support for defensive behavior (reference to product description: ASCPD-144)

  Description:

  The EcuM does not observe the defensive behavior of the calling component and ignores the function argument `caller` of the function `EcuM_GoDown()`. Hence the configuration of allowed users of `EcuM_GoDown()` (Related requirements: ECUM196_Conf, ECUM206_Conf, ECUM207_Conf) is also not implemented.

  Requirements:

  EcuM4047, ECUM196_Conf

- No support for multiple Driver Init Lists.

  Description:

The EcuM module does not provide multiple configuration parameters for EcuMDriverInitListZero and EcuMDriverInitListOne in order for the MCAL modules to be initialized only by the master core.

Requirements:

EcuM4013

- No Det error for unspecified shutdown causes in `EcuM_SelectShutdownCause()`

  Description:

  `EcuM_SelectShutdownCause()` does not report DET error `ECUM_E_INVALID_PAR` for unspecified/not configured shutdown causes.

  Rationale:

  Shutdown causes other than default causes can be added by EcuM configuration but this implementation only supports default shutdown causes.

  Requirements:

  EcuM3009

- No wakeup status notification to ComM and BswM in `EcuM_SetWakeupEvent()` and `EcuM_ValidateWakeupEvent()` `EcuM_WakeupRestart()` `EcuM_GoHalt()`

  Description:

  The wakeup source status changes are not indicated to ComM and BswM from within the calls to the functions `EcuM_SetWakeupEvent()`, `EcuM_ValidateWakeupEvent()`, `EcuM_WakeupRestart()` `EcuM_GoHalt()` Instead the notification calls to BswM and ComM modules are delayed and performed from `EcuM_MainFunction()`.

  Rationale:

  The function `EcuM_SetWakeupEvent()` may be called from an init function like those of the CanTrcv. There the calls of `CanTrcv_Init()` -> `EcuM_SetWakeupEvent()` -> `ComM_EcuM_WakeUpIndication()` -> `ComM_EcuM_PNCWakeUpIndication()` and `BswM_EcuM_CurrentWakeup()` are performed before the ComM and BswM are initialized. The notifications are ignored by the ComM and BswM and a DET error may be raised. The solution is implemented as outlined in AUTOSAR Bugzilla issue http://www.autosar.org/bugzilla/show_bug.cgi?id=53338.

  Requirements:

  EcuM2645, SWS_EcuM_04112

- Shutdown target configuration

  Description:

In contrast to requirements ECUM178_Conf, ECUM179_Conf, shutdown targets cannot be configured other than predefined targets `ECUM_STATE_OFF, ECUM_STATE_RESET, ECUM_STATE_SLEEP`

Rationale:

EcuM shutdown targets other than `ECUM_STATE_OFF, ECUM_STATE_RESET, ECUM_STATE_SLEEP` are not used in EcuM. The APIs which uses shutdown targets are of `EcuM_StateType` and these APIs only allow this three shutdown targets. This is addressed in AUTOSAR Bugzilla issue http://www.autosar.org/bugzilla/show_bug.cgi?id=51379.

Requirements:

ECUM178_Conf, ECUM179_Conf

- Reset loop detection

  Description:

  No support for reset loop detection.

  Rationale:

  Apart from the configuration parameter ECUM171_Conf: `EcuMResetLoopDetection`, the specification of ECU State Manager R4.0 Rev 3 does not have requirements regarding action items for this feature. AUTOSAR Bugzilla Ref: http://www.autosar.org/bugzilla/show_bug.cgi?id=50899

  Requirements:

  ECUM171_Conf

- The parameter `EcuMResetReasonRef` is optional

  Description:

  In contrast to requirement ECUM128_Conf, the multiplicity of the parameter `EcuMResetReasonRef` has been changed from 1 to 0..1.

  Rationale:

  For communication channel wakeup, the wakeup source may not be mapped to an MCU reset reason. Ref: AUTOSAR Bugzilla http://www.autosar.org/bugzilla/show_bug.cgi?id=52629

  Requirements:

  ECUM128_Conf

- Error reporting to Dem is optional

  Description:

According to the requirement EcuM2985, detection of production errors shall not be switched off. In contrast to this requirement, production errors are switched off by the configuration parameter `EcuMIncludeDem`

Rationale:

In AUTOSAR SWS ECU StateManager of AUTOSAR 4.0 Rev 3, the configuration parameter `EcuMIncludeDem` is used to enable/disable the availability of Dem. Also the multiplicity of a similar parameter `EcuMDemEventParameterRefs` is mentioned as 0 to 1. These specification errors are handled in the AUTOSAR Bugzilla http://www.autosar.org/bugzilla/show_bug.cgi?id=50236 and http://www.autosar.org/bugzilla/show_bug.cgi?id=55543 respectively.

Requirements:

EcuM2985

- No debugging support (reference to product description: ASCPD-68)

  Description:

  The feature *Debugging support* is not supported.

  Requirements:

  EcuM4035, EcuM4036, EcuM4037

- No provision of `EcuM_Generated_Types.h`

  Description:

  According to the requirement EcuM2992, the ECU Manager module implementation shall provide a `EcuM_Generated_Types.h` file which contains generated type declarations that fulfill the forward declarations in `EcuM.h`. In contrast to this requirement, `EcuM_Generated_Types.h` is not provided.

  Rationale:

  All the types are statically defined in this implementation.

  Requirements:

  EcuM2992

- No hash value computation

  Description:

  In contrast to section 7.3.4 Checking Configuration Consistency, the hash value computation is not performed and also it is not stored in configuration parameter `EcuMConfigConsistencyHash`.

  Rationale:

The hash value computation feature has not been implemented in this release.

Requirements:

EcuM2794, EcuM2795, EcuM2796, EcuM2798, EcuM2799, ECUM102_Conf

- No Alias for `EcuMShutdownTarget`

  Description:

  In contrast to requirement EcuM2959, there is no alias defined for the parameter `EcuMShutdownTarget`.

  Rationale:

  According to requirement [EcuM624] only `ECUM_STATE_RESET`, `ECUM_STATE_SLEEP`, and `ECUM_STATE_OFF` can be chosen as valid shutdown targets. Also the parameter (`EcuMShutdownTargetId`) for defining the alias is removed in the AUTOSAR 4.1 specification (through RFC #51379). Thus it is not possible (also not valid) to define an alias for `EcuMShutdownTarget`. This is the topic of AUTOSAR Bugzilla http://www.autosar.org/bugzilla/show_bug.cgi?id=58918.

  Requirements:

  EcuM2959

- No Default wakeup source reporting

  Description:

  The requirements EcuM4040 and EcuM2601 implies that when a hardware cannot detect a specific wake-up source for a reset/restart reason, then the ECU Manager module shall report `ECUM_WKSOURCE_RESET` as a default wakeup source. In contrast to these requirements, `ECUM_WKSOURCE_RESET` is not reported as a default wakeup source during `STARTUP` or `WAKEUP`, if a reset/restart reason is not found (by the hardware). This is the topic of AUTOSAR Bugzilla http://www.autosar.org/bugzilla/show_bug.cgi?id=58920.

  Rationale:

  This feature has not been implemented in this release.

  Requirements:

  EcuM2601, EcuM4040_Implicit_2

- DET initialization is not dependent on modules configured in EcuM

  Description:

  Contrary to the requirement EcuM2783, DET initialization does not depend on any of the modules configured in the EcuM. DET will be initialized only if EcuM includes DET and an init service entry of DET in any initialization list exists.

Rationale:

This functionality is not supported.

- AUTOSAR 3.2 support: The client-server interface `EcuM_ApplicationMode` is not provided

  Description:

  Contrary to the AUTOSAR 3.2 specification the client-server interface `EcuM_ApplicationModerequirement` that contains ports to `EcuM_SelectApplicationMode` (EcuM2833) and `EcuM_GetApplicationMode` (EcuM2834) is not provided.

  Rationale:

  This functionality is not supported.

- EcuM_Types.h header file does not include Rte Types related header file

  Description:

  Contrary to the requirement EcuM3025 the file `EcuM_Types.h` does not include the file `Rte_EcuM_Type.h`.

  Rationale:

  When adding AUTOSAR 3.2 support to EcuM, the BSW's view of the default API has to be different to the Rte's/SWC's view of the default API. To resolve this issue distinct translation units for the BSW interface and the Rte/SWC interface have been introduced. Thus `EcuM_Types.h` does not include `Rte_EcuM_Type.h`.

  Requirements:

  EcuM3025

- Disable Interrupts and Enable Interrupts is not available for multicore.

  Description:

  Within the Poll Sequence, WakeupRestart Sequence, all interrupts should be disabled before the `Mcu_SetMode(Mcu_ModeType)` API is called, and enabled back again.

  Rationale:

  This functionality is not supported.

  Requirements:

  EcuM.MasterCore.Poll.Sequence, EcuM.SlaveCore.Poll.Sequence, SWS_EcuM_04026

- Master / Slave Core Shutdown Synchronization during shutdown

Description:

The SWS specifies that a shutdown flag should be set, which can be read by all EcuMs of all slave cores. Then BSW de-initialization on slave core takes places, and the shutdown flag should be unset.

Rationale:

If a flag would be used, this means that there should be a global variable access, from different core contexts, which will result in an Os error and which may lead to cache coherency issues.

Requirements:

SWS_EcuM_04019, SWS_EcuM_04021

- Expected interfaces from CanSM are not available

  Description:

  The SWS specifies that EcuM requires interfaces from the CanSM module in order to fulfil its core functionality.

  Rationale:

  Currently EcuM does not make use of the `CanSM_StartWakeupSource(Std_ReturnType, NetworkHandleType)` and `CanSM_StopWakeupSource(Std_ReturnType, NetworkHandleType)` APIs for wakeup handling.

  Requirements:

  SWS_EcuM_02858

- Additional mandatory expected interfaces

  Description:

  The SWS does not specify anymore some mandatory interfaces for the EcuM which are needed to fulfil its core functionality: Dem_ReportErrorStatus - Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation. DisableAllInterrupts - Depending on the selected value from the Rte configuration parameter InterruptBlockingFunction, when the EcuM exits an exclusive area using the SchM, it needs the DisableAllInterrupts interface. EnableAllInterrupts - Depending on the selected value from the Rte configuration parameter InterruptBlockingFunction, when the EcuM enters an exclusive area using the SchM, it needs the EnableAllInterrupts interface.

  Rationale:

Currently the EcuM is using the mentioned interfaces but they have been removed from the Autosar 4.-3 SWS document as being mandatory interfaces.

Requirements:

SWS_EcuM_02858

- No support for EcuMOSResource

  Description:

  The functionality related to this parameter is not supported by the current implementation. Considering that it is disabled, its multiplicity has been changed from 1..* to 0.

  Requirements:

  ECUM183_Conf

- Parameter `EcuMFlexModuleConfigurationRef` refers also to other BSW modules init structure

  Description:

  Contrary to the requirement ECUM182_Conf that is related to the parameter `EcuMFlexModuleConfigurationRef`, EcuM refers also to the init structure of other BSW modules.

  Rationale:

  Reference to the init structure of BswM, CanTrcv, Cdd, Eth, EthTrcv, FrMultipleConfiguration and Ocu is also needed in parameter `EcuMFlexModuleConfigurationRef`. This is addressed in AUTOSAR 4.-1.3 Software Specification of EcuM.

  Reference to the init structure of Dio is also needed in parameter `EcuMFlexModuleConfigurationRef` since Dio will be initialized by EcuM during the startup phase. This is addressed in AUTOSAR Bugzilla issue http://www.autosar.org/bugzilla/show_bug.cgi?id=51175.

  Reference to the init structure of Fr is also needed in parameter `EcuMFlexModuleConfigurationRef` since Fr could also be initialized by EcuM during the startup phase. This is addressed in AUTOSAR Bugzilla issue http://www.autosar.org/bugzilla/show_bug.cgi?id=53326.

  Reference to the init structure of the following MCAL modules: GtmCfgConfigSet , MclConfigSet , SpmcConfigSet, which are not specified in the EcuM Software Specification document is also needed in the `EcuMFlexModuleConfigurationRef` parameter.

  Requirements:

  ECUM182_Conf

- Multi-Core GoSleep Sequence is similar to the Single-Core GoSleep Sequence

Description:

Activities in the GoSleep Sequence have been updated in Autosar 4.3.0 SWS of EcuM after bugzilla RfC #63167 has been implemented. However these changes were only applied to the Single-Core diagram (Activities in the GoSleep Sequence - Figure 12) while the Multi-Core diagrams of the GoSleep Sequence have not been updated. They should of been updated aswell since this RfC solves a bug which applies to both Single-Core and Multi-Core use-cases. This implementation provides the solution from RfC #63167 not only for the Single-Core scenario but also for the Multi-Core scenario.

Requirements:

SWS_EcuM_04023, SWS_EcuM_04027

- Multi-Core Poll Sequence is similar to the Single-Core Poll Sequence

  Description:

  Activities in the Poll Sequence have been updated in Autosar 4.3.0 SWS of EcuM after bugzilla RfC #63167 has been implemented. However these changes were only applied to the Single-Core diagram (Activities in the Poll Sequence - Figure 14) while the Multi-Core diagrams of the Poll Sequence have not been updated. They should of been updated aswell since this RfC solves a bug which applies to both Single-Core and Multi-Core use-cases. This bugzilla RfC fixes the blocking loop of the Poll Sequence which should react also to validated wakeup events, not only to pending wakeup events. This implementation provides the solution from RfC #63167 not only for the Single-Core scenario but also for the Multi-Core scenario.

  Requirements:

  SWS_EcuM_04025, SWS_EcuM_04029, EcuM3020

- EcuM_MemMap.h is included instead of MemMap.h

  Description:

  Due to the architecture of EcuM_MemMap.h, the preprocessing time is significantly decreased compared to MemMap.h. See ASCPROJECT-2931 for a detailed performance comparison. Also see SWS_-MemMap_00032 which specifies using [Mip]_MemMap.h.

  Requirements:

  EcuM2862

- No support for EcuMWakeupSourcePolling

  Description:

  The functionality related to this parameter is not supported by the current implementation.

  Requirements:

ECUM153_Conf

3.3.4.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- **Shutdown synchronization**

  Description:

  The slave-core EcuM instance should de-initialize its BswM and SchM before sending a signal to indicate that core is ready for shutdown. However only the BswM is de-initialized before the signal is sent.

  Rationale:

  Synchronization data must be sent before the SchM is de-initialized because the EcuM uses the SchM Sender-Receiver interface for core synchronization. If SchM is de-initialized first, then this interface cannot be used anymore.

- **API Reentrancy status in a Multi-Core context**

  Description:

  The following EcuM APIs are specified by the EcuM SWS as Non-Reentrant: - EcuM_SetWakeupEvent() - EcuM_GetPendingWakeupEvents() - EcuM_GetValidatedWakeupEvents() - EcuM_GetExpiredWakeupEvents() - EcuM_ClearWakeupEvent() - EcuM_StartupTwo() - EcuM_MainFunction().

  However, in a Multi-Core context, these APIs are reentrant, since they can be called from different core contexts.

3.3.4.6. Open-source software

EcuM does not use open-source software.

3.3.5. Nm module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 3.0.0
- Module version: 5.12.5.B337087
- Supplier: Elektrobit Automotive GmbH

3.3.5.1. Change log

This chapter lists the changes between different versions.

Module version 5.12.5
2020-06-19

- Improved calculate value functionality in tresos for parameter NmChannelId
- ASCNM-617 Fixed known issue: Nm_RemoteSleepIndication and Nm_RemoteSleepCancellation cannot be used when multicore is enabled, Nm coordinator is not used and NmProvideRemoteSleepCallbacks is set
- ASCNM-623 Fixed known issue: Linker error is reported for Nm if multi-core is enabled while development error detection and coordinator support are disabled

Module version 5.12.4
2020-02-21

- Changed default value for parameter NmComControlEnabled to false

Module version 5.12.3
2019-10-11

- ASCNM-597 Fixed known issue: Linker errors are reported due to incorrect memory mapping

Module version 5.12.2
2019-06-14

- Internal module improvements. This module version update does not affect module functionality.

Module version 5.12.1
2019-02-15

- Internal module improvements. This module version update does not affect module functionality.

Module version 5.12.0
2018-10-26

- Implemented Multi-core support
- ASCNM-566 Fixed known issue: In certain configurations, Nm does not support communication control service for UdpNm

Module version 5.11.4
2018-06-22

- Added extra callback support for Nm_RemoteSleepIndication and Nm_RemoteSleepCancellation functions.

Module version 5.11.3
2018-02-16

- ASCNM-502 Fixed known issue: The API Nm_SetUserData() may cause a null pointer exception
- Updated description for parameters NmCarWakeUpRxEnabled and NmCarWakeUpCallback
- Removed AUTOSAR 3.x compliant symbolic name value macros and updated the logic to only provide AUTOSAR 4.0.2 compliant macros

Module version 5.11.2
2017-09-22

- Added NmNodeDetectionEnabled, NmNodeIdEnabled and NmRepeatMsgIndEnabled as per channel configurable.

Module version 5.11.1
2017-06-30

- Implemented support for car wake up
- ASCNM-473 Fixed known issue: Coordinated shutdown is not aborted while waiting for valid synchronization
- ASCNM-478 Fixed known issue: Nm gets locked in synchronization state. Fix is available only by enabling and configuring `NmWaitForShutdownTime`.

Module version 5.11.0
2017-03-31

- Updated the path of the REFINED_MODULE_DEF parameter

Module version 5.10.0
2016-05-24

- Internal module improvements. This module version update does not affect module functionality: ASCNM-431, ASCNM-432

Module version 5.9.0
2016-02-10

- Added support for Debug & Trace with custom header file configurable via parameter `BaseDbgHeaderFile`

Module version 5.8.0
2015-11-06

- Internal module improvement. This module version update does not affect module functionality: ASCNM-409, ASCNM-411, ASCNM-412, ASCNM-416, ASCNM-417, ASCNM-417, ASCNM-418, ASCNM-419

Module version 5.7.0
2015-02-20

- Add global NmPassiveModeEnabled parameter

Module version 5.6.0
2014-10-02

- Internal module improvement. This module version update does not affect module functionality: ASCNM-402

Module version 5.5.0
2014-04-25

- Internal module improvement. This module version update does not affect module functionality: ASCNM-397, ASCNM-395, ASCNM-393, ASCNM-389, ASCNM-388

Module version 5.4.1
2013-10-18

- ASCNM-355 Fixed known issue: `Nm_MainFunction` is not available if *Coordinator Support* is disabled
- Updated the memory mapping macro names in template file `Nm_ExtraCallbacks.c`

- Added support for function tracing via AUTOSAR Debugging

Module version 5.4.0
2013-06-26

- ASCNM-323 Fixed known issue: Compiler error when `NmComUserDataSupport` is set to true
- ASCNM-324 Fixed known issue: Memory access violation when channels are not configured to be zero-based and consecutive
- ASCNM-321 Fixed known issue: *Nm Coordinator* does not shutdown the channel if parameter `NmShutdownDelayTimer` is set to zero
- ASCNM-291 Fixed known issue: Compiler errors may occur if parameter `NmStandardBusType` has the same value in all channels
- ASCNM-320 Fixed known issue: `Nm_StateChangeNotification` cannot be modified as it is not provided as a template function
- ASCNM-354 Fixed known issue: Compiler errors may occur if parameter `NmStateReportEnabled` is set to false for all channels and parameter `NmStateChangeIndEnabled` is set to true

Module version 5.3.0
2013-02-15

- ASCNM-300 Fixed known issue: A compiler error occurs when symbolic names according to AUTOSAR R4.0.3 are used
- Changed the reference path of `ComMChannel` in parameter `NmComMChannelRef` to `/AUTOSAR/EcucDefs/ComM/ComMConfigSet/ComMChannel`
- Implemented memory allocation keywords in compliance to AUTOSAR 4.0.3
- ASCNM-276 Fixed known issue: The Nm may access an incorrect or non-existent channel when channels are not configured to be zero-based and consecutive

Module version 5.2.0
2012-10-12

- Added support for *State Change Notification*
- ASCNM-261 Fixed known issue: A compiler error occurs when `Nm_Init()` is called without any arguments and `NmCoordinatorSupportEnabled` and `NmDevErrorDetect` are both disabled
- Changed the top-level structure of the software-component description in the arxml files from `/AUTOSAR/Nm` to `/AUTOSAR_Nm`
- Removed the type `Nm_ReturnType` and the associated return values from `NmStack_Types.h`

Module version 5.1.1
2012-06-27

- ASCNM-235 Fixed known issue: NM has to support `_NON_` zero-based and `_NON_` consecutive Nm channel IDs
- Added templates for the extra callback notifications for OEM-specific extensions
- Added enumeration type `Nm_BusNmType` in header file `NmStack_types.h`

Module version 5.1.0
2012-03-16

- ASCNM-199 Fixed known issue: Nm does not support FrNm when communication control service `NmComControlEnabled` is activated
- ASCNM-217 Fixed known issue: Signature of `Nm_GetUserData()` is wrong
- ASCNM-228 Fixed known issue: Nm throws compilation error when bus type is not selected as CAN or FlexRay
- Updated to AUTOSAR 4.0.3 version

Module version 5.0.0
2011-08-22

- Initial AUTOSAR 4.0 version

3.3.5.2. New features

- No new features have been added since the last release.

3.3.5.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- Additional callout in callback function `Nm_StateChangeNotification()`

  Description:

  The callback function `Nm_StateChangeNotification()` forwards the state changes to the additional callout function `Nm_StateChangeNotificationCallout()`. Parameter `NmStateChangeNotificationCallout` controls this behavior.

This additional callout function can be implemented in order to handle the notified state changes from the BusNm modules.

- Function tracing support via AUTOSAR Debugging

  Description:

  The module Nm supports tracing of function entry and exit via the EB Dbg module.

  Function tracing records following parameters for each function:

  - function name
  - values of the function arguments
  - point in time of function invocation
  - point in time of function termination
  - return value of the function

- `ComUserDataSupport` configuration in modules Nm, CanNm, FrNm, UdpNm

  Description:

  The value of `NmComUserDataSupport`, should be equal to `FrNmComUserDataSupport`, `CanNmComUserDataSupport`, `UdpNmComUserDataSupport` in case the modules are added into configuration (FrNm, CanNm, UdpNm). A XPath check is added for the node `NmComUserDataSupport` in `Nm.xdm.m4`.

3.3.5.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- No support for LINNM (reference to product description: ASCPD-105)

  Description:

  The *NM Coordinator* functionality does not support LINNM as a standard bus type.

  Requirements:

  Nm220_Conf

- Coordination of nested subbusses (reference to product description: ASCPD-106)

  The *NmCoordinatorSyncSupport* functionality is not supported in current implementation.

  Description:

  The requirement Nm256 demands that a *NM Ccoordinator* shall support two or more *NM Coordinators* connected to the same NM cluster.

The requirement Nm257 describes that one or more passive *NM Coordinators* send Nm messages only if the node has a network management request pending or a connected network. In this case the connected network is not ready to sleep and is coordinated actively by that coordinator.

The requirements under section 7.2.8 *Coordination of nested subbusses* in the AUTOSAR specification (Rev 4.0.3) describe the detailed functionality required to coordinate the nested subbusses that use an active or passive *Nm Coordinator*.

In contrast to these requirements, coordination of nested subbusses that use active or passive *NM Coordinators* is not supported.

Requirements:

Nm256, Nm257, Nm258, Nm259, Nm260, Nm261, Nm262, Nm263, Nm264, Nm265, Nm266, Nm267, Nm237_Conf and Nm236_Conf

- Tracing of variables is not supported via AUTOSAR Debugging

  Description:

  The Nm module does not provide support for tracing global variables.

  Requirements:

  Nm189, Nm190, Nm191, Nm240

- Scope of evaluation of coordination algorithm

  Description:

  The requirement Nm171 describes that the evaluation of coordination algorithm shall be carried out in all API calls of the Nm including the main function of Nm. In contrast to this requirement, the evaluation and actions of coordination algorithm are not performed in all API calls of the Nm and only performed in `Nm_MainFunction()` and in some API calls (e.g. `Nm_NetworkRelease()` and `Nm_RemoteSleepIndication`).

  Rationale:

  The requirement Nm171 is implementation-specific. See the Bugzilla entry http://www.autosar.org/bugzilla/show_bug.cgi?id=58474. where the description of a similar requirement Nm170 is corrected as follows: [SWS_NM_00170]: There are no limitation where the actions performed by the coordinator algorithm shall take place. This can be done either by the Nm main function (`Nm_MainFunction()`) or module indication/callbacks. So it is clear that it is up to the implementer to perform the actions of coordination algorithm on some APIs. It may not necessary to perform the actions of coordination algorithm in all the APIs of Nm. Additionally note that Nm170 and Nm171 are close together.

  Requirements:

Nm171

- Additional provision of callback API `Nm_StateChangeNotificationCallout()`

  Description:

  The requirement Nm095 demands that the Nm module shall not provide interface functions beyond those specified in the AUTOSAR specification. In contrast to this requirement, an additional (user-specific) callout function `Nm_StateChangeNotificationCallout()` is provided to notify state changes within the <BusNm> modules.

  Rationale:

  It is at least one customer requirement which is not specified in the AUTOSAR specification.

  Requirements:

  Nm095

- No call of `<BusNm>_NetworkRequest()` for the network which throws error on `<BusNm>_NetworkRelease()`

  Description:

  The requirement Nm236 states that if Nm receives `E_NOT_OK` on invocation of `<BusNm>_NetworkRelease()`, Nm shall abort the shutdown and call `<BusNm>_NetworkRequest()` for all the networks in the coordinated clusters. This includes the own network which throws the error on `<BusNm>_NetworkRelease()`. In contrast to this requirement, `<BusNm>_NetworkRequest()` is not called for the own network which throws `E_NOT_OK` on invocation of `<BusNm>_NetworkRelease()`.

  Additionally `E_NOT_OK` is the correct return value as Nm236 (in AUTOSAR 4.0 Rev 3) talks about `NM_E_NOT_OK`. See http://www.autosar.org/bugzilla/show_bug.cgi?id=56744

  Rationale:

  The use case where `<BusNm>_NetworkRelease()` returns `E_NOT_OK` is invalid. See the RFC http://www.autosar.org/bugzilla/show_bug.cgi?id=60589.

  Requirements:

  Nm236

3.3.5.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Implementation-specific parameter range limitations

  Description:

  Parameter `NmChannelId`: Range limited from 0..255 to 0...254.

  Rationale:

  This limitation allows for a more efficient implementation.

- No support for link time configuration

  Description:

  The Nm Interface module can only be configured at pre-compile time. Link time configuration is not supported.

  Rationale:

  Source code can be optimized with respect to code size and execution speed more aggressively if only pre-compile time configuration is supported.

  Requirements:

  Nm195, Nm247

3.3.5.6. Open-source software

Nm does not use open-source software.

4. ACG8 Mode Management user guide

4.1. Overview

This user guide describes the ACG8 Mode Management in EB tresos AutoCore. ACG8 Mode Management comprises the following modules:

- EcuM
- BswM
- ComM
- Nm

In this user guide you learn about the mode management modules EcuM and BswM. You also find information regarding specific features of ComM and Nm. General information about ComM, Nm and related <bus>Nm modules is provided as part of the network and state management concepts chapter in the EB tresos AutoCore Generic 8 documentation. In addition, the BswM Editor is described here. This is a tool provided by Elektrobit Automotive GmbH to assist you when configuring the BswM module.

The mode management modules EcuM and BswM provide options that support the **BSW distribution** feature for use in a multi-core project environment. For more information about the general concepts related to multi-core and BSW distribution in AUTOSAR, and the support provided in EB tresos AutoCore Generic, see section *Multi-core support* in chapter *Concepts* in the EB tresos AutoCore Generic 8 documentation.

- Section 4.2, “Background information” provides some basic information about the EcuM and BswM modules. It also provides information on specific features of ComM and Nm.
- Section 4.3, “Configuring the EcuM (ECU State Manager) module and the BswM (BSW Mode Manager) module” provides some integration notes and help for configuring the EcuM and BswM modules.
- Section 4.4, “BswM Editor user's guide” describes how to use the editor for configuring the BswM module.
- Section 4.5, “Configuring the ComM module” describes how to configure the ComM for specific use cases.
- Section 4.6, “Configuring the Nm module” describes how to configure the Nm for specific use cases.

4.2. Background information

4.2.1. ECU State Manager module (EcuM)

4.2.1.1. EcuM Flex support

EB tresos AutoCore provides the flexible ECU management variant of the `EcuM` module. This variant of the `EcuM` module handles the early `startup` phases, the late `shutdown` phases and the `sleep` phases in which the whole OS scheduling is locked by the `EcuM` taking the OS resource `RES_SCHEDULER`. All other phases and states are completely configurable and handled by the `BswM` module as depicted in Figure 4.1, “The basic software stack states”.

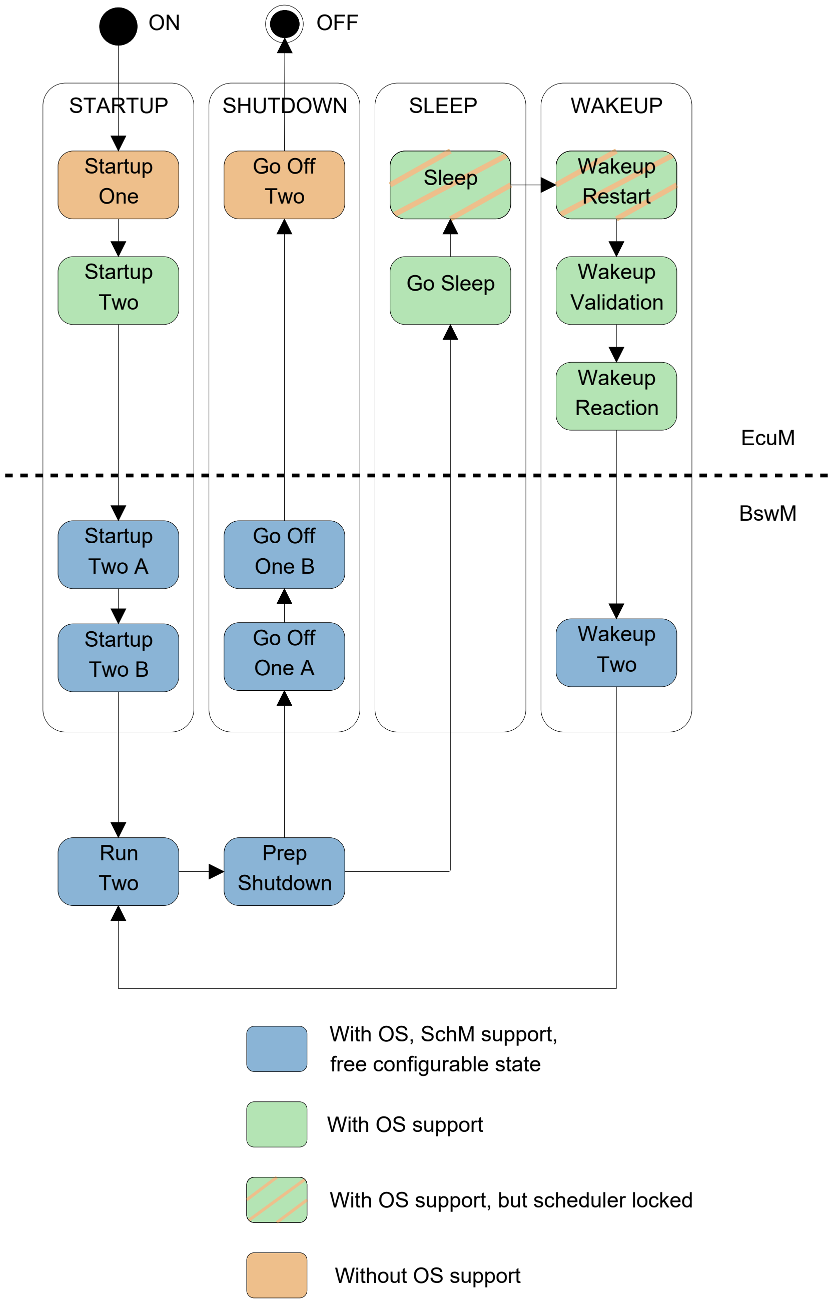


Figure 4.1. The basic software stack states

As the states of the `BswM` module are completely configurable, the states highlighted in blue in Figure 4.1, “The basic software stack states” show an example of a configuration. However, you may change and adapt this configuration to your system's needs.

4.2.1.2. EcuM Fixed support

Some basic support of the ECU State Manager with a fixed state machine has been implemented in order to allow the integration of modules that are based on AUTOSAR 3.2 in projects that are based AUTOSAR 4.0 or later. Most notably the handling of RUN and POST_RUN requests has been implemented. The **EcuM Fixed support** feature is enabled by configuring the EcuMFixedConfiguration container.

4.2.1.2.1. Functional overview

The main difference between the EcuM flexible and the EcuM fixed functionality is an extension of the EcuM state machine. The state machine is now controlled by five additional API functions that implement the request/release/kill behavior of RUN and POST_RUN states. A sixth API function can be called to get the status of wakeup sources. In addition, a new callback function for synchronization with the BswM module is provided.

After startup, the EcuM does not shut down until at least EcuMRunMinimumDuration/EcuMPeriodMainFunction calls to EcuM_MainFunction() have been performed. During this time, other modules have the possibility to issue RUN or POST_RUN requests. This then keeps the EcuM alive until all such requests are eventually released again.

Because some of the modules subject to the EcuM fixed state machine may have to be initialized by the BswM, some mechanism is needed to inform the EcuM about when these modules are ready. This is necessary, because the initialization may be performed asynchronously within BswM. The vendor specific callback function Std_ReturnType EcuM_GoRun(void) is provided by the EcuM and has to be called by BswM when all modules are initialized. Until this callback is called, EcuM_MainFunction() will exit without further action if **EcuM Fixed support** is enabled.

All other EcuM actions, especially sleep / wakeup / shutdown handling, are left to the EcuM flexible implementation, with the only exception of additional calls to Rte in order to signal the mode switches. The mode switch RTE_MODE_EcuM_Mode_WAKE_SLEEP has been added, at the end of the wakeup procedure ("Wakeup Sequence II" in the SWS). Restart of the system can then be handled by Rte/BswM.

The intended usage is as follows:

- System starts.
- EcuM starts.
- EcuM calls BswM_Init().
- BswM does its initialization tasks, possibly asynchronously.
- EcuM_MainFunction() processes EcuM Flex tasks only.
- After some time, BswM initialization has finished -> BswM calls EcuM_GoRun().
- EcuM_MainFunction() processes EcuM Fixed specific tasks, too.

- After some time: system goes to sleep.
- System wakes up again.
- `EcuM_MainFunction()` processes EcuM Flex tasks only, especially wakeup validation.
- After wakeup validation has been performed, `EcuM` signals `WAKE_SLEEP` mode to `Rte`.
- `BswM` does re-initialization tasks, possibly asynchronously.
- After some time, `BswM` re-initialization has finished -> `BswM` calls `EcuM_GoRun()` (e.g. in WAKEUP-II state).
- Again, `EcuM_MainFunction()` processes EcuM Fixed specific tasks, too.

4.2.1.2.2. Configuration options

The following additional configuration options are available:

- `EcuMFixedConfiguration`: Enables **EcuM Fixed support**
- `EcuMFixedConfiguration/EcuMRunMinimumDuration`: Timeout for `RUN` requests
- `EcuMFixedConfiguration/EcuMFixedUserConfig`: List of users which can request a `RUN` or `POST_RUN` state
- `EcuMDemEventParameterRefs/ECUM_E_ALL_RUN_REQUESTS_KILLED`: EventId reported to DEM when `EcuM_KillAllRUNRequests()` is called.

4.2.1.3. Multi-core and BSW distribution support

In the `EcuM`, multi-core support is implemented based on the BSW distribution "Master-Satellite" concept defined by AUTOSAR. In this "Master-Satellite" implementation, the functionality of the `EcuM` module is distributed across multiple cores.

4.2.1.3.1. Functional overview

The basic states for a multi-core system are depicted in Figure 4.2, “The basic software stack states for a system with multi-core”.

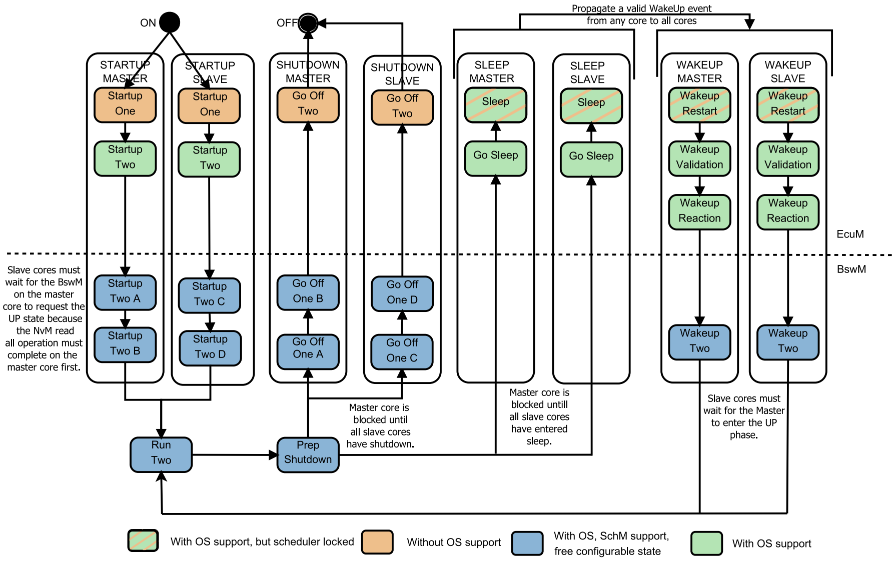

ON
OFF
Propagate a valid WakeUp event
from any core to all cores
STARTUP MASTER
Startup One
Startup Two
STARTUP SLAVE
Startup One
Startup Two
SHUTDOWN MASTER
Go Off Two
SHUTDOWN SLAVE
Go Off Two
SLEEP MASTER
Sleep
Go Sleep
SLEEP SLAVE
Sleep
Go Sleep
WAKEUP MASTER
Wakeup Restart
Wakeup Validation
Wakeup Reaction
WAKEUP SLAVE
Wakeup Restart
Wakeup Validation
Wakeup Reaction
EcuM
BswM
Slave cores must wait for the BswM on the master core to request the UP state because the NvM read all operation must complete on the master core first.
Startup Two A
Startup Two B
Startup Two C
Startup Two D
Go Off One B
Go Off One A
Go Off One D
Go Off One C
Wakeup Two
Wakeup Two
Master core is blocked until all slave cores have shutdown.
Master core is blocked untill all slave cores have entered sleep.
Slave cores must wait for the Master to enter the UP phase.
Run Two
Prep Shutdown
With OS support, but scheduler locked
Without OS support
With OS, SchM support, free configurable state
With OS support

Figure 4.2. The basic software stack states for a system with multi-core

As in the single-core case, the states of the BswM module are completely configurable. The states highlighted in blue in Figure 4.2, “The basic software stack states for a system with multi-core” show an example of a configuration. However, you may change and adapt this configuration to your system's needs.

On a multi-core system, state transitions may encounter delays due to synchronization between the master and slave(s). This synchronization is required when transitioning between the phases mentioned below:

- STARTUP - > UP
- UP -> SLEEP
- UP -> SHUTDOWN
- SLEEP(WAKEUP) -> UP

Synchronization between master and slave cores is done via the SchM sender-receiver API for messaging between the cores. This is a deviation from the AUTOSAR standard as the standard specifies a synchronization method that will result in Os errors (RfC#61904).

To signal to the slave cores that the master core is ready, the master core shall send the value TRUE to the EcuM_MasterCoreReadyPort. To signal to the master core that a slave core is ready, a slave core shall send its coreId to the EcuM_SlaveCoreReadyPort. The EcuM instances need to be configured in the Rte configuration, otherwise the code that is generated for the SchM sender-receiver functions will be empty.

4.2.1.3.2. Configuration options

To use the **BSW Distribution** feature in the EcuM, configure the parameters EcuMEnableMulticore and if needed, EcuMMasterCoreId. If you do not enable EcuMMasterCoreId an implicit value for the master

EcuM core, core-ID 0, is used. Refer to Section 4.3, “Configuring the EcuM (ECU State Manager) module and the BswM (BSW Mode Manager) module” for further integration and configuration instructions.

4.2.1.4. Post-build selectable support

The EcuM is implemented with variant handling functionality, allowing multiple module configuration variants to be loaded at the same time.

4.2.1.4.1. Functional overview

If the EcuM is configured with multiple variants, one EcuM_variant-name_PBCfg.c file is generated for each variant. This file contains the post-build configuration data structure. A specific variant can be selected by using the EcuM_DeterminePbConfiguration() callout function.

This callout function is invoked by EcuM_Init(), which is responsible for loading the post-build data structure.

It is not possible to retrieve the post-build data structure from the PbCfgM module, since EcuM initializes the PbCfgM.

4.2.1.4.2. Configuration options

The following AUTOSAR configuration parameters support post-build selectable variants:

- EcuMDefaultAppMode
- EcuMDefaultState
- EcuMDefaultResetModeRef
- EcuMDefaultSleepModeRef

For the **EcuM Fixed** extension, the EcuMRunMinimumDuration parameter supports post-build selectable variants.

In addition to the AUTOSAR configuration parameters above, the driver initialization lists EcuMDriverInitListOne and EcuMDriverRestartList support post-build selectable configurations.

For information on how to migrate the EcuM module from ACG-8.5.2 to ACG-8.6.0, see Section 4.3.2.1, “Migrating the EcuM from ACG-8.5.2 to ACG-8.6.0”.

4.2.2. BSW Mode Manager module (BswM)

4.2.2.1. Mode arbitration

The BswM module arbitrates mode requests from software components or from basic software modules, applies configurable rules to these requests, and performs configurable actions. In contrast to the EcuM, the BswM does not provide any state dependent handling. The arbitration, rules and actions that are configured in the BswM are executed independently of any state. If state dependent handling is needed, define a state variable externally and then check this state variable within the BswM rules.

Figure 4.3, “Code example of a BswM rule” gives an example of one rule configured in the BswM with the parameter BswMRule. In such a rule, a logical expression is configured via the parameter BswMLogicalExpression. This logical expression consists of one or more conditions. Each condition is configured by the parameter BswMModeCondition. Request ports are configured to use the rule that has been defined. When a request is received via the request port, the logical expression is evaluated and results either in TRUE or in FALSE. Depending on this result, an action list that is configured with the parameter BswMActionList is executed. Each action list itself includes one or more actions. An action could be for example a user callout. This is configured using the parameter BswMAction.

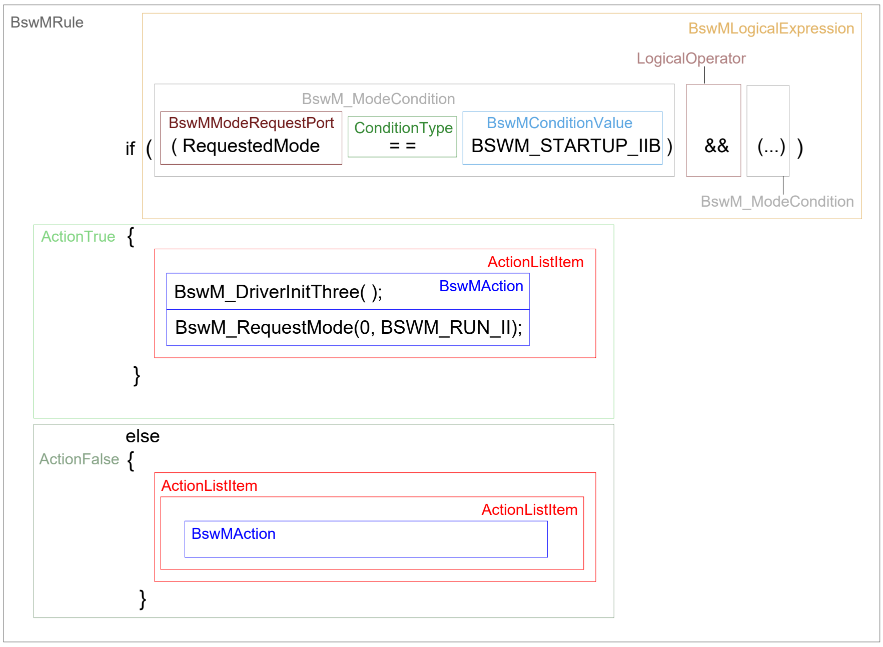


Figure 4.3. Code example of a BswM rule

Figure 4.4, “Sample containers, references and multiplicity of a BswM rule” shows the configuration containers, the references and their multiplicity in a BswM rule.

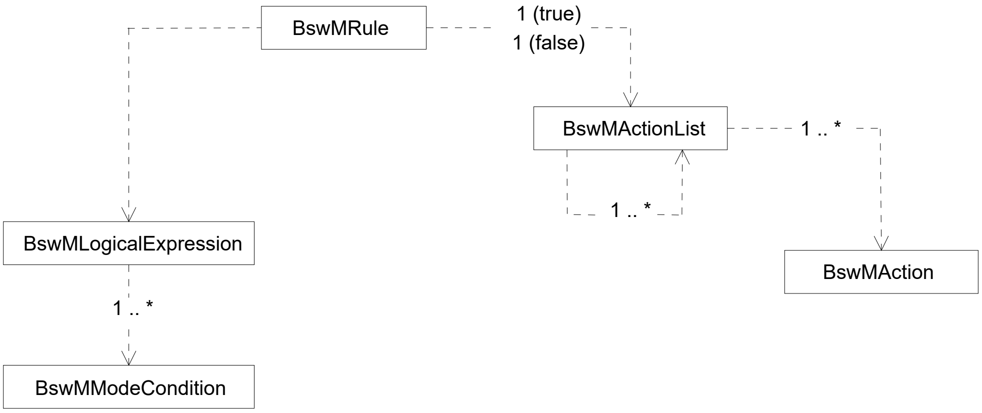


Figure 4.4. Sample containers, references and multiplicity of a `BswM` rule

4.2.2.2. Queued mode arbitration

4.2.2.2.1. Functional overview

Concurrent processing of mode arbitration requests is a challenging topic for an integrator because reentrancy issues or race conditions are common when the arbitration of a mode request port depends on the execution of an action list.

According to the AUTOSAR Specification of Basic Software Mode Manager, the system integrator is responsible for ensuring that the execution of an action list in the context of an immediate request does not jeopardize the system's consistency.

In order to minimize the risk that such issues affect a complex project, you can use the queued mode arbitration feature to better control the processing of arbitration requests. Instead of allowing concurrent execution, this feature ensures that the processing of an immediate request of any type or the processing of the `BswM_MainFunction()` cannot be interrupted by other requests.

This is achieved by queuing the requests that interrupt the processing of the existing immediate request or the processing of the `BswM_MainFunction()`. The queued requests are executed in the order of their arrival after the processing that they interrupted was completed.

Note that the queued arbitration requests are not executed from their original context anymore, but in the context of the request that they interrupted.

4.2.2.2.2. Configuration options

The queued mode arbitration feature can be enabled via the optional container **BswMQueuedArbitration**. The following configuration parameters can be used to fine-tune this feature.

The BswMArbitrationQueueLength parameter controls the size of the arbitration queue. By default, the value of this parameter is the maximum number of mode request ports in all BswM configurations. You can change this value to fit the project's requirements. If the arbitration queue is full and a new request needs to be enqueued, the new request is rejected. In this case, a Det error is reported if BswMDevErrorDetect is set to TRUE. Note that a larger queue increases the RAM usage of BswM.

The BswMQueueDeferredRequests parameter can be used to exclude or include deferred mode request ports from being queued. If set to FALSE, only immediate mode request ports are queued. If set to TRUE, both immediate and deferred mode request ports are queued.

The BswMDelayMainFunction parameter can be used to handle the situation when the BswM_MainFunction() interrupts the execution of an immediate mode request. If set to FALSE, the BswM_MainFunction() executes even if there is an immediate request being processed. Thus, the arbitration feature is bypassed. If set to TRUE, the BswM_MainFunction() is delayed until the processing of the immediate mode request is completed. Note that any other request that interrupts the processing of the immediate request that the BswM_MainFunction() interrupted, is also processed before the BswM_MainFunction().

4.2.2.2.3. Configuration examples

4.2.2.2.3.1. Example 1

An immediate generic request I_GEN_1 is made. While I_GEN_1 is processed, an immediate generic request I_GEN_2 and a deferred BswMCanSMIndication request D_CANSM are made from an interrupt context. Depending on the queued mode arbitration configuration parameters, the following outcomes are possible:

- BswMQueuedArbitration is disabled

  Both I_GEN_2 and D_CANSM interrupt the execution of I_GEN_1.

- BswMQueuedArbitration is enabled, BswMArbitrationQueueLength = 2, BswMQueueDeferredRequests = TRUE

  I_GEN_2 and D_CANSM are enqueued, and their mode request handler functions return. After I_GEN_1 finishes processing both arbitration and execution of action lists, I_GEN_2 and D_CANSM are also processed, in this order. Note that both I_GEN_2 and D_CANSM are processed from the context of the BswM_RequestMode corresponding to I_GEN_1.

- BswMQueuedArbitration is enabled, BswMArbitrationQueueLength = 1, BswMQueueDeferredRequests = TRUE

I_GEN_2 is enqueued but D_CANSM is not because the queue is not large enough. Thus, a Det error is reported. The service ID corresponds to `BswM_CanSM_CurrentState`, the error ID is `BSWM_E_QUEUE_FULL`.

After I_GEN_1 finishes processing both the arbitration and execution of action lists, I_GEN_2 is processed from the context of the `BswM_RequestMode` corresponding to I_GEN_1.

- `BswMQueuedArbitration` is enabled, `BswMArbitrationQueueLength` = 1, `BswMQueueDeferredRequests` = FALSE

  I_GEN_2 is enqueued but D_CANSM is processed as soon as it arrives, from its original `BswM_CanSM_CurrentState` context. After I_GEN_1 finishes processing, I_GEN_2 is unqueued and processed from the context of the `BswM_RequestMode` corresponding to I_GEN_1.

4.2.2.2.3.2. Example 2

An immediate generic request I_GEN_1 is made. While I_GEN_1 is processed, a generic deferred request D_GEN is made. The `SchM` triggers the `BswM_MainFunction()`. Afterwards, another generic immediate request I_GEN_2 is performed. Depending on the mode arbitration configuration parameters, the following outcomes are possible:

- `BswMQueuedArbitration` is disabled

  D_GEN, `BswM_MainFunction()` and I_GEN_2 interrupt I_GEN_1.

- `BswMQueuedArbitration` is enabled, `BswMQueueDeferredRequests` = TRUE, `BswMDelayMainFunction` = FALSE

  D_GEN is queued. `BswM_MainFunction()` interrupts and executes while I_GEN_1 is processed. I_GEN_2 is queued. After I_GEN_1 finishes, D_GEN and I_GEN_2 are processed in the order as they arrived. `BswM_MainFunction()` does not arbitrate D_GEN in this cycle because D_GEN was queued before `BswM_MainFunction()` executed and the port was not updated.

- `BswMQueuedArbitration` is enabled, `BswMQueueDeferredRequests` = FALSE, `BswMDelayMainFunction` = FALSE

  D_GEN is executed. The port is only updated because it is deferred. `BswM_MainFunction()` interrupts and executes while I_GEN_1 is processed and I_GEN_2 is queued. `BswM_MainFunction()` also arbitrates D_GEN as D_GEN updated the port. After I_GEN_1 finishes, I_GEN_2 is processed.

- `BswMQueuedArbitration` is enabled, `BswMQueueDeferredRequests` = TRUE, `BswMDelayMainFunction` = TRUE

  First, D_GEN is queued. Then, the `BswM_MainFunction()` is delayed because I_GEN_1 has not yet finished processing. I_GEN_2 is also queued. After I_GEN_1 finishes processing, first D_GEN is executed (port update). Then I_GEN_2 is processed. `BswM_MainFunction()` is last to be triggered, even if it occurred before I_GEN_2. `BswM_MainFunction()` arbitrates D_GEN as D_GEN updated the port before `BswM_MainFunction()` executed.

Note that the `BswM_MainFunction()` is first delayed (i.e. returns without performing any arbitration or control), and is later triggered by the `BswM_RequestMode` corresponding to I_GEN_1.

4.2.2.2.3.3. Example 3

If the `BswM_MainFunction()` is interrupted by an immediate generic request I_GEN_1, then I_GEN_1 is queued and only processed after `BswM_MainFunction()` finishes. Deferred requests are queued only if `BswMQueueDeferredRequests` = TRUE. This is the same behavior as in the above examples.

4.2.2.3. Multi-core and BSW distribution support

In the `BswM`, multi-core support is implemented based on the "multi partition" support concept defined by AUTOSAR. In this "multi partition" support implementation, one instance of the `BswM` is assigned to each partition.

4.2.2.3.1. Functional overview

The `BswM` mainly interacts with the state managers of the functional clusters, e.g. with the `ComM`, and should therefore be locally available on the same cores. Therefore, the `BswM` is distributed over multiple partitions that contain BSW modules. These independent `BswM` instances have partition specific configuration sets. The synchronization of the different instances is done via BSW modes.

All `BswM` instances are independent of each other and have their own configuration. This means that if a mode request is triggered in a context of a certain `BswM` instance, only that instance will perform the mode control and mode arbitration. Sometimes, however, it might be necessary to propagate a mode request to all `BswM` instances. This situation typically arises with single instance modules that can only trigger a mode change to one `BswM` instance only(on the partition they are located). This can, however, be achieved by using a `SchM` switch action on the partition that receives the mode change.

As an example, suppose there are 2 partitions, each on its own core. Both of these partitions have a `BswM` instance but only the first one has a `ComM` instance.

The `ComM` triggers a `BswM_ComM_CurrentMode(Network, RequestedMode)` mode request which needs to produce changes on both `BswM` partitions. Suppose there is a rule that triggers changes to the first `BswM` instance using an action list `ComM_ActionList`. In order to also notify the second `BswM` instance, a `SchM` switch action needs to be added to the `ComM_ActionList`. A `BswM` switch port is also required to configure this action and the user must use the proper mode declaration group provided by the `ComM` for this type of mode request. A custom mode declaration group can also be used.

The SchM switch action will switch the mode of the switch port to a value corresponding to the RequestedMode if the ComM request. On the second BswM instance, a BSW mode notification port needs to be added. This port needs to reference the same mode declaration group via the switch interface specified on the switch port from the first BswM instance. A rule needs to be added on the second BswM instance. This rule will verify that the BSW mode notification port is equal to the required mode declaration group value that was switched by the first instance via the switch port.

In order to finish the setup, the RTE Editor's BSW Mode Mapping tool needs to be used in order to connect the provided mode group corresponding to the switch port on the first instance to the required mode group corresponding to the BSW mode notification port on the second instance.

For a better understanding of how this works and what mode arbitration, rule, mode conditions, logical expressions, interfaces and ports do, read from the AUTOSAR_SWS_BSWModeManager.pdf chapter called "Functional specification".

4.2.2.3.2. Configuration options

To use the **BSW Distribution** feature in the BswM, configure the parameter BswMPartitionRef. Refer to Section 4.3, “Configuring the EcuM (ECU State Manager) module and the BswM (BSW Mode Manager) module” for further integration and configuration instructions.

4.2.2.4. Post-build selectable support

BswM is variant-aware and allows you to configure different values in different variants for several configuration parameters. These parameters are either existing post-build changeable containers that can have a different number of instances in different selectable variants. Or they are existing configuration parameters with the implementation config class set to PostBuild that can have different values in different selectable variants.

If there are multiple post-build selectable variants defined, the BswM produces different BswM_{variant}.xgen files consisting of the specific variant configuration. In this situation, BswM_PBcfg.c aggregates all the generated BswM_{variant}_PBcfg.c files.

If no different BswM configurations are required in a project that features multiple post-build selectable variants, you can use the POST_BUILD_VARIANT_USED configuration parameter in order for BswM to produce a single BswM.xgen file. If you set the POST_BUILD_VARIANT_USED to false, the behavior of BswM is the same as in a project without post-build selectable variants.

Note that if the selected IMPLEMENTATION_CONFIG_VARIANT is VariantPreCompile, you cannot set the POST_BUILD_VARIANT_USED parameter to true.

4.2.3. Communication Manager (ComM)

4.2.3.1. Overview

This chapter describes specific features of the `ComM` module. For background information on the communication management, see chapter *Concepts* in the EB tresos AutoCore Generic documentation.

4.2.3.2. BSW distribution

The `ComM` provides mechanisms for handling communication mode and network management when the communication stack is distributed over multiple cores and one bus type (e.g CAN, FR, IP, LIN) is mapped to one core. For the general concept of BSW distribution of the communication mode management, see chapter *Multi-core support* in the EB tresos AutoCore Generic documentation.

4.2.3.2.1. Functional overview

In a multi-core context, the communication management functionality is processed on the master core. This means that both channel and partial network state machines are processed on this master core.

The `ComM` interacts with `Nm` and `BusSm` during the communication startup or communication shutdown. With a distributed communication stack, the `ComM` uses direct function calls when calling the `Nm` because `ComM` and `Nm` are mapped to the same core. The `ComM` uses cross-core `SchM_Call()` operations when calling a bus-specific state manager `BusSm`.

In case of partial networking, the `ComM` receives and transmits partial network `Com` signals. As the signals are processed locally on a core, the `ComM` provides implementations for satellites. The number of satellites depends on the number of cores that process the partial network signals. The satellites only provide the functionality for exchanging partial network information (PNI) between the master where the PNI is processed and the satellites where the PNI is received or transmitted by using core-local calls of `Com_SendSignal` and `Com_ReceiveSignal`.

Depending on the project use case, the diagnostic communication requests to `ComM` can be triggered from the same core or from a different core. In order to support both cases, the calls between `Dcm` and `ComM` are handled by using `SchM_Call()`.

You can enable or disable the multi-core support for `ComM`. When multi-core support is disabled, the `ComM` always uses direct function calls because all involved modules are mapped to the same core. When multi-core support is enabled, the `ComM` always uses `SchM_Call()`. You must ensure that the integrated modules are compatible and have multi-core support enabled.

The `ComM` multi-core BSW distribution support is enabled with the configuration parameter `ComMMultiCoreSupport`. When this parameter is enabled, the generated basic software module description includes:

- all required and provided client-server entries, based on the number of configured channels
- all implementations for the ComM satellites, based on the use of partial networking and the PNI signals
- all required and provided variable data prototypes, based on the configured PNI signals

The ComM generates multiple BswImplementations based on the following information:

- assignment of the ComM master core
- mapping of the partial network signals to PDUs
- mapping of the Com main processing functions

For information on how to configure the ComM BSW distribution support, see Section 4.5, “Configuring the ComM module”

4.2.4. Generic Network Management Interface module (Nm)

4.2.4.1. Overview

In this chapter, you find information describing specific features of the Nm module. Background information about the network management concept and the Nm module is provided in the Concepts chapter of the EB tresos AutoCore Generic documentation. You find instructions relating to the configuration of Nm features in the parameter descriptions in the Nm module references chapter. You can also view the parameter descriptions in the **Description** view in EB tresos Studio.

4.2.4.2. Network management state information

Network management state information (NMS) is a feature that gives you the possibility to send user data information regarding the state of the bus as part of an Nm PDU.

4.2.4.2.1. Functional overview

The basic principle of this feature is to handle the Nm user data as a signal and to use existing mechanisms from the communication stack in order to ensure consistency of the user data. For this purpose, a Com signal is configured in the Nm module.

When you enable this feature the lower <bus>Nm calls a callback function Nm_StateChangeNotification(). Within this function, the user data is updated by calling Com_SendSignal() with the identifier of the related Com signal configured in Nm.

The data sent within this signal is described in the table below:

Value	Name	Description
1	NM_RM_BSM	NM in state RepeatMessage(transition from BusSleepMode)
2	NM_RM_PBSM	NM in state RepeatMessage(transition fromPrepareBusSleepMode)
4	NM_NO_RM	NM in state NormalOperation(transition from RepeatMessage)
8	NM_NO_RS	NM in state NormalOperation(transition from ReadySleep)
16	NM_RM_RS	NM in state RepeatMessage(transition from ReadySleep)
32	NM_RM_NO	NM in state RepeatMessage(transition from NormalOperation)

Table 4.1. NMS signal data

4.2.4.2.2. Configuration options

You can use the network management state information (NMS) feature by enabling the configuration parameters `NmStateChangeIndEnabled` and `NmStateReportEnabled`. The parameter `NmStateReportSignalRef` references the related `Com` signal. See the parameter descriptions for further information.

4.2.4.3. BSW distribution

The `Nm` provides mechanisms to call the bus-specific network management modules mapped to different cores with one bus type processed on one core. For the general concept for BSW distribution of the mode management for communication, see chapter *Multi-core support* in the EB tresos AutoCore Generic documentation.

4.2.4.3.1. Functional overview

The `Nm` module must be mapped to the same core as the `ComM` master where the channel and partial network state machines are processed. All function calls between `ComM` and `Nm` are done on the local core via direct function calls.

When multi-core support is enabled, the `Nm` interacts with bus-specific network management modules (`BusNm`s) cross-core by using `SchM_Call()` operations.

The `Nm` module provides a set of synchronous APIs that are meant to be used for OEM-specific extensions. If you want to use these APIs in your project, you should always use the corresponding APIs provided by `BusNm` modules on the local core. The goal is to avoid cross-core synchronous interfaces because they could cause blocking.

The `Nm` BSW distribution support is enabled with configuration parameter `NmMultiCoreSupport`. If this parameter is enabled, the generated basic software module description includes all required and provided client-server entries. These entries are generated based on the number of configured channels. When the multi-core support is enabled in the `Nm`, you must also enable the multi-core support for the `BusNm`s modules that are used in your project. This allows you to connect the required client-server connections in the `Rte`.

If the calling context of a `SchM_Call()` is unknown, a dummy context is used. You must ensure that the dummy context and relevant events are mapped to tasks with correctly assigned priorities. For example, in the case of `Nm`, the functions are normally called from the context of the `ComM` main processing function. In this case, you must make sure that the dummy context used has the same priority as the `ComM` context. For the cases where a function is executed from the context of an interrupt, you must make sure that the dummy context used has the highest priority in the system, similar to an interrupt.

4.3. Configuring the EcuM (ECU State Manager) module and the BswM (BSW Mode Manager) module

4.3.1. Overview

The following sections provide you with instructions on how to configure the `EcuM` and `BswM` modules.

NOTE

Before configuring the `EcuM` module, execute the Service Needs Calculator
Many parameters of the `EcuM` module are set automatically by the **Service Needs Calculator**. Therefore, execute the **Service Needs Calculator** before configuring the `EcuM` module.

Instructions on how to execute the **Service Needs Calculator** are available in the EB tresos AutoCore Generic 8 Base documentation.

In this chapter you find only instructions on how to set those parameters that are not set by the **Service Needs Calculator**. For the best possible results, read through the instructions below step-by-step in the order presented.

This guide is intended for users who have basic knowledge of the following:

- Integration of AUTOSAR BSW modules
- Startup behavior of an ECU
- State management of an ECU

4.3.2. General configuration/integration hints

Two template files for all `EcuM` callout functions are generated in the folder `workspace/<yourProjectName>/output/generated/templates` when you generate your project. To include these files into the build process, copy the template files after generation into your application folder. If necessary, adapt the template files to your specific needs. Further information about generating and building your application can be found in the build environment chapter of the EB tresos AutoCore Generic 8 Base documentation.

As illustrated in Figure 4.4, “Sample containers, references and multiplicity of a `BswM` rule”, each `BswMRule` references exactly one `BswMLogicalExpression` and one `BswMActionList`. This is so, because only the `BswMRuleTrueActionList` is used. Therefore, these two references are not listed each time in the configuration examples.

You must configure the interfaces that are provided by the `BswM` module. At least the following three interfaces which are configured by the parameter `BswMModeRequestPort` have to be configured.

BswMModeRequestPort	BswMModeRequestSource
BswM_ReqPort	BswMGenericRequest
EcuM_WakeupSourcePort_Power	BswMEcuMWakeupSource
BswM_CurrentModePort	BswMComMIndication

Table 4.2. BswM interface configuration (`BswMModeRequestPort`)

WARNING

Map `BswM_MainFunction()` to a standard task (not extended)

If `Rte_Start()` is called from `BswM` context, map the `BswM_MainFunction()` to a standard task (not extended). Otherwise, a deadlock occurs.

4.3.2.1. Migrating the `EcuM` from ACG-8.5.2 to ACG-8.6.0

1. Migrating configuration parameters

   ▶ Import the `EcuM` module.

   For details on how to import configuration data, see the `EB tresos Studio user's guide`.

   ▶ For each `EcuMDriverInitItem` container in `EcuMDriverInitListOne` and `EcuMDriverRestartList`, you need to configure a Boolean parameter `EcuMEnableDriver`.

   If enabled, the corresponding `EcuMDriverInitItem` service is initialized by the `EcuM`. This parameter is enabled by default for each new `EcuMDriverInitItem` container.

2. Migrating API changes

   ▶ The access to the `EcuM` post-build data structure changed.

   Therefore, the `EcuM_PBcfg.h` header file must be included in order for the callout function `EcuM_DeterminePbConfiguration()` to return a pointer to the `EcuM` post-build data structure.

   ▶ The definitions of the following callout functions changed: `EcuM_AL_DriverInitOne()` and `EcuM_AL_DriverRestart()`.

EcuM provides an example of the definitions in the generated template file EcuM_Callout_Stubs.c.

4.3.2.2. Integration notes for multi-core projects

Multi-core is enabled in the Os by configuring the OsNumberOfCores parameter to a value higher than 1 and by assigning Os Applications to different core IDs. The initial core from which the application starts will always be the core-ID 0 or the OS_CORE_ID_MASTER (as specified by the Os SWS).

The master EcuM instance needs to be connected with each one of the slave EcuM instances. In order to achieve this, connect the EcuM slave ports with the master EcuM instance by adding new configuration parameters to the Rte configuration under the **BSW Module Instances** list, related to the master EcuM instance. The EcuM ports need to be connected in the following manner:

1. Add a new container for each slave EcuM instance in the **RteBswRequiredSenderReceiverConnection** tab with the following connections:

   - RteBswProvidedVariableDataPrototypeRef: add a reference to the EcuM_SlaveCoreReadyPort. Each slave core shall write to this variable data prototype to signal that they are ready. The slave core shall pass their coreId.
   - RteBswRequiredVariableDataPrototypeRef: add a reference to the EcuM_SlaveCoreSyncPort. The master core shall read from this variable to determine if the slave cores are ready.
   - RteBswProvidedDataModInstRef: add a reference to the slave EcuM instance that you need to synchronize with the master EcuM instance.

Each slave EcuM instance needs to add connection with the master EcuM instance, by connecting the EcuM master ports with each EcuM slave instances. In the Rte configuration, under the **BSW Module instances** list, in the container of each slave EcuM instance, the EcuM ports need to be connected in the following manner:

1. Add a new container in the **RteBswRequiredSenderReceiverConnection** tab with the following connections:

   - RteBswProvidedVariableDataPrototypeRef: add a reference to the EcuM_MasterCoreReadyPort. The master core shall write to this variable data prototype to signal the slave cores that it is ready.
   - RteBswRequiredVariableDataPrototypeRef: add a reference to the EcuM_MasterCoreSyncPort. The slave cores shall read from their port to determine if the master core is ready.
   - RteBswProvidedDataModInstRef: add a reference to the master EcuM instance in order to synchronize it with the current slave EcuM instance.

2. Also, the ports related to the sleep mode need to be connected, for this add a new container in the **RteBswRequiredSenderReceiverConnection** tab with the following connections:

- RteBswProvidedVariableDataPrototypeRef: add a reference to the EcuM_SetSleepMode. The master core shall write to the SetSleepMode port to inform the slave cores of the currently selected sleep mode.
- RteBswRequiredVariableDataPrototypeRef: add a reference to the EcuM_GetSleepMode. Each slave core shall read from their GetSleepMode port to determine the current sleep mode.
- RteBswProvidedDataModInstRef: add a reference to the master EcuM instance in order to synchronize it with the current slave EcuM instance.

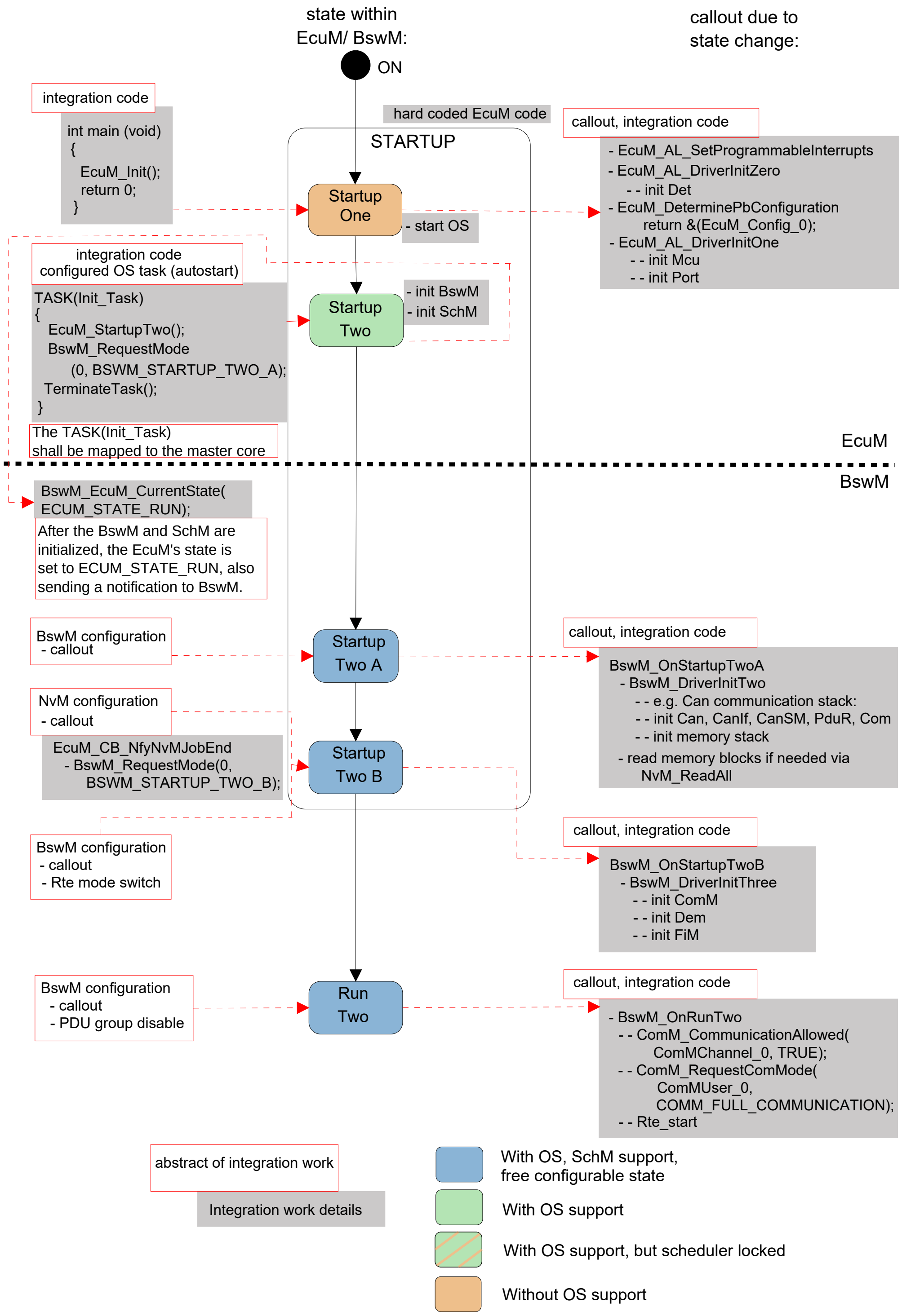
To set up multiple instances of the BswM module proceed as follows:

1. Configure an OS-Application for each partition on which a BswM instance is required in the Os module configuration.
2. Assign the OS-Applications to the desired cores.
3. In the BswM module configuration, add a BswM instance configuration for each partition as needed.
4. In each BswM instance configuration, enable the BswMPartitionRef parameter and select the OS-Application reference that you configured earlier in the Os module.
5. Finally, set up the BSW partitioning in the Rte.

4.3.3. Configuring EcuM and BswM for single-core projects

4.3.3.1. Configuring the startup states

The startup state of the BswM and the EcuM is initiated after power on or reset. Figure 4.5, “The startup state of the EcuM and of the BswM” gives an overview of the startup states, including the static EcuM states and the configurable BswM states.



```
int main (void)
{
   EcuM_Init();
   return 0;
}
```

```
- EcuM_AL_SetProgrammableInterrupts
- EcuM_AL_DriverInitZero
  - - init Det
- EcuM_DeterminePbConfiguration
      return &(EcuM_Config_0);
- EcuM_AL_DriverInitOne
   - - init Mcu
   - - init Port
```

```
TASK(Init_Task)
{
   EcuM_StartupTwo();
   BswM_RequestMode
        (0, BSWM_STARTUP_TWO_A);
  TerminateTask();
}
```

The TASK(Init_Task) shall be mapped to the master core

```
BswM_EcuM_CurrentState(
ECUM_STATE_RUN);
```

After the BswM and SchM are initialized, the EcuM's state is set to ECUM_STATE_RUN, also sending a notification to BswM.

```
BswM_OnStartupTwoA
  - BswM_DriverInitTwo
     - - e.g. Can communication stack:
     - - init Can, CanIf, CanSM, PduR, Com
     - - init memory stack
  - read memory blocks if needed via
      NvM_ReadAll
```

```
EcuM_CB_NfyNvMJobEnd
  - BswM_RequestMode(0,
       BSWM_STARTUP_TWO_B);
```

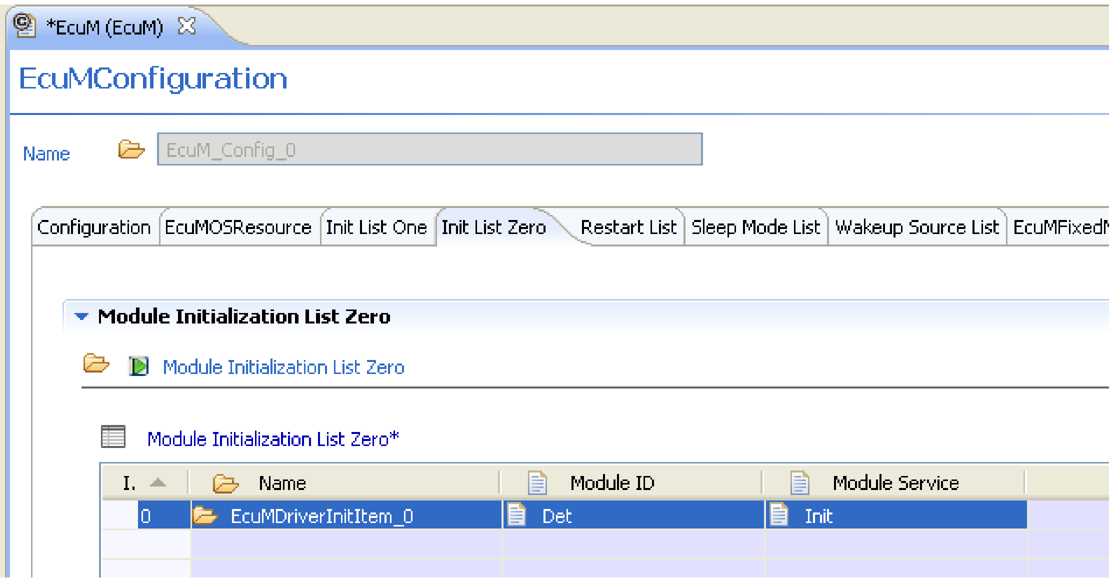
```
BswM_OnStartupTwoB
  - BswM_DriverInitThree
     - - init ComM
     - - init Dem
     - - init FiM
```

```
- BswM_OnRunTwo
  - - ComM_CommunicationAllowed(
          ComMChannel_0, TRUE);
  - - ComM_RequestComMode(
           ComMUser_0,
           COMM_FULL_COMMUNICATION);
  - - Rte_start
```

Figure 4.5. The startup state of the `EcuM` and of the `BswM`

After executing the startup code, the main function is called. Therefore, you must provide the main function and its implementation as integration code as shown in Figure 4.5, “The startup state of the `EcuM` and of the `BswM`”.

- Configure the **Module Initialization List Zero**:



EcuMConfiguration

Name: EcuM_Config_0

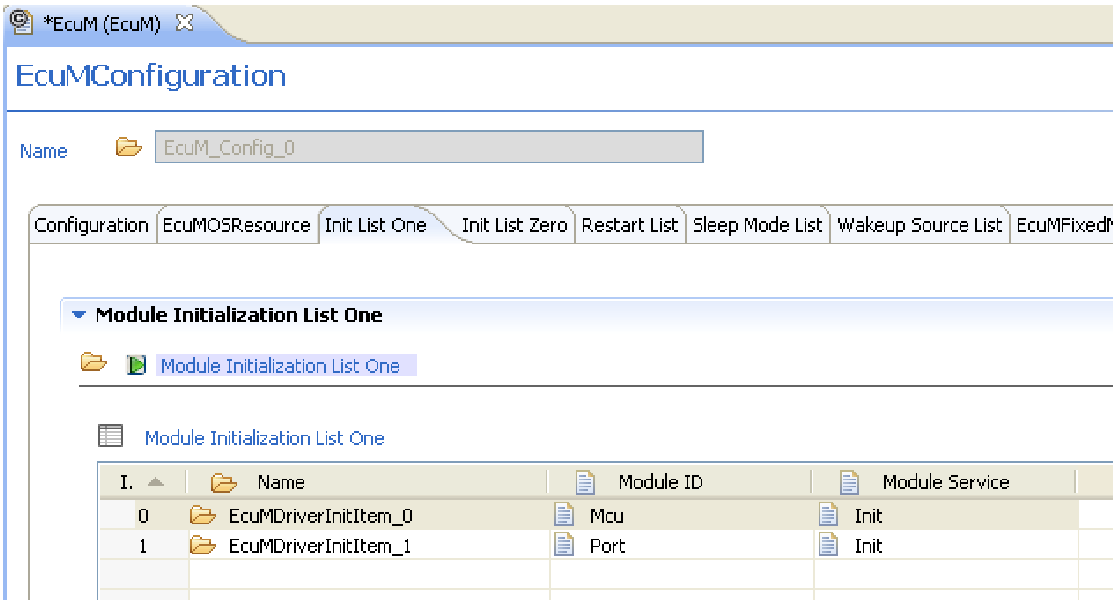
Module Initialization List Zero

Module Initialization List Zero

Module Initialization List Zero*

I.	Name	Module ID	Module Service
0	EcuMDriverInitItem_0	Det	Init

- Configure the **Module Initialization List One**:



EcuMConfiguration

Name: EcuM_Config_0

Module Initialization List One

Module Initialization List One

Module Initialization List One

I.	Name	Module ID	Module Service
0	EcuMDriverInitItem_0	Mcu	Init
1	EcuMDriverInitItem_1	Port	Init

The `EcuM` enters the state `ECUM_STATE_STARTUP_ONE` and calls the callout functions shown in Figure 4.5, “The startup state of the `EcuM` and of the `BswM`”.

- Use the template files as described in Section 4.3.2, “General configuration/integration hints” as base for your callout implementation. If you have configured the *Module Initialization List Zero* and the *Module Initialization List One* as described above, the template file includes all configured init function calls.

Now the `EcuM` starts the `Os`.

- Configure the *Init_Task* within the `Os` and enable the parameter `OsTaskAutostart` for this task.

The `EcuM` starts the `Os` at the end of the `startup one` state. After the `Os` has been started, all tasks with the parameter `OsTaskAutostart` set to `TRUE` are executed. Therefore the `Init_Task` is executed and the integrator must provide the implementation according to Figure 4.5, “The startup state of the `EcuM` and of the `BswM`”.

The `EcuM` initializes the BSW Scheduler (`SchM`) and the `BswM` modules. The `Init_Task` includes a call to the `BswM` module for requesting a new mode:

```
BswM_RequestMode(0, BSWM_STARTUP_TWO_A);
```

The state handling is now given to the `BswM` module.

The general configuration structure of the `BswM` is illustrated by Figure 4.4, “Sample containers, references and multiplicity of a `BswM` rule”. The general settings of some configuration parameters are summarized in Section 4.3.2, “General configuration/integration hints”.

- In addition to the general settings, configure the `BswM` in the following way for the `startup` state:

BswMRule	BswMModeCondition	BswMAction
BswMRule_StartupTwoA	BswM_Cond_StartupTwoA ▶ BswMConditionMode: BswM_ReqPort ▶ BswMConditionValue: BswMBswMode ▶ BswMBswModeSourceType: BSWM_GENERIC_REQUEST ▶ BswMBswRequestedMode: BSWM_STARTUP_TWO_A	BswM_Act_StartupTwoA ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: `BswM_OnStartupTwoA()`
BswMRule_StartupTwoB	BswM_Cond_StartupTwoB ▶ BswMConditionMode: BswM_ReqPort ▶ BswMConditionValue: BswMBswMode ▶ BswMBswModeSourceType: BSWM_GENERIC_REQUEST ▶ BswMBswRequestedMode: BSWM_STARTUP_TWO_B	BswM_Act_StartupTwoB ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: `BswM_OnStartupTwoB()` BswM_Act_Request_RunTwo ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: BswM_RequestMode(0, BSWM_RUN_TWO)

BswMRule	BswMModeCondition	BswMAction
BswMRule_RunTwo	BswM_Cond_RunTwo ▶ BswMConditionMode: BswM_ReqPort ▶ BswMConditionValue: BswMBswMode ▶ BswMBswModeSourceType: BSWM_GENERIC_REQUEST ▶ BswMBswRequestedMode: BSWM_RUN_TWO	BswM_Act_RunTwo ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: `BswM_OnRunTwo()`
		BswM_Act_RunTwo_Pdu ▶ BswMAvailableActions: BswMPduGroupSwitch ▶ BswMEnabledPduGroupRef: your configured PDU groups
		BswM_Act_NotifyRteRun ▶ BswMAvailableActions: BswMRteSwitch ▶ BswMRteSwitchInterfaceRef: reference to MODE-SWITCH-INTERFACE ▶ BswMSwitchedMode: reference to MODE-DECLARATION-GROUP

4.3.3.2. Configuring the `shutdown` states

The `shutdown` state is initiated after a request from a software component or from the `ComM` module. Figure 4.6, “The `shutdown` states” gives an overview of the `shutdown` states, including the static `EcuM` state and the configurable `BswM` states:

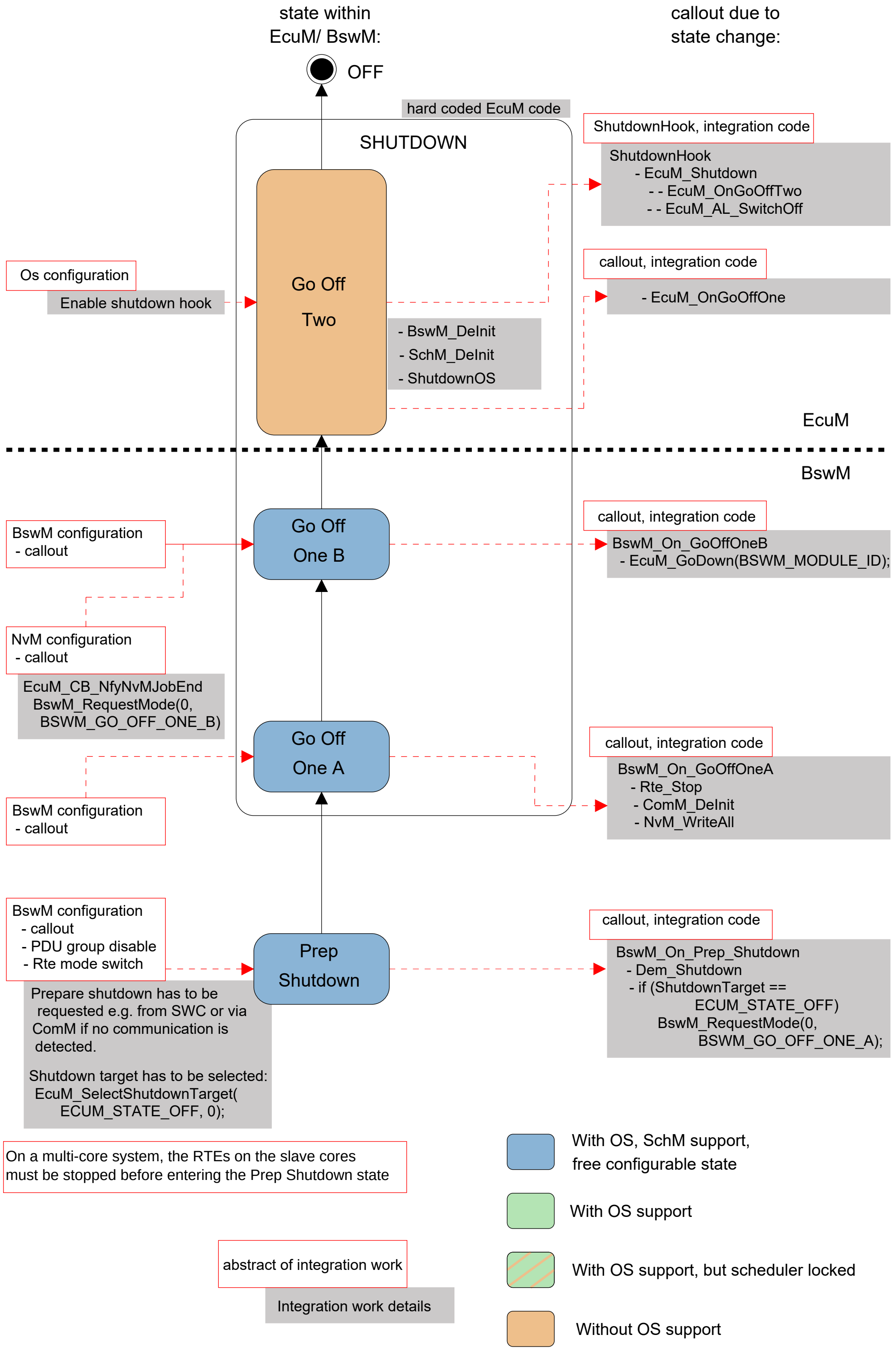


Figure 4.6. The `shutdown` states

When requesting shutdown, set the shutdown target to ECUM_STATE_OFF. The callout function BswM_On_Prep_Shutdown evaluates this parameter and requests go off one A from the BswM module. The BswM configuration is summarized in the table below and the needed callout functions are shown in Figure 4.6, “The shutdown states”.

In the callout function BswM_On_GoOffOneB, the BswM module delegates the shutdown sequence to the EcuM module by calling the function EcuM_GoDown. The EcuM de-initializes the BswM module and the BSW Scheduler (SchM), and shuts down the Os.

To ensure that the shutdown hook is called, enable the parameter OsShutdownHook in the Os configuration.

Configure the BswM parameters in the following way for the shutdown state:

BswMRule	**BswMModeCondition**	**BswMAction**
BswMRule_NoCom	BswM_Cond_NoCom ▶ BswMConditionMode: BswM_CurrentModePort ▶ BswMConditionValue: BswMBswMode ▶ BswMBswModeSourceType: BSWM_COMM_INDICATION ▶ BswMBswRequestedMode: COMM_NO_COMMUNICATION	BswM_Act_PrepShutdown ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: BswM_On_Prep_Shutdown()
		BswM_Act_NotifyRteRun ▶ BswMAvailableActions: BswMRteSwitch ▶ BswMRteSwitchInterfaceRef: reference to MODE-SWITCH-INTERFACE ▶ BswMSwitchedMode: reference to MODE-DECLARATION-GROUP
		BswM_Act_PrepShutdown_Pdu ▶ BswMAvailableActions: BswMPduGroupSwitch ▶ BswMDisablePduGroupRef: enter your configured PDU groups
BswMRule_GoOffOneA	BswM_Cond_GoOffOneA ▶ BswMConditionMode: BswM_ReqPort ▶ BswMConditionValue: BswMBswMode ▶ BswMBswModeSourceType: BSWM_GENERIC_REQUEST ▶ BswMBswRequestedMode: BSWM_GO_OFF_ONE_A	BswM_Act_GoOffOneA ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: BswM_OnGoOffOneA()

BswMRule	BswMModeCondition	BswMAction
BswMRue_GoOffOneB	BswM_Cond_GoOffOneB ▶ BswMConditionMode: BswM_ReqPort ▶ BswMConditionValue: BswMBswMode ▶ BswMBswModeSourceType: BSWM_GENERIC_REQUEST ▶ BswMBswRequestedMode: BSWM_GO_OFF_ONE_B	BswM_Act_GoOffOneB ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: `BswM_OnGoOffOneB()`

4.3.3.3. Configuring the `sleep` states

The `sleep` state is initiated after a request from a software component or from the `ComM` module. Figure 4.7, “The `sleep` states” gives an overview of the `sleep` states, including the static `EcuM` states and the configurable `BswM` states.

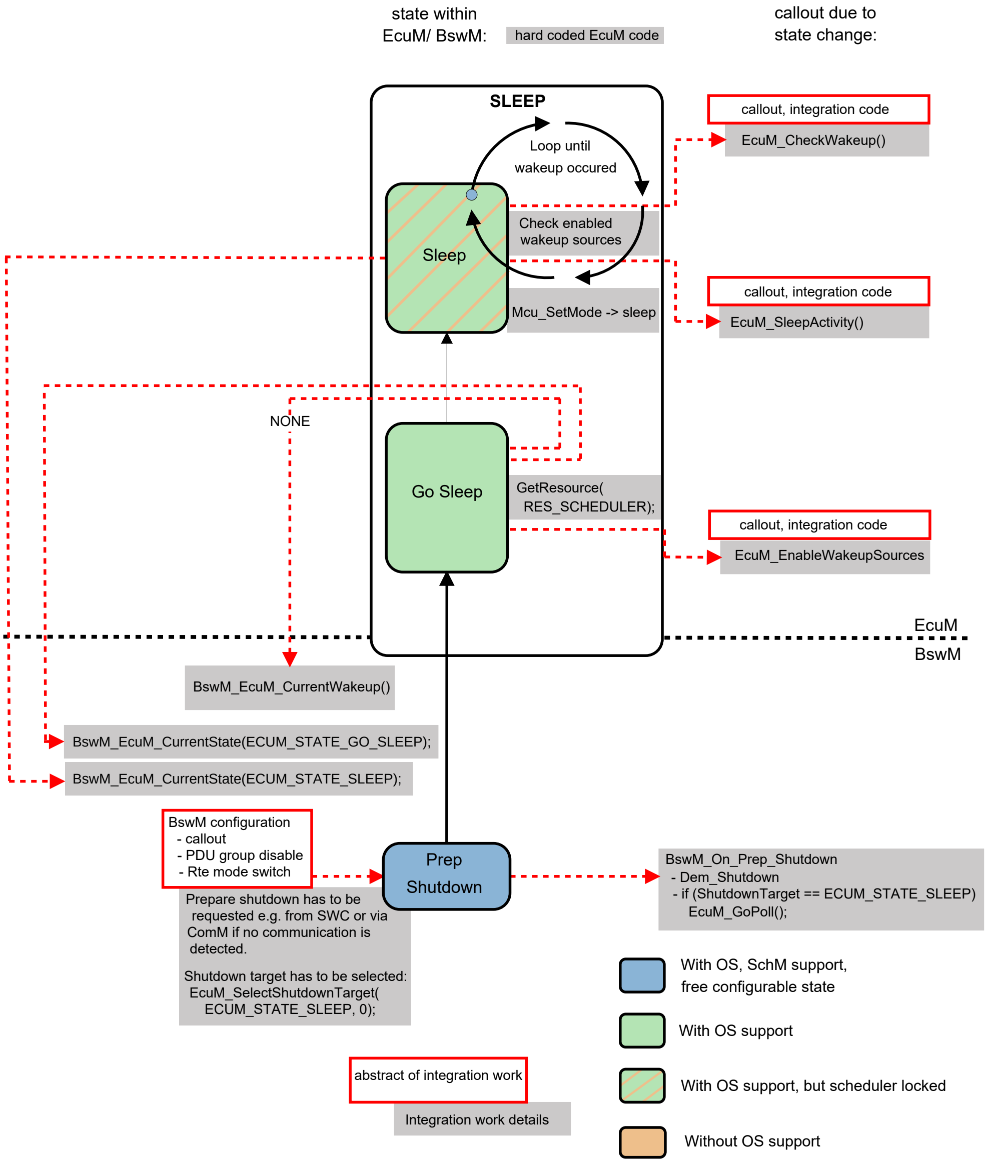
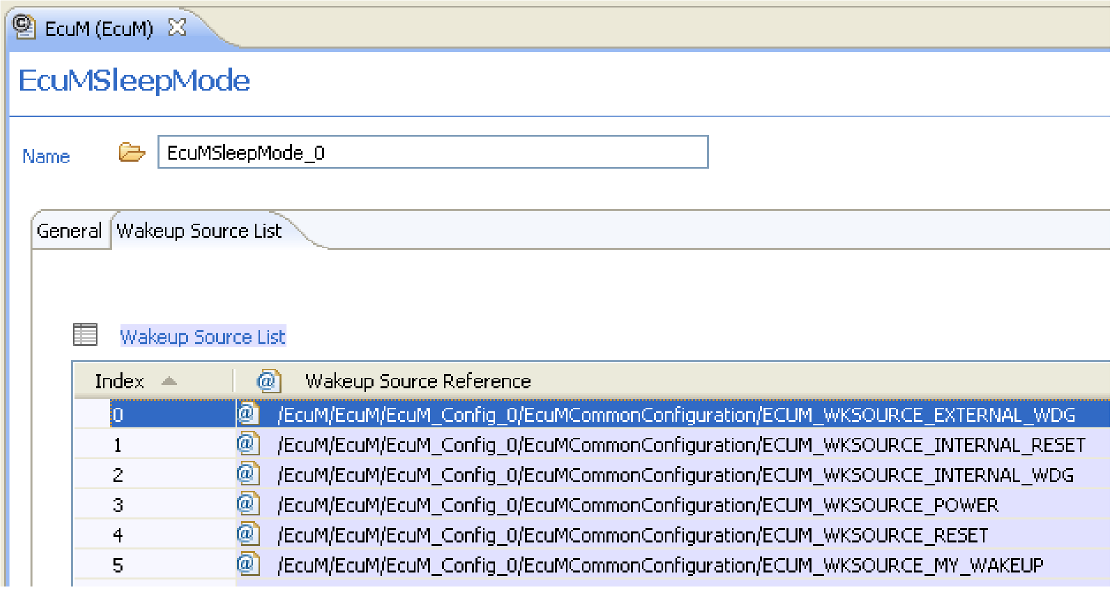


Figure 4.7. The `sleep` states

To configure the `sleep` state, you do not need to configure the `BswM` module, because the `prepare shutdown` configuration as described in Section 4.3.3.2, “Configuring the `shutdown` states” is also used for the `sleep` state.

- When requesting `sleep`, set the shutdown target to `ECUM_STATE_SLEEP`. The callout function `BswM_On_Prep_Shutdown` evaluates this parameter and calls the `EcuM` function `EcuM_GoPoll`.
- Alternatively, call the `EcuM` function `EcuM_GoHalt`.
- To configure the `EcuM` module for the `sleep` mode, configure the `sleep` mode list by registering all `wakeup` sources for this `sleep` mode, as shown in Figure 4.8, “Configuring the **Wakeup Source List**”.



Name: EcuMSleepMode_0

Index	Wakeup Source Reference
0	/EcuM/EcuM/EcuM_Config_0/EcuMCommonConfiguration/ECUM_WKSOURCE_EXTERNAL_WDG
1	/EcuM/EcuM/EcuM_Config_0/EcuMCommonConfiguration/ECUM_WKSOURCE_INTERNAL_RESET
2	/EcuM/EcuM/EcuM_Config_0/EcuMCommonConfiguration/ECUM_WKSOURCE_INTERNAL_WDG
3	/EcuM/EcuM/EcuM_Config_0/EcuMCommonConfiguration/ECUM_WKSOURCE_POWER
4	/EcuM/EcuM/EcuM_Config_0/EcuMCommonConfiguration/ECUM_WKSOURCE_RESET
5	/EcuM/EcuM/EcuM_Config_0/EcuMCommonConfiguration/ECUM_WKSOURCE_MY_WAKEUP

Figure 4.8. Configuring the **Wakeup Source List**

4.3.3.4. Configuring the wakeup states

The wakeup state is initiated after a pre-configured wakeup event occurred. Figure 4.9, “The wakeup states” gives an overview of the wakeup states, including the static EcuM states and the configurable BswM states.

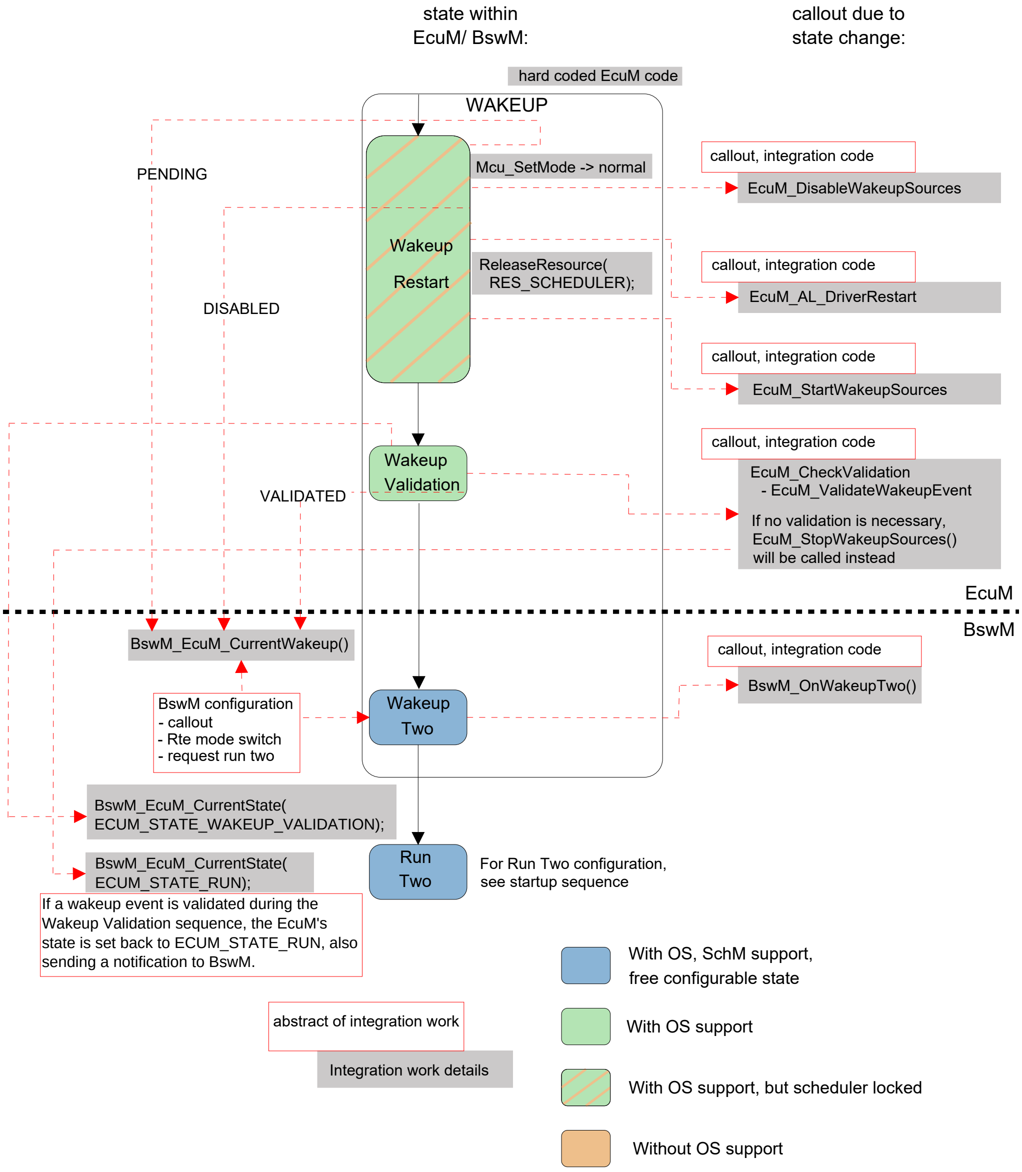
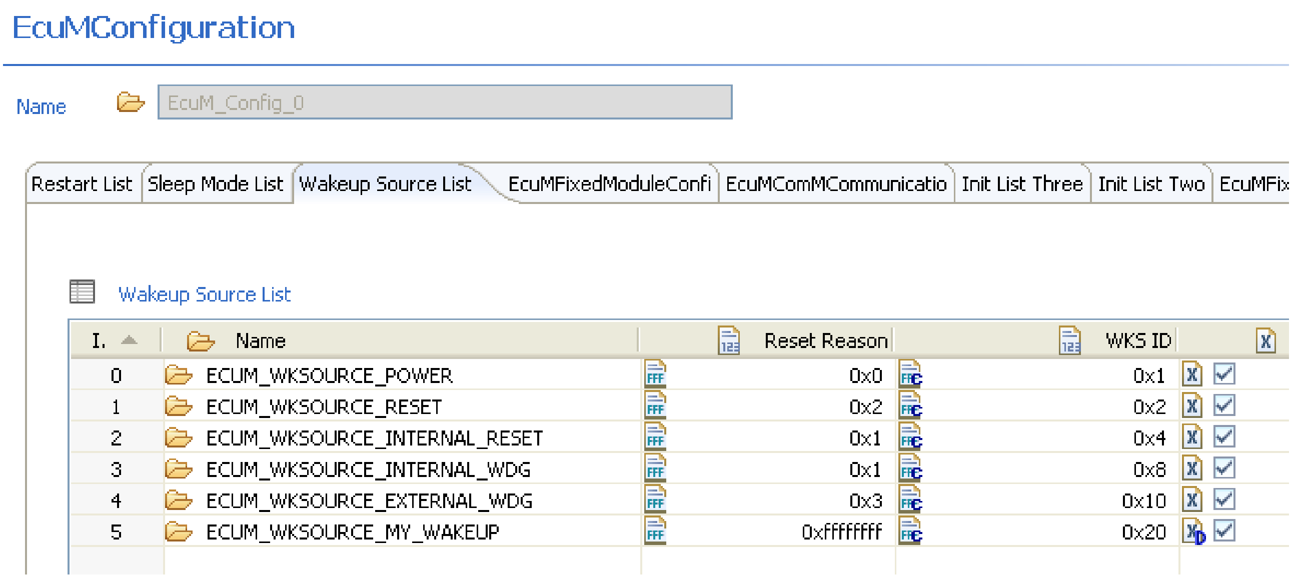


Figure 4.9. The `wakeup` states

During `sleep` mode, the `EcuM` reacts on wakeup events which are configured for the `sleep` mode. Therefore, the wakeup events have to be configured in the `EcuM` *Wakeup Source List* as shown in Figure 4.10, “Configuring the **Sleep Mode List**”. Additionally, the wakeup events must be referenced from the *Sleep Mode List* configuration as shown in Figure 4.8, “Configuring the **Wakeup Source List**”.



EcuMConfiguration

Name EcuM_Config_0

Restart List | Sleep Mode List | Wakeup Source List | EcuMFixedModuleConfi | EcuMComMCommunicatio | Init List Three | Init List Two | EcuMFix

Wakeup Source List

I.	Name	Reset Reason	WKS ID	
0	ECUM_WKSOURCE_POWER	0x0	0x1	☑
1	ECUM_WKSOURCE_RESET	0x2	0x2	☑
2	ECUM_WKSOURCE_INTERNAL_RESET	0x1	0x4	☑
3	ECUM_WKSOURCE_INTERNAL_WDG	0x1	0x8	☑
4	ECUM_WKSOURCE_EXTERNAL_WDG	0x3	0x10	☑
5	ECUM_WKSOURCE_MY_WAKEUP	0xffffffff	0x20	☑

Figure 4.10. Configuring the **Sleep Mode List**

If a wakeup event occurs, the `EcuM` module signals this to the `BswM` module by calling `BswM_EcuM_CurrentWakeup` with the wakeup status *PENDING*. After setting the `Mcu` to normal processing speed, the `wakeup` sources are disabled. This disablement is signaled to the `BswM` module by calling `BswM_EcuM_CurrentWakeup` with the `wakeup` status `DISABLED`. In this configuration example, these two indications are not further processed by the `BswM` module.

After validating a wakeup event successfully, the `BswM_EcuM_CurrentWakeup` is called again, with the `wakeup` status `VALIDATED`. This indication brings the `BswM` into the state `wakeup two` and hands over the further processing of the wakeup handling to the `BswM` module.

Configure the `BswM` in the following way for the `wakeup` state:

BswMRule	**BswMModeCondition**	**BswMAction**
BswMRule_WakeupTwo	BswM_Cond_WakeupTwo ▶ BswMConditionMode: EcuM_WakeupSourcePort_Power ▶ BswMConditionValue: BswMBswMode ▶ BswMBswModeSourceType: BSWM_ECUM_WAKEUP_SOURCE ▶ BswMBswRequestedMode: ECUM_WKSTATUS_VALIDATED	BswM_Act_WakeUpTwo ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: `BswM_OnWakeupTwo()`
		BswM_Act_NotifyRteRun ▶ BswMAvailableActions: BswMRteSwitch ▶ BswMRteSwitchInterfaceRef: reference to MODE-SWITCH-INTERFACE ▶ BswMSwitchedMode: reference to MODE-DECLARATION-GROUP

BswMRule	BswMModeCondition	BswMAction
		BswM_Act_RequestRunTwo ▶ BswMAvailableActions: BswMUserCallout ▶ BswMUserCalloutFunction: BswM_RequestMode(0, BSWM_RUN_TWO)

4.3.4. Configuring EcuM and BswM for multi-core projects

4.3.4.1. Configuring `startup` on systems with multiple cores

To configure `startup` on a multi-core system, take the following steps:

On the master core:

1. Configure the `TASK(Init_Task_Master)` on the master to be triggered as a consequence of the `StartOs()` and inside it call `EcuM_StartupTwo()`.
2. Configure in `BswM` a rule and action list that shall trigger STARTUP_TWO_A.
3. STARTUP_TWO_A shall trigger the `NvM_ReadAll()` operation.
4. When `NvM_ReadAll()` is complete, the NvM Job end notification callout is called e.g. `EcuM_CB_NfyNvMJobEnd()`. Request STARTUP_TWO_B from this callout.
5. Configure in `BswM` a rule and action list that shall trigger STARTUP_TWO_B.
6. STARTUP_TWO_B shall request BSWM_RUN_TWO.
7. Configure in `BswM` a rule and action list that shall trigger BSWM_RUN_TWO.
8. `BswM_OnRunTwo()` is called.

On the slave core:

1. Configure the `TASK(Init_Task_Slave)` on the slave to be triggered as a consequence of the `StartOs()` and inside it call `EcuM_StartupTwo()`.
2. Configure in `BswM` a rule and action list that shall trigger STARTUP_TWO_C.
3. When `NvM_ReadAll()` is complete on the master core, the BSWM_STARTUP_TWO_D is requested by the master core on the slave core.
4. Configure in `BswM` a rule and action list that shall trigger STARTUP_TWO_D.
5. STARTUP_TWO_D shall request BSWM_RUN_TWO.
6. Configure in `BswM` a rule and action list that shall trigger BSWM_RUN_TWO.

7. `BswM_OnRunTwo()` is called.

Below you have a diagram depicting an example on how the `startup` is working on an ECU with one master core and one slave core.

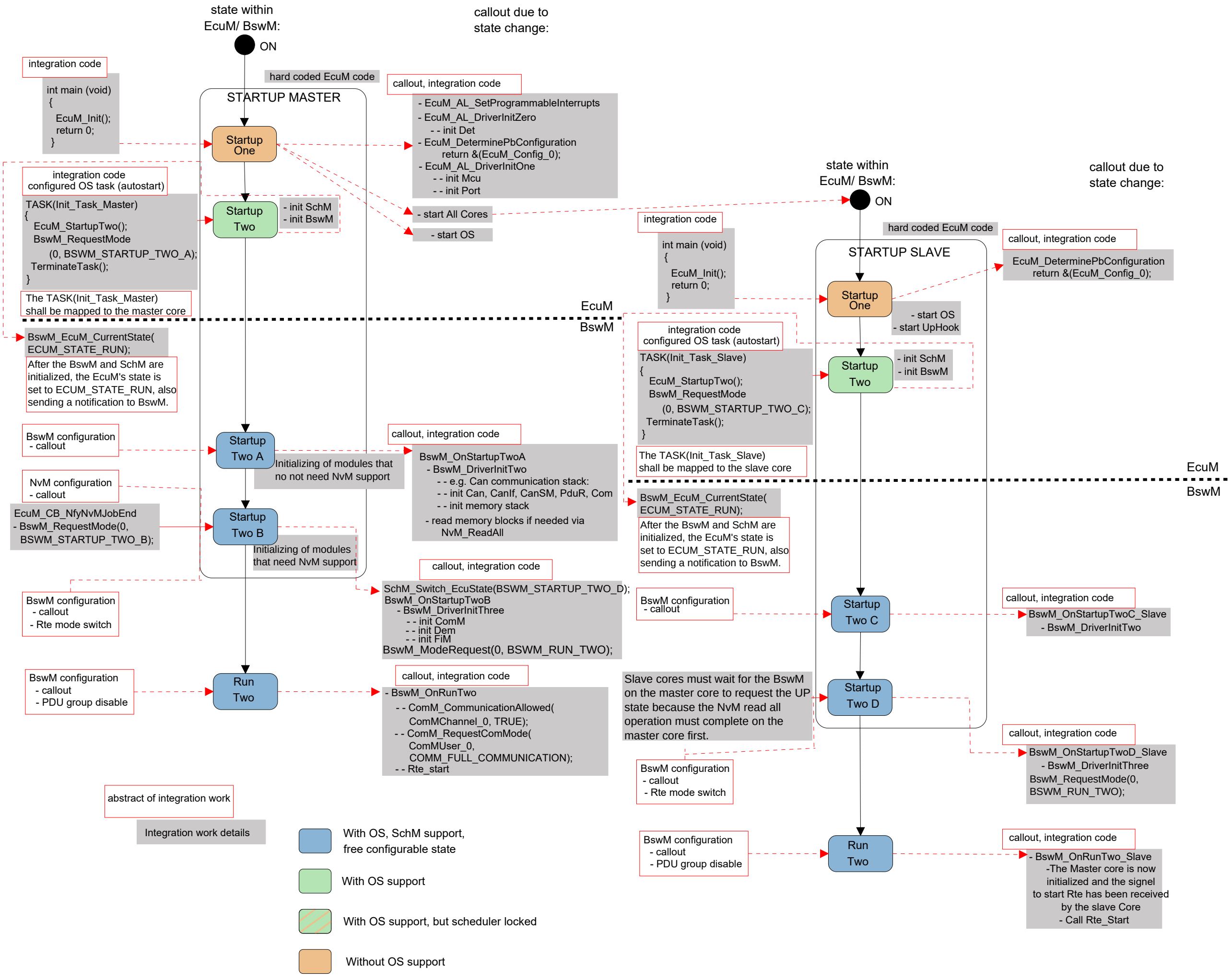


Figure 4.11. The startup state of the `EcuM` and of the `BswM` with multi-core

4.3.4.2. Configuring `shutdown` on systems with multiple cores

To `shutdown` a multi-core system, a synchronization mechanism between the slave core and the master core is needed.

To configure `shutdown` on a multi-core system, take the following steps:

On the master core:

1. Configure all the action to be done before going to state `PrepShutdown`.
2. Configure BSWM_GO_OFF_ONE_A to be called from state `PrepShutdown`.
3. Configure all the actions to be done in GO_OFF_ONE_A state e.g. `Rte_Stop()`, `NvM_WriteAll()`.
4. When `NvM_WriteAll()` is complete, configure the NvM Job End callout e.g. `EcuM_CB_NfyNvMJobEnd()` to be called and BSWM_GO_OFF_ONE_B is requested from `BswM`.
5. Configure in `BswM` a rule and action list that shall trigger GO_OFF_ONE_B.
6. GO_OFF_ONE_B shall pass the notification that the `NvM_WriteAll()` is finished to the slave core.
7. The master core shall wait until the slave core signals that it is ready for `shutdown`.
8. GO_OFF_ONE_B shall call `EcuM_GoDown()`.

On the slave core:

1. Configure all the action to be done before going to state `PrepShutdown`.
2. Configure in `BswM` a rule and action list that shall trigger GO_OFF_ONE_C.
3. Wait for GO_OFF_ONE_D notification from the master core.
4. Configure in `BswM` a rule and action list that shall trigger GO_OFF_ONE_D.
5. GO_OFF_ONE_D shall call `EcuM_GoDown()`.

The callout `EcuM_AL_SwitchOff()` must be configured and the actual `shutdown` (POWER OFF/RESET) must be done via integration code since it is hardware specific. Also depending on project needs the `shutdown` might require additional operations.

Below you have a diagram depicting an example on how the `shutdown` is working on an ECU with one master core and one slave core.

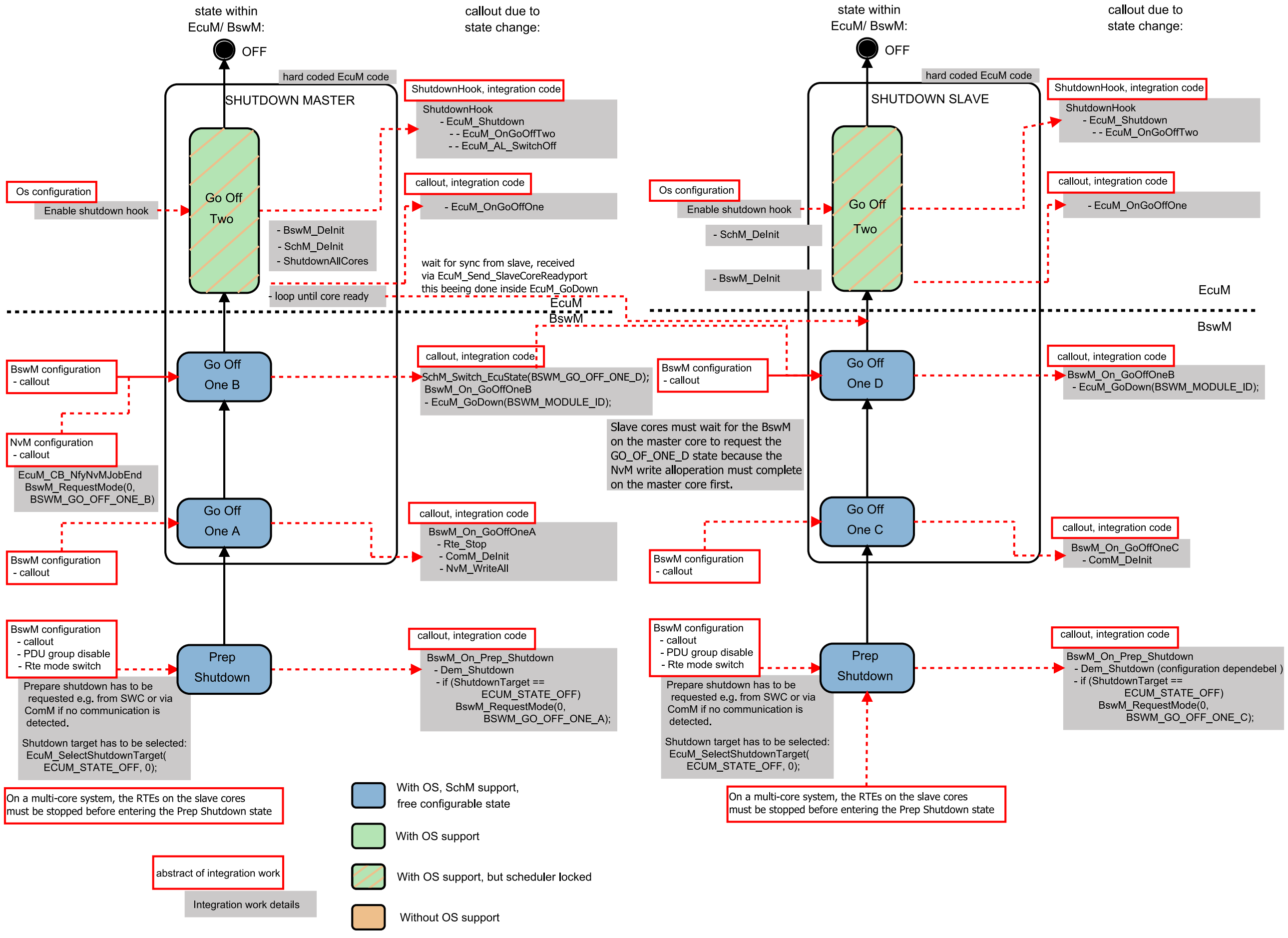


Figure 4.12. The `shutdown` states on a system with multi-core

4.3.4.3. Configuring `sleep` on systems with multiple cores

To configure `sleep` on a multi-core system, take the following steps:

On the master core:

1. Configure the master core to notify the slave core to go to `sleep` mode, when the master is entering `PrepShutdown` state via mode switch interface (inter-core communication).
2. Configure `BswM` to call `EcuM_GoPoll()` as the last step before entering `sleep` state.

On the slave core:

1. When the notification to go to the `sleep` state comes from the master core, `PrepShutdown` must be executed on the slave core.
2. Configure `BswM` to trigger the `PrepShutdown` when the notification is received.
3. Configure `BswM` to call `EcuM_GoPoll()` as the last step before entering `sleep` state.

Below you have a diagram depicting an example on how the `sleep` mode is working on an ECU with one master core and one slave core.

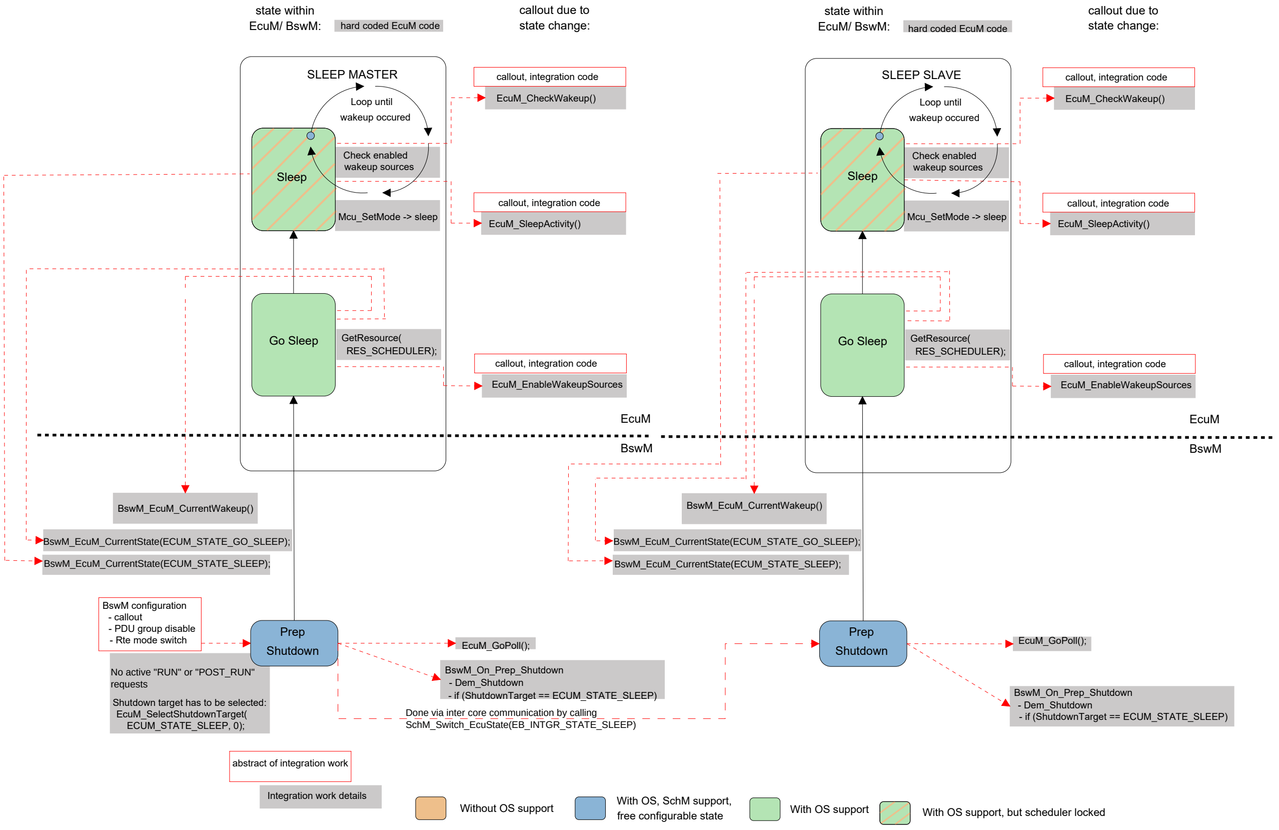


Figure 4.13. The `sleep` states of a system with multi-core

4.3.4.4. Configuring the `wakeup` state on systems with multiple cores

The `wakeup` state is initiated after a pre-configured wakeup event occurs. To configure `wakeup` on a multi-core system, take the following steps:

On the master core:

1. The callout `EcuM_DisableWakeupSources()` should be configured as it is used for changing the hardware state so it will not report the same wakeup event again.
2. The callout `EcuM_AL_DriverRestart()` should be configured if any driver or CDD requires restart when returning from low power.
3. The callout `EcuM_StartWakeupSources()` should be configured as it is used for changing the hardware state so it can validate a pending wakeup event.

4. The callout `BswM_OnWakeupTwo()` shall be configured so that when a pending wakeup event on the master was validated, a set wakeup event is triggered on the slave core.
5. The callout `EcuM_CheckValidation()` is used to poll for a pending wakeup event in order to determine if it was validated or not.

On the slave core:

1. The callout `EcuM_DisableWakeupSources()` should be configured as it is used for changing the hardware state so it will not report the same wakeup event again.
2. The callout `EcuM_AL_DriverRestart()` should be configured if any driver or CDD requires restart when returning from low power.
3. The callout `EcuM_StartWakeupSources()` should be configured as it is used for changing the hardware state so it can validate a pending wakeup event.
4. The callout `EcuM_CheckValidation()` is used to poll for a pending wakeup event in order to determine if it was validated or not.
5. The callout `BswM_OnWakeupTwo()` shall be configured if integration code must be executed as part of wakeup two.

Below you have a diagram depicting an example on how the wake-up is working on an ECU with one master core and one slave core.

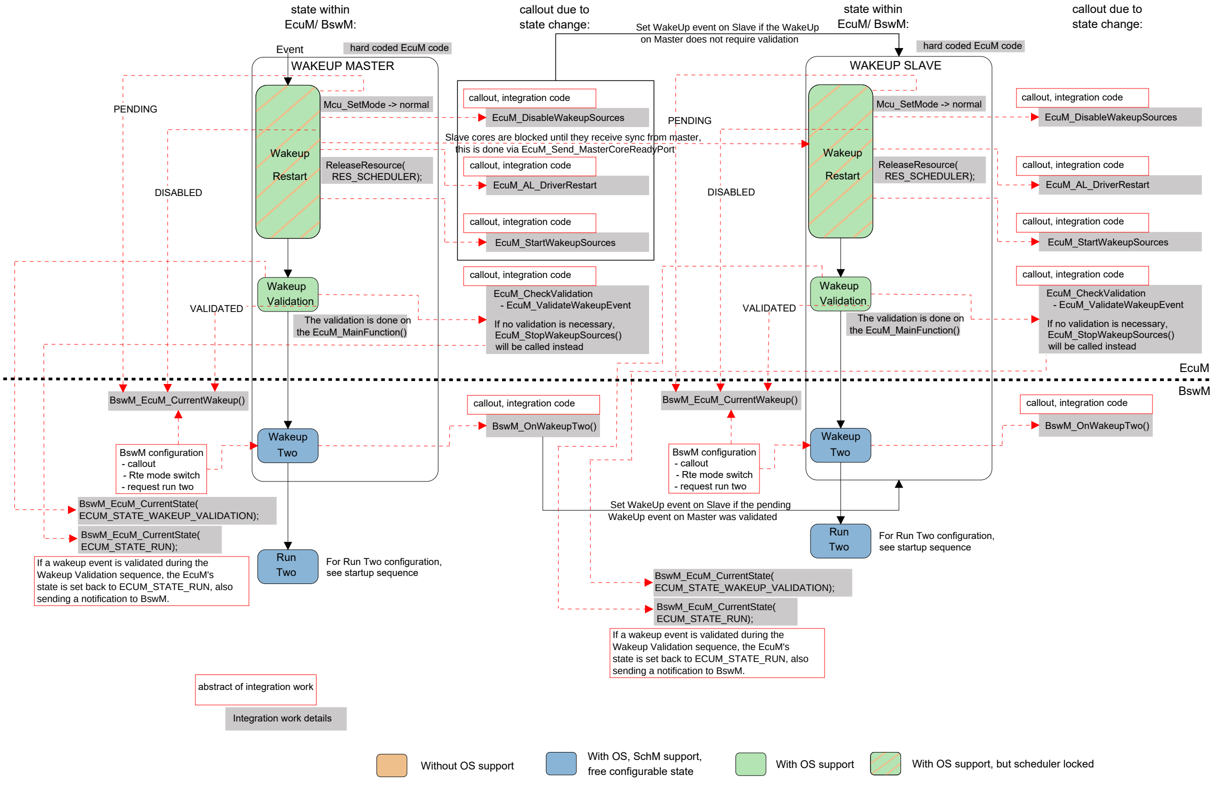

state within
EcuM/ BswM:
callout due to
state change:
Set WakeUp event on Slave if the WakeUp
on Master does not require validation
state within
EcuM/ BswM:
callout due to
state change:
Event
hard coded EcuM code
WAKEUP MASTER
PENDING
DISABLED
VALIDATED
Wakeup
Restart
Mcu_SetMode -> normal
ReleaseResource(
RES_SCHEDULER);
Wakeup
Validation
The validation is done on
the EcuM_MainFunction()
Wakeup
Two
Run
Two
For Run Two configuration,
see startup sequence
callout, integration code
EcuM_DisableWakeupSources
Slave cores are blocked until they receive sync from master,
this is done via EcuM_Send_MasterCoreReadyPort
callout, integration code
EcuM_AL_DriverRestart
callout, integration code
EcuM_StartWakeupSources
callout, integration code
EcuM_CheckValidation
- EcuM_ValidateWakeupEvent
If no validation is necessary,
EcuM_StopWakeupSources()
will be called instead
BswM_EcuM_CurrentWakeup()
BswM configuration
- callout
- Rte mode switch
- request run two
callout, integration code
BswM_OnWakeupTwo()
BswM_EcuM_CurrentState(
ECUM_STATE_WAKEUP_VALIDATION);
BswM_EcuM_CurrentState(
ECUM_STATE_RUN);
If a wakeup event is validated during the
Wakeup Validation sequence, the EcuM's
state is set back to ECUM_STATE_RUN, also
sending a notification to BswM.
hard coded EcuM code
WAKEUP SLAVE
PENDING
DISABLED
VALIDATED
Wakeup
Restart
Mcu_SetMode -> normal
ReleaseResource(
RES_SCHEDULER);
Wakeup
Validation
The validation is done on
the EcuM_MainFunction()
Wakeup
Two
Run
Two
For Run Two configuration,
see startup sequence
callout, integration code
EcuM_DisableWakeupSources
callout, integration code
EcuM_AL_DriverRestart
callout, integration code
EcuM_StartWakeupSources
callout, integration code
EcuM_CheckValidation
- EcuM_ValidateWakeupEvent
If no validation is necessary,
EcuM_StopWakeupSources()
will be called instead
EcuM
BswM
BswM_EcuM_CurrentWakeup()
BswM configuration
- callout
- Rte mode switch
- request run two
callout, integration code
BswM_OnWakeupTwo()
Set WakeUp event on Slave if the pending
WakeUp event on Master was validated
BswM_EcuM_CurrentState(
ECUM_STATE_WAKEUP_VALIDATION);
BswM_EcuM_CurrentState(
ECUM_STATE_RUN);
If a wakeup event is validated during the
Wakeup Validation sequence, the EcuM's
state is set back to ECUM_STATE_RUN, also
sending a notification to BswM.
abstract of integration work
Integration work details
Without OS support
With OS, SchM support,
free configurable state
With OS support
With OS support, but scheduler locked

Figure 4.14. The `wakeup` states of a system with multi-core

NOTE

Synchronising wake-up between cores

Depending on the core on which the wake-up event occurs, the other cores should be notified to wake up, too. One way to achieve this is to ensure that the core that receives the wake-up event sets wake-up events for the other cores. This can be done immediately after the wake-up event occurs if no validation is needed or after the validation has taken place if validation is needed.

4.4. BswM Editor user's guide

4.4.1. Overview

The purpose of the BswM Editor user's guide is to introduce the new editor provided by Elektrobit (EB). This new editor is provided to simplify the configuration of the `BswM` module. In particular, the editor simplifies the configuration of `BswM` mode request ports, mode switch ports, and rules.

For this user's guide, it is assumed that you are already familiar with the `BswM` module itself. Information about the `BswM` module is provided in a separate user's guide in the EB tresos AutoCore Generic documentation.

- Section 4.4.2, “Background information” provides an overview of the concepts of the BswM Editor.
- Section 4.4.3, “Using the BswM Editor” provides instructions on how to use the BswM Editor.
- Section 4.4.4, “Other notes” provides other notes regarding the BswM Editor.

4.4.2. Background information

The BswM Editor provides a text-based view of the `BswM` mode request ports, mode switch ports, and rules. The text editor for the `BswM` configuration language provides:

- Syntax highlighting: keywords are highlighted showing correct syntax.
- Smart code completion: use the key combination **Ctrl**+**Space** to see valid options when editing.
- Error and warning markers: warning or error symbols are shown at the start of lines that need to be corrected or completed.

Also, an outline view is provided which shows the dependencies between `BswM` ports and the rules that reference them. With the help of the outline view, you can see the functionality of a complex configuration.

When you start the BswM Editor, the project's `BswM` ECU configuration, which is stored inside `BswM.xdm` file, is converted into a special configuration language and loaded into the editor. When you use the BswM Editor, all the changes are continuously saved internally. You can notice the *star* next to the title of the **BswM Editor** tab.

If you close the **BswM Editor** tab, a confirmation window appears and you can save the changes to the BswM configuration. If you select **Yes**, the changes that are performed in the editor are converted and stored within `BswM.xdm`.

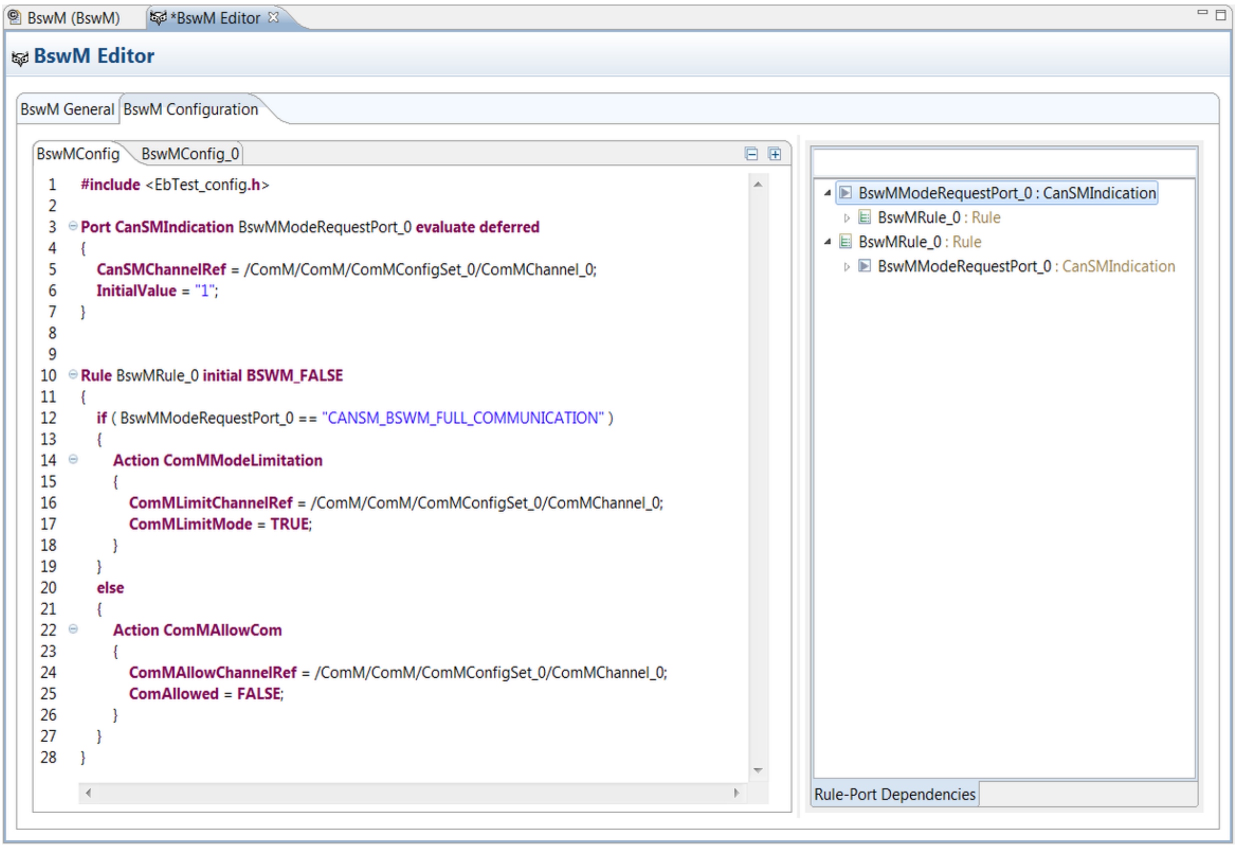
4.4.3. Using the BswM Editor

- Section 4.4.3.1, "Starting the BswM Editor" describes basic steps on how to use the editor.
- Section 4.4.3.2, "Creating a new mode request port" provides an example on how to create a new mode request port.
- Section 4.4.3.3, "Creating a new mode switch port" provides an example on how to create a new mode switch port.
- Section 4.4.3.5, "Creating a new rule" provides an example on how to create a new rule.

4.4.3.1. Starting the BswM Editor

To use the BswM Editor, add the `BswM` module to your project configuration and proceed as follows:

1. Open the **Generic Editor** of the BswM module. This loads the current configuration for the `BswM` module.
2. Switch to the **Configuration** tab and add a configuration. Multiple configurations are supported.
3. Address any errors that you have in the current configuration. You can only start the BswM Editor if the existing configuration is valid.
4. Save the `BswM` module configuration.
5. Open the BswM Editor of the `BswM` module.
6. Configure the options on the **General** tab according to your project requirements.
7. Switch to your **BswM Configuration** tab.
8. Use this context-sensitive editor to create mode requests, switch ports, and rules. Figure 4.15, "BswM Editor" shows the editor with an example mode request. Further examples are given in later sections.



```
#include <EbTest_config.h>

Port CanSMIndication BswMModeRequestPort_0 evaluate deferred
{
   CanSMChannelRef = /ComM/ComM/ComMConfigSet_0/ComMChannel_0;
   InitialValue = "1";
}


Rule BswMRule_0 initial BSWM_FALSE
{
   if ( BswMModeRequestPort_0 == "CANSM_BSWM_FULL_COMMUNICATION" )
   {
      Action ComMModeLimitation
      {
         ComMLimitChannelRef = /ComM/ComM/ComMConfigSet_0/ComMChannel_0;
         ComMLimitMode = TRUE;
      }
   }
   else
   {
      Action ComMAllowCom
      {
         ComMAllowChannelRef = /ComM/ComM/ComMConfigSet_0/ComMChannel_0;
         ComAllowed = FALSE;
      }
   }
}
```

Figure 4.15. BswM Editor

9. Click the **Close** button on the **BswM Editor** tab *BswM Editor ☒ to end the editing session.
10. Save your changes when prompted.
11. Save the `BswM` module configuration to permanently store your changes in the `BswM.xdm` file.

4.4.3.2. Creating a new mode request port

You can define the following mode request ports at any point within the editor pane:

- `BswMBswModeNotification`
- `BswMCanSMIndication`
- `BswMComMIndication`
- `BswMComPncRequest`
- `BswMDcmComModeCurrentState`
- `BswMDcmComModeRequest`
- `BswMDcmResetModeRequest`

- `BswMDcmSessionModeRequest`
- `BswMEcuMIndication`
- `BswMEcuMWakeupSource`
- `BswMEthIfPortGroupLinkStateChg`
- `BswMEthSMIndication`
- `BswMFrSMIndication`
- `BswMGenericRequest`
- `BswMLinScheduleIndication`
- `BswMLinSMIndication`
- `BswMLinTpModeRequest`
- `BswMNvMJobModeIndication`
- `BswMNvMRequest`
- `BswMRteSwitchAckNotification`
- `BswMSdClientServiceCurrentState`
- `BswMSdConsumedEventGroupCurrentState`
- `BswMSdEventHandlerCurrentState`
- `BswMSwcModeNotification`
- `BswMSwcModeRequest`
- `BswMTimer`

The following code defines a new `BswMGenericRequest` mode request port:

```
Port GenericRequest BswM_ReqPort evaluate deferred
{
  ModeRequesterId = 0;
  RequestedModeMax = 5;
  InitialValue = "BSWM_STARTUP_TWO_A";
}
```

The definition of this mode request port begins with the keyword `GenericRequest`. The user-defined name `BswM_ReqPort` follows. After the keyword `evaluate`, put either `deferred` or `immediate`. Use `deferred` if the value of the mode request port should be evaluated when the `BswM` main function is executed. Use `immediate` if the value of the mode request port should be immediately evaluated when the port is updated.

The following syntax diagram shows an example of the possible options for a generic request. Keywords and syntax are shown in clear boxes while objects that you provide are highlighted in gray.

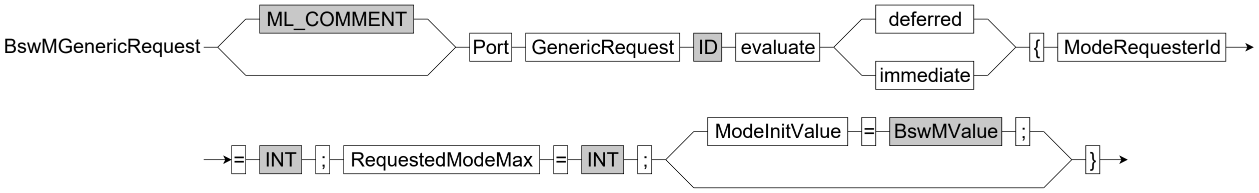


Figure 4.16. BswMGenericRequest

A generic mode request port has the following attributes:

- `ModeRequesterId`: This attribute corresponds to the configuration parameter `BswMModeRequesterId`.
- `RequestedModeMax`: This attribute corresponds to the configuration parameter `BswMRequestedModeMax`.
- `InitialValue`: This attribute corresponds to the configuration parameter `BswMModeInitValue`.

As you can see in the example above, the mode request port data structure keywords correspond almost completely to the ECU configuration parameter names. The prefix `BswM` has been removed though.

4.4.3.3. Creating a new mode switch port

The following example defines a `BswMSwitchPort`:

```
SwitchPort BswMSwitchPort_Rte
{
  SwitchInterfaceRef = /BswMMode/PortInterfaces/BswMMode;
  DataMapping = /BswMMode/DataTypeMappingSets/BswMModeMapping;
  ModeSwitchQueueLength = 1;
}
```

The definition of a switch port begins with the keyword `SwitchPort` followed by the user-defined name. The following attributes must be defined for each switch port:

- `SwitchInterfaceRef`: This is the path to the mode switch interface to be used.
- `DataMapping`: This is the path to the data type mapping for the mode declaration group used within the referenced mode switch interface.
- `ModeSwitchQueueLength`: The number of mode switches that can be queued.

Switch ports are referenced to by `RteSwitch` and `SchMSwitch` actions, which are defined within rule action lists.

The following syntax diagram shows an example of the possible options for a switch port. Keywords and syntax are shown in clear boxes while objects that you provide are highlighted in gray.

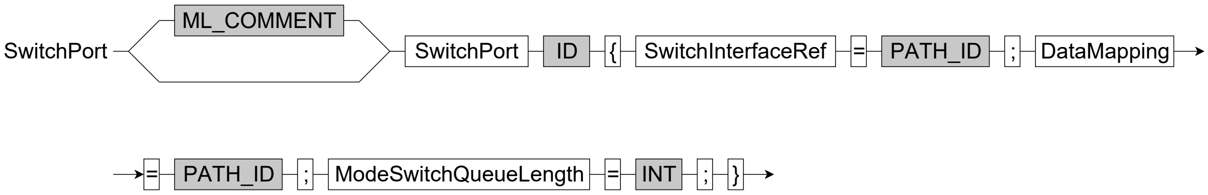


Figure 4.17. SwitchPort

4.4.3.4. Creating a new send port

The following example defines a `BswMSendPort`:

```
SendPort BswMSendPort_A
{
  VariableDataPrototypeRef = /CounterDemo/IfCounter/CounterValue;
}
```

The definition of a send port begins with the keyword `SendPort` followed by the user-defined name of the send port. The following attribute must be defined for each send port:

- `VariableDataPrototypeRef`: This is the path to the variable data prototype to be used.

The following syntax diagram shows an example of the possible options for a send port. Keywords and syntax are shown in clear boxes while objects that you provide are highlighted in gray.

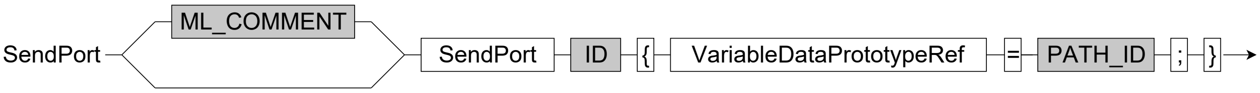


Figure 4.18. SendPort

4.4.3.5. Creating a new rule

In the example below, a rule is defined that executes a triggered action list if the mode request port `BswM_ReqPort` is set to `BSWM_STARTUP_TWO_A`:

```
Rule BswMRule_0 initial BSWM_UNDEFINED
{
  if triggered ( BswM_ReqPort == "BSWM_STARTUP_TWO_A" )
  {
    Action UserCallout
    {
      UserCalloutFunction = "BswM_OnStartupTwoA()";
    }
    Action RteSwitch
    {
      RteSwitchPortRef = BswMSwitchPort_Rte;
      SwitchedMode =
         /BswMMode/ModeDeclarationGroups/BswMModeGroup/BSWM_STARTUP_TWO_A;
    }
  }
  else triggered
  {
    Action UserCallout
    {
      UserCalloutFunction = "MyCalloutFunction()";
    }
  }
}
```

A rule begins with the keyword Rule. The user-defined name follows. The keyword initial is followed by either BSWM_TRUE, BSWM_FALSE or BSWM_UNDEFINED. This defines how the BswM rule's triggered action list(s) shall be executed when the rule is first evaluated. If the rule does not have a triggered action list, the initial value does not have any influence on the rule's behavior.

A rule must have an if-statement. The if-statement begins with the keyword if. If the optional keyword triggered is added, the true action list is executed (i.e. the rule's result must change before the true action list can be executed again). The rule's logical expression follows. In this example, a primitive expression is defined, but complex expressions with logical *and*, *or*, *xor*, and *nand* can be used. Within the body of the if-statement, the true action list is defined. An action list can execute multiple actions, (nested) action lists or (nested) rules. In this example, a user-callout action is executed, followed by an Rte switch action.

The else-statement is optional. This element defines the false action list. Like the true action list, the optional keyword triggered can be defined after the keyword else and before the statement body.

The following syntax diagram shows an example of the possible options for a rule. Keywords and syntax are shown in clear boxes while objects to be provided by the user are highlighted in gray.

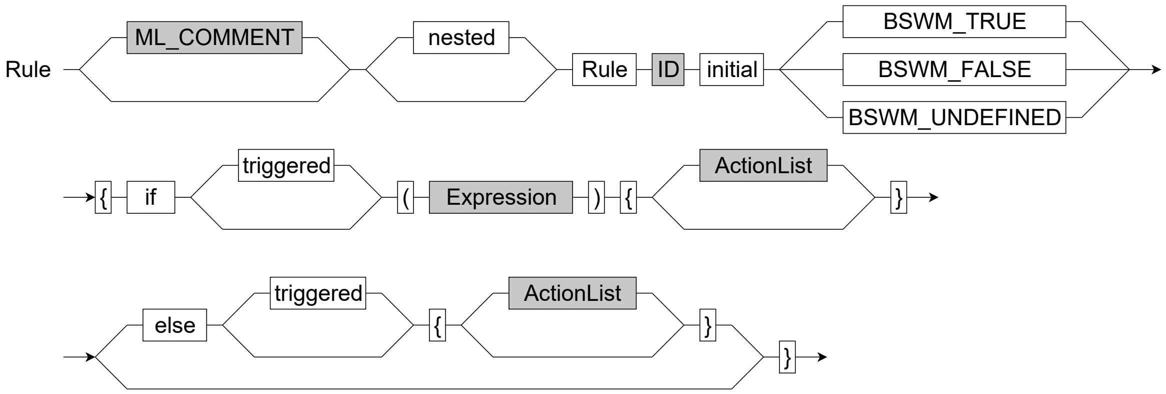


Figure 4.19. Rule

4.4.4. Other notes

In order to simplify the understanding of the BswM configuration, the names of logical expressions, mode conditions, actions, and action lists are not used at all. As a result, the previous names of all logical expressions, mode conditions, actions, and action lists are replaced if the editor is closed and the configuration is saved back to the project's `BswM.xdm`.

The replacement starts with the name of the container followed by a specific infix. For logical expressions, the container names are followed by an infix related to the rule which the logical expression references (ie. `BswMLogicalExpression_{Rule}`)

For mode conditions, the container names are followed by an infix related to the mode request port it references and also the mode value (ie. `BswMModeCondition_{Port}_{Value}`).

For action lists, the container names are followed by an infix related to the rule that references them and also followed by an infix which specifies if it is a true or false action list (e. `BswMActionList_{Rule}_TrueActionList`)

For actions, the container names are followed by the action type (e. `BswMAction_{ActionType}`).

All replacements are ending with the index of the respective item (e.g. `BswMLogicalExpression_MyRule_0`).

The names of rules and mode request ports remain the same.

The BswM Editor also performs some optimizations in order to reduce the size of the configuration stored in the `BswM.xdm` file.

For example, if an action or mode condition is used with the same structure in two different rules in the BswM Editor then only one container will be created for that action or mode condition.

This optimization does not apply to action lists, and logical expressions. Even if two action lists have the same structure used in two different rules, two different containers will be created. This is done in order to keep the configuration intuitive.

In order to keep the configuration valid, the BswM Editor automatically adjusts the values of the `BswMMaxNumRules` and `BswMMaxNumActionLists` parameters. This adjustments might happen if new rules are added or the optimization process changes the number of action lists.

As BswM Generic Editor is more flexible and it is easy to create configurations that are invalid, BswM Editor grammar contains some workarounds to open incomplete configurations and save as many cases as possible. The only case that BswM Editor cannot save is when configuration contains Rules, ModeRequestPorts, SendPorts or SwitchPorts that do not have a name. In that case BswM Editor will open and show the configuration, but it will not be able to save it until the names will be provided. In order to indicate to the user that his action is required the place that should contain name will be marked with `!!no-name!!` sequence. The following code shows ModeRequestPort that do not have a name:

```
Port GenericRequest !!no-name!! evaluate deferred{
    ModeRequesterId=0;
    InitialValue="0023";
}
```

The user should replace such indicator with a name in order to save the configuration.

Please be aware that if you remove item's name in Generic Editor and if the item is referenced by other items, the relations between the items will be lost and in order to repair the configuration you will need to manually update names and also proper references in BswM Editor.

4.5. Configuring the ComM module

4.5.1. Configuring ComM for BSW distribution

In this section, you configure the `ComM` for BSW distribution. For more information on the `ComM` BSW distribution, see Section 4.2.3.2, “BSW distribution”.

Configuring ComM for BSW distribution

Prerequisite:

- All integrated modules are compatible and have BSW distribution enabled.
- In the `Rte`, the following is configured:
  - The module instances, i.e. the `BswImplementation`s, are properly mapped to the cores via `OsApplication`s.
  - Events are properly mapped to tasks.
  - Provided and required client-server connections are properly connected.
  - Provided and required sender-receiver connections are properly connected.

Step 1
To enable the `ComM` BSW distribution support, enable the configuration parameter `ComMMultiCoreSupport`.

Step 2
To define the `ComM` master core, reference the corresponding `EcuCPartition` in configuration parameter `ComMMasterCoreEcuCPartitionRef`.

Step 3
For each asynchronous interface between `ComM` and `BusSm`, connect the provided and required client-server entries for each channel. Based on this information, the `Rte` generates the `SchM_Call()` operations that are used for inter-core communication.

Step 4
For each asynchronous interface between `ComM` and `Dcm`, connect the provided and required client-server entries for each channel. Based on this information, the `Rte` generates the `SchM_Call()` operations that are used for inter-core communication.

4.6. Configuring the Nm module

4.6.1. Configuring Nm for BSW distribution

In this section, you configure the `Nm` for BSW distribution. For more information on the `Nm` BSW distribution, see Section 4.2.4.3, “BSW distribution”.

Configuring Nm for BSW distribution

Prerequisite:

- All BusNms modules that are used in the project have multi-core support enabled.
- In the Rte, the following is configured:
  - The module instances, i.e. the BswImplementations, are properly mapped to the cores via OsApplications.
  - Events are properly mapped to tasks.
  - Provided and required client-server connections are properly connected.
  - Provided and required sender-receiver connections are properly connected.

Step 1
To enable the Nm BSW distribution support, enable the configuration parameter NmMultiCoreSupport.

Step 2
For each asynchronous interface between Nm and BusNm, connect the provided and required client-server entries for each channel. Based on this information, the Rte generates the SchM_Call() operations that are used for inter-core communication.

5. ACG8 Mode Management module references

5.1. Overview

This chapter provides module references for the ACG8 Mode Management product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 Mode Management user's guide.

5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter BswMCompuConstText of the BswM module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function cxpath:getCompuMethodsVT() in the range field which provides the allowed values.

5.2. BswM

5.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
BswMConfig	1..n	This container contains the configuration parameters and sub containers of the AUTOSAR BswM module.
BswMGeneral	1..1	General configuration parameters of the Basic SW Mode Manager.
ReportToDem	1..1	**Label:** Production error handling Production error handling
BswMDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1
POST_BUILD_VARIANT_USED	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT
Label	Config Variant

Description	Specifies the variant of the BswM used in the current project. If EcuM module from EB is used it is mandatory to have the PbcfgM module present in the project too.
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile
Range	VariantPreCompile VariantPostBuild
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild

Parameter Name	**POST_BUILD_VARIANT_USED**
Label	Post Build Variant Used
Description	This parameter can be used to bypass the postbuild selectable variant awareness of BswM. If set to FALSE, BswM will not be variant aware (default behaviour). If set to TRUE, BswM will be variant aware and will produce variant dependent artifacts (ie. different .xgen and PBcfg files for each variant). This parameter is useful for the situations where postbuild selectable variants exist in the project but the BswM doesn't have any variant specific configuration.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild

5.2.1.1. BswMConfig

Containers included		
Container name	**Multiplicity**	**Description**
BswMArbitration	1..1	This container includes all configuration sub- containers and parameters related to the mode arbitration functionality of the BswM. The arbitration functionality detects changes in the request- and indication-states of available sources and computes if, and if yes which actions the BswM shall perform as reaction.

Containers included		
BswMDataTypeMappingSets	0..1	Collection of references to DataTypeMappingSet.
BswMModeControl	1..1	This container includes all configuration sub-containers and parameters related to the mode control functionality of the BswM.

Parameters included	
Parameter name	**Multiplicity**
BswMPartitionRef	0..1

Parameter Name	BswMPartitionRef	
Description	This references the partition the BswM shall run inside. On a single-core ECU it is not required to configure this parameter. In case of a multi-core ECU it is mandatory to configure this parameter even if there is only one BswM instance.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	Link:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.2. BswMArbitration

Containers included		
Container name	**Multiplicity**	**Description**
BswMLogicalExpression	0..n	This container describes the logical expressions that can be used for the mode arbitration. The logical expressions are built of a set of arguments and a logical operator. Each argument can either be a mode condition or a sub-expression to allow definition of more complex logical expressions. The AND and OR operator accept an unlimited number of arguments; NAND and XOR are not associative, so only two argument are allowed.

Containers included		
		Note that the order of evaluation of the expressions is not defined.
BswMModeCondition	0..n	This container describes the BswM mode conditions that can be used either by itself to form a rule or as a part of a logical expression. A mode condition compares a mode request port state with some predefined value and results in a boolean value.
BswMModeRequestPort	0..n	Each instance of this container defines a mode request interface that is used to requests or indicate modes from/to the BswM. These interfaces are implemented as ports or as ordinary C-functions based upon if the request is made by an SW-C or a BSW module. *There are different types of mode requests:* ▶ Mode requests from the SW-C:s ▶ Mode Requests from other BSW modules such as the DCM. ▶ State/mode indications from the RTE or other BSW modules such as the bus specific State Managers *Note : BswM treats all request and indications in the exact same way.*
BswMRule	0..n	Each instance of this container describes a BswM arbitration rule. The rule either consists of a simple mode condition or a more complex logical expression which in turn calculates a boolean value from multiple mode conditions. This container also references the action lists that shall be invoked when the rule is evaluated to `True` or `False`.

5.2.1.3. BswMLogicalExpression

Parameters included	
Parameter name	**Multiplicity**
BswMLogicalOperator	0..1

Parameters included	
BswMArgumentRef	1..n

Parameter Name	BswMLogicalOperator	
Description	This parameter specifies the logical operator to be used in the logical expression. If the expression only consists of a single condition this parameter shall not be used.	
Multiplicity	0..1	
Type	ENUMERATION	
Range	BSWM_AND BSWM_NAND BSWM_OR BSWM_XOR	
Configuration class	**PreCompile:**	VariantPreCompile
	Link:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	BswMArgumentRef	
Description	This is a choice reference either to a mode condition or a sub-expression.	
Multiplicity	1..n	
Type	CHOICE-REFERENCE	

Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMArbitration/BswMModeCondition/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMArbitration/BswMLogicalExpression/*[(asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())) and (node:name(.) != node:name(node:current()/../..))])	

Configuration class	**PreCompile:**	VariantPreCompile
	Link:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.4. BswMModeCondition

Containers included		
Container name	**Multiplicity**	**Description**
BswMConditionValue	1..1	This container holds the parameters and references necessary to identify the mode type and the value that the mode request is compared to.

Parameters included	
Parameter name	**Multiplicity**
BswMConditionType	1..1
BswMConditionMode	1..1

Parameter Name	BswMConditionType
Multiplicity	1..1
Type	ENUMERATION
Range	BSWM_EQUALS BSWM_EQUALS_NOT
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	BswMConditionMode
Description	This parameter references the mode request port that is used for the condition.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMArbitration/BswMModeRequestPort/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.5. BswMConditionValue

Containers included		
Container name	**Multiplicity**	**Description**
BswMBswMode	1..1	This container defines the value and type of a mode in the BSW.
BswMCompuScaleModeValue	1..1	This container contains parameters used to define a mode value.
BswMModeDeclaration	1..1	When the mode corresponds to a mode request or mode indication interface the mode is defined by a mode declaration. The mode declarations are defined in the SW-C Template and hence a foreign reference to the corresponding Mode Declaration is used.

5.2.1.6. BswMBswMode

Parameters included	
Parameter name	**Multiplicity**
BswMBswRequestedMode	1..1

Parameter Name	**BswMBswRequestedMode**	
Description	This parameter contains the symbolic name (as a string) of a certain mode/state that can be requested/indicated by the BSW modules.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.7. BswMCompuScaleModeValue

Parameters included	
Parameter name	**Multiplicity**
BswMCompuConstText	1..1
BswMCompuMethodRef	1..1

Parameter Name	**BswMCompuConstText**
Description	The value of this parameter shall match the VT member of a CompuConst defined within the referenced CompuMethod (BswMCompuMethodRef). The interval value of the corresponding CompuScale shall be used as the mode request value.
Multiplicity	1..1
Type	STRING
Range	asc_bswm:getCompuMethodsVT(../BswMCompuMethodRef)
Configuration class	**VariantPostBuild:** VariantPostBuild **VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMCompuMethodRef**
Description	This is a foreign reference to the CompuMethod used for mode requests. The CompuMethod category shall be TEXTTABLE.
Multiplicity	1..1
Type	FOREIGN-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild **VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.8. BswMModeDeclaration

Parameters included	
Parameter name	**Multiplicity**
BswMModeValueRef	1..1

Parameter Name	**BswMModeValueRef**
Description	This is a foreign reference to the Mode Declaration used for the mode requests corresponding to this condition.
Multiplicity	1..1
Type	FOREIGN-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild

Origin	AUTOSAR_ECUC

5.2.1.9. BswMModeRequestPort

Containers included		
Container name	**Multiplicity**	**Description**
BswMModeInitValue	0..1	This container defines the initial mode value that is used by BswM for the corresponding mode request after initialization. This container is optional and shall only be used for Mode Requests that do not already have an initial value.
BswMModeRequestSource	1..1	This choice container specifies the source of the mode request or state/mode indication. The requester of a mode can be both SW-C:s and other BSW Modules, such as the bus specific State Managers.

Parameters included	
Parameter name	**Multiplicity**
BswMRequestProcessing	1..1

Parameter Name	**BswMRequestProcessing**	
Description	This parameter defines if the processing of the mode arbitration shall be done immediately when a mode request is received or if it shall be deferred to the processing of the main function of BswM.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	BSWM_DEFERRED BSWM_IMMEDIATE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.10. BswMModeInitValue

Containers included		
Container name	**Multiplicity**	**Description**

Containers included		
BswMBswModeInitValue	1..1	This is the choice for a initial mode value used for the initialization of mode requests in case the request is made by a BSW module.
BswMCompuScaleModeValue	1..1	This container contains parameters used to define a mode value.
BswMSwcModeInitValue	1..1	This is the choice for a foreign reference to the Mode Declaration used for the initialization of mode requests in case the request is made by a SW-C.

5.2.1.11. BswMBswModeInitValue

Parameters included	
Parameter name	**Multiplicity**
BswMBswModeInitValueMode	1..1

Parameter Name	BswMBswModeInitValueMode	
Description	This parameter defines the initial mode value that is used by BswM for the corresponding mode request after initialization.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.12. BswMCompuScaleModeValue

Parameters included	
Parameter name	**Multiplicity**
BswMCompuConstText	1..1
BswMCompuMethodRef	1..1

Parameter Name	BswMCompuConstText

Description	The value of this parameter shall match the VT member of a CompuConst defined within the referenced CompuMethod (BswMCompuMethodRef). The interval value of the corresponding CompuScale shall be used as the mode request value.	
Multiplicity	1..1	
Type	STRING	
Range	asc_bswm:getCompuMethodsVT(../BswMCompuMethodRef)	
Configuration class	**VariantPostBuild:**	VariantPostBuild
	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**BswMCompuMethodRef**	
Description	This is a foreign reference to the CompuMethod used for mode requests. The CompuMethod category shall be TEXTTABLE.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
	VariantPreCompile:	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.13. BswMSwcModeInitValue

Parameters included	
Parameter name	**Multiplicity**
BswMSwcModeInitValueRef	1..1

Parameter Name	**BswMSwcModeInitValueRef**	
Description	This is a foreign reference to the Mode Declaration used for the initialization of mode requests.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild

Origin	AUTOSAR_ECUC

5.2.1.14. BswMModeRequestSource

Containers included		
Container name	**Multiplicity**	**Description**
BswMBswModeNotification	1..1	The source of the mode request is a Bsw Module.
BswMCanSMIndication	1..1	This is an indication of the current state of the CAN State Manager.
BswMComMIndication	1..1	This is an indication of the current communication mode of a channel in the Communication Manager.
BswMComMPncRequest	1..1	This is a request of the current communication mode of a Partial Network Cluster in the Communication Manager.
BswMDcmApplicationUpdatedIndication	1..1	This is a request to update application data from the DCM. This container does not contain any parameters since there are no further configuration needed for this type of request.
BswMDcmComModeCurrentState	1..1	The source of the mode request is the Diagnostic Communication Manager.
BswMDcmComModeRequest	1..1	The source of the mode request is the Diagnostic Communication Manager.
BswMDcmResetModeRequest	1..1	This is a reset mode request from the DCM. This container does not contain any parameters since there are no further configuration needed for this type of request.
BswMDcmSessionModeRequest	1..1	This is a session mode request from the DCM. This container does not contain any parameters since there are no further configuration needed for this type of request.
BswMEcuMIndication	1..1	This is a notification of the current operation mode of the ECU State Manager. This container does not contain any parameters since there are no further configuration needed for this type of request.
BswMEcuMWakeupSource	1..1	This is a notification of the current state of an ECU State Manager wakeup source.
BswMEthIfPortGroupLinkStateChg	1..1	This is an indication from the EthIf if the link state of a Ethernet interface switch port group has changed.

Containers included		
BswMEthSMIndication	1..1	This is an indication of the current state of the Ethernet State Manager.
BswMFrSMIndication	1..1	This is an indication of the current state of the FlexRay State Manager.
BswMGenericRequest	1..1	This mode request originates from a requester that is not among the list of standardized mode requesters (i.e. the different resource managers).
BswMJ1939DcmBroadcastStatus	1..1	This is a notification of the desired broadcast status per network, triggered via DM13. Note that these ports can only have values 0 or 1 due to how the BswM_J1939DcmBroadcastStatus transmits information (via a single 16 bit parameter where each bit represent a ComM channel). This imposes further restrictions regarding mode conditions or initial values.
BswMJ1939NmIndication	1..1	This is an indication of the current state of the J1939 network management module.
BswMLinScheduleIndication	1..1	This is an indication of the currently active LIN Schedule Table for a specific LIN Interface.
BswMLinSMIndication	1..1	This is an indication of the current state of the LIN State Manager.
BswMLinTpModeRequest	1..1	This is a LinTp mode request from the LinIf. This port corresponds to a call of the `BswM_LinTp_RequestMode` API.
BswMNmIfCarWakeUpIndication	1..1	This is an indication of a CarWakeup from the NmIf.
BswMNvMJobModeIndication	1..1	Indicates the current status of the multiblock job. The job is identified via BswMNvmService, e.g. 0x0c for NvmReadAll, 0x0d for NvmWriteAll. Possible Values are:
BswMNvMRequest	1..1	This is a mode request for a NvM Block.
BswMRteSwitchAckNotification	1..1	This is a notification from the RTE that a mode transition has been completed.
BswMSdClientServiceCurrentState	1..1	Used by Service Discovery module to indicate current state of the Client Service (available/down).
BswMSdConsumedEventGroupCurrentState	1..1	Used by Service Discovery to indicate current status of the EventHandler (requested/released).
BswMSdEventHandlerCurrentState	1..1	Used by Service Discovery to indicate current status of the EventHandler (requested/released).

Containers included		
BswMSwcModeNotification	1..1	This is a mode switch notification associated with a RTE switch interface.
BswMSwcModeRequest	1..1	The source of the mode request is a SW Component.
BswMTimer	1..1	This is a timer which can be used for time dependent rules. In order to start/stop this timer, a BswMTimerControl action with BswMTimerAction = BSWM_TIMER_START/BSWM_-TIMER_STOP must reference it. This mode request port can be used normally in mode conditions. Note that the initial value of this mode request port is BSWM_TIMER_STOPPED and is not configurable. Mode conditions referencing this mode request port shall only use BSWM_TIMER_STOPPED, BSWM_TIMER_STARTED or BSWM_TIMER_EXPIRED as mode values.
BswMWdgMRequestPartitionReset	1..1	This is a Partition Reset request from from the WdgM. This port corresponds to a call of the `BswM_WdgM_RequestPartitionReset` API.

5.2.1.15. BswMBswModeNotification

Parameters included	
Parameter name	**Multiplicity**
BswMBswModeDeclarationGroupPrototypeRef	1..1

Parameter Name	BswMBswModeDeclarationGroupPrototypeRef	
Description	This is a foreign reference to the Mode Declaration Group Prototype.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.16. BswMCanSMIndication

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
BswMCanSMChannelRef	1..1

Parameter Name	BswMCanSMChannelRef
Description	This is a reference to the CanSM channel handle that the mode request corresponds to.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.17. BswMComMIndication

Parameters included	
Parameter name	**Multiplicity**
BswMComMChannelRef	1..1

Parameter Name	BswMComMChannelRef
Description	This is a reference to the Communication Manager channel handle that the indication corresponds to.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.18. BswMComMPncRequest

Parameters included	
Parameter name	**Multiplicity**
BswMComMPncRef	1..1

Parameter Name	BswMComMPncRef

Description	This is a reference to the Communication Manager PNC handle of the Partial Network Cluster that the request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.19. BswMDcmApplicationUpdatedIndication

5.2.1.20. BswMDcmComModeCurrentState

Parameters included	
Parameter name	**Multiplicity**
BswMDcmComMChannelRef	1..1

Parameter Name	**BswMDcmComMChannelRef**	
Description	This is a reference to the Communication Manager channel handle that the indication corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.21. BswMDcmComModeRequest

Parameters included	
Parameter name	**Multiplicity**
BswMDcmComMChannelRef	1..1
BswMDcmComMNetwork	1..1

Parameter Name	**BswMDcmComMChannelRef**
Description	This is a reference to the Communication Manager channel handle that the indication corresponds to.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMDcmComMNetwork**
Description	This configuration parameter is not used. The reference to the ComM channel shall be defined within the parameter BswMDcmComMChannelRef.
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.22. BswMDcmResetModeRequest

5.2.1.23. BswMDcmSessionModeRequest

5.2.1.24. BswMEcuMIndication

5.2.1.25. BswMEcuMWakeupSource

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
BswMEcuMWakeupSrcRef	1..1

Parameter Name	BswMEcuMWakeupSrcRef	
Description	This is a reference to the ECU State Manager Wakeup Source that the indication corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.26. BswMEthIfPortGroupLinkStateChg

Parameters included	
Parameter name	**Multiplicity**
BswMEthIfSwitchPortGroupRef	1..1

Parameter Name	BswMEthIfSwitchPortGroupRef	
Description	This is a reference to the Ethernet Interface Switch Port Group that the indication corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.27. BswMEthSMIndication

Parameters included	
Parameter name	**Multiplicity**
BswMEthSMChannelRef	1..1

Parameter Name	BswMEthSMChannelRef	
Description	This is a reference to the ComM channel respectively the EthSM channel that the mode request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.28. BswMFrSMIndication

Parameters included	
Parameter name	**Multiplicity**
BswMFrSMChannelRef	1..1

Parameter Name	BswMFrSMChannelRef	
Description	This is a reference to the FlexRay Cluster handle that the mode request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.29. BswMGenericRequest

Parameters included	
Parameter name	**Multiplicity**
BswMModeRequesterId	1..1
BswMRequestedModeMax	1..1

Parameter Name	BswMModeRequesterId

Description	This parameters identifies the different users of the generic mode request interface.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMRequestedModeMax**	
Description	This parameter is not used in the current BswM implementation. The upper limit for the requested modes is 65535. The BswM_ModeType is statically defined as uint16.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.30. BswMJ1939DcmBroadcastStatus

Parameters included	
Parameter name	**Multiplicity**
BswMJ1939DcmChannelRef	1..1

Parameter Name	**BswMJ1939DcmChannelRef**	
Description	Reference to the communication channel which is affected by this mode request.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.31. BswMJ1939NmIndication

Parameters included	
Parameter name	**Multiplicity**
BswMJ1939NmChannelRef	1..1
BswMJ1939NmNodeRef	1..1

Parameter Name	**BswMJ1939NmChannelRef**	
Description	This is a reference to the J1939Nm channel handle that the mode request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMJ1939NmNodeRef**	
Description	This is a reference to the node that the mode request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.32. BswMLinScheduleIndication

Parameters included	
Parameter name	**Multiplicity**
BswMLinScheduleRef	1..1

Parameter Name	**BswMLinScheduleRef**
Description	This is a reference to the LIN Schedule Table handle that the mode request corresponds to.
Multiplicity	1..1

Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.33. BswMLinSMIndication

Parameters included	
Parameter name	**Multiplicity**
BswMLinSMChannelRef	1..1

Parameter Name	**BswMLinSMChannelRef**	
Description	This is a reference to the LIN channel handle that the mode request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.34. BswMLinTpModeRequest

Parameters included	
Parameter name	**Multiplicity**
BswMLinTpChannelRef	1..1

Parameter Name	**BswMLinTpChannelRef**	
Description	This is a reference to the LIN Interface Channel that the mode request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile

	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.35. BswMNmIfCarWakeUpIndication

Parameters included	
Parameter name	**Multiplicity**
BswMNmChannelRef	1..1

Parameter Name	**BswMNmChannelRef**	
Description	This is a reference to the channel handle that the indication corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.36. BswMNvMJobModeIndication

Parameters included	
Parameter name	**Multiplicity**
BswMNvmService	1..1

Parameter Name	**BswMNvmService**	
Description	Identifies the Nvm job which is related to the mode request.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	NvmReadAll	
	NvmWriteAll	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild

Origin	AUTOSAR_ECUC

5.2.1.37. BswMNvMRequest

Parameters included	
Parameter name	**Multiplicity**
BswMNvMBlockRef	1..1

Parameter Name	**BswMNvMBlockRef**	
Description	This is a reference to the NvM Block Descriptor that the request corresponds to.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.38. BswMRteSwitchAckNotification

Parameters included	
Parameter name	**Multiplicity**
BswMSwitchPortRef	1..1

Parameter Name	**BswMSwitchPortRef**	
Description	References the switch port which will receive the notification.	
Multiplicity	1..1	
Type	REFERENCE	
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMSwitchPort/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.39. BswMSdClientServiceCurrentState

Parameters included	
Parameter name	**Multiplicity**
BswMSdClientMethodsRef	1..1

Parameter Name	**BswMSdClientMethodsRef**	
Description	This is a reference to a client service in the Sd module.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.40. BswMSdConsumedEventGroupCurrentState

Parameters included	
Parameter name	**Multiplicity**
BswMSdConsumedEventGroupRef	1..1

Parameter Name	**BswMSdConsumedEventGroupRef**	
Description	This is a reference to an eventGroup that is defined within a client service in the Sd module.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.41. BswMSdEventHandlerCurrentState

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
BswMSdEventHandlerRef	1..1

Parameter Name	BswMSdEventHandlerRef	
Description	This is a reference to an event handler that is defined within a server service in the Sd module.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.42. BswMSwcModeNotification

Parameters included	
Parameter name	**Multiplicity**
BswMSwcModeNotificationModeDeclarationGroupPrototypeRef	1..1

Parameter Name	BswMSwcModeNotificationModeDeclarationGroupPrototypeRef	
Description	This is a foreign reference to the ModeDeclarationGroupPrototype.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.43. BswMSwcModeRequest

Parameters included	
Parameter name	**Multiplicity**
BswMSwcModeRequestModeDeclarationGroupPrototypeRef	0..1
BswMSwcModeRequestVariableDataPrototypeRef	0..1

Parameter Name	BswMSwcModeRequestModeDeclarationGroupPrototypeRef

Description	This is a foreign reference to the ModeDeclarationGroupPrototype. As the mode request is SR-Communication the BswM shall provide a SR-Interface which corresponds to the ModeDeclarationGroupPrototype.
Multiplicity	0..1
Type	FOREIGN-REFERENCE
Configuration class	**PreCompile:** VariantPreCompile **Link:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMSwcModeRequestVariableDataPrototypeRef**
Description	This is a foreign reference to the VariableDataPrototype. It's the SWC's responsability to define a SR-Interface.
Multiplicity	0..1
Type	FOREIGN-REFERENCE
Configuration class	**PreCompile:** VariantPreCompile **Link:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.44. BswMTimer

5.2.1.45. BswMWdgMRequestPartitionReset

Parameters included	
Parameter name	**Multiplicity**
BswMWdgMOsApplicationRef	0..1
BswMWdgMRequestPartitionResetRef	1..1

Parameter Name	**BswMWdgMOsApplicationRef**
Description	This is a reference to the Os application that requests the reset. If this parameter is not enabled (eg. the default application is used), then the BswM will react to a call of BswM_WdgM_RequestPartitionReset only if the application id is 0 (default application).
Multiplicity	0..1

Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	Link:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**BswMWdgMRequestPartitionResetRef**	
Description	This is a reference to the partition that shall be reset. This parameter is not used. Instead, the BswMWdgMOsApplicationRef parameter can be used to specify the application for which the reset shall be done.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.46. BswMRule

Parameters included	
Parameter name	**Multiplicity**
BswMNestedExecutionOnly	1..1
BswMRuleInitState	1..1
BswMRuleExpressionRef	1..1
BswMRuleFalseActionList	0..1
BswMRuleTrueActionList	0..1

Parameter Name	**BswMNestedExecutionOnly**
Description	This parameter determines if the current rule is independent or subordinate. If BswMNestedExecutionOnly = FALSE (default), the current rule is considered independent and it will be evaluated either at the BswM_MainFunction (if its logical expressions reference deferred mode request ports) or as soon as the state of one of the referenced mode request ports changes (in case they are immediate). If BswMNestedExecutionOnly = TRUE, the current rule is considered subordinate and it will only evaluated as part of an action list execution.

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMRuleInitState**	
Description	This parameter is a part of the reset/initialization behavior of BswM. Action lists are executed when the result of a rule evaluation have changed since the last evaluation. This parameter defines the previous evaluation result of a rule to be used after initialization of the BswM. If this parameter is set to `BSWM_TRUE`, the evaluation result is treated as changed if the rule is evaluated to false. If this parameter is set to `BSWM_FALSE`, the evaluation result is treated as changed if the rule is evaluated to true. If this parameter is set to `BSWM_UNDEFINED`,the evaluation result is always treated as changed at the first evaluation of the rule after initialization.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	BSWM_FALSE	
	BSWM_TRUE	
	BSWM_UNDEFINED	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMRuleExpressionRef**
Description	This is a reference to the logical expression that is evaluated for each rule.
Multiplicity	1..1
Type	REFERENCE

Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMArbitration/BswMLogicalExpression/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMRuleFalseActionList**
Description	This is a reference to the action list that shall be executed when the rule is evaluated to `False`.
Multiplicity	0..1
Type	REFERENCE
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMActionList/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
Configuration class	**PreCompile:** VariantPreCompile **PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMRuleTrueActionList**
Description	This is a reference to the action list that shall be executed when the rule is evaluated to `True`.
Multiplicity	0..1
Type	REFERENCE
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMActionList/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
Configuration class	**PreCompile:** VariantPreCompile **PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.47. BswMDataTypeMappingSets

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
BswMDataTypeMappingSetRef	1..n

Parameter Name	BswMDataTypeMappingSetRef	
Description	Reference to DataTypeMappingSet.	
Multiplicity	1..n	
Type	FOREIGN-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	Link:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.48. BswMModeControl

Containers included		
Container name	**Multiplicity**	**Description**
BswMAction	0..n	Each container of this type defines an action. These actions can be part of one or several action lists.
BswMActionList	0..n	Each instance of this container defines an action list that is invoked based on the BswM Rules. An action list contains a list of numbered action items to be processed. An action list can also include other action lists.
BswMRteModeRequestPort	0..n	This container defines a mode request port which the BswM may utilize to send a mode request to a SW-C which is acting as a mode-manager. If this container is referenced by a BswMRteModeRequest, the BswM shall create a corresponding PPort in its service description.
BswMSwitchPort	0..n	This container specifies PPorts and/or providedModeDeclarationGroups , which the BswM has to create in its SWCD resp. BSWMD.

5.2.1.49. BswMAction

Containers included		
Container name	**Multiplicity**	**Description**

Containers included		
BswMAvailableActions	1..1	Choice container including the available actions to be used in the action lists.

5.2.1.50. BswMAvailableActions

Containers included		
Container name	Multiplicity	Description
BswMComMAllowCom	1..1	This container includes all parameters for the action to allow or to block communication for a ComM Channel. ComM_CommunicationAllowed is called when this action is configured.
BswMComMModeLimitation	1..1	This container includes all parameters related to a limitation of communication mode for a ComM Channel. ComM_LimitChannelToNoComMode is called when this action is configured.
BswMComMModeSwitch	1..1	This container includes all parameters related to a switch of communication mode for a ComM User. ComM_RequestComMode is called when this action is configured.
BswMDeadlineMonitoringControl	1..1	This container includes all parameters related to enabling and disabling of deadline monitoring for one or several PDUs in COM.
BswMEcuMGoDown	1..1	This container defines the UserId which shall be forwarded to the GoDown request.
BswMEcuMSelectShutdownTarget	1..1	This container defines the shutdown target.
BswMEthIfSwitchPortGroupRequestMode	1..1	This container includes all parameters related to requesting a mode for the EthIfSwtPortGroup. The EthIf_SwitchPortGroupRequestMode API is called when this action is executed.
BswMJ1939DcmStateSwitch	1..1	This container includes all parameters related to a switch of the J1939 Diagnostic Communication Managers network state for a J1939 node. J1939Dcm_SetState is called when this action is configured.
BswMJ1939RmStateSwitch	1..1	This container includes all parameters related to a switch of the J1939 Request Managers network state for a J1939 node. J1939Rm_SetState is called when this action is configured.

Containers included		
BswMLinScheduleSwitch	1..1	This container includes all parameters related to a switch of LIN schedule table. LinSM_ScheduleRequest is called when this action is configured.
BswMNMControl	1..1	This container includes all parameters related to enabling and disabling of Network Management communication. Disabling of NM communication can be requested by DCM.
BswMPduGroupSwitch	1..1	This container includes references to the PDU groups that shall be enabled and disabled. Com_IpduGroupControl is called when this action is configured.
BswMPduRouterControl	1..1	This container includes all parameters related to enabling and disabling of routing of Routing Path Groups in the PDU Router. PduR_EnableRouting or PduR_DisableRouting is called when this action is configured.
BswMRteModeRequest	1..1	This container defines a mode request that the BswM may send to a SW-C which is acting as a mode-manager. RTE_-Write is called when this action is configured.
BswMRteSwitch	1..1	This container defines a mode switch indication that the BswM provides to the SW-C that need to be notified about the mode switch. Rte_Switch is called when this action is configured.
BswMSchMSwitch	1..1	This container defines a mode switch indication that the BswM provides to the SW-C that need to be notified about the mode switch. SchM_Switch is called when this action is configured.
BswMSdClientServiceModeRequest	1..1	This container includes all parameters related to the selection of an client service of Sd. Sd_ClientServiceSetState is called when this action is configured.
BswMSdConsumedEventGroupModeRequest	1..1	This container includes all parameters related to the selection of a consumed EventGroup of Sd. Sd_ConsumedEventGroupSetState is called when this action is configured.
BswMSdServerServiceModeRequest	1..1	This container includes all parameters related to the selection of a server service of Sd. Sd_ServerServiceSetState is called when this action is configured.
BswMSwitchIPduMode	1..1	This container includes all parameters related to the selection of the transmission mode an I-PDU to be sent by COM. Com_SwitchIpduTxMode is called when this action is configured.

Containers included		
BswMTimerControl	1..1	This container includes all parameters for the action to start or to stop a timer.
BswMTriggerIPduSend	1..1	This container includes all parameters related to the triggering of an I-PDU to be sent by COM. Com_TriggerIPDUSend is called when this action is configured.
BswMUserCallout	1..1	This container includes all details needed for a user defined function call.

5.2.1.51. BswMComMAllowCom

Parameters included	
Parameter name	**Multiplicity**
BswMComAllowed	1..1
BswMComMAllowChannelRef	1..1

Parameter Name	BswMComAllowed	
Description	The parameter BswMComMAllowChannelRef refers to a channel which will allow or block communication using the function ComM_CommunicationAllowed()	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	BswMComMAllowChannelRef	
Description	This is a reference to the ComM Channel for which communication shall be allowed or blocked.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.52. BswMComMModeLimitation

Parameters included	
Parameter name	**Multiplicity**
BswMComMLimitMode	1..1
BswMComMLimitChannelRef	1..1

Parameter Name	BswMComMLimitMode	
Description	This parameter specifies if the channel referenced by the BswMComMLimitChannelRef shall be limited to NoCom (true) or if the limitation shall be lifted (false).	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	BswMComMLimitChannelRef	
Description	This is a reference to the ComM channel for which the communication mode should be limited.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.53. BswMComMModeSwitch

Parameters included	
Parameter name	**Multiplicity**
BswMComMRequestedMode	1..1
BswMComMUserRef	1..1

Parameter Name	BswMComMRequestedMode
Description	This parameter specifies if the requested communication mode.

Multiplicity	1..1	
Type	ENUMERATION	
Range	BSWM_FULL_COM BSWM_NO_COM	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMComMUserRef**	
Description	This is a reference to the ComM User that is associated to the Communication channel for which the communication mode should be requested.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.54. BswMDeadlineMonitoringControl

Parameters included	
Parameter name	**Multiplicity**
BswMDisabledDMPduGroupRef	0..n
BswMEnabledDMPduGroupRef	0..n

Parameter Name	**BswMDisabledDMPduGroupRef**	
Description	This is a reference to a PDU Group for which the Deadline Monitoring should be disabled.	
Multiplicity	0..n	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	PostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMEnabledDMPduGroupRef**

Description	This is a reference to a PDU Group for which the Deadline Monitoring should be enabled.	
Multiplicity	0..n	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	PostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.55. BswMEcuMGoDown

Parameters included	
Parameter name	**Multiplicity**
BswMEcuMUserIdRef	1..1

Parameter Name	**BswMEcuMUserIdRef**	
Description	This is a reference to a EcuM UserId.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.56. BswMEcuMSelectShutdownTarget

Parameters included	
Parameter name	**Multiplicity**
BswMEcuMShutdownTarget	1..1
BswMEcuMResetModeRef	0..1
BswMEcuMShutdownTargetRef	1..1
BswMEcuMSleepModeRef	0..1

Parameter Name	**BswMEcuMShutdownTarget**
Description	Selects the EcuM shutdown target type.

Multiplicity	1..1
Type	ENUMERATION
Range	ECUM_STATE_SLEEP ECUM_STATE_RESET ECUM_STATE_OFF
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMEcuMResetModeRef**
Description	This is a reference to a reset mode.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPreCompile **Link:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMEcuMShutdownTargetRef**
Description	This configuration parameter is not used. The configuration parameter BswMEcuMShutdownTargetRef has been renamed to BswMEcuMSleepModeRef and the parameter BswMEcuMResetModeRef has been introduced. Rationale: A BswMEcuMShutdownTarget may reference a SleepMode or a ResetMode.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMEcuMSleepModeRef**
Description	This is a reference to a sleep mode.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPreCompile **Link:** VariantPostBuild

Origin	AUTOSAR_ECUC

5.2.1.57. BswMEthIfSwitchPortGroupRequestMode

Parameters included	
Parameter name	**Multiplicity**
BswMEthTrcvMode	1..1
BswMEthIfSwitchPortGroupRef	1..1

Parameter Name	**BswMEthTrcvMode**	
Description	This parameter contains the mode which will be requested.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	BSWM_ETHTRCV_MODE_ACTIVE	
	BSWM_ETHTRCV_MODE_DOWN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMEthIfSwitchPortGroupRef**	
Description	This is a reference to the Ethernet Interface Switch Port Group which will receive the request.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.58. BswMJ1939DcmStateSwitch

Parameters included	
Parameter name	**Multiplicity**
BswMJ1939DcmRequestedState	1..1

Parameters included	
BswMJ1939DcmChannelRef	1..1
BswMJ1939DcmNodeRef	1..1

Parameter Name	**BswMJ1939DcmRequestedState**	
Description	This parameter describes the communication state of the J1939 Diagnostic Communication Manager and corresponds to the parameter "newState" of the function J1939Dcm_SetState.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	BSWM_J1939DCM_STATE_OFFLINE BSWM_J1939DCM_STATE_ONLINE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMJ1939DcmChannelRef**	
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId. This reference corresponds to the parameter "channel" of the function J1939Dcm_SetState.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMJ1939DcmNodeRef**	
Description	This reference points to a J1939NmNode and provides access to the unique J1939NmNodeId.This reference corresponds to the parameter "node" of the function J1939Dcm_SetState.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild

Origin	AUTOSAR_ECUC

5.2.1.59. BswMJ1939RmStateSwitch

Parameters included	
Parameter name	**Multiplicity**
BswMJ1939RmRequestedState	1..1
BswMJ1939RmChannelRef	1..1
BswMJ1939RmNodeRef	1..1

Parameter Name	**BswMJ1939RmRequestedState**
Description	This parameter describes the communication state of the J1939 Request Manager and corresponds to the parameter "new state" of the function J1939Rm_SetState.
Multiplicity	1..1
Type	ENUMERATION
Range	BSWM_J1939RM_STATE_OFFLINE BSWM_J1939RM_STATE_ONLINE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMJ1939RmChannelRef**
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId. This reference corresponds to the parameter "channel" of the function J1939Rm_SetState.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMJ1939RmNodeRef**

Description	This reference points to a J1939NmNode and provides access to the unique J1939NmNodeId. This reference corresponds to the parameter "node" of the function J1939Rm_SetState.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.60. BswMLinScheduleSwitch

Parameters included	
Parameter name	**Multiplicity**
BswMLinScheduleRef	1..1

Parameter Name	**BswMLinScheduleRef**
Description	This is a reference to the LIN schedule table that the LIN SM shall change to.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.61. BswMNMControl

Parameters included	
Parameter name	**Multiplicity**
BswMNMAction	1..1
BswMComMNetworkHandleRef	1..1

Parameter Name	**BswMNMAction**
Description	This parameter specifies if the communication of the corresponding NM channel should be enabled or disabled.
Multiplicity	1..1

Type	ENUMERATION
Range	BSWM_NM_DISABLE BSWM_NM_ENABLE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMComMNetworkHandleRef**
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.62. BswMPduGroupSwitch

Parameters included	
Parameter name	**Multiplicity**
BswMPduGroupSwitchReinit	1..1
BswMDisabledPduGroupRef	0..n
BswMEnabledPduGroupRef	0..n

Parameter Name	**BswMPduGroupSwitchReinit**
Description	This parameter defines if the values of timers, memorized last values etc. are retained or reinitialized during a PDU Group Switch.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	BswMDisabledPduGroupRef	
Description	This is a reference to a PDU Group that should be disabled.	
Multiplicity	0..n	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	PostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	BswMEnabledPduGroupRef	
Description	This is a reference to a PDU Group that should be enabled.	
Multiplicity	0..n	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	PostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.63. BswMPduRouterControl

Parameters included	
Parameter name	**Multiplicity**
BswPduRouterAction	1..1
BswMPduRoutingPathGroupRef	1..n

Parameter Name	BswPduRouterAction	
Description	This parameter specifies if the routing of the corresponding PDU should be enabled or disabled.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	BSWM_PDUR_DISABLE	
	BSWM_PDUR_ENABLE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	BswMPduRoutingPathGroupRef
Description	This is a reference to the PDU Routing Path Group for which the routing in the PDU Router should be enabled or disabled.
Multiplicity	1..n
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPreCompile **PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.64. BswMRteModeRequest

Parameters included	
Parameter name	**Multiplicity**
BswMRequestedModeRef	1..1
BswMRteModeRequestPortRef	1..1

Parameter Name	BswMRequestedModeRef
Description	This is a foreign reference to the Mode Declaration used for the mode request
Multiplicity	1..1
Type	FOREIGN-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild **VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	BswMRteModeRequestPortRef
Description	This is a reference to a BswMRteModeRequestPort.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMRteModeRequestPort/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
Configuration class	**VariantPostBuild:** VariantPostBuild **VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

5.2.1.65. BswMRteSwitch

Parameters included	
Parameter name	**Multiplicity**
BswMSwitchedMode	1..1
BswMRteSwitchPortRef	1..1

Parameter Name	**BswMSwitchedMode**	
Description	This parameter references a Mode Declaration of a Mode Declaration Group.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Range	asc_bswm:getCompatibleModeDeclarations(../BswMRteSwitchPortRef)	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMRteSwitchPortRef**	
Description	This is a reference to the BswMSwitchPort.	
Multiplicity	1..1	
Type	REFERENCE	
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMSwitchPort/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.66. BswMSchMSwitch

Parameters included	
Parameter name	**Multiplicity**
BswMSchMSwitchedMode	1..1
BswMSchMSwitchPortRef	1..1

Parameter Name	**BswMSchMSwitchedMode**
Description	This parameter contains the integer value that corresponds to a certain mode in a Mode Declaration Group.
Multiplicity	1..1
Type	FOREIGN-REFERENCE
Range	asc_bswm:getCompatibleModeDeclarations(../BswMSchMSwitchPortRef)
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMSchMSwitchPortRef**
Description	This is a reference to the BswMSwitchPort.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMSwitchPort/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.67. BswMSdClientServiceModeRequest

Parameters included	
Parameter name	**Multiplicity**
BswMSdClientMethodsRef	1..1
BswMSdClientServiceState	1..1

Parameter Name	**BswMSdClientMethodsRef**
Description	This is a reference to a client service in the Sd module.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile

	VariantPostBuild: VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMSdClientServiceState**
Description	This parameter specifies if the corresponding client service shall be released or requested.
Multiplicity	1..1
Type	ENUMERATION
Range	BSWM_SD_CLIENT_SERVICE_RELEASED BSWM_SD_CLIENT_SERVICE_REQUESTED
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.68. BswMSdConsumedEventGroupModeRequest

Parameters included	
Parameter name	**Multiplicity**
BswMSdConsumedEventGroupRef	1..1
BswMSdConsumedEventGroupState	1..1

Parameter Name	**BswMSdConsumedEventGroupRef**
Description	This is a reference to an eventGroup that is defined within a client service in the Sd module.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMSdConsumedEventGroupState**
Description	This parameter specifies if the corresponding consumed event group shall be released or requested.
Multiplicity	1..1

Type	ENUMERATION
Range	BSWM_SD_CONSUMED_EVENTGROUP_RELEASED BSWM_SD_CONSUMED_EVENTGROUP_REQUESTED
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.69. BswMSdServerServiceModeRequest

Parameters included	
Parameter name	**Multiplicity**
BswMSdServerMethodsRef	1..1
BswMSdServerServiceState	1..1

Parameter Name	**BswMSdServerMethodsRef**
Description	This is a reference to a server service in the Sd module.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMSdServerServiceState**
Description	This parameter specifies if the corresponding server service shall be down or available.
Multiplicity	1..1
Type	ENUMERATION
Range	BSWM_SD_SERVER_SERVICE_DOWN BSWM_SD_SERVER_SERVICE_AVAILABLE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.70. BswMSwitchIPduMode

Parameters included	
Parameter name	**Multiplicity**
BswMSwitchIPduModeValue	1..1
BswMSwitchIPduModeRef	1..1

Parameter Name	**BswMSwitchIPduModeValue**	
Description	This parameter defines which transmission mode shall be selected during this call.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMSwitchIPduModeRef**	
Description	This is a reference to an I-PDU for which the transmission mode shall be set.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.71. BswMTimerControl

Parameters included	
Parameter name	**Multiplicity**
BswMTimerAction	1..1
BswMTimerValue	1..1
BswMTimerRef	1..1

Parameter Name	**BswMTimerAction**
Description	Specify the action for the timer. The timer can be started or stopped.
Multiplicity	1..1

Type	ENUMERATION
Default value	BSWM_TIMER_START
Range	BSWM_TIMER_START BSWM_TIMER_STOP
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMTimerValue**
Description	Specify the timer value (in seconds) that is used when the timer is started. Note that this value must be a multiple of BswMMainFunctionPeriod.
Multiplicity	1..1
Type	FLOAT
Configuration class	**PreCompile:** VariantPreCompile **Link:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMTimerRef**
Description	Specify the Timer for which the timer action shall be executed.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMArbitration/BswMModeRequestPort/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current()) and BswMModeRequestSource = 'BswMTimer'])
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.72. BswMTriggerIPduSend

Parameters included	
Parameter name	**Multiplicity**
BswMTriggeredIPduRef	1..n

Parameter Name	BswMTriggeredIPduRef	
Description	This is a reference to an I-PDU that should be triggered for transmission.	
Multiplicity	1..n	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	PostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.73. BswMUserCallout

Parameters included	
Parameter name	**Multiplicity**
BswMUserCalloutFunction	1..1

Parameter Name	BswMUserCalloutFunction	
Description	Specifies a complete function call (including parameters) the BswM will make when this action is executed. A semicolon for ending the statement is not needed. Note that the return value of this function is completely ignored.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.74. BswMActionList

Containers included		
Container name	**Multiplicity**	**Description**
BswMActionListItem	1..n	This container defines an item in an action list.

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
BswMActionListExecution	1..1

Parameter Name	BswMActionListExecution
Description	This parameter controls if the corresponding action list shall be executed every time the rule is evaluated or only when the result of the evaluation changes. This parameter does not have an effect when this action list is executed within another action list. Possible values: ▶ BSWM_CONDITION: action list shall be executed every time the rule is evaluated ▶ BSWM_TRIGGER: action list shall be executed every time the result of the evaluation changes
Multiplicity	1..1
Type	ENUMERATION
Range	BSWM_CONDITION BSWM_TRIGGER
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.75. BswMActionListItem

Parameters included	
Parameter name	**Multiplicity**
BswMAbortOnFail	1..1
BswMActionListItemIndex	1..1
BswMActionListItemRef	1..1
BswMReportFailToDemRef	0..1

Parameter Name	BswMAbortOnFail
Description	This parameter defines if the execution of the action list shall be aborted if this specific action returns `E_NOT_OK`. *Note that this is only applicable for actions that have `E_NOT_OK` as a possible return value.*

Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMActionListItemIndex**
Description	This parameter defines the index of the action in the action list. It is used define in which order the actions shall be performed.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMActionListItemRef**
Description	The action item can either be an atomic action or a reference to another action list or rule.
Multiplicity	1..1
Type	CHOICE-REFERENCE

Range	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMAction/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])
	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMModeControl/BswMActionList/*[(asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())) and (node:name(.) != node:name(node:current()/../../..))])
	node:paths(as:modconf('BswM')[1]/BswMConfig/*/BswMArbitration/BswMRule/*[asc_bswm:getBswMConfig(.) = asc_bswm:getBswMConfig(node:current())])

Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	BswMReportFailToDemRef	
Description	If the reference is given, the DEM event shall be reported failed if this specific action returns E_NOT_OK; it shall be reported passed if this specific action returns E_OK.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
	PreCompile:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.76. BswMRteModeRequestPort

Containers included		
Container name	**Multiplicity**	**Description**
BswMRteModeRequestPortInterfaceRef	1..1	This is an instance reference to the variable data prototype used for the mode request.

Parameters included	
Parameter name	**Multiplicity**
BswMRteModeRequestPortInterfaceMappingRef	0..1

Parameter Name	BswMRteModeRequestPortInterfaceMappingRef	
Description	This is a foreign reference to the variable and parameter interface mapping used for the mode request.	
Multiplicity	0..1	
Type	FOREIGN-REFERENCE	
Configuration class	**Link:**	VariantPostBuild
	PreCompile:	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.77. BswMRteModeRequestPortInterfaceRef

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
TARGET	1..1
CONTEXT	0..n

Parameter Name	TARGET
Multiplicity	1..1
Type	REFERENCE
Origin	AUTOSAR_ECUC

Parameter Name	CONTEXT
Multiplicity	0..n
Type	REFERENCE
Range	SW-COMPONENT-PROTOTYPE* PORT-PROTOTYPE
Origin	AUTOSAR_ECUC

5.2.1.78. BswMSwitchPort

Parameters included	
Parameter name	Multiplicity
BswMModeSwitchInterfaceRef	1..1
BswMModeSwitchQueueLength	1..1

Parameter Name	BswMModeSwitchInterfaceRef	
Description	Reference to the ModeSwitchInterface of this BswMModeSwitchPort.	
Multiplicity	1..1	
Type	FOREIGN-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	BswMModeSwitchQueueLength
Description	This parameters defines the queue length of the mode switch port.
Multiplicity	1..1

Type	INTEGER	
Default value	1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.79. BswMGeneral

Containers included		
Container name	**Multiplicity**	**Description**
BswMQueuedArbitration	0..1	This optional container controls the queued mode arbitration feature which ensures that the processing of an immediate request of any type or the processing of the BswM_-MainFunction cannot be interrupted by other requests. This is achieved by queueing the requests which interrupt the processing of the existing immediate request or the processing of the BswM_MainFunction. The queued requests are executed in the order of their arrival after the processing they interrupted has been completed. If the container is disabled (by default), BswM's behaviour remains the same as before. If the container is enabled, the queued mode arbitration feature is enabled and can be finely tuned via the provided parameters.
BswMUserIncludeFiles	0..1	Collection of header file names which shall be included by the BswM.

Parameters included	
Parameter name	**Multiplicity**
BswMBinaryGenerationVerified	1..1
BswMCanSMEnabled	1..1
BswMComMEnabled	1..1
BswMDcmEnabled	1..1
BswMLegacyDcmEnabled	1..1
BswMDevErrorDetect	1..1
BswMEcuMEnabled	1..1
BswMEthIfEnabled	1..1

Parameters included	
BswMEthSMEnabled	1..1
BswMFrSMEnabled	1..1
BswMGenericRequestEnabled	1..1
BswMJ1939DcmEnabled	1..1
BswMJ1939NmEnabled	1..1
BswMLinSMEnabled	1..1
BswMLinTPEnabled	1..1
BswMLogicalExpressionsOptimization	1..1
BswMActionsOptimization	1..1
BswMMainFunctionPeriod	0..1
BswMMaxNumActionLists	1..1
BswMMaxNumRules	1..1
BswMNmEnabled	1..1
BswMNvMEnabled	1..1
BswMRelocatablePbcfgEnable	1..1
BswMRteUsage	1..1
BswMSchMEnabled	1..1
BswMSdEnabled	1..1
BswMVersionInfoApi	1..1
BswMWdgMEnabled	1..1

Parameter Name	**BswMBinaryGenerationVerified**
Description	If this parameter is TRUE and the selected variant is VariantPostBuild then symbolic names are not allowed as valid values for the BswMBswModeInitValueMode parameter.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMCanSMEnabled**

Description	Enable/disable CanSM module related BswM API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMComMEnabled**
Description	Enable/disable ComM module related BswM API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMDcmEnabled**
Description	Enable/disable Dcm module related BswM API: ▶ `BswM_Dcm_CommunicationMode_CurrentState` ▶ `BswM_Dcm_ApplicationUpdated (stub only)`
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMLegacyDcmEnabled**

Description	Enable/disable the 4.0.2 legacy Dcm related API: ▶ `BswM_Dcm_RequestCommunicationMode` ▶ `BswM_Dcm_RequestSessionMode` ▶ `BswM_Dcm_RequestResetMode` Note that a 4.0.2 compatible Dcm is needed in order for the BswM to produce compilable code.
Multiplicity	1..1
Type	BOOLEAN
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMDevErrorDetect**
Description	Switches the Development Error Detection and Notification ON or OFF. ▶ `true` : Enabled / ON ▶ `false`: Disabled / OFF
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMEcuMEnabled**
Description	Enables/Disables EcuM-related API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**BswMEthIfEnabled**
Description	Enables/Disables EthIf-related API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMEthSMEnabled**
Description	Enables/Disables EthSM-related API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMFrSMEnabled**
Description	Enables/Disables FrSM-related API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild

Origin	AUTOSAR_ECUC

Parameter Name	**BswMGenericRequestEnabled**
Description	Enables/Disables GenericRequest-related API `true` : Enabled `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMJ1939DcmEnabled**
Description	Enable/disable J1939Dcm module related BswM API: true: Enabled false: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMJ1939NmEnabled**
Description	Enable/disable J1939Nm module related BswM API. true: Enabled false: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMLinSMEnabled**
Description	Enables/Disables LinSM-related API

	▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMLinTPEnabled**
Description	Enables/Disables LinTP-related API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMLogicalExpressionsOptimization**
Description	This parameter controls the optimization of logical expressions w.r.t ROM consumption. If set to TRUE, BswM will only generate linktime artifacts for the expressions which are directly referenced by rules, ignoring unused or nested expressions. If set to FALSE, BswM will generate linktime artifacts for all expressions. This parameter is ignored and treated as TRUE if IMPLEMENTATION_CONFIG_VARIANT = VariantPreCompile. IMPORTANTE NOTE: If BswMLogicalExpressionsOptimization = TRUE, rules cannot be assigned a previously unused expression or an expression which is part of another logical expresson at postbuild time.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild

Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMActionsOptimization**
Description	This parameter controls the optimization of actions w.r.t ROM consumption. If set to TRUE, BswM will only generate linktime artifacts for the actions which are actually used and ignoring all others. If set to FALSE, BswM will generate linktime artifacts for all actions. This parameter is ignored and treated as TRUE if IMPLEMENTATION_CONFIG_VARIANT = VariantPreCompile. IMPORTANTE NOTE: If BswMActionsOptimization = TRUE, action list items cannot be assigned previously unused actions at postbuild time.
Multiplicity	1..1
Type	BOOLEAN
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMMainFunctionPeriod**
Description	The cycle time of the periodic main function of BswM defined in seconds .
Multiplicity	0..1
Type	FLOAT
Configuration class	**PreCompile:** VariantPreCompile **Link:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMMaxNumActionLists**
Description	The maximum number of BswM action lists supported by this configuration.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMMaxNumRules**
Description	The maximum number of BswM rules supported by this configuration.
Multiplicity	1..1

Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMNmEnabled**
Description	Enables/Disables NM-related API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMNvMEnabled**
Description	Enables/Disables NvM-related API ▶ `true` : Enabled ▶ `false`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMRelocatablePbcfgEnable**
Description	Enables/disable support for relocatable post-build configuration. ▶ True: Postbuild configuration relocatable in memory. ▶ False: Postbuild configuration not relocatable in memory.
Multiplicity	1..1

Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMRteUsage**
Label	Enable Rte Usage
Description	This parameter enables the usage of the RTE for this module. For an easy integration it is recommended to disable the usage of the RTE at the beginning of the integration work.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMSchMEnabled**
Description	This parameter is disabled and its default value is always 'true' as the SchM interface is always needed by the BswM.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**BswMSdEnabled**
Description	enable/disable Sd module related BswM_Sd_CurrentState API:
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile

	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMVersionInfoApi**	
Description	Switches the possibility to read the version information with the service `BswM_GetVersionInfo()`. ▶ true: Enabled ▶ false: Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**BswMWdgMEnabled**	
Description	Enables/Disables WdgM-related API ▶ `true` : Enabled ▶ `false`: Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.2.1.80. BswMQueuedArbitration

Parameters included	
Parameter name	**Multiplicity**
BswMArbitrationQueueLength	1..1
BswMQueueDeferredRequests	1..1
BswMDelayMainFunction	1..1

Parameter Name	BswMArbitrationQueueLength	
Description	This parameter controls the size of the arbitration queue. By default, the value of this parameter is the maximum number of mode request ports in all BswM configurations but this can be manually changed in order to fit the project's requirements. In case the arbitration queue is full and a new request needs to be enqueued, it will be rejected and a DET error will be reported if BswMDevErrorDetect is set to TRUE. Note that a larger queue will also increase the RAM usage of BswM.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=255	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	BswMQueueDeferredRequests	
Description	This parameter can be used to exclude or include deferred mode request ports from being queued. If set to FALSE, only immediate mode request ports will be queued. If set to TRUE, both immediate and deferred mode request ports will be queued.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	BswMDelayMainFunction
Description	This parameter can be used to handle the situation when the BswM_MainFunction interrupts the execution of an immediate mode request. If set to FALSE, the BswM_MainFunction will execute even if there is an immediate request being processed (thus bypassing the arbitration feature). If set to TRUE, the BswM_MainFunction will be delayed until the processing of the immediate mode request is completed. Note that any other request interrupting the processing of the immediate request the BswM_MainFunction interrupted will also be processed before the BswM_MainFunction.

Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.81. BswMUserIncludeFiles

Parameters included	
Parameter name	**Multiplicity**
BswMUserIncludeFile	1..n

Parameter Name	**BswMUserIncludeFile**
Multiplicity	1..n
Type	STRING
Configuration class	**PreCompile:** VariantPreCompile **PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.2.1.82. ReportToDem

Parameters included	
Parameter name	**Multiplicity**
BswMActionFailedReportToDem	1..1
BswMActionFailedReportToDemDetErrorId	1..1

Parameter Name	**BswMActionFailedReportToDem**
Label	BswM Action Failure
Description	Selects the handling of the production error: *BSWM_E_ACTION_FAILED* - DEM: All errors are reported to the Diagnostics Event Manager (Dem). - DET: All errors are reported to the Development Error Tracer (Det) if enabled.

	▶ `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMActionFailedReportToDemDetErrorId**
Label	BswM Action Failed DemToDet ErrorId
Description	This parameter is used to report Dem to Det. It shall be checked that the variable `BswMActionFailedReportToDemDetErrorId` is set to a value between 15 and 255; the default value shall be 15. A preprocessor define `BSWM_E_ACTION_FAILED` shall be generated holding the value of `BswMActionFailedReportToDemDetErrorId`. Dependencies: ▶ The identifier to report Dem to Det must not be zero-based.
Multiplicity	1..1
Type	INTEGER
Default value	15
Range	<=255 >=15
Configuration class	**PreCompile:** VariantPreCompile **PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.2.1.83. BswMDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
BswMDefProgEnabled	1..1
BswMPrecondAssertEnabled	1..1
BswMPostcondAssertEnabled	1..1
BswMStaticAssertEnabled	1..1
BswMUnreachAssertEnabled	1..1
BswMInvariantAssertEnabled	1..1

Parameter Name	**BswMDefProgEnabled**
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module BswM. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module BswM. Dependency on parameter(s): ▶ Enable Development Error Detection (`BswMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`BswMDefProgEnabled`): must be enabled
Multiplicity	1..1

Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module BswM. Dependency on parameter(s): ▶ Enable Development Error Detection (`BswMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`BswMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMStaticAssertEnabled**
Label	Enable Static Assertions
Description	Enables handling of static assertion checks reported from the module BswM. Dependency on parameter(s): ▶ Enable Development Error Detection (`BswMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`BswMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMUnreachAssertEnabled**
Label	Enable Unreachable Code Assertions
Description	Enables handling of unreachable code assertion checks reported from the module BswM. Dependency on parameter(s): ▶ Enable Development Error Detection (`BswMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`BswMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile **VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**BswMInvariantAssertEnabled**
Label	Enable Invariant Assertions
Description	Enables handling of invariant assertion checks reported from functions of the module BswM. Dependency on parameter(s): ▶ Enable Development Error Detection (`BswMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`BswMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile

	VariantPostBuild:	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.2.1.84. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL

Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	15
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**

Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	42
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

5.2.1.85. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the BswM can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.2. Application programming interface (API)

5.2.2.1. Macro constants

5.2.2.1.1. BSWM_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.2.2.1.2. BSWM_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.2.2.1.3. BSWM_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.2.2.1.4. BSWM_E_INVALID_PAR

Purpose	Error Code.
Value	0x03U
Description	A parameter was invalid.

5.2.2.1.5. BSWM_E_NO_INIT

Purpose	Error Code.
Value	0x01U
Description	A service was called prior to initialization.

5.2.2.1.6. BSWM_E_NULL_POINTER

Purpose	Error Code.
Value	0x02U
Description	A null pointer was passed as an argument.

5.2.2.1.7. BSWM_E_PARAM_CONFIG

Purpose	Error Code.
Value	0x06U
Description	The provided configuration is inconsistent.

5.2.2.1.8. BSWM_E_PARAM_POINTER

Purpose	Error Code.
Value	0x07U
Description	A parameter pointer was invalid.

5.2.2.1.9. BSWM_E_QUEUE_FULL

Purpose	Error Code.
Value	0x09U
Description	A new mode request needs to be queued but the mode arbitration queue has reached its maximum capacity.

5.2.2.1.10. BSWM_E_REQ_MODE_OUT_OF_RANGE

Purpose	Error Code.
Value	0x05U
Description	A requested mode was out of range.

5.2.2.1.11. BSWM_E_REQ_USER_OUT_OF_RANGE

Purpose	Error Code.
Value	0x04U
Description	A requesting user was out of range.

5.2.2.1.12. BSWM_E_WRONG_CONTEXT

Purpose	Error code.
Value	0x08U
Description	The API has been called from an invalid partition context.

5.2.2.1.13. BSWM_MODULE_ID

Purpose	AUTOSAR module identification.
Value	42U

5.2.2.1.14. BSWM_NO_CFGCLASSMIX_REQUIRED

Purpose	
Value	

5.2.2.1.15. BSWM_SID_CANSM_CURRENT_STATE

Purpose	AUTOSAR API service ID.
Value	0x05U
Description	Definition of service ID for BswM_CanSM_CurrentState().

5.2.2.1.16. BSWM_SID_COMM_CURRENT_MODE

Purpose	AUTOSAR API service ID.
Value	0x0EU
Description	Definition of service ID for BswM_ComM_CurrentMode().

5.2.2.1.17. BSWM_SID_COMM_CURRENT_PNC_MODE

Purpose	AUTOSAR API service ID.
Value	0x15U
Description	Definition of service ID for BswM_ComM_CurrentPNCMode().

5.2.2.1.18. BSWM_SID_DCM_APPLICATION_UPDATED

Purpose	AUTOSAR API service ID.
Value	0x14U
Description	Definition of service ID for BswM_Dcm_ApplicationUpdated().

5.2.2.1.19. BSWM_SID_DCM_COMMUNICATIONMODE_CURRENT_STATE

Purpose	AUTOSAR API service ID.
Value	0x06U
Description	Definition of service ID for BswM_Dcm_CommunicationMode_CurrentState().

5.2.2.1.20. BSWM_SID_DCM_REQUEST_COMMUNICATION_MODE

Purpose	AUTOSAR API service ID.

Value	0xFEU
Description	Definition of service ID for BswM_Dcm_RequestCommunicationMode().

5.2.2.1.21. BSWM_SID_DCM_REQUEST_RESET_MODE

Purpose	AUTOSAR API service ID.
Value	0x07U
Description	Definition of service ID for BswM_Dcm_RequestResetMode().

5.2.2.1.22. BSWM_SID_DCM_REQUEST_SESSION_MODE

Purpose	AUTOSAR API service ID.
Value	0x08U
Description	Definition of service ID for BswM_Dcm_RequestSessionMode().

5.2.2.1.23. BSWM_SID_DEINIT

Purpose	AUTOSAR API service ID.
Value	0x04U
Description	Definition of service ID for BswM_Deinit().

5.2.2.1.24. BSWM_SID_ECUM_CURRENT_STATE

Purpose	AUTOSAR API service ID.
Value	0x0FU
Description	Definition of service ID for BswM_EcuM_CurrentState().

5.2.2.1.25. BSWM_SID_ECUM_CURRENT_WAKEUP

Purpose	AUTOSAR API service ID.
Value	0x10U
Description	Definition of service ID for BswM_EcuM_CurrentWakeup().

5.2.2.1.26. BSWM_SID_ETHIF_PORTGROUPLINKSTATECHG

Purpose	AUTOSAR API service ID.
Value	0x26U
Description	Definition of service ID for BswM_EthIf_PortGroupLinkStateChg().

5.2.2.1.27. BSWM_SID_ETHSM_CURRENT_STATE

Purpose	AUTOSAR API service ID.
Value	0x0DU
Description	Definition of service ID for BswM_EthSM_CurrentState().

5.2.2.1.28. BSWM_SID_FRSM_CURRENT_STATE

Purpose	AUTOSAR API service ID.
Value	0x0CU
Description	Definition of service ID for BswM_FrSM_CurrentState().

5.2.2.1.29. BSWM_SID_GET_VERSION_INFO

Purpose	AUTOSAR API service ID.
Value	0x01U
Description	Definition of service ID for BswM_GetVersionInfo().

5.2.2.1.30. BSWM_SID_INIT

Purpose	AUTOSAR API service ID.
Value	0x00U
Description	Definition of service ID for BswM_Init().

5.2.2.1.31. BSWM_SID_INVALID

Purpose	
Value	0xFFU

5.2.2.1.32. BSWM_SID_J1939DCMBROADCASTSTATUS

Purpose	AUTOSAR API service ID.
Value	0x1BU
Description	Definition of service ID for BswM_J1939DcmBroadcastStatus().

5.2.2.1.33. BSWM_SID_J1939NM_STATECHANGENOTIFICATION

Purpose	AUTOSAR API service ID.
Value	0x18U
Description	Definition of service ID for BswM_J1939Nm_StateChangeNotification().

5.2.2.1.34. BSWM_SID_LINSM_CURRENT_SCHEDULE

Purpose	AUTOSAR API service ID.
Value	0x0AU
Description	Definition of service ID for BswM_LinSM_CurrentSchedule().

5.2.2.1.35. BSWM_SID_LINSM_CURRENT_STATE

Purpose	AUTOSAR API service ID.
Value	0x09U
Description	Definition of service ID for BswM_LinSM_CurrentState().

5.2.2.1.36. BSWM_SID_LINTP_REQUESTMODE

Purpose	AUTOSAR API service ID.
Value	0x0BU
Description	Definition of service ID for BswM_LinTp_RequestMode().

5.2.2.1.37. BSWM_SID_MAINFUNCTION

Purpose	AUTOSAR API service ID.

Value	0x03U
Description	Definition of service ID for BswM_MainFunction().

5.2.2.1.38. BSWM_SID_NM_CAR_WAKEUP_INDICATION

Purpose	AUTOSAR API service ID.
Value	0x24U
Description	Definition of service ID for BswM_NmIf_CarWakeUpIndication().

5.2.2.1.39. BSWM_SID_NVM_CURRENT_BLOCK_MODE

Purpose	AUTOSAR API service ID.
Value	0x16U
Description	Definition of service ID for BswM_NvM_CurrentBlockMode().

5.2.2.1.40. BSWM_SID_NVM_CURRENT_JOB_MODE

Purpose	AUTOSAR API service ID.
Value	0x17U
Description	Definition of service ID for BswM_NvM_CurrentJobMode().

5.2.2.1.41. BSWM_SID_REQUESTMODE

Purpose	AUTOSAR API service ID.
Value	0x02U
Description	Definition of service ID for bswm_requestmode().

5.2.2.1.42. BSWM_SID_SD_CLIENTSERVICE_CURRENTSTATE

Purpose	AUTOSAR API service ID.
Value	0x1FU
Description	Definition of service ID for BswM_Sd_ClientServiceCurrentState().

5.2.2.1.43. BSWM_SID_SD_CONSUMEDEVENTGROUP_CURRENTSTATE

Purpose	AUTOSAR API service ID.
Value	0x21U
Description	Definition of service ID for BswM_Sd_ConsumedEventGroupCurrentState().

5.2.2.1.44. BSWM_SID_SD_EVENTHANDLER_CURRENTSTATE

Purpose	AUTOSAR API service ID.
Value	0x20U
Description	Definition of service ID for BswM_Sd_EventHandlerCurrentState().

5.2.2.1.45. BSWM_SID_TRIGGER_SLAVE_RTE_STOP

Purpose	AUTOSAR API service ID.
Value	0x13U
Description	Definition of service ID for BswM_TriggerSlaveRTEStop().

5.2.2.1.46. BSWM_SID_TRIGGER_STARTUP_PHASE2

Purpose	AUTOSAR API service ID.
Value	0x12U
Description	Definition of service ID for BswM_TriggerStartUpPhase2().

5.2.2.1.47. BSWM_SID_WDGM_REQUEST_PARTITION_RESET

Purpose	AUTOSAR API service ID.
Value	0x11U
Description	Definition of service ID for BswM_WdgM_RequestPartitionReset().

5.2.2.1.48. BSWM_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	1U

5.2.2.1.49. BSWM_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	15U

5.2.2.1.50. BSWM_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	3U

5.2.2.1.51. BSWM_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.2.2.2. Functions

5.2.2.2.1. BswM_Deinit

Purpose	Deinitializes the BSW Mode Manager.
Synopsis	`void BswM_Deinit ( void );`
Service ID	4
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	After a call of BswM_Deinit no mode processing shall be performed by BswM even if any mode requests are made or the BswM main function is called Precondition: None

5.2.2.2.2. BswM_GetVersionInfo

Purpose	Return the modules version information.
Synopsis	`void BswM_GetVersionInfo ( Std_VersionInfoType * VersionInfoPtr );`

Parameters (out)	`VersionInfoPtr`	- Pointer to struct to be filled with the version information.
Description	This function provides the information to module vendor ID, module ID and software version major.minor.patch Precondition: BSWM_VERSION_INFO_API = STD_ON	

5.2.2.2.3. BswM_Init

Purpose	Initializes the BSW Mode Manager.	
Synopsis	`void **BswM_Init** ( const BswM_ConfigType * ConfigPtr );`	
Service ID	0	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ConfigPtr`	- Pointer to a selected configuration structure.
Description	This routine initializes the BSW Mode Manager. After execution of this routine the BSW Mode Manager is ready to arbitrate incoming mode requests Precondition: None	

5.2.2.2.4. BswM_IsValidConfig

Purpose	Validates the post-build configuration data structure passed to the BswM_Init function.	
Synopsis	`Std_ReturnType **BswM_IsValidConfig** ( const void * voidPtr );`	
Parameters (in)	`voidPtr`	The pointer to a BswM post-build data structure. If a NULL_PTR is passed, the BswM will attempt to retrieve the BswM post-build configuration from the PbcfgM module.
Parameters (in,out)	`voidPtr`	The pointer to a BswM post-build data structure. If a NULL_PTR is passed, the BswM will attempt to retrieve the BswM post-build configuration from the PbcfgM module.
Return Value		

E_OK	When the pre-compile, link-time and platform hash values stored within the post-build structure correspond to the hash values of the compiled source files. Otherwise, E_NOT_OK will be returned.	

5.2.2.2.5. BswM_RequestMode

Purpose	Generic function call to request modes.	
Synopsis	`void` **`BswM_RequestMode`** `( BswM_UserType requesting_user , BswM_ModeType requested_mode );`	
Service ID	2	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`requesting_user`	The user that requests the mode
	`requested_mode`	The requested mode.
Description	This function is used to request modes and shall only be used by other BSW modules that does not have a specific mode request interface. Precondition: None	

5.2.3. Integration notes

5.2.3.1. Exclusive areas

This section describes the exclusive areas used by the `BswM` module.

5.2.3.1.1. SCHM_BSWM_EXCLUSIVE_AREA

Protected data structures	This memory section is used to protect all shared data within the same partition from mutual access. Note that each BswM partition will have its own exclusive area.
Recommended locking mechanism	These exclusive areas must always be protected by a locking mechanism. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Re-

	fer to the section Mapping exclusive areas in the basic software modules in the Integration notes section for details.

5.2.3.2. Production errors

The BswM supports reporting of production errors for action list items. If you configure the BswMReportFailToDemRef parameter of an action list item to a valid DemEventParameter reference, the status is reported as follows:

- If the configured action of the action list item returns E_OK, the Dem error BSWM_E_ACTION_FAILED with a DEM_EVENT_STATUS_PASSED status is reported.
- If the configured action of the action list item returns E_NOT_OK, the Dem error BSWM_E_ACTION_FAILED with a DEM_EVENT_STATUS_FAILED is reported.

You can use the same reference to a DemEventParameter for multiple action list items.

NOTE

Availability of error report

The error report is only available if the BswMActionListItemRef parameter references a BswMAction. If the BswMActionListItemRef parameter references a BswMActionList or a BswMRule, no Dem errors are reported.

5.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section Memory mapping and compiler abstraction in the Integration notes section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CONFIG_DATA_UNSPECIFIED
VAR_INIT_UNSPECIFIED
VAR_NO_INIT_UNSPECIFIED
CONST_UNSPECIFIED
INTERNAL_VAR_POWER_ON_INIT_UNSPECIFIED
CODE
VAR_NO_INIT_8
VAR_POWER_ON_INIT_32

5.2.3.4. Integration requirements

WARNING ⚠ **Integration requirements list is not exhaustive**

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.2.3.4.1. intgr.BswM.EB_INTREQ_BswM_0001

Description	The PbcfgM module support shall always be used in case the selected IMPLEMENTATION_CONFIG_VARIANT is VariantPostBuild.
Rationale	The EcuM calls the BswM_Init API with NULL_PTR as a parameter. Consequently, the BswM must obtain the the post-build configuration via the PbcfgM.

5.2.3.4.2. intgr.BswM.EB_INTREQ_BswM_0002

Description	The BswM shall reject a configuration in which the selected IMPLEMENTATION_CONFIG_VARIANT is VariantPostBuild and a symbolic name value is configured as an initial value of a mode request port via BswMBswModeInitValueMode.
Rationale	In order to support the binary generation of the BswM's post-build configuration, the BswM now requires that the initial value defined within the parameter BswMBswModeInitValueMode be a numerical constant if the selected variant is VariantPostBuild. This constraint is due to the fact the numerical value of the symbolic value can only be known at runtime but the XGEN generation of the binary representation of the post-build configuration needs to be done before.

5.2.3.4.3. intgr.BswM.EB_INTREQ_BswM_0003

Description	The BswMSchMEnabled parameter shall be read only and its default value set to 'TRUE'.
Rationale	The SchM interface is always needed by the BswM after Bsw distribution support was added.

5.2.3.4.4. intgr.BswM.EB_INTREQ_BswM_0004

Description	It falls within the user's responsibilities to be aware and resolve any reentrancy issues that might occur when immediate mode request ports are configured.

Rationale	The execution of both user callout functions and predefined BswM actions as a result of the mode arbitration of immediate mode request ports is potentially dangerous and can lead to unexpected behaviour. In case of the predefined BswM actions, the user must consult the API specification of the functions which are called by the BswM and verify their reentrancy status. As an example, the BswMPduGroupSwitch action triggers the execution of the Com_IpduGroupControl() API which is not reentrant. Configuring such an action within a rule whose logical expression uses an immediate mode request port might lead to unexpected behaviour.

5.2.3.4.5. intgr.BswM.EB_INTREQ_BswM_0005

Description	BswM API's shall not be called from CAT1 interrupts if multiple BswM instances are configured.
Rationale	In CAT1 interrupts the BswM cannot use the GetApplicationID API to determine the exact instance that will arbitrate the request.

5.2.3.4.6. intgr.BswM.EB_INTREQ_BswM_BswMPduGroupSwitch_BswMDeadlineMonitoringControl

Description	BswMPduGroupSwitch and BswMDeadlineMonitoringControl actions can only be configured in one BswM instance
Rationale	The Com module only has one instance and the API's corresponding to the BswMPduGroupSwitch and BswMDeadlineMonitoringControl actions are not designed to be called concurrently. As the BswM allows multiple instances, only one of them can safely call the two Com API's.

5.2.3.4.7. intgr.BswM.EB_INTREQ_BswMLogicalExpressionsOptimization

Description	If configuration parameter BswMLogicalExpressionsOptimization is set to TRUE, the parameter BswMRuleExpressionRef cannot reference previously unused logical expressions or logical expressions which are part of other logical expressions at PostBuild time.
Rationale	Linktime artifacts for each logical expression are needed in order for the BswMRuleExpressionRef parameter to be changed at PostBuild time. However, this increases ROM consumption. If changing BswMRuleExpressionRef in the above described manner is not required at PostBuild time, the linktime ROM consumption can be improved by setting BswMLogicalExpressionsOptimization to TRUE.

5.2.3.4.8. intgr.BswM.EB_INTREQ_BswMActionsOptimization

Description	If configuration parameter BswMActionsOptimization is set to TRUE, the parameter BswMActionListItemRef cannot reference a previously unused action at PostBuild time.
Rationale	Linktime artifacts for each action are needed in order for the BswMActionListItemRef parameter to be changed at PostBuild time. However, this increases ROM consumption. If changing BswMActionListItemRef in the above described manner is not required at PostBuild time, the linktime ROM consumption can be improved by setting BswMActionsOptimization to TRUE.

5.2.3.4.9. intgr.BswM.EB_INTREQ_BswMSchMSwitchPostbuild

Description	At PostBuild time a BswMSchMSwitch action cannot be configured for a rule referencing different mode request ports than the original rule it was mapped to.
Rationale	If at PostBuild time a BswMSchMSwitch is mapped to a rule referencing different mode request ports than the one it was mapped initially the basic software module description requires an update. Updating the basic software module description at PostBuild time is not supported in the ACG.

5.3. ComM

5.3.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
ComMDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
ComMConfigSet	1..1	This container is the base for a multiple configuration set.
ComMGeneral	1..1	**Label:** General Configuration General configuration parameters of the Communication Manager.
CommonPublishedInformation	1..1	**Label:** Common Published Information

Containers included		
		Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT	
Label	Configuration Variant	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	VariantPostBuild	
Range	VariantPostBuild	
Configuration class	**VariantPostBuild:**	VariantPostBuild

5.3.1.1. ComMDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
ComMDefProgEnabled	1..1
ComMPrecondAssertEnabled	1..1
ComMPostcondAssertEnabled	1..1
ComMStaticAssertEnabled	1..1
ComMUnreachAssertEnabled	1..1
ComMInvariantAssertEnabled	1..1

Parameter Name	ComMDefProgEnabled
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module ComM. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows:

	1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module ComM. Dependency on parameter(s): ▶ Enable Development Error Detection (`ComMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`ComMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module ComM. Dependency on parameter(s): ▶ Enable Development Error Detection (`ComMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`ComMDefProgEnabled`): must be enabled
Multiplicity	1..1

Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMStaticAssertEnabled**
Label	Enable Static Assertions
Description	Enables handling of static assertion checks reported from the module ComM. Dependency on parameter(s): ▶ Enable Development Error Detection (`ComMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`ComMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMUnreachAssertEnabled**
Label	Enable Unreachable Code Assertions
Description	Enables handling of unreachable code assertion checks reported from the module ComM. Dependency on parameter(s): ▶ Enable Development Error Detection (`ComMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`ComMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMInvariantAssertEnabled**	
Label	Enable Invariant Assertions	
Description	Enables handling of invariant assertion checks reported from functions of the module ComM. Dependency on parameter(s): ▶ Enable Development Error Detection (`ComMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`ComMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.2. ComMConfigSet

Containers included		
Container name	**Multiplicity**	**Description**
ComMChannel	1..n	This container contains the configuration (parameters) of the bus channel(s). The channel parameters shall be harmonized within the whole communication stack.
ComMPnc	0..n	This container contains the configuration of the partial network cluster (PNC).
ComMUser	0..n	This container contains a list of user. For each user the Rte creates ports for requesting communication modes and for notification of mode changes.

Parameters included	
Parameter name	**Multiplicity**
ComMPncEnabled	0..1

Parameter Name	**ComMPncEnabled**
Description	Defines whether in this configuration set the partial networking is enabled.

	▶ `TRUE`: Enabled ▶ `FALSE`: Disabled
Multiplicity	0..1
Type	BOOLEAN
Default value	true
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.3.1.3. ComMChannel

Containers included		
Container name	**Multiplicity**	**Description**
ComMNetworkManagement	1..1	**Label:** Network Management Configuration This container contains the configuration parameters of the networkmanagement.
ComMUserPerChannel	0..n	This container contains a list of identifiers that are needed to refer to a user in the system which is linked to a channel.

Parameters included	
Parameter name	**Multiplicity**
ComMBusType	1..1
ComMChannelId	1..1
ComMFullCommRequestNotificationEnabled	1..1
ComMGlobalNvmBlockDescriptor	1..1
ComMMainFunctionPeriod	1..1
ComMNoCom	1..1
ComMNoWakeup	1..1
ComMPncGatewayType	1..1
ComMManageReference	0..1

Parameter Name	**ComMBusType**
Label	Bus Type
Description	This parameter identifies the bus type of the network channel. Dependencies:

	▶ If the bus type of the network channel is set to *COMM_BUS_TYPE_INTERNAL* the Network Management Variant has to be set to *NONE* and if the bus type of the network channel is set to *COMM_BUS_TYPE_LIN* the Network Management Variant has to be set to *LIGHT*
Multiplicity	1..1
Type	ENUMERATION
Default value	COMM_BUS_TYPE_CAN
Range	COMM_BUS_TYPE_CAN COMM_BUS_TYPE_ETH COMM_BUS_TYPE_FR COMM_BUS_TYPE_INTERNAL COMM_BUS_TYPE_LIN
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMChannelId**
Label	Network Channel ID
Description	Identification number of the network channel. Dependencies: ▶ The identifier of the network channel must be zero-based, consecutive and unique.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMFullCommRequestNotificationEnabled**
Description	Defines if the optional SenderReceiver Port of Interface `ComM_CurrentChannelRequest` will be provided for the network channel. ▶ `TRUE`: Enabled ▶ `FALSE`: Disabled
Multiplicity	1..1
Type	BOOLEAN

Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMGlobalNvmBlockDescriptor**	
Label	Enable Storage for NoWakeUp Inhibition	
Description	If this parameter is set to "true", the NoWakeUp inhibition state of the channel shall be stored (in some implementation specific way) in the block pointed to by ComMGlobalNvmBlockDescriptor. Dependencies: ▶ If the parameter is set to true, a valid Nvm block reference must be given in the (existing, i.e. multiplicity 1) ComMGlobalNvmBlockDescriptor pointing to a sufficiently big Nvm block.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMMainFunctionPeriod**	
Label	Main Function Period	
Description	Specifies the period the MainFunction has to be triggered in seconds. ComM scheduling shall be at least as fast as the communication stack and a schedule longer than 100ms makes no sense for communication.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.020	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMNoCom**
Label	No Change To Communication
Description	If this parameter is enabled it is not possible to switch the mode of the network channel to *Full Communication* or *Silent Communication*.

	This parameter just defines the initial value. The limitation can be modified by calling `ComM_LimitChannelToNoComMode()`. ▶ `TRUE`: Enabled - Not allowed to switch to Communication Modes above. ▶ `FALSE`: Disabled - Allowed to switch Communication Modes above. Dependencies: ▶ Mode Limitation must be enabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMNoWakeup**
Label	Prevent Channel Wake Up
Description	Defines if an ECU is not allowed to wake-up the network channel. ▶ `TRUE`: Enabled (not allowed to wake-up) ▶ `FALSE`: Disabled. This is the default/init value of the runtime variable that can be changed during runtime using `ComM_PreventWakeUp()`.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMPncGatewayType**
Label	Partial Network Gateway Type
Description	Identifies the Partial Network Gateway behaviour of a ComMChannel.
Multiplicity	1..1
Type	ENUMERATION
Default value	COMM_GATEWAY_TYPE_ACTIVE
Range	COMM_GATEWAY_TYPE_ACTIVE

	COMM_GATEWAY_TYPE_PASSIVE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMManageReference**	
Description	Reference to the managing channel. ▶ `ImplementationType`: COMM_ChannelType	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.4. ComMNetworkManagement

Parameters included	
Parameter name	**Multiplicity**
ComMNmLightTimeout	0..1
ComMNmVariant	1..1
ComMPncNmRequest	1..1

Parameter Name	**ComMNmLightTimeout**
Label	Network Management Light Timeout
Description	Defines the timeout (in seconds) after state *ready sleep* is left. Range: greater than 0 and less than or equal to 255 Dependencies: ▶ Even though the AUTOSAR Specification allows a minimum value of 0, the possible minimum value has been restricted to 0.001 which is also the minimum value of ComMMainFunctionPeriod. Rationale: A timeout value configured to be less than the value of the Main Function Period cannot really be ensured. ▶ The value of this parameter must be greater or equal the value of ComMMainFunctionPeriod.

Multiplicity	0..1	
Type	FLOAT	
Default value	10.000	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMNmVariant**	
Label	Network Management Variant	
Description	Defines the functionality of the networkmanagement. Shall be harmonized with NM configuration. ▶ `NONE` : No NM available ▶ `LIGHT` : No AUTOSAR NM available but functionality to shut down a network channel (not synchronized). ▶ `PASSIVE` : AUTOSAR NM available but ECU is not allowed to keep the bus awake. ▶ `FULL` : AUTOSAR NM available (default)	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	FULL	
Range	FULL LIGHT NONE PASSIVE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMPncNmRequest**
Label	Pnc Network Request
Description	If this parameter equals true then every time a FULL Communication is requested due to a change in the PNC state machine to PNC_REQUESTED Nm shall be called using the API Nm_NetworkRequest. Dependencies: ▶ Channel is not limited to COMM_NO_COM. Refer <function>ComM_LimitChannelToNoComMode()</function>.

	▶ Wake up is not inhibited. Refer parameter `ComMWakeupInhibitionEnabled`	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.5. ComMUserPerChannel

Parameters included	
Parameter name	**Multiplicity**
ComMUserChannel	1..1

Parameter Name	**ComMUserChannel**	
Label	User Reference	
Description	Reference to the ComMUser that corresponds to this channel user. ▶ `ImplementationType`: COMM_UserHandleType	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.6. ComMPnc

Containers included		
Container name	**Multiplicity**	**Description**
ComMPncComSignal	0..n	Represents the PncComSignals which are used to communicate the EIRA and ERA status of this PNC.

Parameters included	
Parameter name	**Multiplicity**
ComMPncId	1..1

Parameters included	
ComMChannelPerPnc	1..n
ComMUserPerPnc	0..n
ComMPncEthIfSwitchPortGroupRef	0..255

Parameter Name	ComMPncId	
Label	PNC ID	
Description	Partial network cluster identification number.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	ComMChannelPerPnc	
Description	Reference to the ComMChannel that is required for this PNC. ▶ `ImplementationType`: COMM_ChannelType	
Multiplicity	1..n	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	ComMUserPerPnc	
Label	User Per PNC	
Description	Reference to the ComMUsers that correspond to this PNC. ▶ `ImplementationType`: COMM_UserHandleType	
Multiplicity	0..n	
Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	ComMPncEthIfSwitchPortGroupRef
Label	EthIf Switch port group references
Description	Reference to the PortGroups that correspond to this PNC
Multiplicity	0..255

Type	REFERENCE	
Configuration class	**PostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.7. ComMPncComSignal

Parameters included	
Parameter name	**Multiplicity**
ComMPncComSignalDirection	1..1
ComMPncComSignalKind	1..1
ComMPncComSignalChannelRef	1..1
ComMPncComSignalRef	1..1

Parameter Name	**ComMPncComSignalDirection**	
Label	PNC Com Signal Direction	
Description	Indicates the communication direction of this PncComSignal.	
Multiplicity	1..1	
Type	ENUMERATION	
Range	RX TX	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMPncComSignalKind**	
Label	PNC Com Signal Kind	
Description	Indicates whether this PncComSignal represents EIRA or ERA PNC information.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	EIRA	
Range	EIRA ERA	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMPncComSignalChannelRef**
Label	PNC Com Signal Channel Reference
Description	Reference to the ComMChannel which is used to determine whether this PncComSignal shall participate in the active or passive role (via the parameter ComMPncGatewayType of the ComMChannel). Not applicable if ComMPncComSignalKind is EIRA. Dependencies: - The parameter `ComMPncComSignalKind` must be set to 'ERA'. - The parameter `ComMPncGatewayEnabled` must be set to 'true'.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** | VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMPncComSignalRef**
Label	PNC Com Signal Reference
Description	Reference to the ComSignal which is used to transport the partial network channel request information.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** | VariantPostBuild
Origin	AUTOSAR_ECUC

5.3.1.8. ComMUser

Parameters included	
Parameter name	**Multiplicity**
ComMUserIdentifier	1..1
ComMUserEcucPartitionRef	0..1

Parameter Name	**ComMUserIdentifier**
Label	User Identifier
Description	Numeric identifier of one user of the Communication Manager.

	Dependencies: ▶ The user identifier must be zero-based and consective.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMUserEcucPartitionRef**	
Description	Denotes in which "EcucPartition" the requester is executed. When the partition is stopped, the communication request shall be cancelled in the ComM to avoid a stay-awake situation of the bus due to a stopped partition.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.3.1.9. ComMGeneral

Containers included		
Container name	**Multiplicity**	**Description**
ComMServiceAPI	1..1	**Label:** Service API Parameters Container for configuration of the service API of ComM.

Parameters included	
Parameter name	**Multiplicity**
ComMMultiCoreSupport	1..1
ComMMasterCoreEcuCPartitionRef	1..1
ComMDcmUsage	1..1
ComM0PncVectorAvoidance	1..1
ComMDevErrorDetect	1..1
ComMDirectUserMapping	0..1
ComMEcuGroupClassification	1..1
ComMModeLimitationEnabled	1..1

Parameters included	
ComMPncGatewayEnabled	0..1
ComMPncPrepareSleepTimer	0..1
ComMPncSupport	1..1
ComMPnInfoOffset	0..1
ComMRelocatablePbcfgEnable	1..1
ComMRteUsage	1..1
ComMSynchronousWakeUp	1..1
ComMTMinFullComModeDuration	1..1
ComMVersionInfoApi	1..1
ComMWakeupInhibitionEnabled	1..1
ComMGlobalNvMBlockDescriptor	0..1
ComMReleaseNoneChannels	1..1
ComMPncReleaseChannel	1..1
ComMDefineOrder	1..1

Parameter Name	ComMMultiCoreSupport	
Label	ComM multicore support	
Description	Enables MultiCoreSupport.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	ComMMasterCoreEcuCPartitionRef	
Label	Master Core EcuC Partition Reference	
Description	Master Core EcuC Partition Ref	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	ComMDcmUsage

Label	Enable Dcm Usage	
Description	This parameter enables the usage of the Dcm. In case this parameter is set to `false`: - ComM does not provide the API functions `ComM_DCM_ActiveDiagnostic()` and `ComM_DCM_InactiveDiagnostic` - ComM does not call the API functions `Dcm_ComM_FullComModeEntered()`, `Dcm_ComM_SilentComModeEntered()` and `Dcm_ComM_NoComModeEntered()`	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**ComM0PncVectorAvoidance**	
Label	Enable ComM0PncVectorAvoidance	
Description	Avoids sending of 0-PNC-Vectors in case ComMPncGatewayEnabled is enabled - `TRUE`: Enabled - `FALSE`: Disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMDevErrorDetect**
Label	Enable Development Error Detection
Description	Switches the Development Error Detection and Notification `ON` or `OFF`. - `TRUE`: Enabled - `FALSE`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	true

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMDirectUserMapping**	
Label	Direct User Mapping	
Description	If this parameter is set to true the configuration tool shall automatically create a ComMUser per ComMPnc and a ComMUser per ComMChannel. The shortName of the generated ComMUsers shall follow the following naming convention: ▶ `PNCUser_ComMPncId`, e.g. PNCUser_13 ▶ `ChannelUser_ComMChannelId`, e.g. ChannelUser_25 Restriction: ComMUser, which are created due to this configuration parameter, shall not be used by SWCs (only available for BswM).	
Multiplicity	0..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMEcuGroupClassification**
Label	ECU Group Classification
Description	This parameter is interpreted as a bitfield where each bit enables/disables a mode inhibition. ▶ `Bit 0`: Wake Up inhibition active ▶ `Bit 1`: Limit to *No Communication* mode For Example: `00000011`: Wake up inhibition and limitation to *No Communication* mode active Dependencies: ▶ Mode Limitation must be enabled.
Multiplicity	1..1
Type	INTEGER
Default value	3

Range	<=255 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMModeLimitationEnabled**
Label	Mode Limitation
Description	This parameter enables the mode limitation functionality. ▶ `TRUE`: Enabled ▶ `FALSE`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMPncGatewayEnabled**
Label	Support for Partial Network Gateway
Description	Enables or disables support of Partial Network Gateway. ▶ `False`: Partial Networking Gateway is disabled ▶ `True`: Partial Networking Gateway is enabled
Multiplicity	0..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMPncPrepareSleepTimer**
Label	PNC PrepareSleep Timer
Description	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
Multiplicity	0..1
Type	FLOAT
Range	<=63.0

	>=0.0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMPncSupport**	
Label	Support for Partial Network Cluster (PNC)	
Description	Enables or disables support of partial networking. ▶ `False`: Partial Networking is disabled ▶ `True`: Partial Networking is enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ComMPnInfoOffset**	
Label	PnInfoOfset	
Description	Position of partial network information inside NM frame. Ex. For a NM frame CBV BYTE1 BYTE2 ...BYTE7. ▶ A value 1 for ComMPnInfoOffset, lead to interpreting BYTE1 as holding PN ID 8 to 15. ▶ A value 2 for ComMPnInfoOffset, lead to interpreting BYTE2 as holdign PN ID 16 to 23.	
Multiplicity	0..1	
Type	INTEGER	
Default value	1	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**ComMRelocatablePbcfgEnable**
Label	ComMRelocatablePbcfgEnable
Description	Enables/disables support for relocatable postbuild configuration. ▶ True: Postbuild configuration relocatable in memory.

	▶ False: Postbuild configuration not relocatable in memory.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMRteUsage**
Label	Enable Rte Usage
Description	This parameter enables the usage of the RTE for this module. For an easy integration it is recommended to disable the usage of the RTE at the beginning of the integration work.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMSynchronousWakeUp**
Label	Synchronous Wake Up
Description	If this parameter is enabled, a wake up of one channel shall lead to a wake up of all network channels. ▶ `TRUE`: Enabled ▶ `FALSE`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMTMinFullComModeDuration**
Label	Minimum Full Communication Mode Duration Time
Description	Minimum time duration in seconds the ComM spends in the Full Communication mode.

Multiplicity	1..1
Type	FLOAT
Default value	5
Range	<=65.000 >=0.001
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMVersionInfoApi**
Label	Enable Version Info API
Description	Switches the possibility to read the published information with the service `ComM_GetPublishedInformation()`. ▶ `TRUE`: Enabled ▶ `FALSE`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMWakeupInhibitionEnabled**
Label	Wakeup Inhibition
Description	Enable wake up inhibition functionality. ▶ `TRUE`: Enabled ▶ `FALSE`: Disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMGlobalNvMBlockDescriptor**
Label	Global NVRAM Block Descriptor

Description	Reference to NVRAM block containing the non volatile data. Index of the NvMBlockDescriptor will be used by the ComM module for the identification of the NvM block
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**ComMReleaseNoneChannels**
Label	ComMReleaseNoneChannels
Description	Specifies if all channels with attribute ComMBusType set to COMM_BUS_TYPE_INTERNAL shall switch from Ready Sleep state to No Communication. ▶ `TRUE`If ComMBusType = (COMM_BUS_TYPE_CAN or COMM_BUS_TYPE_FR or COMM_BUS_TYPE_LIN or COMM_BUS_TYPE_ETH) and ComMNmVariant = NONE, state transition from Ready Sleep state to No Communication is not performed. The transition from Ready Sleep state to No Communication state is performed only in case ComMBusType = COMM_BUS_TYPE_INTERNAL. ▶ `FALSE`When entering ready sleep state all channels with ComMNmVariant = NONE go immediatly to No Communication.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMPncReleaseChannel**
Label	ComMPncReleaseChannel
Description	Configures when the Full Communication requests shall be released. When this parameter is set to COMM_NO_INTERNAL_AND_EXTERNAL_REQUESTS, all Full Communication requests shall be released in the PNC_NO_COMMUNICATION state (Autosar 4.1.1 and Autosar 3.2.3). When this parameter is set to COMM_NO_INTERNAL_REQUESTS, all Full Communication requests shall be released after leaving the PNC_REQUESTED state (Autosar 4.2.1, see Rfc #63643).

Multiplicity	1..1	
Type	ENUMERATION	
Default value	COMM_NO_INTERNAL_REQUESTS	
Range	COMM_NO_INTERNAL_REQUESTS COMM_NO_INTERNAL_AND_EXTERNAL_REQUESTS	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**ComMDefineOrder**	
Label	ComMDefineOrder	
Description	Defines the order to be used for mode declaration group ComMMode. ▶ `EXPLICIT_ORDER` - Explicit order is used. ▶ `ALPHABETIC_ORDER` - Alphabetical order is used.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	EXPLICIT_ORDER	
Range	EXPLICIT_ORDER ALPHABETIC_ORDER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.1.10. ComMServiceAPI

Parameters included	
Parameter name	**Multiplicity**
ComMEnableASR32ServiceAPI	1..1
ComMEnableASR40ServiceAPI	1..1
ComMDefaultASRServiceAPI	1..1

Parameter Name	**ComMEnableASR32ServiceAPI**
Label	Enable AUTOSAR 3.2 service API
Description	Configures whether the AUTOSAR 3.2 service API shall be provided.

	- `TRUE` = Enables AUTOSAR 3.2 service API. - `FALSE` = Disables AUTOSAR 3.2 service API.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMEnableASR40ServiceAPI**
Label	Enable AUTOSAR 4.0 service API
Description	Configures whether the AUTOSAR 4.0 service API shall be provided. - `TRUE` = Enables AUTOSAR 4.0 service API. - `FALSE` = Disables AUTOSAR 4.0 service API. ;
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**ComMDefaultASRServiceAPI**
Label	Default AUTOSAR service API
Description	Defines the default AUTOSAR service API. - `AUTOSAR_32` = AUTOSAR 3.2 service API is the default one. - `AUTOSAR_40` = AUTOSAR 4.0 service API is the default one. - `NONE` = No default AUTOSAR service API is provided.
Multiplicity	1..1
Type	ENUMERATION
Default value	AUTOSAR_40
Range	AUTOSAR_32 AUTOSAR_40 NONE

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.3.1.11. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**	
Label	AUTOSAR Major Version	
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	4	
Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	19
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version

Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	12
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.3.1.12. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the ComM can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.3.2. Application programming interface (API)

5.3.2.1. Type definitions

5.3.2.1.1. ComM_ASR32_InhibitionStatusType

Purpose	Inhibition status of ComM.
Type	`uint8`
Description	Defines whether a mode inhibition is active or not. ▶ Bit 0 (LSB): Wake Up inhibition active. ▶ Bit 1 (MSB): Limit to "No Communication" mode.

5.3.2.1.2. ComM_ASR32_ModeType

Purpose	Current mode of the Communication Manager (main state of the state machine).

Type	`uint8`
Description	If Rte usage is enabled in the ComM, the type ComM_ModeType is defined in Rte_-Types.h and is made public indirectly by including Rte_Types.h in ComM.h.

5.3.2.1.3. ComM_ASR32_UserHandleType

Purpose	Type for user of the communication manager.
Type	`uint8`
Description	Wraps the ComM_UserHandleType. Range: - 0 ... 255

5.3.2.1.4. ComM_ASR40_InhibitionStatusType

Purpose	Inhibition status of ComM.
Type	`uint8`
Description	Defines whether a mode inhibition is active or not. - Bit 0 (LSB): Wake Up inhibition active. - Bit 1 (MSB): Limit to "No Communication" mode.

5.3.2.1.5. ComM_ASR40_ModeType

Purpose	Current mode of the Communication Manager (main state of the state machine).
Type	`uint8`
Description	If Rte usage is enabled in the ComM, the type ComM_ModeType is defined in Rte_-Types.h and is made public indirectly by including Rte_Types.h in ComM.h.

5.3.2.1.6. ComM_ASR40_UserHandleType

Purpose	Type for user of the communication manager.
Type	`uint8`
Description	Wraps the ComM_UserHandleType.

	Range: ▶ 0 ... 255

5.3.2.1.7. ComM_BusSM_RequestComMode_FctPtr

Purpose	Function pointer type for BusSM Request.
Type	`Std_ReturnType(*)(uint8, uint8)`

5.3.2.1.8. ComM_Dcm_Notif_FctPtr

Purpose	Function pointer type for Dcm notifications.
Type	`Std_ReturnType(*)(uint8)`

5.3.2.1.9. ComM_Dcm_Notif_Type

Purpose	Type for Comm to Dcm notification function pointers.	
Type	struct	
Members	`ComM_Dcm_Notif_FctPtr FullComNotif`	
	`ComM_Dcm_Notif_FctPtr SilentComNotif`	
	`ComM_Dcm_Notif_FctPtr NoComNotif`	

5.3.2.1.10. ComM_InhibitionStatusType

Purpose	Inhibition Status Type.
Type	`ComM_ASR40_InhibitionStatusType`
Description	Provide AUTOSAR 4.0 InhibitionStatusType as default to other BSW modules

5.3.2.1.11. ComM_InitStatusType

Purpose	Initialization status of ComM.

Type	`uint8`

5.3.2.1.12. ComM_ModeType

Purpose	Mode Type.
Type	`ComM_ASR40_ModeType`
Description	Provide AUTOSAR 4.0 ComM_ModeType as default to other BSW modules

5.3.2.1.13. ComM_Multicore_SchmSend_FctPtr

Purpose	Function pointer type Master Core send signal.
Type	`Std_ReturnType(*)(const uint8 *data)`

5.3.2.1.14. ComM_NvDataType

Purpose	Inhibition status of ComM to be stored in NvM.	
Type	struct	
Members	`uint16 InhibitionCounter`	number of rejected Full ComMode request
	`ComM_ASR40_InhibitionStatusType ECUGroupClassification`	Current ECU group classification.
	`uint8 ChannelWakeUpInhibition`	Defines whether a WakeUp inhibition is active or not to Persistently stored. This array is derived by calculating the number of channels configured in ComM. Each bit corresponds to a channel. E.g : If there are more than 8 channels, size of COMM_NUM_BYTES_NVM will greater than 1

5.3.2.1.15. ComM_PncModeType

Purpose	Current mode of a PNC.
Type	`uint8`

5.3.2.1.16. ComM_RxSignal_Struct_Type

Purpose	Type for Rx signals configuration.	
Type	struct	
Members	`Com_SignalIdType ComHandleId`	

5.3.2.1.17. ComM_StateType

Purpose	State and sub-state of ComM state machine.
Type	`uint8`
Description	Range: 0 : COMM_NO_COM_NO_PENDING_REQUEST 1 : COMM_NO_COM_REQUEST_PENDING 2 : COMM_FULL_COM_NETWORK_REQUESTED 3 : COMM_FULL_COM_READY_SLEEP 4 : COMM_SILENT_COM

5.3.2.1.18. ComM_TxSignal_Struct_Type

Purpose	Type for Tx signals configuration.	
Type	struct	
Members	`Com_SignalIdType ComHandleId`	
	`uint8 TxSignalKind`	

5.3.2.1.19. ComM_UserHandleType

Purpose	User Handle Type.
Type	`ComM_ASR40_UserHandleType`
Description	Provide AUTOSAR 4.0 UserHandleType as default to other BSW modules

5.3.2.2. Macro constants

5.3.2.2.1. COMM_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.3.2.2.2. COMM_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	1U

5.3.2.2.3. COMM_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.3.2.2.4. COMM_BUS_TYPE_CAN

Purpose	Bus types used as values for COMM_BUSTYPE_OF_CHANNEL().
Value	0U

5.3.2.2.5. COMM_BUS_TYPE_ETH

Purpose	
Value	1U

5.3.2.2.6. COMM_BUS_TYPE_FR

Purpose	
Value	2U

5.3.2.2.7. COMM_BUS_TYPE_INTERNAL

Purpose	
Value	3U

5.3.2.2.8. COMM_BUS_TYPE_LIN

Purpose	
Value	4U

5.3.2.2.9. COMM_DET_REPORT_ERROR

Purpose	Macro for reporting an error to DET.
Value	((void)Det_ReportError(COMM_MODULE_ID, (InstanceId), (ApiId), (ErrorId)))

5.3.2.2.10. COMM_EIRA_ERA_ACTIVE

Purpose	
Value	0U

5.3.2.2.11. COMM_ERA_PASSIVE

Purpose	
Value	1U

5.3.2.2.12. COMM_E_BUSSM_REQUESTCOMMODE

Purpose	Master BusSm Request Com Mode error Id.
Value	246U

5.3.2.2.13. COMM_E_DCM_COMM_FULLCOMMODEENTERED

Purpose	Dcm Full Com Entered Notification error Id.
Value	245U

5.3.2.2.14. COMM_E_DCM_COMM_NOCOMMODEENTERED

Purpose	Dcm No Com Entered Notification error Id.
Value	243U

5.3.2.2.15. COMM_E_DCM_COMM_SILENTCOMMODEENTERED

Purpose	Dcm Silent Com Entered Notification error Id.

Value	244U

5.3.2.2.16. COMM_E_MASTER_DATA_GET

Purpose	Master Data Get error Id.
Value	247U

5.3.2.2.17. COMM_E_MASTER_DATA_SET

Purpose	Master Data Set error Id.
Value	248U

5.3.2.2.18. COMM_E_MODE_LIMITATION

Purpose	Function call has been successfully but mode can not be granted because of mode inhibition.
Value	0x02U

5.3.2.2.19. COMM_E_NOT_INITED

Purpose	Error Code.
Value	1U
Description	API service used without module initialization.

5.3.2.2.20. COMM_E_SLAVE_DATA_GET

Purpose	Slave Data Get error Id.
Value	249U

5.3.2.2.21. COMM_E_SLAVE_DATA_SET

Purpose	Slave Data Set error Id.

Value	250U

5.3.2.2.22. COMM_E_UNINIT

Purpose	Return value when ComM not initialized.
Value	3U

5.3.2.2.23. COMM_E_WRONG_PARAMETERS

Purpose	Error Code.
Value	2U
Description	API service used with wrong parameters, e.g. a NULL pointer.

5.3.2.2.24. COMM_FULL_COMMUNICATION

Purpose	Full Communication mode value for ComM_ModeType.
Value	2U

5.3.2.2.25. COMM_FULL_COM_NETWORK_REQUESTED

Purpose	Full_Com_Network_Requested value for ComM_StateType.
Value	2U

5.3.2.2.26. COMM_FULL_COM_READY_SLEEP

Purpose	Full_Com_Ready_Sleep value for ComM_StateType.
Value	3U

5.3.2.2.27. COMM_INIT

Purpose	COM Manager is initialized and usable.
Value	1U

5.3.2.2.28. COMM_INSTANCE_ID

Purpose	Id of instance of ComM.
Value	COMM_MASTER_CORE_ID

5.3.2.2.29. COMM_LIMIT_NOCOMM

Purpose	Macros for Communication Inhibition.
Value	2U

5.3.2.2.30. COMM_LIMIT_NOCOMM_MASK

Purpose	
Value	253U

5.3.2.2.31. COMM_MODULE_ID

Purpose	AUTOSAR module identification.
Value	12U

5.3.2.2.32. COMM_NM_FULL_VARIANT

Purpose	Nm variant types used for values of COMM_NM_VARIANT_OF_CHANNEL().
Value	0U

5.3.2.2.33. COMM_NM_LIGHT_VARIANT

Purpose	
Value	1U

5.3.2.2.34. COMM_NM_NONE_VARIANT

Purpose	
Value	2U

5.3.2.2.35. COMM_NM_PASSIVE_VARIANT

Purpose	
Value	3U

5.3.2.2.36. COMM_NO_COMMUNICATION

Purpose	No Communication mode value for ComM_ModeType.
Value	0U

5.3.2.2.37. COMM_NO_COM_NO_PENDING_REQUEST

Purpose	No_Com_No_Pending_Request value for ComM_StateType.
Value	0U

5.3.2.2.38. COMM_NO_COM_REQUEST_PENDING

Purpose	No_Com_Pending_Request value for ComM_StateType.
Value	1U

5.3.2.2.39. COMM_NO_INTERNAL_AND_EXTERNAL_REQUESTS

Purpose	
Value	0U

5.3.2.2.40. COMM_NO_INTERNAL_REQUESTS

Purpose	
Value	1U

5.3.2.2.41. COMM_NUM_CHANNELS_BYTES

Purpose	

Value	(((COMM_NUM_CHANNELS - 1U) / 8U) + 1U)

5.3.2.2.42. COMM_NUM_PNC_BYTES

Purpose	
Value	(((COMM_NUM_PNC - 1U) / 8U) + 1U)

5.3.2.2.43. COMM_PREVENT_WAKEUP

Purpose	
Value	1U

5.3.2.2.44. COMM_PREVENT_WAKEUP_MASK

Purpose	
Value	254U

5.3.2.2.45. COMM_SID_BUS_SM_INDICATION

Purpose	AUTOSAR API service ID.
Value	51U
Description	Definition of service ID for ComM_BusSM_ModeIndication().

5.3.2.2.46. COMM_SID_COMMUNICATIONALLOWED

Purpose	AUTOSAR API service ID.
Value	53U
Description	Definition of service ID for ComM_CommunicationAllowed().

5.3.2.2.47. COMM_SID_DCM_ACTIVE_DIAGNOSTIC

Purpose	AUTOSAR API service ID.

Value	31U
Description	Definition of service ID for ComM_DCM_ActiveDiagnostic().

5.3.2.2.48. COMM_SID_DCM_INACTIVE_DIAGNOSTIC

Purpose	AUTOSAR API service ID.
Value	32U
Description	Definition of service ID for ComM_DCM_InactiveDiagnostic().

5.3.2.2.49. COMM_SID_DEINIT

Purpose	AUTOSAR API service ID.
Value	2U
Description	Definition of service ID for ComM_DeInit().

5.3.2.2.50. COMM_SID_ECUM_PNC_WAKE_UP_INDICATION

Purpose	AUTOSAR API service ID.
Value	55U
Description	Definition of service ID for ComM_EcuM_WakeupIndiciation().

5.3.2.2.51. COMM_SID_ECUM_WAKE_UP_INDICATION

Purpose	AUTOSAR API service ID.
Value	42U
Description	Definition of service ID for ComM_EcuM_WakeupIndiciation().

5.3.2.2.52. COMM_SID_GET_CURRENT_COM_MODE

Purpose	AUTOSAR API service ID.
Value	8U
Description	Definition of service ID for ComM_GetCurrentComMode().

5.3.2.2.53. COMM_SID_GET_INHIBITION_STATUS

Purpose	AUTOSAR API service ID.
Value	4U
Description	Definition of service ID for ComM_GetInhibitionStatus().

5.3.2.2.54. COMM_SID_GET_MAX_COM_MODE

Purpose	AUTOSAR API service ID.
Value	6U
Description	Definition of service ID for ComM_GetMaxComMode().

5.3.2.2.55. COMM_SID_GET_REQUESTED_COM_MODE

Purpose	AUTOSAR API service ID.
Value	7U
Description	Definition of service ID for ComM_GetRequestedComMode().

5.3.2.2.56. COMM_SID_GET_STATE

Purpose	AUTOSAR API service ID.
Value	52U
Description	Definition of service ID for ComM_GetState().

5.3.2.2.57. COMM_SID_GET_STATUS

Purpose	AUTOSAR API service ID.
Value	3U
Description	Definition of service ID for ComM_GetStatus().

5.3.2.2.58. COMM_SID_GET_VERSION_INFO

Purpose	AUTOSAR API service ID.

Value	16U
Description	Definition of service ID for ComM_GetVersionInfo().

5.3.2.2.59. COMM_SID_INIT

Purpose	AUTOSAR API service ID.
Value	1U
Description	Definition of service ID for ComM_Init().

5.3.2.2.60. COMM_SID_LIMIT_CHANNEL_TO_NO_COM

Purpose	AUTOSAR API service ID.
Value	11U
Description	Definition of service ID for ComM_LimitChannelToNoComMOde().

5.3.2.2.61. COMM_SID_LIMIT_ECU_TO_NO_COM

Purpose	AUTOSAR API service ID.
Value	12U
Description	Definition of service ID for ComM_LimitECUToNoComMode().

5.3.2.2.62. COMM_SID_MAIN_FUNCTION

Purpose	AUTOSAR API service ID.
Value	0x60U
Description	Definition of service ID for ComM_MainFunction_<Channel_ID>().

5.3.2.2.63. COMM_SID_MASTER_DATA_GET

Purpose	Master Data Get SID.
Value	247U

5.3.2.2.64. COMM_SID_MASTER_DATA_SET

Purpose	Master Data Set SID.
Value	248U

5.3.2.2.65. COMM_SID_NM_BUS_SLEEP_MODE

Purpose	AUTOSAR API service ID.
Value	26U
Description	Definition of service ID for ComM_Nm_BusSleepMode().

5.3.2.2.66. COMM_SID_NM_NETWORK_MODE

Purpose	AUTOSAR API service ID.
Value	24U
Description	Definition of service ID for ComM_Nm_NetworkMode().

5.3.2.2.67. COMM_SID_NM_NETWORK_START_IND

Purpose	AUTOSAR API service ID.
Value	21U
Description	Definition of service ID for ComM_Nm_NetworkStartIndication().

5.3.2.2.68. COMM_SID_NM_PREPARE_BUS_SLEEP

Purpose	AUTOSAR API service ID.
Value	25U
Description	Definition of service ID for ComM_Nm_PrepareBusSleepMode().

5.3.2.2.69. COMM_SID_NM_RESTART_IND

Purpose	AUTOSAR API service ID.

Value	27U
Description	Definition of service ID for ComM_Nm_RestartIndication().

5.3.2.2.70. COMM_SID_PREVENT_WAKE_UP

Purpose	AUTOSAR API service ID.
Value	9U
Description	Definition of service ID for ComM_PreventWakeUp().

5.3.2.2.71. COMM_SID_READ_INHIBIT_COUNTER

Purpose	AUTOSAR API service ID.
Value	13U
Description	Definition of service ID for ComM_ReadInhibitCounter().

5.3.2.2.72. COMM_SID_REQUEST_COM_MODE

Purpose	AUTOSAR API service ID.
Value	5U
Description	Definition of service ID for ComM_GetRequestComMode().

5.3.2.2.73. COMM_SID_RESET_INHIBIT_COUNTER

Purpose	AUTOSAR API service ID.
Value	14U
Description	Definition of service ID for ComM_ResetInhibitCounter().

5.3.2.2.74. COMM_SID_SET_ECU_GROUP_CLASS

Purpose	AUTOSAR API service ID.
Value	15U
Description	Definition of service ID for ComM_SetECUGroupClassification().

5.3.2.2.75. COMM_SID_SLAVE_DATA_GET

Purpose	Slave Data Get SID.
Value	249U

5.3.2.2.76. COMM_SID_SLAVE_DATA_SET

Purpose	Slave Data Set SID.
Value	250U

5.3.2.2.77. COMM_SILENT_COM

Purpose	Silent Communication value for ComM_StateType.
Value	4U

5.3.2.2.78. COMM_SILENT_COMMUNICATION

Purpose	Silent Communication mode value for ComM_ModeType.
Value	1U

5.3.2.2.79. COMM_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	5U

5.3.2.2.80. COMM_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	19U

5.3.2.2.81. COMM_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	4U

5.3.2.2.82. COMM_UNINIT

Purpose	The COM Manager is not initialized or not usable. Default value after reset.
Value	0U

5.3.2.2.83. COMM_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.3.2.2.84. ComM_GetCurrentComMode

Purpose	Wrapping macro for ComM_GetCurrentComMode to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_GetCurrentComMode

5.3.2.2.85. ComM_GetInhibitionStatus

Purpose	Wrapping macro for ComM_GetInhibitionStatus to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_GetInhibitionStatus

5.3.2.2.86. ComM_GetMaxComMode

Purpose	Wrapping macro for ComM_GetMaxComMode to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_GetMaxComMode

5.3.2.2.87. ComM_GetRequestedComMode

Purpose	Wrapping macro for ComM_GetRequestedComMode to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_GetRequestedComMode

5.3.2.2.88. ComM_LimitChannelToNoComMode

Purpose	Wrapping macro for ComM_LimitChannelToNoComMode to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_LimitChannelToNoComMode

5.3.2.2.89. ComM_LimitECUToNoComMode

Purpose	Wrapping macro for ComM_LimitECUToNoComMode to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_LimitECUToNoComMode

5.3.2.2.90. ComM_PreventWakeUp

Purpose	Wrapping macro for ComM_PreventWakeUp to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_PreventWakeUp

5.3.2.2.91. ComM_ReadInhibitCounter

Purpose	Wrapping macro for ComM_ReadInhibitCounter to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_ReadInhibitCounter

5.3.2.2.92. ComM_RequestComMode

Purpose	Wrapping macro for ComM_RequestComMode to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_RequestComMode

5.3.2.2.93. ComM_ResetInhibitCounter

Purpose	Wrapping macro for ComM_ResetInhibitCounter to provide AUTOSAR 4.0 API as default to other BSW modules.

Value	ComM_ASR40_ResetInhibitCounter

5.3.2.2.94. ComM_SetECUGroupClassification

Purpose	Wrapping macro for ComM_SetECUGroupClassification to provide AUTOSAR 4.0 API as default to other BSW modules.
Value	ComM_ASR40_SetECUGroupClassification

5.3.2.2.95. PNC_FULL_COMMUNICATION

Purpose	PNC is able to communicate.
Value	4U

5.3.2.2.96. PNC_NO_COMMUNICATION

Purpose	PNC does not communicate.
Value	3U

5.3.2.2.97. PNC_PREPARE_SLEEP

Purpose	PNC is active with no deadline monitoring.
Value	2U

5.3.2.2.98. PNC_READY_SLEEP

Purpose	PNC is requested by a remote ComM user.
Value	1U

5.3.2.2.99. PNC_REQUESTED

Purpose	PNC is requested by a local ComM user.
Value	0U

5.3.2.3. Objects

5.3.2.3.1. ComM_BusSmRequestComModeMcFpList

Purpose	
Type	`const ComM_BusSM_RequestComMode_FctPtr`

5.3.2.3.2. ComM_BusTypeOfChannel

Purpose	Bus type of channel given by index.
Type	`const uint8`

5.3.2.3.3. ComM_ChanState

Purpose	ComM Channel State.
Type	`ComM_StateType`
Description	This variable holds the current state of the ComM Channel.

5.3.2.3.4. ComM_ChannelInhibitionStatusInit

Purpose	Default ComMNoCom.
Type	`const ComM_ASR40_InhibitionStatusType`
Description	Default value configured for ComMNoCom

5.3.2.3.5. ComM_ChannelMode

Purpose	ComM Channel Mode.
Type	`ComM_ASR40_ModeType`
Description	This variable holds the current mode of the ComM Channel.

5.3.2.3.6. ComM_ChannelModePrevious

Purpose	ComM Channel Mode.

Type	ComM_ASR40_ModeType
Description	This variable holds the current mode of the ComM Channel before ComM_ClearUser-Request is called.

5.3.2.3.7. ComM_Communication_Allowed

Purpose	number of bytes needed to store CommunicationAllowed flags
Type	`uint8`

5.3.2.3.8. ComM_DcmActiveStatus

Purpose	Flag to be set if DCM indicated active communication.
Type	`uint8`

5.3.2.3.9. ComM_Dcm_Notif_FpList

Purpose	
Type	`const ComM_Dcm_Notif_Type`

5.3.2.3.10. ComM_EB_NoWakeupNvStorage

Purpose	Persistent storage of WakeUp inhibition status of channels in NvM.
Type	`const uint8`

5.3.2.3.11. ComM_EB_NvRAM

Purpose	
Type	ComM_NvDataType

5.3.2.3.12. ComM_EB_NvROM

Purpose	Inhibition status of ComM in ROM.

Type	`const ComM_NvDataType`
Description	Default Inhibition status of ComM to be stored in NvM

5.3.2.3.13. ComM_InitStatus

Purpose	ComM Initialization status.
Type	`ComM_InitStatusType`
Description	This variable holds the current initialization status of the ComM module.

5.3.2.3.14. ComM_NetReqNoNmTimeoutMs

Purpose	NetReqNoNmTimeout.
Type	`const uint16`
Description	These variables hold the timeout value for Full Com Network Request state in ms. After this time the Full Com Network Request state is left and the the transition to the state "ready sleep" is performed. This timeout is only relevant and used for channels of Nm variant NONE and LIGHT.

5.3.2.3.15. ComM_NmVariantOfChannel

Purpose	NM variant of channel given by index.
Type	`const uint8`

5.3.2.3.16. ComM_PNCRequestedComMode

Purpose	ComM PNC requested mode.
Type	`ComM_ASR40_ModeType`
Description	This variable holds the current PNC requested mode.

5.3.2.3.17. ComM_PNCRequestedComMode0PncVectorAvoidance

Purpose	ComM PNC requested mode.
Type	`ComM_ASR40_ModeType`

Description	This variable holds the current PNC requested mode.

5.3.2.3.18. ComM_PncRxActiveEraMask

Purpose	Bit array of Rx Eira Com signal masks.
Type	`const uint8 *const`
Description	A PNC may not have references to all RX ERA Com signals. Therefore, the bits corresponding to PNCs which do not reference a Com signal have to be masked out from the newly received ERA before being merged to the existing ERA bits. This array contains references to bit arrays for each RX ERA Com signal. If a Com Signal is associated to a PNC, then the bit corresponding to that PNC in the bit array will be set to 1. If a Com Signal is not associated to a PNC, then the corresponding bit in the bit array will be set to 0.

5.3.2.3.19. ComM_PncRxEiraMask

Purpose	Bit array of Rx Eira Com signal masks.
Type	`const uint8 *const`
Description	A PNC may not have references to all RX EIRA Com signals. Therefore, the bits corresponding to PNCs which do not reference a Com signal have to be masked out from the newly received EIRA before being merged to the existing EIRA bits. This array contains references to bit arrays for each RX EIRA Com signal. If a Com Signal is associated to a PNC, then the bit corresponding to that PNC in the bit array will be set to 1. If a Com Signal is not associated to a PNC, then the corresponding bit in the bit array will be set to 0.

5.3.2.3.20. ComM_PncRxUnfilteredEraMask

Purpose	
Type	`const uint8 *const`

5.3.2.3.21. ComM_PncStateGWE

Purpose	Current state of PNC only available if PNC Gateway is enabled.
Type	`ComM_PncModeType`

5.3.2.3.22. ComM_PncZeroVectorAvoidanceRelease

Purpose	Request the network every time when entering Pnc_Requested state if ComM0PncVectorAvoidance configured as TRUE.
Type	`const uint8`

5.3.2.3.23. ComM_ReadySleepNoNmTimeoutMs

Purpose	Ready Sleep Timeout.
Type	`const uint16`
Description	These variables hold the timeout value for ready sleep state in ms. After this time the ready sleep state is left and the the transition to the state "no communication" is performed. This timeout is only relevant and used for channels of Nm variant NONE and LIGHT. For channels of NmVariant LIGHT the timeout value is given by the configuration parameter ComMNmLightTimeout.

5.3.2.3.24. ComM_RequestedComMode

Purpose	ComM Requested Mode.
Type	`ComM_ASR40_ModeType`
Description	This variable holds the currently requested mode.

5.3.2.3.25. ComM_RxActiveEra

Purpose	This variable holds the received ERA signal.
Type	`uint8`

5.3.2.3.26. ComM_RxActiveEraSignalStatus

Purpose	Holds the Status flag indicating changes in the ERA Active signals.
Type	`boolean`

5.3.2.3.27. ComM_RxComSignalActiveEraCfg

Purpose	Array of Rx ERA ComSignal Handle Ids sorted in the ascending order.

Type	`const ComM_RxSignal_Struct_Type`

5.3.2.3.28. ComM_RxComSignalEiraCfg

Purpose	Array of Rx EIRA ComSignal Handle Ids sorted in the ascending order.
Type	`const ComM_RxSignal_Struct_Type`

5.3.2.3.29. ComM_RxComSignalUnfilteredEraCfg

Purpose	Array of Rx ERA ComSignal Handle Ids sorted in the ascending order.
Type	`const ComM_RxSignal_Struct_Type`

5.3.2.3.30. ComM_RxEira

Purpose	This variable holds the received EIRA signal.
Type	`uint8`

5.3.2.3.31. ComM_RxEiraSignalStatus

Purpose	Holds the Status flag indicating changes in the EIRA signals.
Type	`boolean`

5.3.2.3.32. ComM_RxUnfilteredEra

Purpose	
Type	`uint8`

5.3.2.3.33. ComM_RxUnfilteredEraSignalStatus

Purpose	Holds the Status flag indicating changes in the ERA Passive signals.
Type	`boolean`

5.3.2.3.34. ComM_StoreNoWakeUpInhibition

Purpose	Storage for NoWakeUpInhibition in the NvM block pointed by ComMGlobalNvmBlockDescriptor.
Type	`const ComM_ASR40_InhibitionStatusType`

5.3.2.3.35. ComM_TxEiraEraActive

Purpose	This variable holds data for sending Eira signal.
Type	`uint8`

5.3.2.3.36. ComM_TxEraPassive

Purpose	This variable holds data for sending Eira signal.
Type	`uint8`

5.3.2.3.37. ComM_UserRequestedComMode

Purpose	ComM User Requested Mode.
Type	`ComM_ASR40_ModeType`
Description	This variable holds the current user requested mode.

5.3.2.4. Functions

5.3.2.4.1. ComM_ASR32_GetCurrentComMode

Purpose	Retrieve the current mode of ComM via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_GetCurrentComMode`** `( ComM_ASR32_UserHandleType User , ComM_ASR32_ModeType * ComMode );`	
Service ID	COMM_SID_GET_CURRENT_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`User`	Handle of the user for which the query shall be performed.

	ComMode	Current communication mode.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Getting of Current Communication Mode failed.
Description	This function retrieves the current communication mode of the ComM state machine.	

5.3.2.4.2. ComM_ASR32_GetInhibitionStatus

Purpose	Retrieve inhibition status via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_GetInhibitionStatus`** `( NetworkHandleType Channel , ComM_ASR32_InhibitionStatusType * Status );`	
Service ID	COMM_SID_GET_INHIBITION_STATUS	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	Channel	Network channel on which the mode transition has occurred.
Parameters (out)	Status	Current inhibition status of ComM.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Getting of Inhibition Status failed.
Description	This function returns the inhibition status of a ComM channel.	

5.3.2.4.3. ComM_ASR32_GetMaxComMode

Purpose	Retrieve the maximum allowed communication mode via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_GetMaxComMode`** `( ComM_ASR32_UserHandleType User , ComM_ASR32_ModeType * ComMode );`	
Service ID	COMM_SID_GET_MAX_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	User	Handle of the user for which the query shall be performed.

Parameters (out)	ComMode	Maximum allowed communication mode.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Getting of Maximum Communication Mode failed.
Description	This function retrieves the maximum allowed communication mode of the corresponding user.	

5.3.2.4.4. ComM_ASR32_GetRequestedMode

Purpose	Retrieve the communication mode requested by a user via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_GetRequestedMode`** `( ComM_ASR32_UserHandleType User , ComM_ASR32_ModeType * ComMode );`	
Service ID	COMM_SID_GET_REQUESTED_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	User	Handle of the user for which the query shall be performed.
Parameters (out)	ComMode	Communication mode that has been requested most recently by that user.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Getting of the Requested Communication Mode failed.
Description	This function retrieves the communication mode that has been requested most recently by the given user.	

5.3.2.4.5. ComM_ASR32_LimitChannelToNoComMode

Purpose	Limit channel to "No Communication" via AUTOSAR 3.2 API.
Synopsis	`Std_ReturnType` **`ComM_ASR32_LimitChannelToNoComMode`** `( NetworkHandleType Channel , boolean Status );`
Service ID	COMM_SID_LIMIT_CHANNEL_TO_NO_COM

Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Channel`	Channel on which a limitations will be en(-dis)abled.
	`Status`	En(-dis)ables channel limitation.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Limiting of channel failed.
Description	This function en(-dis)ables a limitation for a channel.	

5.3.2.4.6. ComM_ASR32_LimitECUToNoComMode

Purpose	Limit ECU to "No Communication" via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_LimitECUToNoComMode`** `( boolean Status );`	
Service ID	COMM_SID_LIMIT_ECU_TO_NO_COM	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Status`	En(-dis)ables limitation.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Limiting of ECU failed.
Description	This function en(-dis)ables a limitation for a ECU.	

5.3.2.4.7. ComM_ASR32_PreventWakeUp

Purpose	Changes Wakeup inhibition status for requested channel via AUTOSAR 3.2 API.
Synopsis	`Std_ReturnType` **`ComM_ASR32_PreventWakeUp`** `( NetworkHandleType Channel , boolean Status );`
Service ID	COMM_SID_PREVENT_WAKE_UP
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	`Channel`	on which a Wake Up inhibition will be en(-dis)abled
	`Status`	En(-dis)ables Wake up Inhibition.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Change of Wakeup Inhibition Status failed
Description	This function allows user to enable or disable the Wake Up inhibition for a requested channel.	

5.3.2.4.8. ComM_ASR32_ReadInhibitCounter

Purpose	Return the amount of rejected user requests via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_ReadInhibitCounter`** `( uint16 * CounterValue );`	
Service ID	COMM_SID_READ_INHIBIT_COUNTER	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`CounterValue`	Amount of rejected user requests.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Reading of Inhibit Counter failed.
Description	This function returns the amount of rejected "Full Communication" user requests.	

5.3.2.4.9. ComM_ASR32_RequestComMode

Purpose	Request of a communication mode by a user via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_RequestComMode`** `( ComM_ASR32_UserHandleType User , ComM_ASR32_ModeType ComMode );`	
Service ID	COMM_SID_REQUEST_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`User`	Handle of the user requesting the communication mode.

	`ComMode`	Desired communication mode.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Requesting of Communication Mode failed.
Description	This function requests the given communication mode for the given user. If Rte usage is enables in the ComM configuration the declaration of this function is provided by Rte_ComM.h.	

5.3.2.4.10. ComM_ASR32_ResetInhibitCounter

Purpose	Reset inhibition counter via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_ResetInhibitCounter`** `( void );`	
Service ID	COMM_SID_RESET_INHIBIT_COUNTER	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Resetting of Inhibit Counter failed.
Description	This function resets the "Inhibited Full Communication Request Counter".	

5.3.2.4.11. ComM_ASR32_SetECUGroupClassification

Purpose	Change the ECU Group Classification status via AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`ComM_ASR32_SetECUGroupClassification`** `( ComM_ASR32_InhibitionStatusType Status );`	
Service ID	COMM_SID_SET_ECU_GROUP_CLASS	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Status`	Inhibition status.
Return Value	Standard Return Code	
	`E_OK`	No Errors.

	E_NOT_OK	Setting of the ECU Group Classification failed.
Description	This function changes the ECU Group Classification status.	

5.3.2.4.12. ComM_ASR40_GetCurrentComMode

Purpose	Retrieve the current mode of ComM.	
Synopsis	`Std_ReturnType` **`ComM_ASR40_GetCurrentComMode`** `( ComM_ASR40_UserHandleType User , ComM_ASR40_ModeType * ComMode );`	
Service ID	COMM_SID_GET_CURRENT_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	User	Handle of the user for which the query shall be performed.
	ComMode	Current communication mode.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Getting of Current Communication Mode failed.
Description	This function retrieves the current communication mode of the ComM state machine.	

5.3.2.4.13. ComM_ASR40_GetInhibitionStatus

Purpose	Retrieve inhibition status.	
Synopsis	`Std_ReturnType` **`ComM_ASR40_GetInhibitionStatus`** `( NetworkHandleType Channel , ComM_ASR40_InhibitionStatusType * Status );`	
Service ID	COMM_SID_GET_INHIBITION_STATUS	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	Channel	Network channel on which the mode transition has occurred.
Parameters (out)	Status	Current inhibition status of ComM.

Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Getting of Inhibition Status failed.
Description	This function returns the inhibition status of a ComM channel.	

5.3.2.4.14. ComM_ASR40_GetMaxComMode

Purpose	Retrieve the maximum allowed communication mode.	
Synopsis	Std_ReturnType **ComM_ASR40_GetMaxComMode** (ComM_ASR40_UserHandleType User , ComM_ASR40_ModeType * ComMode);	
Service ID	COMM_SID_GET_MAX_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	User	Handle of the user for which the query shall be performed.
Parameters (out)	ComMode	Maximum allowed communication mode.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Getting of Maximum Communication Mode failed.
Description	This function retrieves the maximum allowed communication mode of the corresponding user.	

5.3.2.4.15. ComM_ASR40_GetRequestedComMode

Purpose	Retrieve the communication mode requested by a user.	
Synopsis	Std_ReturnType **ComM_ASR40_GetRequestedComMode** (ComM_ASR40_UserHandleType User , ComM_ASR40_ModeType * ComMode);	
Service ID	COMM_SID_GET_REQUESTED_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	User	Handle of the user for which the query shall be performed.

Parameters (out)	`ComMode`	Communication mode that has been requested most recently by that user.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Getting of the Requested Communication Mode failed.
Description	This function retrieves the communication mode that has been requested most recently by the given user.	

5.3.2.4.16. ComM_ASR40_LimitChannelToNoComMode

Purpose	Limit channel to "No Communication".	
Synopsis	`Std_ReturnType` **`ComM_ASR40_LimitChannelToNoComMode`** `( NetworkHandleType Channel , boolean Status );`	
Service ID	COMM_SID_LIMIT_CHANNEL_TO_NO_COM	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Channel`	Channel on which a limitations will be en(-dis)abled.
	`Status`	En(-dis)ables channel limitation.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Limiting of channel failed.
Description	This function en(-dis)ables a limitation for a channel.	

5.3.2.4.17. ComM_ASR40_LimitECUToNoComMode

Purpose	Limit ECU to "No Communication".
Synopsis	`Std_ReturnType` **`ComM_ASR40_LimitECUToNoComMode`** `( boolean Status );`
Service ID	COMM_SID_LIMIT_ECU_TO_NO_COM
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	Status	En(-dis)ables limitation.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Limiting of ECU failed.
Description	This function en(-dis)ables a limitation for a ECU.	

5.3.2.4.18. ComM_ASR40_PreventWakeUp

Purpose	Changes Wakeup inhibition status for requested channel.	
Synopsis	Std_ReturnType **ComM_ASR40_PreventWakeUp** (NetworkHandleType Channel , boolean Status);	
Service ID	COMM_SID_PREVENT_WAKE_UP	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	Channel	on which a Wake Up inhibition will be en(-dis)abled
	Status	En(-dis)ables Wake up Inhibition.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Change of Wakeup Inhibition Status failed
Description	This function allows user to enable or disable the Wake Up inhibition for a requested channel.	

5.3.2.4.19. ComM_ASR40_ReadInhibitCounter

Purpose	Return the amount of rejected user requests.	
Synopsis	Std_ReturnType **ComM_ASR40_ReadInhibitCounter** (uint16 * CounterValue);	
Service ID	COMM_SID_READ_INHIBIT_COUNTER	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	CounterValue	Amount of rejected user requests.

Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Reading of Inhibit Counter failed.
Description	This function returns the amount of rejected "Full Communication" user requests.	

5.3.2.4.20. ComM_ASR40_RequestComMode

Purpose	Request of a communication mode by a user.	
Synopsis	`Std_ReturnType` **`ComM_ASR40_RequestComMode`** `( ComM_ASR40_UserHandleType User , ComM_ASR40_ModeType ComMode );`	
Service ID	COMM_SID_REQUEST_COM_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`User`	Handle of the user requesting the communication mode.
	`ComMode`	Desired communication mode.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	Requesting of Communication Mode failed.
Description	This function requests the given communication mode for the given user. If Rte usage is enables in the ComM configuration the declaration of this function is provided by Rte_ComM.h.	

5.3.2.4.21. ComM_ASR40_ResetInhibitCounter

Purpose	Reset inhibition counter.
Synopsis	`Std_ReturnType` **`ComM_ASR40_ResetInhibitCounter`** `( void );`
Service ID	COMM_SID_RESET_INHIBIT_COUNTER
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Return Value	Standard Return Code

	E_OK	No Errors.
	E_NOT_OK	Resetting of Inhibit Counter failed.
Description	This function resets the "Inhibited Full Communication Request Counter".	

5.3.2.4.22. ComM_ASR40_SetECUGroupClassification

Purpose	Change the ECU Group Classification status.	
Synopsis	Std_ReturnType **ComM_ASR40_SetECUGroupClassification** (ComM_ASR40_InhibitionStatusType Status);	
Service ID	COMM_SID_SET_ECU_GROUP_CLASS	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	Status	Inhibition status.
Return Value	Standard Return Code	
	E_OK	No Errors.
	E_NOT_OK	Setting of the ECU Group Classification failed.
Description	This function changes the ECU Group Classification status.	

5.3.2.4.23. ComM_BitArrayClear

Purpose	Clear the x-th bit in a bit array.	
Synopsis	void **ComM_BitArrayClear** (uint8 * arr , uint16 idx);	
Parameters (in)	arr	array of uint8
	idx	number of bit to be set to 0

5.3.2.4.24. ComM_BitArrayIsBitSet

Purpose	Test the x-th bit in a bit array.	
Synopsis	boolean **ComM_BitArrayIsBitSet** (const uint8 * arr , uint16 idx);	
Parameters (in)	arr	array of uint8

	`idx`	number of bit to be tested
Return Value	TRUE or FALSE whether x-th bit is set or not.	

5.3.2.4.25. ComM_BitArraySet

Purpose	Set the x-th bit in a bit array.	
Synopsis	`void` **`ComM_BitArraySet`** `( uint8 * arr , uint16 idx );`	
Parameters (in)	`arr`	array of uint8
	`idx`	number of bit to be set to 1

5.3.2.4.26. ComM_BusSM_ModeIndication

Purpose	Indicate that BusSM has changed its communication mode.	
Synopsis	`void` **`ComM_BusSM_ModeIndication`** `( NetworkHandleType Channel , uint8 * ComMode );`	
Service ID	COMM_SID_BUS_SM_INDICATION	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	Network channel on which the mode transition has occurred.
	`ComMode`	New communication mode of the BusSM.
Description	This function provides an indication that the BusSM has changed its communication mode. The new mode is supplied as a parameter.	

5.3.2.4.27. ComM_ChannelNoComInhibitionStatus

Purpose	Evaluate NoCom Inhibition for ComM Channels This function checks the ComM channel NoCom Inhibition is set.	
Synopsis	`boolean` **`ComM_ChannelNoComInhibitionStatus`** `( uint8 instIdx );`	
Parameters (in)	`instIdx`	index of state machine instance
Return Value	Status of NoCom inhibition setting.	
	`TRUE`	if inhibition is set.
	`FALSE`	if inhibition is not set.

5.3.2.4.28. ComM_ChannelWakeUpInhibitionStatus

Purpose	Evaluate WakeUp Inhibition for ComM Channels This function checks the ComM channel WakeUp Inhibition is set.	
Synopsis	boolean **ComM_ChannelWakeUpInhibitionStatus** (uint8 instIdx);	
Parameters (in)	instIdx	index of state machine instance
Return Value	Status of WakeUp inhibition setting.	
	TRUE	if inhibition is set.
	FALSE	if inhibition is not set.

5.3.2.4.29. ComM_CommunicationAllowed

Purpose	Indication to ComM when communication is allowed .	
Synopsis	void **ComM_CommunicationAllowed** (NetworkHandleType Channel , boolean Allowed);	
Parameters (in)	Channel	Network channel on which communication is allowed.
	Allowed	communication is allowed (true) or not (false)
Description	EcuM or BswM shall indicate to ComM when communication is allowed.	

5.3.2.4.30. ComM_DCM_ActiveDiagnostic

Purpose	Indicate an active diagnostic session by DCM.	
Synopsis	void **ComM_DCM_ActiveDiagnostic** (NetworkHandleType Channel);	
Service ID	COMM_SID_DCM_ACTIVE_DIAGNOSTIC	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	Network channel needed for Diagnostic communication.
Description	This function provides an indication that DCM is currently handling an active diagnostic session on the channel given by ComM's configuration. This implies that for the respective channels, network mode must be requested.	

5.3.2.4.31. ComM_DCM_InactiveDiagnostic

Purpose	Indicate an inactive diagnostic session by DCM.	
Synopsis	`void ComM_DCM_InactiveDiagnostic ( NetworkHandleType Channel );`	
Service ID	COMM_SID_DCM_INACTIVE_DIAGNOSTIC	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	Network channel no longer needed for Diagnostic communication.
Description	This function provides an indication that DCM is currently not handling an active diagnostic session on the channel given by ComM's configuration. This implies that for the respective channels, network mode can be released when no other user requires the respective channel.	

5.3.2.4.32. ComM_DeInit

Purpose	Deinitialize ComM module.
Synopsis	`void ComM_DeInit ( void );`
Service ID	COMM_SID_DEINIT
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function deinitializes (terminates) the ComM module. De-init shall only be performed if all channels controlled by ComM are in "No Communication" state. The function call will be ignored if the state is not "No Communication".

5.3.2.4.33. ComM_EcuM_PNCWakeUpIndication

Purpose	Notification of a wake up on the corresponding partial network cluster.
Synopsis	`void ComM_EcuM_PNCWakeUpIndication ( PNCHandleType PNCid );`
Service ID	COMM_SID_ECUM_PNC_WAKE_UP_INDICATION
Sync/Async	Synchronous

Reentrancy	Reentrant	
Parameters (in)	`PNCid`	Partial network index
Description	This function provides an indication that the ECU State Manager has received a wakeup event for the given PNC.	

5.3.2.4.34. ComM_EcuM_WakeUpIndication

Purpose	Indicate that the ECU State Manager has received a wakeup.	
Synopsis	`void ComM_EcuM_WakeUpIndication ( NetworkHandleType Channel );`	
Service ID	COMM_SID_ECUM_WAKE_UP_INDICATION	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	Network channel on which the mode transition has occurred.
Description	This function provides an indication that the ECU State Manager has received a wakeup event for the given channel.	

5.3.2.4.35. ComM_EvaluatePnc

Purpose	Evaluate ComM Partial Network Cluster This function checks the ComM PNC mode and emits specific event related to the mode of the PNC.	
Synopsis	`void ComM_EvaluatePnc ( uint8 Pnc , ComM_ASR40_ModeType ComMode );`	
Parameters (in)	`Pnc`	Partial Network Cluster on which the mode transition has occurred.
	`ComMode`	Maximum allowed communication mode.

5.3.2.4.36. ComM_GetState

Purpose	Return current state.	
Synopsis	`Std_ReturnType ComM_GetState ( NetworkHandleType Channel , ComM_StateType * State );`	
Service ID	COMM_SID_GET_STATE	

Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Channel`	
Parameters (out)	`State`	(state of ComM.)
Return Value	Standard Return Code	
	`E_OK`	Function Successfully return current state of ComM state machine.
	`E_NOT_OK`	Return of current state of ComM state machine failed .
	`COMM_E_UNINIT`	ComM not initialized
Description	This function returns current state, including sub-state, of the ComM state machine.	

5.3.2.4.37. ComM_GetStatus

Purpose	Return the initialization status of the ComM module.	
Synopsis	`Std_ReturnType` **`ComM_GetStatus`** `( ComM_InitStatusType * Status );`	
Service ID	COMM_SID_GET_STATUS	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`Status`	Current initialization status of ComM.
Return Value	Standard Return Code	
	`E_OK`	Function serviced successfully.
	`E_NOT_OK`	Function execution failed.
Description	This function returns the initialization status of the ComM module.	

5.3.2.4.38. ComM_GetVersionInfo

Purpose	Get version information of the ComM module.
Synopsis	`void` **`ComM_GetVersionInfo`** `( Std_VersionInfoType * VersionInfoPtr );`
Service ID	COMM_SID_GET_VERSION_INFO
Sync/Async	Synchronous

Reentrancy	Non-Reentrant	
Parameters (out)	`VersionInfoPtr`	Pointer to where to store the version information of this module.
Description	This service returns the version information of this module. The version information includes: ▶ Module Id ▶ Vendor Id ▶ Vendor specific version numbers	

5.3.2.4.39. ComM_Init

Purpose	Initialize the ComM module.	
Synopsis	`void` **`ComM_Init`** `( const ComM_ConfigType * ConfigPtr );`	
Service ID	COMM_SID_INIT	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`ConfigPtr`	- Pointer to a selected configuration structure.
Description	This function initializes the ComM module by setting all internal state variables to defined values. Precondition: This API function has to be called before any other function (except ComM_GetStatus()) can be called.	

5.3.2.4.40. ComM_IsValidConfig

Purpose	Validates the post-build configuration data structure.	
Synopsis	`Std_ReturnType` **`ComM_IsValidConfig`** `( const void * voidConfigPtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`voidConfigPtr`	- pointer to a ComM post-build data structure. If a NULL_PTR is passed, the ComM will attempt to retrieve the ComM post-

		build configuration from the PbcfgM module.
Return Value	Standard Return Code	
	`E_OK`	When the pre-compile, link-time and platform hash values stored within the post-build structure correspond to the hash values of the compiled source files.
	`E_NOT_OK`	Otherwise, E_NOT_OK will be returned.
Description	This function validates the post-build configuration data structure passed to the ComM_Init function.	

5.3.2.4.41. ComM_Nm_BusSleepMode

Purpose	Notify that the network management has entered Bus Sleep Mode.	
Synopsis	`void` **`ComM_Nm_BusSleepMode`** `( NetworkHandleType Channel );`	
Service ID	COMM_SID_NM_BUS_SLEEP_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	Network channel on which the mode transition has occurred.
Description	This function provides a notification that the network management has entered Bus Sleep Mode.	

5.3.2.4.42. ComM_Nm_NetworkMode

Purpose	Notify that the network management has entered Network Mode.	
Synopsis	`void` **`ComM_Nm_NetworkMode`** `( NetworkHandleType Channel );`	
Service ID	COMM_SID_NM_NETWORK_MODE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	Network channel on which the mode transition has occurred.
Description	This function provides a notification that the network management has entered Network Mode.	

5.3.2.4.43. ComM_Nm_NetworkStartIndication

Purpose	Indicate reception of NM PDU in Bus Sleep Mode.	
Synopsis	`void` **`ComM_Nm_NetworkStartIndication`** `( NetworkHandleType Channel );`	
Service ID	COMM_SID_NM_NETWORK_START_IND	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Channel`	Network channel on which the NM PDU has been received.
Description	This function provides an indication that a NM PDU has been received in the Bus Sleep Mode, which indicates that some nodes in the network have already entered the Network Mode.	

5.3.2.4.44. ComM_Nm_PrepareBusSleepMode

Purpose	Notify that the network management has entered Prepare Bus Sleep Mode.	
Synopsis	`void` **`ComM_Nm_PrepareBusSleepMode`** `( NetworkHandleType Channel );`	
Service ID	COMM_SID_NM_PREPARE_BUS_SLEEP	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`Channel`	Network channel on which the mode transition has occurred.
Description	This function provides a notification that the network management has entered Prepare Bus Sleep Mode.	

5.3.2.4.45. ComM_Nm_RestartIndication

Purpose	Indicate a restart of NM.
Synopsis	`void` **`ComM_Nm_RestartIndication`** `( NetworkHandleType Channel );`
Service ID	COMM_SID_NM_RESTART_IND
Sync/Async	Synchronous

Reentrancy	Reentrant	
Parameters (in)	`Channel`	Network channel on which the NM PDU has been received.
Description	This function provides an indication that NmIf has started to shut down the coordinated buses, and not all coordinated buses have indicated bus sleep state, and on at least one of the coordinated buses NM is restarted.	

5.3.2.4.46. ComM_PncMainFunction

Purpose	Main processing function of PNC.
Synopsis	`void ComM_PncMainFunction ( void );`

5.3.2.4.47. ComM_PncProcessTimers

Purpose	Timer processing function of PNC.
Synopsis	`void ComM_PncProcessTimers ( void );`

5.3.2.4.48. ComM_ProcessTimers

Purpose	Processes the timer for a channel.	
Synopsis	`void ComM_ProcessTimers ( uint8 Channel );`	
Parameters (in)	`Channel`	Channel for which the timers are processed
Description	This function decreases the timer of the specified channel and emits an event, when the timer expires.	

5.3.2.4.49. ComM_SetChannelComMInhibitionStatus

Purpose	Set Inhibition status for ComM Channels This function set and resets the ComM channel Inhibition.	
Synopsis	`void ComM_SetChannelComMInhibitionStatus ( uint8 instIdx , ComM_ASR40_InhibitionStatusType Inhibitiontype , boolean Status );`	
Parameters (in)	`instIdx`	index of state machine instance

	Inhibitiontype	Inhibition type of ComM channel.
	Status	to be set.

5.3.3. Integration notes

5.3.3.1. Exclusive areas

This section describes the exclusive areas used by the ComM module.

5.3.3.1.1. SCHM_COMM_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the EB tresos AutoCore Generic documentation. Refer to the section Mapping exclusive areas in the basic software modules in the Integration notes section for details.

5.3.3.2. Production errors

Production errors are not reported by the ComM module.

5.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section Memory mapping and compiler abstraction in the Integration notes section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
VAR_CLEARED_16
VAR_INIT_16
VAR_INIT_8

CONST_32
CONST_16
CONST_8
VAR_CLEARED_8
VAR_CLEARED_UNSPECIFIED
VAR_INIT_UNSPECIFIED
VAR_POWER_ON_CLEARED_UNSPECIFIED
CONFIG_DATA_UNSPECIFIED
CONST_UNSPECIFIED
CODE

5.3.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the ComM module.

5.4. EcuM

5.4.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
EcuMConfiguration	1..1	**Label:** Configuration This container contains the configuration (parameters) of the ECU State Manager

Containers included		
EcuMDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
EcuMFixedGeneral	0..1	*The functionality related to this parameter is not supported by the current implementation.* This container holds the general, pre-compile configuration parameters for the EcuMFixed.
EcuMFlexGeneral	1..1	This container holds the general, pre-compile configuration parameters for the flexible ECU management.
EcuMGeneral	1..1	**Label:** Pre-Compile Configuration Parameter This container holds the general, pre-compile configuration parameters.
ReportToDem	1..1	**Label:** Production error handling Production error handling
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by Common-PublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT	
Label	Config Variant	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	VariantPostBuild	
Range	VariantPostBuild	
Configuration class	**VariantPostBuild:**	VariantPostBuild

5.4.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	15
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	4
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ModuleId
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	10
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	VendorId
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

5.4.1.2. EcuMConfiguration

Containers included		
Container name	Multiplicity	Description
EcuMCommonConfiguration	1..1	This container contains the common configuration (parameters) of the ECU State Manager

Containers included		
EcuMFixedConfiguration	0..1	This container contains the configuration (parameters) of the EcuMFixed. Only applicable if EcuMFixed is implemented. Dependency: ▶ The parameter `EcuMRteUsage` shall be enabled.
EcuMFlexConfiguration	1..1	**Label:** EcuM Flex Configuration This container contains the configuration (parameters) of the EcuMFlex.

5.4.1.3. EcuMCommonConfiguration

Containers included		
Container name	**Multiplicity**	**Description**
EcuMDefaultShutdownTarget	1..1	**Label:** Default Shutdown Target This container describes the default shutdown target to be selected by EcuM. The actual shutdown target may be overridden by the `EcuM_SelectShutdownTarget()` service.
EcuMDriverInitListOne	0..1	**Label:** Module Initialization List One Container for Init Block I. This container holds a list of module IDs that will be initialized. Each module in the list will be called for initialization in the list order. All modules in this list are initialized before the OS is started and so these modules require no OS support.
EcuMDriverInitListZero	0..1	**Label:** Module Initialization List Zero Container for Init Block 0. This container holds a list of module IDs that will be initialized. Each module in the list will be called for initialization in the list order. All modules in this list are initilialized before the post-build configuration has been loaded and the OS is initialized. Therefore, these modules may not use post-build configuration.
EcuMDriverRestartList	0..1	List of module IDs that shall be re-initialized during WakeupRestart. Each module in the list will be called for initialization in the list order by the callout `EcuM_AL_DriverRestart()`. Restriction:

Containers included		
		▶ Entries in this list must appear in the same order as in the list of `EcuM_DriverInitListOne` and `EcuM_DriverInitListZero`. EcuMDriverRestartList shall only contain a subset of the combined list of init block 0 and init block 1 drivers.
EcuMSleepMode	1..n	This container describes the configured sleep mode. The name of this container specifies the symbolic name of the sleep mode.
EcuMWakeupSource	5..32	Configuration of Wakeup Sources. The short name of this container will be available as pre-processor define which expands to values corresponding to the bit positions(wakeup Source ID) of this wakeup Source. *Attention:* For the first five wakeup sources standardized names will be used: ▶ `Wakeup Source ID: 0`: ECUM_WKSOURCE_POWER ▶ `Wakeup Source ID: 1`: ECUM_WKSOURCE_RESET ▶ `Wakeup Source ID: 2`: ECUM_WKSOURCE_INTERNAL_RESET ▶ `Wakeup Source ID: 3`: ECUM_WKSOURCE_INTERNAL_WDG ▶ `Wakeup Source ID: 4`: ECUM_WKSOURCE_EXTERNAL_WDG
EcuMDemEventParameterRefs	0..1	References to DemEventParameter elements which shall be invoked using the API `Dem_ReportErrorStatus()` in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in this container and can be extended by vendor specific error references.

Parameters included	
Parameter name	**Multiplicity**
EcuMConfigConsistencyHash	1..1
EcuMDefaultAppMode	1..1
EcuMOSResource	0..n

Parameters included	
EcuMDefaultAppModeBaseId	0..1

Parameter Name	EcuMConfigConsistencyHash	
Label	Configuration Consistency Hash	
Description	*The functionality related to this parameter is not supported by the current implementation.* A hash value is generated across all pre-compile and link-time parameters of all BSW modules. This hash value is compared against a field in the `EcuM_ConfigType` and hence allows checking the consistency of the entire configuration. Dependency: ▶ The maximal value for `EcuMConfigConsistencyHash` is 18446744073709551615 (= 0xFFFFFFFFFFFFFFFF = (2^64)-1) according to EcuM R4.0, but EB tresos Studio limits the maximum value to 9223372036854775807 (= 0x7FFFFFFFFFFFFFFF = (2^63)-1).	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=9223372036854775807 >=0	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcuMDefaultAppMode	
Label	OS Application Mode	
Description	Reference to the default application mode of the OS, loaded when the ECU comes out of reset.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcuMOSResource

Description	*The functionality related to this parameter is not supported by the current implementation.* Reference to the OS resource which is used to bring the ECU into sleep mode. In case of multi core Ecu's each core shall have an own OsResource.
Multiplicity	0..n
Type	REFERENCE
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMDefaultAppModeBaseId**
Label	Offset OS Index Value
Description	Parameter used to set the offset value for the EcuMDefaultAppMode.
Multiplicity	0..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.4.1.4. EcuMDefaultShutdownTarget

Parameters included	
Parameter name	**Multiplicity**
EcuMDefaultState	1..1
EcuMDefaultResetModeRef	0..1
EcuMDefaultSleepModeRef	0..1

Parameter Name	**EcuMDefaultState**
Label	Default Shutdown State
Description	This parameter describes the state of the default shutdown target selected when the ECU comes out of reset. If EcuMStateSleep is selected, the parameter `EcuMDefaultSleepModeRef` selects the specific sleep mode.

	▶ If `EcuMStateReset` is selected then the default reset mode implicitely selected as `ECUM_RESET_MCU` since the current SWS do not have the configuration parameter reference to EcuM Reset Modes.
Multiplicity	1..1
Type	ENUMERATION
Default value	EcuMStateOff
Range	EcuMStateOff EcuMStateReset EcuMStateSleep
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMDefaultResetModeRef**
Label	Default Reset Mode
Description	Reference to the default reset mode. Dependency: ▶ Applicable only if `EcuMDefaultShutdownTarget` is selected as `EcuMStateReset`.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMDefaultSleepModeRef**
Label	Default Sleep Mode
Description	Reference to the default sleep mode. Dependency: ▶ Applicable only if `EcuMDefaultShutdownTarget` is selected as `EcuMStateSleep`.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.5. EcuMDriverInitListOne

Containers included		
Container name	**Multiplicity**	**Description**
EcuMDriverInitItem	1..n	This container describes one entry in a driver init list.

5.4.1.6. EcuMDriverInitItem

Parameters included	
Parameter name	**Multiplicity**
EcuMEnableVendorApiInfix	1..1
EcuMModuleHeaderFile	1..1
EcuMModuleID	1..1
EcuMModuleInitConfigStr	0..1
EcuMModuleService	1..1
EcuMEnableDriver	1..1

Parameter Name	EcuMEnableVendorApiInfix
Label	EcuMEnableVendorApiInfix
Description	Enable this parameter if `VendorApiInfix` is available for a module. This generate the module service in the following format: <function>EcuMModuleID_VendorId_VendorApiInfix_EcuMModuleService()</function> Also this considers the existance of header file: `EcuMModuleID_VendorId_VendorApiInfix.h` Dependency: - The module shall be configured in `EcuMFlexModuleConfigurationRef`
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	EcuMModuleHeaderFile
Label	Module Header File
Description	Base name of the header file to be included for the usage of the module.

	If this parameter is left blank, the EcuM will include the header file with base name as entered in `EcuMModuleID`. E.g if the `Can` is configured to be initialized and the `EcuMModuleHeaderFile` is kept blank, the EcuM will include `Can.h`, else the EcuM will include the header with the file name assembeled from the base name defined by `EcuMModuleHeaderFile` and the postfix `.h`.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMModuleID**	
Label	Module ID	
Description	Specify the module short name (aka module ID) of the module to be initialized; e.g. *Adc* The name must not be empty and follow the naming conventions for valid C identifiers. It is assumed that there exists: ▶ A header file named after the value of this parameter; e.g. `Adc.h` ▶ A initialization function (declared in the mentioned header file) prefixed with the value of this parameter; e.g. `Adc_Init()`	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMModuleInitConfigStr**
Label	Module Config String
Description	To find the init-configuration of most Autosar modules, the EcuM evaluates the `EcuMModuleConfigurationRefs` given in this configuration. The name of the multiple-configuration container these references refer to also is the linker-symbol of the configuration defined in that container. That symbol is then passed to the Init-function. While initializing a module which does not use the multiple configuration container name as symbol for the configuration object, you cannot use the `EcuMMod-`

	`uleConfigurationRef` to determine the pointer, which is to be passed to the Init function. In this case, you *must* enable and provide the parameter. The parameter is then used as parameter for the Init-method of the very module. It is also possible to *override* the standard configuration of any module using a parameter in it's Init function using this parameter. If this parameter is given, the `EcuMModuleConfigurationRef` is ignored. The string given here must evaluate to a C address, e.g. as plain number, or as address operator followed by a constant name. Some Autosar modules do not have a parameter in the signature of their *MOD_-Init()* method. For those modules that do not require a parameter, do not set this parameter but disable it.
Multiplicity	0..1
Type	STRING
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMModuleService**
Label	Module Service
Description	The service to be called to initialize that module, e.g. Init, PreInit, Start etc. If the service is Init and the parameter `EcuMModuleConfigurationRef` has been set for that module, the corresponding pointer to the init structure (< Module >_ConfigType) shall be passed as an argument.
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMEnableDriver**
Label	Enable Driver
Description	If enabled, this DriverInitItem will be used by the EcuM_AL_DriverInitOne() function.
Multiplicity	1..1
Type	BOOLEAN
Default value	true

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.4.1.7. EcuMDriverInitListZero

Containers included		
Container name	**Multiplicity**	**Description**
EcuMDriverInitItem	1..n	This container describes one entry in a driver init list.

5.4.1.8. EcuMDriverInitItem

Parameters included	
Parameter name	**Multiplicity**
EcuMEnableVendorApiInfix	1..1
EcuMModuleHeaderFile	1..1
EcuMModuleID	1..1
EcuMModuleService	1..1

Parameter Name	**EcuMEnableVendorApiInfix**	
Label	EcuMEnableVendorApiInfix	
Description	Enable this parameter if `VendorApiInfix` is available for a module. This generate the module service in the following format: <function>EcuMModuleID_VendorId_VendorApiInfix_EcuMModuleService()</function> Also this considers the existance of header file: `EcuMModuleID_VendorId_VendorApiInfix.h` Dependency: ▶ The module shall be configured in `EcuMFlexModuleConfigurationRef`	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMModuleHeaderFile**
Label	Module Header File

Description	Base name of the header file to be included for the usage of the module. If this parameter is left blank, the EcuM will include the header file with base name as entered in `EcuMModuleID`. E.g if the `Can` is configured to be initialized and the `EcuMModuleHeaderFile` is kept blank, the EcuM will include `Can.h`, else the EcuM will include the header with the file name assembeled from the base name defined by `EcuMModuleHeaderFile` and the postfix `.h`.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMModuleID**	
Label	Module ID	
Description	Specify the module short name (aka module ID) of the module to be initialized; e.g. *Adc* The name must not be empty and follow the naming conventions for valid C identifiers. It is assumed that there exists: ▶ A header file named after the value of this parameter; e.g. `Adc.h` ▶ A initialization function (declared in the mentioned header file) prefixed with the value of this parameter; e.g. `Adc_Init()`	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMModuleService**
Label	Module Service
Description	The service to be called to initialize that module, e.g. Init, PreInit, Start etc. If the service is Init and the parameter EcuMModuleConfigurationRef has been set for that module, the corresponding pointer to the init structure (< Module >_ConfigType) and in case of multiple instantiation an uint8 value to identify the instance of the module(<MSN>_CtrlIdx) shall be passed as an argument.

Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.9. EcuMDriverRestartList

Containers included		
Container name	**Multiplicity**	**Description**
EcuMDriverInitItem	1..n	This container describes one entry in a driver init list. Restriction: ▶ The EcuMDriverInitItem name in EcuMDriverRestartList shall be the same for all EcuM configuration containers

5.4.1.10. EcuMDriverInitItem

Parameters included	
Parameter name	**Multiplicity**
EcuMEnableVendorApiInfix	1..1
EcuMModuleHeaderFile	1..1
EcuMModuleID	1..1
EcuMModuleInitConfigStr	0..1
EcuMModuleService	1..1
EcuMEnableDriver	1..1

Parameter Name	**EcuMEnableVendorApiInfix**
Label	EcuMEnableVendorApiInfix
Description	Enable this parameter if VendorApiInfix is available for a module. This generate the module service in the following format: <function>EcuMModuleID_VendorId_VendorApiInfix_EcuMModuleService()</function> Also this considers the existance of header file: EcuMModuleID_VendorId_VendorApiInfix.h Dependency:

	▶ The module shall be configured in `EcuMFlexModuleConfigurationRef`
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMModuleHeaderFile**
Label	Module Header File
Description	Base name of the header file to be included for the usage of the module. If this parameter is left blank, the EcuM will include the header file with base name as entered in `EcuMModuleID`. E.g if the `Can` is configured to be initialized and the `EcuMModuleHeaderFile` is kept blank, the EcuM will include `Can.h`, else the EcuM will include the header with the file name assembeled from the base name defined by `EcuMModuleHeaderFile` and the postfix `.h`.
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMModuleID**
Label	Module ID
Description	Specify the module short name (aka module ID) of the module to be initialized; e.g. *Adc* The name must not be empty and follow the naming conventions for valid C identifiers. It is assumed that there exists: ▶ A header file named after the value of this parameter; e.g. `Adc.h` ▶ A initialization function (declared in the mentioned header file) prefixed with the value of this parameter; e.g. `Adc_Init()`
Multiplicity	1..1
Type	STRING

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMModuleInitConfigStr**	
Label	Module Config String	
Description	To find the init-configuration of most Autosar modules, the EcuM evaluates the `EcuMModuleConfigurationRefs` given in this configuration. The name of the multiple-configuration container these references refer to also is the linker-symbol of the configuration defined in that container. That symbol is then passed to the Init-function. While initializing a module which does not use the multiple configuration container name as symbol for the configuration object, you cannot use the `EcuMModuleConfigurationRef` to determine the pointer, which is to be passed to the Init function. In this case, you *must* enable and provide the parameter. The parameter is then used as parameter for the Init-method of the very module. It is also possible to *override* the standard configuration of any module using a parameter in it's Init function using this parameter. If this parameter is given, the `EcuMModuleConfigurationRef` is ignored. The string given here must evaluate to a C address, e.g. as plain number, or as address operator followed by a constant name. Some Autosar modules do not have a parameter in the signature of their *MOD_*-Init() method. For those modules that do not require a parameter, do not set this parameter but disable it.	
Multiplicity	0..1	
Type	STRING	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMModuleService**
Label	Module Service
Description	The service to be called to initialize that module, e.g. Init, PreInit, Start etc. If the service is Init and the parameter `EcuMModuleConfigurationRef` has been set for that module, the corresponding pointer to the init structure (< Module >_ConfigType) shall be passed as an argument.
Multiplicity	1..1

Type	STRING	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMEnableDriver**	
Label	Enable Driver	
Description	If enabled, this DriverInitItem will be used by the EcuM_AL_DriverRestart() function.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.4.1.11. EcuMSleepMode

Parameters included	
Parameter name	**Multiplicity**
EcuMSleepModeId	1..1
EcuMSleepModeSuspend	1..1
EcuMSleepModeMcuModeRef	1..1
EcuMWakeupSourceMask	1..n

Parameter Name	**EcuMSleepModeId**	
Label	Sleep Mode ID	
Description	Unique ID for sleep mode. The ID has to be zero-based and consecutive.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMSleepModeSuspend**
Description	Set this flag to true, if the CPU is suspended, halted, or powered off in the sleep mode. If the CPU keeps running in this sleep mode, then this flag must be set to false.

Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMSleepModeMcuModeRef**
Label	Sleep Mcu Mode
Description	Reference to the corresponding MCU mode for this sleep mode.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMWakeupSourceMask**
Label	Wakeup Source Reference
Description	Reference to a wakeup source that shall be enabled for this sleep mode.
Multiplicity	1..n
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.12. EcuMWakeupSource

Parameters included	
Parameter name	**Multiplicity**
EcuMValidationTimeout	0..1
EcuMWakeupSourceId	1..1
EcuMWakeupSourcePolling	1..1
EcuMComMChannelRef	0..1
EcuMComMPNCRef	0..1
EcuMResetReasonRef	0..1

Parameter Name	EcuMValidationTimeout	
Label	Validation Timeout (s)	
Description	The validation timeout is the period in seconds for which the ECU State Manager will wait for the validation of a wakeup event. If the timeout parameter is disabled the ECU Manager does not validate the wakeup source. Reported wakeup events of wakeup sources with disabled timeout parameter are considered to be always valid. The validation timeout must always be disabled for all default wakeup sources.	
Multiplicity	0..1	
Type	FLOAT	
Default value	0.0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcuMWakeupSourceId	
Label	Wakeup Source ID	
Description	This parameter is the identifier of a wakeup source. The numerical value ranging from 0 to 31 defines the position corresponding to this wakeup source in all instances of `EcuM_WakeupSourceType` bitfield.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcuMWakeupSourcePolling	
Label	Wakeup Source Polling	
Description	*The functionality related to this parameter is not supported by the current implementation.*	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcuMComMChannelRef	
Label	ComM Channel	
Description	Reference to a Network (channel) defined in the Communication Manager. No reference indicates that the wakeup source is not a communication channel.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcuMComMPNCRef	
Label	ComM PNC	
Description	This is a reference to a one or more PNC's defined in the Communication Manager. No reference indicates that the wakeup source is not assigned to a partial network.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	EcuMResetReasonRef	
Description	This parameter describes the mapping of reset reasons detected by the MCU driver into wakeup sources.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.13. EcuMDemEventParameterRefs

Parameters included	
Parameter name	**Multiplicity**
ECUM_E_ALL_RUN_REQUESTS_KILLED	0..1
ECUM_E_CONFIGURATION_DATA_INCONSISTENT	0..1

Parameters included	
ECUM_E_RAM_CHECK_FAILED	0..1

Parameter Name	**ECUM_E_ALL_RUN_REQUESTS_KILLED**	
Description	Reference to the DemEventParameter which shall be issued when the error ECUM_E_ALL_RUN_REQUESTS_KILLED has occurred. Dependency on parameter(s): ▶ `EcuMFixedConfiguration` is configured to enable EcuM Fixed support. Further notes: ▶ Activation: Thrown, if the service EcuM_KillAllRunRequests is issued. ▶ Healing: None. The error resides in memory until it is deleted. ▶ Trigger debounce: None. The error is reported on first occurrence. ▶ Rate of diagnostic checks: None. This error is thrown by every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the *Integration notes* section of the module references.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ECUM_E_CONFIGURATION_DATA_INCONSISTENT**	
Description	Reference to the DemEventParameter which shall be issued when the error "ECUM_E_CONFIGURATION_DATA_INCONSISTENT" has occured. *The functionality related to this parameter is not supported by the current implementation.*	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**ECUM_E_RAM_CHECK_FAILED**
Description	Reference to the DemEventParameter which shall be issued when the error ECUM_E_RAM_CHECK_FAILED has occured.

	Dependency on parameter(s): - `EcuMRamChkFailedReportToDem`: Select DEM to enable the reporting of ECUM_E_RAM_CHECK_FAILED. Further notes: - Activation: Thrown, if the RAM check during wakeup failed. - Healing: Dependent on the implementation of callout function `EcuM_CheckRamHash()` that must be implemented by user. - Trigger debounce: Dependent on the implementation of callout function `EcuM_CheckRamHash()` that must be implemented by user. - Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the *Integration notes* section of the module references.
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.14. EcuMFixedConfiguration

Containers included		
Container name	**Multiplicity**	**Description**
EcuMDriverInitListThree	0..1	*The functionality related to this parameter is not supported by the current implementation.* Container for Init Block III. This container holds a list of module IDs that will be initialized. Each module in the list will be called for initialization in the list order. All modules in this list are initilialized after the OS is started and so these modules may use OS support. These modules may also rely on the Nvram ReadAll job to have provided all data.
EcuMDriverInitListTwo	0..1	*The functionality related to this parameter is not supported by the current implementation.* Container for Init Block II. This container holds a list of module IDs that will be initialized. Each module in the list will be

Containers included		
		called for initialisation in the list order. All modules in this list are initilialized after the OS is started and so these modules may use OS support. These modules may not rely on the Nvram ReadAll job to have provided all data.
EcuMFixedUserConfig	1..n	These containers describe the identifiers that are needed to refer to a software component or another appropriate entity in the system which is designated to request the RUN state. Application requestors refer to entities above RTE, system requestors to entities below RTE (e.g. Communication Manager).
EcuMTTII	0..n	*The functionality related to this parameter is not supported by the current implementation.* This container contains parameters for the TTII protocol. The list must contain at least on element when `ECUM_TTII_ENABLED` is set to true. Restriction: ▶ The `EcuMTTII` name shall be the same for all EcuM configuration containers
EcuMWdgM	0..1	*The functionality related to this parameter is not supported by the current implementation.* This container holds the configuration parameters for the interaction between the Watchdog Manager (WdgM) and EcuM. The WdgM mode to be selected in a specific Sleep Mode of EcuM is configured in the EcuMSleepMode container.

Parameters included	
Parameter name	**Multiplicity**
EcuMNvramReadallTimeout	1..1
EcuMNvramWriteallTimeout	1..1
EcuMRunMinimumDuration	1..1
EcuMFixedModuleConfigurationRef	0..n
EcuMComMCommunicationAllowedList	0..n
EcuMNormalMcuModeRef	1..1

Parameter Name	EcuMNvramReadallTimeout

Description	*The functionality related to this parameter is not supported by the current implementation.* Period given in seconds for which the ECU State Manager will wait until it considers a ReadAll job of the NVRAM Manager as failed. Dependency: ▶ The value shall not be smaller than `EcuMMainFunctionPeriod`.
Multiplicity	1..1
Type	FLOAT
Default value	0.0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMNvramWriteallTimeout**
Description	*The functionality related to this parameter is not supported by the current implementation.* Period given in seconds for which the ECU State Manager will wait until it considers a WriteAll job of the NVRAM Manager as failed. Dependency: ▶ The value shall not be smaller than `EcuMMainFunctionPeriod`.
Multiplicity	1..1
Type	FLOAT
Default value	0.0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMRunMinimumDuration**
Description	Duration given in seconds for which the ECU State Manager will stay in RUN state even when no one requests RUN. This duration should be long at least as long as a SW-Cs needs to request RUN. Dependency: ▶ The value shall not be smaller than `EcuMMainFunctionPeriod`.
Multiplicity	1..1

Type	FLOAT
Default value	0.0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMFixedModuleConfigurationRef**
Description	*The functionality related to this parameter is not supported by the current implementation.* References to the init structure of the corresponding BSW module.
Multiplicity	0..n
Type	CHOICE-REFERENCE
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMComMCommunicationAllowedList**
Description	*The functionality related to this parameter is not supported by the current implementation.* References to the ComMChannels for which EcuM has to call ComM_CommunicationAllowed.
Multiplicity	0..n
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMNormalMcuModeRef**
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter is a reference to the normal MCU mode to be restored after a sleep. Restriction: ▶ This parameter shall have the same value in all configurations
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild

Origin	AUTOSAR_ECUC

5.4.1.15. EcuMDriverInitListThree

Containers included		
Container name	**Multiplicity**	**Description**
EcuMDriverInitItem	1..n	*The functionality related to this parameter is not supported by the current implementation.* This container describes one entry in a driver init list. Restriction: ▶ The `EcuMDriverInitItem` name in `EcuMDriverInitListThree` shall be the same for all EcuM configuration containers

5.4.1.16. EcuMDriverInitItem

Parameters included	
Parameter name	**Multiplicity**
EcuMModuleID	1..1
EcuMModuleService	1..1

Parameter Name	**EcuMModuleID**
Description	*The functionality related to this parameter is not supported by the current implementation.* Specify the module short name (aka module ID) of the module to be initialized; e.g. *Adc* The name must not be empty and follow the naming conventions for valid C identifiers. It is assumed that there exists: ▶ A header file named after the value of this parameter; e.g. `Adc.h` ▶ A initialization function (declared in the mentioned header file) prefixed with the value of this parameter; e.g. `Adc_Init()` Restriction:

	▶ This parameter for a `DriverInitItem` in `EcuMDriverInitListThree` shall have the same value in all configurations
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMModuleService**
Description	*The functionality related to this parameter is not supported by the current implementation.* The service to be called to initialize that module, e.g. Init, PreInit, Start etc. If the service is Init and the parameter `EcuMModuleConfigurationRef` has been set for that module, the corresponding pointer to the init structure (< Module >_ConfigType) shall be passed as an argument. Restriction: ▶ This parameter for a `DriverInitItem` in `EcuMDriverInitListThree` shall have the same value in all configurations
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.17. EcuMDriverInitListTwo

Containers included		
Container name	**Multiplicity**	**Description**
EcuMDriverInitItem	1..n	*The functionality related to this parameter is not supported by the current implementation.* This container describes one entry in a driver init list. Restriction: ▶ The `EcuMDriverInitItem` name in `EcuMDriverInitListTwo` shall be the same for all EcuM configuration containers

5.4.1.18. EcuMDriverInitItem

Parameters included	
Parameter name	**Multiplicity**
EcuMModuleID	1..1
EcuMModuleService	1..1

Parameter Name	**EcuMModuleID**
Description	*The functionality related to this parameter is not supported by the current implementation.* Specify the module short name (aka module ID) of the module to be initialized; e.g. *Adc* The name must not be empty and follow the naming conventions for valid C identifiers. It is assumed that there exists: ▶ A header file named after the value of this parameter; e.g. `Adc.h` ▶ A initialization function (declared in the mentioned header file) prefixed with the value of this parameter; e.g. `Adc_Init()` Restriction: ▶ This parameter for a `DriverInitItem` in `EcuMDriverInitListTwo` shall have the same value in all configurations
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMModuleService**
Description	*The functionality related to this parameter is not supported by the current implementation.* The service to be called to initialize that module, e.g. Init, PreInit, Start etc. If the service is Init and the parameter `EcuMModuleConfigurationRef` has been set for that module, the corresponding pointer to the init structure (< Module >_ConfigType) shall be passed as an argument. Restriction:

	▶ This parameter for a `DriverInitItem` in `EcuMDriverInitListTwo` shall have the same value in all configurations
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.19. EcuMFixedUserConfig

Parameters included	
Parameter name	**Multiplicity**
EcuMFixedUser	1..1

Parameter Name	**EcuMFixedUser**
Description	Parameter used to identify one user. A unique ID needs to be provided for each user. The ID also has to be zero-based and consecutive.
Multiplicity	1..1
Type	INTEGER
Default value	0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.20. EcuMTTII

Parameters included	
Parameter name	**Multiplicity**
EcuMDivisor	1..1
EcuMSleepModeRef	1..1
EcuMSuccessorRef	0..1

Parameter Name	**EcuMDivisor**
Description	*The functionality related to this parameter is not supported by the current implementation.* Initial value for the Divisor Counter of this TTII step.

	This Counter is decremented every time the ECU has been woken up by the TTII Wakeup Source. When this counter has elapsed the succeeding TTII step is executed. *This parameter is not used in this version of the EcuM.* Restriction: ▶ This parameter for an EcuM TTII shall have the same value in all configurations ▶ The maximal value for `EcuMDivisor` is 18446744073709551615 (= 0xFFFFFFFFFFFFFFFF = (2^64)-1) according to EcuM configuration but EB tresos Studio limits the maximum value to 9223372036854775807 (= 0x7FFFFFFFFFFFFFFF = (2^63)-1).
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=9223372036854775807 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMSleepModeRef**
Description	*The functionality related to this parameter is not supported by the current implementation.* This configuration parameter is a reference to a configured sleep mode that is used for TTII. Restriction: ▶ This parameter for an EcuM TTII shall have the same value in all configurations
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMSuccessorRef**

Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter is a reference to the next sleep mode in the TTII protocol. Restriction: ▶ This parameter for an EcuM TTII shall have the same value in all configurations	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.21. EcuMWdgM

Parameters included	
Parameter name	**Multiplicity**
EcuMWdgMPostRunModeRef	1..1
EcuMWdgMRunModeRef	1..1
EcuMWdgMShutdownModeRef	1..1
EcuMWdgMStartupModeRef	1..1
EcuMWdgMWakeupModeRef	1..1
EcuMSupervisedEntityRef	1..1

Parameter Name	**EcuMWdgMPostRunModeRef**	
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter references the WdgM mode to be set when entering the POST RUN state of EcuM.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMWdgMRunModeRef**

Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter references the WdgM mode to be set when entering the RUN state of EcuM.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMWdgMShutdownModeRef**	
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter references the WdgM mode to be set when leaving the GO OFF I state of EcuM.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMWdgMStartupModeRef**	
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter references the WdgM mode to be set when entering the STARTUP II state of EcuM.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMWdgMWakeupModeRef**
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter references the WdgM mode to be set when entering the WAKEUP I state of EcuM.

Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMSupervisedEntityRef**	
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter references the Supervised Entity ID that way configured for EcuM in the Watchdog Manager. The EcuM updates this associated alive counter cyclically after it has initialized the WdgM in all states except SLEEP. The cyclic update process is suspended in the SLEEP mode but will be activated after the SLEEP mode is left. Configure the supervised entity of the EcuM in the WdgM so that the WdgM causes a reset if the alive counter is not updated in the WdgM modes WdgMStartupMode, WdgMWakeupMode, WdgMRunMode, and WdgMPostRunMode. During the time in which the WdgMShutdownMode is active no alive counter is updated any more bcause the WdgM is not supervising the counter any more.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.22. EcuMFlexConfiguration

Containers included		
Container name	**Multiplicity**	**Description**
EcuMAlarmClock	0..n	*The functionality related to this parameter is not supported by the current implementation.* These containers describe the configured alarm clocks. The name of these containers allows giving a symbolic name to one alarm clock.
EcuMFlexUserConfig	1..n	Describes the identifiers that are needed to refer to a particular software component or another appropriate entity in the system which uses the EcuMFlex Interfaces.

Containers included		
EcuMGoDownAllowedUsers	0..1	*The functionality related to this parameter is not supported by the current implementation.* This container describes the collection of allowed users which are allowed to call the EcuM_GoDown API.
EcuMResetMode	1..n	These containers describe the configured reset modes. The name of these containers allows one of the following symbolic names to be given to the different reset modes: - ECUM_RESET_MCU - ECUM_RESET_WDGM - ECUM_RESET_IO
EcuMSetClockAllowedUsers	0..1	*The functionality related to this parameter is not supported by the current implementation.* This container describes the collection of allowed users, which are allowed to call the `EcuM_SetClock()` API.
EcuMShutdownCause	4..255	These containers describe the configured shut down or reset causes. The name of these containers allows to give one of the following symbolic names to the different shut down causes - ECUM_CAUSE_ECU_STATE - ECU state machine entered a state for shutdown - ECUM_CAUSE_WDGM - WdgM detected failure - ECUM_CAUSE_DCM - Dcm requests shutdown - and values from configuration
EcuMShutdownTarget	1..3	These containers describe the configured shut down targets. The name of these containers allows to give symbolic names to the different shutdown targets. Current implementation supports only three predefined shutdown targets - `ECUM_STATE_SLEEP, ECUM_STATE_RESET, ECUM_STATE_OFF.`

Parameters included	
Parameter name	**Multiplicity**
EcuMFlexModuleConfigurationRef	0..n
EcuMNormalMcuModeRef	1..1

Parameter Name	**EcuMFlexModuleConfigurationRef**
Description	This parameter contains a reference to the init structure of the corresponding BSW module. Dependency: ▶ The reference in the list shall not be repeated.
Multiplicity	0..n
Type	CHOICE-REFERENCE
Configuration class	**PostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMNormalMcuModeRef**
Label	Normal Mcu Mode
Description	This parameter is a reference to the normal MCU mode to be restored after a sleep.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.23. EcuMAlarmClock

Parameters included	
Parameter name	**Multiplicity**
EcuMAlarmClockId	1..1
EcuMAlarmClockTimeOut	1..1
EcuMAlarmClockUser	1..1

Parameter Name	**EcuMAlarmClockId**
Description	*The functionality related to this parameter is not supported by the current implementation.* Unique ID for this alarmclock.
Multiplicity	1..1
Type	INTEGER

Default value	0
Range	<=255 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMAlarmClockTimeOut**
Description	*The functionality related to this parameter is not supported by the current implementation.* Parameter allows to define a timout in seconds for this alarm clock.
Multiplicity	1..1
Type	FLOAT
Default value	0.0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMAlarmClockUser**
Description	*The functionality related to this parameter is not supported by the current implementation.* Parameter allows an alarm to be assigned to the user.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.24. EcuMFlexUserConfig

Parameters included	
Parameter name	**Multiplicity**
EcuMFlexUser	1..1
EcuMFlexEcucPartitionRef	0..1

Parameter Name	**EcuMFlexUser**

Description	Parameter used to identify one user. Restriction: ▶ User ID shall be in the range 0 to 255 and shall be unique
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMFlexEcucPartitionRef**
Description	Denotes in which "EcucPartition" the user of the EcuM is executed. This reference parameter has no effect as it is only used by the Alarm Clock functionality which is not supported by the current implementation.
Multiplicity	0..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.25. EcuMGoDownAllowedUsers

Parameters included	
Parameter name	**Multiplicity**
EcuMGoDownAllowedUserRef	1..n

Parameter Name	**EcuMGoDownAllowedUserRef**
Description	*The functionality related to this parameter is not supported by the current implementation.* These parameters describe the references to users which are allowed to call the EcuM_GoDown API.
Multiplicity	1..n
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.26. EcuMResetMode

Parameters included	
Parameter name	**Multiplicity**
EcuMResetModeId	1..1

Parameter Name	**EcuMResetModeId**	
Description	This ID identifies the reset mode in services like `EcuM_SelectShutdownTarget()`.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.27. EcuMSetClockAllowedUsers

Parameters included	
Parameter name	**Multiplicity**
EcuMSetClockAllowedUserRef	1..n

Parameter Name	**EcuMSetClockAllowedUserRef**	
Description	*The functionality related to this parameter is not supported by the current implementation.* References to the users which are allowed to call the `EcuM_SetClock()` API.	
Multiplicity	1..n	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.28. EcuMShutdownCause

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
EcuMShutdownCauseId	1..1

Parameter Name	**EcuMShutdownCauseId**
Description	Unique ID for identifing this shut down cause.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.29. EcuMShutdownTarget

Parameters included	
Parameter name	**Multiplicity**
EcuMShutdownTargetId	1..1

Parameter Name	**EcuMShutdownTargetId**
Description	This ID identifies this shut down target in services like `EcuM_SelectShutdownTarget()`.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.30. EcuMDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
EcuMDefProgEnabled	1..1

Parameters included	
EcuMPrecondAssertEnabled	1..1
EcuMPostcondAssertEnabled	1..1
EcuMStaticAssertEnabled	1..1
EcuMUnreachAssertEnabled	1..1
EcuMInvariantAssertEnabled	1..1

Parameter Name	EcuMDefProgEnabled	
Label	Enable Defensive Programming	
Description	Enables or disables the defensive programming feature for the module EcuM. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	EcuMPrecondAssertEnabled
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module EcuM. Dependency on parameter(s): ▶ Enable Development Error Detection (`EcuMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`EcuMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMPostcondAssertEnabled**	
Label	Enable Postcondition Assertions	
Description	Enables handling of postcondition assertion checks reported from the module EcuM. Dependency on parameter(s): ▶ Enable Development Error Detection (`EcuMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`EcuMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMStaticAssertEnabled**	
Label	Enable Static Assertions	
Description	Enables handling of static assertion checks reported from the module EcuM. Dependency on parameter(s): ▶ Enable Development Error Detection (`EcuMDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`EcuMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMUnreachAssertEnabled**

Label	Enable Unreachable Code Assertions
Description	Enables handling of unreachable code assertion checks reported from the module EcuM. Dependency on parameter(s): - Enable Development Error Detection (`EcuMDevErrorDetect`): must be enabled - Enable Defensive Programming (`EcuMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMInvariantAssertEnabled**
Label	Enable Invariant Assertions
Description	Enables handling of invariant assertion checks reported from functions of the module EcuM. Dependency on parameter(s): - Enable Development Error Detection (`EcuMDevErrorDetect`): must be enabled - Enable Defensive Programming (`EcuMDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.4.1.31. EcuMFixedGeneral

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
EcuMIncludeComM	1..1
EcuMIncludeNvM	1..1
EcuMIncludeNvramMgr	1..1
EcuMIncludeWdgM	1..1
EcuMTTIIEnabled	1..1
EcuMTTIIWakeupSourceRef	1..1

Parameter Name	**EcuMIncludeComM**
Description	*The functionality related to this parameter is not supported by the current implementation.* This configuration parameter defines whether the communication manager is supported by EcuM. This feature is presented for development purpose to compile out the communication manager in the early debugging phase.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMIncludeNvM**
Description	*The functionality related to this parameter is not supported by the current implementation.* This configuration parameter defines whether the non volatile memory manager is supported by EcuM. This feature is presented for development purpose to compile out the volatile memory manager in the early debugging phase.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMIncludeNvramMgr**
Description	*The functionality related to this parameter is not supported by the current implementation.*

	If this parameter is enabled, the EcuM will be compiled with NvM (Non-Volatile RAM manager) support; the EcuM will initialize the NvRam-manager and also implement safe start-up and shutdown of the NvRam, regarding the consistency of NvRam caches and data. Dependency: ▶ NvM must be initialized in init list 2.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMIncludeWdgM**
Description	*The functionality related to this parameter is not supported by the current implementation.* This configuration parameter defines whether the watchdog manager is supported by EcuM. This feature is presented for development purpose to compile out the watchdog manager in the early debugging phase Dependency: ▶ WdgM shall be in the init list.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

Parameter Name	**EcuMTTIIEnabled**
Description	*The functionality related to this parameter is not supported by the current implementation.* Switch to enable or disable Time Triggered Increased Inoperation (TTII).
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMTTIIWakeupSourceRef**	
Description	*The functionality related to this parameter is not supported by the current implementation.* References to the initial sleep mode to be used by TTII when TTII is activated after a RUN mode.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

5.4.1.32. EcuMFlexGeneral

Parameters included	
Parameter name	**Multiplicity**
EcuMAlarmClockPresent	1..1
EcuMEnableDefBehaviour	0..1
EcuMResetLoopDetection	1..1
EcuMAlarmWakeupSource	0..1
EcuMStoredShutdownOperations	1..1

Parameter Name	**EcuMAlarmClockPresent**	
Description	*The functionality related to this parameter is not supported by the current implementation.* Flag indicates whether the optional AlarmClock feature is present.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMEnableDefBehaviour**

Description	*The functionality related to this parameter is not supported by the current implementation.* Switches the defensive behaviour on or off.	
Multiplicity	0..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMResetLoopDetection**	
Description	*The functionality related to this parameter is not supported by the current implementation.* If this flag is false, no reset loop detection is performed.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMAlarmWakeupSource**	
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter describes the reference to the EcuMWakeupSource being used for the EcuM AlarmClock.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMStoredShutdownOperations**
Description	*This parameter denotes the number of shutdown operations for which information are stored in the EcuM for later retrieval.* The parameter is not specified by AUTOSAR but it is an EB specific enhancement to the EcuM
Multiplicity	1..1

Type	INTEGER
Default value	1
Range	<=255 >=1
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.4.1.33. EcuMGeneral

Containers included		
Container name	**Multiplicity**	**Description**
EcuMServiceAPI	1..1	**Label:** Service API Parameters Container for configuration of the service API of EcuM. Check "Enable Rte Usage" in order to enable this configuration item.

Parameters included	
Parameter name	**Multiplicity**
EcuMDevErrorDetect	1..1
EcuMIncludeDem	1..1
EcuMIncludeDet	1..1
EcuMMainFunctionPeriod	1..1
EcuMProvideShutdownHook	1..1
EcuMRelocatablePbcfgEnable	1..1
EcuMRteUsage	1..1
EcuMEnableMulticore	1..1
EcuMMasterCoreId	0..1
EcuMInitialCoreId	0..1
EcuMStartInitialCore	0..1
EcuMUseBoottargetSec	1..1
EcuMVersionInfoApi	1..1

Parameter Name	**EcuMDevErrorDetect**

Label	Enable Development Error Detection	
Description	Enable development error detection. If false, no debug artifacts (e.g. calls to Det) remain in the executable object. Initialization of Det, however is controlled by configuration of optional BSW modules. Dependency: ▶ `EcuMIncludeDet` has to be enabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMIncludeDem**	
Label	Include Dem	
Description	This parameter must be enabled to report production errors to Diagnostics Event Manager(DEM). Dem must be initialized before reporting any errors to it. The initialization consist of `Dem_PreInit()` and `Dem_Init()`. `Dem_PreInit()` shall be configured in `EcuMDriverInitListOne` with proper post build parameter. Further initialization of DEM by calling `Dem_Init()` is controlled by BswM module. Configure `Dem_PreInit()` in `EcuMDriverInitListOne`: ▶ EcuMModuleID : Dem ▶ EcuMModuleService : PreInit ▶ EcuMModuleInitConfigStr : disable and let `EcuMFlexModuleConfigurationRef` refer to the Dem configuration container ▶ EcuMModuleHeaderFile : leave field blank	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMIncludeDet**

Label	Include Det	
Description	According to AUTOSAR this parameter controls the initialization of the Det (Development Error Tracer) module. But in the current EcuM implementation the initialization and the start if the Det is controlled via the initialization lists just like for other AUTOSAR modules. Dependency: ▶ Det must have an entry for its initialization and start in one of the initialization lists.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMMainFunctionPeriod**	
Description	This parameter defines the schedule period of `EcuM_MainFunction`. Unit: [s]	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.02	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	AUTOSAR_ECUC	

Parameter Name	**EcuMProvideShutdownHook**	
Label	Provide ShutdownHook() implementation	
Description	This EB specific parameter enables the provision of a minimalistic implementation of the `ShutdownHook()` function for the Os. The `ShutdownHook()` implementation provided by default only calls `EcuM_Shutdown()`. If you need a more complex implementation of the `ShutdownHook()` function, disable this parameter and provide an application specific implementation.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPostBuild:**	VariantPostBuild

Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMRelocatablePbcfgEnable**
Label	EcuMRelocatablePbcfgEnable
Description	Enables/disables support for relocatable postbuild configuration. ▶ True: Postbuild configuration relocatable in memory. ▶ False: Postbuild configuration not relocatable in memory.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMRteUsage**
Label	Enable Rte Usage
Description	This parameter enables the usage of the RTE for this module. For an easy integration it is recommended to disable the usage of the RTE at the beginning of the integration work.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMEnableMulticore**
Label	Enable Multi-core functionality
Description	*This parameter enables the multicore functionality.* If multiple Os configurations are used, this parameter must always be set to true. The parameter is not specified by AUTOSAR but it is an EB specific enhancement to the EcuM
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMMasterCoreId**	
Label	EcuM Master Core ID	
Description	*This parameter maps the EcuM master core instance to a specific Os Core ID.*	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMInitialCoreId**	
Label	EcuM Initial Core ID	
Description	The ID of the initial core that is started when the system boots up. This parameter can be used to specify the logical core identifier of the core which is initially started when the system boots up. *Note:* If this parameter is not configured, the EcuM assumes that the initial core is the core designated by OS_CORE_ID_MASTER	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMStartInitialCore**
Label	EcuM Start Initial Core
Description	This parameter enables calling the OS API *StartCore(coreID)* upon EcuM initialization, also for the core that boots up the system. *Note:* If this parameter is not enabled, the EcuM avoids to invoke StartCore() for the core that boots up the system, since it is already started by the hardware and for this reason some Os solutions are triggering an error.
Multiplicity	0..1

Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMUseBoottargetSec**
Label	Enable Boottarget Section
Description	This parameter enables the mapping of the variable EcuM_BootTarget to the memory section `SEC_ECUM_BOOTTARGET` instead of `SEC_VAR_CLEARED_UNSPECIFIED`. This might be needed if you access EcuM_BootTarget for a bootloader or boot manager. Please see the comments in EcuM_BootTarget.h for implementation details.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMVersionInfoApi**
Label	Enable Version Info API
Description	Switches the version info API on or off
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	AUTOSAR_ECUC

5.4.1.34. EcuMServiceAPI

Parameters included	
Parameter name	**Multiplicity**
EcuMEnableASR32ServiceAPI	1..1
EcuMEnableASR40ServiceAPI	1..1

Parameters included	
EcuMDefaultASRServiceAPI	1..1

Parameter Name	**EcuMEnableASR32ServiceAPI**
Label	Enable AUTOSAR 3.2 service API
Description	Configures whether the AUTOSAR 3.2 service API shall be provided. ▶ `TRUE` = Enables AUTOSAR 3.2 service API. ▶ `FALSE` = Disables AUTOSAR 3.2 service API.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMEnableASR40ServiceAPI**
Label	Enable AUTOSAR 4.0 service API
Description	Configures whether the AUTOSAR 4.0 service API shall be provided. ▶ `TRUE` = Enables AUTOSAR 4.0 service API. ▶ `FALSE` = Disables AUTOSAR 4.0 service API.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPostBuild:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMDefaultASRServiceAPI**
Label	Default AUTOSAR service API
Description	Defines the default AUTOSAR service API. ▶ `AUTOSAR_32` = AUTOSAR 3.2 service API is the default one. ▶ `AUTOSAR_40` = AUTOSAR 4.0 service API is the default one. ▶ `NONE` = No default AUTOSAR service API is provided.
Multiplicity	1..1
Type	ENUMERATION

Default value	AUTOSAR_40	
Range	AUTOSAR_32 AUTOSAR_40 NONE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

5.4.1.35. ReportToDem

Parameters included	
Parameter name	**Multiplicity**
EcuMRamChkFailedReportToDem	1..1
EcuMRamChkFailedReportToDemDetErrorId	1..1
EcuMCfgDataInconsistentReportToDem	1..1
EcuMCfgDataInconsistentReportToDemDetErrorId	1..1

Parameter Name	**EcuMRamChkFailedReportToDem**	
Label	Ram Check Failure	
Description	Selects the handling of the production error: *ECUM_E_RAM_CHECK_FAILED* ▶ DEM: All errors are reported to the Diagnostics Event Manager (Dem). ▶ DET: All errors are reported to the Development Error Tracer (Det) if enabled. ▶ DISABLE: Production errors are not reported at all.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	DEM	
Range	DEM DET DISABLE	
Configuration class	**VariantPostBuild:**	VariantPostBuild
Origin	Elektrobit Automotive GmbH	

Parameter Name	**EcuMRamChkFailedReportToDemDetErrorId**

Label	EcuM Ram Check Failure DemToDet ErrorId
Description	This parameter is used to report Dem to Det. It shall be checked that the variable `EcuMRamChkFailedReportToDemDetErrorId` is set to a value between 30 and 255; the default value shall be 30. A preprocessor define ECUM_E_DEMTODET_RAM_CHECK_FAILED shall be generated holding the value of `EcuMRamChkFailedReportToDemDetErrorId`. Dependencies: ▶ The identifier to report Dem to Det must not be zero-based.
Multiplicity	1..1
Type	INTEGER
Default value	30
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMCfgDataInconsistentReportToDem**
Label	Inconsistent Configuration Data
Description	*The functionality related to this parameter is not supported by the current implementation.* Selects the handling of the production error: *ECUM_E_CONFIGURATION_DATA_INCONSISTENT* ▶ `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ▶ `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ▶ `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPostBuild:** VariantPostBuild

Origin	Elektrobit Automotive GmbH

Parameter Name	**EcuMCfgDataInconsistentReportToDemDetErrorId**
Label	EcuM Configuration Data Inconsistent DemToDet ErrorId
Description	This parameter is used to report Dem to Det. It shall be checked that the variable `EcuMCfgDataInconsistentReportToDemDetErrorId` is set to a value between 30 and 255; the default value shall be 32. A preprocessor define ECUM_E_DEMTODET_CONFIGURATION_DATA_INCONSISTENT shall be generated holding the value of `EcuMCfgDataInconsistentReportToDemDetErrorId`. Dependencies: ▶ The identifier to report Dem to Det must not be zero-based.
Multiplicity	1..1
Type	INTEGER
Default value	32
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPostBuild
Origin	Elektrobit Automotive GmbH

5.4.1.36. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the EcuM can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

5.4.2. Recommended configurations

5.4.2.1. EcuMRecConfiguration

Containers included	
Container name	**Container definition**
EcuM_Config_0	EcuMConfiguration

Parameters included	
Parameter name	**Value**

5.4.2.1.1. EcuM_Config_0

Containers included	
Container name	**Container definition**
EcuMFlexConfiguration	EcuMFlexConfiguration
EcuMCommonConfiguration	EcuMCommonConfiguration

Parameters included	
Parameter name	**Value**

5.4.2.1.2. EcuMFlexConfiguration

Containers included	
Container name	**Container definition**
ECUM_CAUSE_UNKNOWN	EcuMShutdownCause
ECUM_CAUSE_ECU_STATE	EcuMShutdownCause
ECUM_CAUSE_WDGM	EcuMShutdownCause
ECUM_CAUSE_DCM	EcuMShutdownCause

Containers included	
ECUM_RESET_MCU	EcuMResetMode
ECUM_RESET_WDG	EcuMResetMode
ECUM_RESET_IO	EcuMResetMode
ECUM_STATE_SLEEP	EcuMShutdownTarget
ECUM_STATE_OFF	EcuMShutdownTarget
ECUM_STATE_RESET	EcuMShutdownTarget

Parameters included	
Parameter name	**Value**

5.4.2.1.3. ECUM_CAUSE_UNKNOWN

Parameters included	
Parameter name	**Value**
EcuMShutdownCauseId	0

5.4.2.1.4. ECUM_CAUSE_ECU_STATE

Parameters included	
Parameter name	**Value**
EcuMShutdownCauseId	1

5.4.2.1.5. ECUM_CAUSE_WDGM

Parameters included	
Parameter name	**Value**
EcuMShutdownCauseId	2

5.4.2.1.6. ECUM_CAUSE_DCM

Parameters included	
Parameter name	**Value**

Parameters included	
EcuMShutdownCauseId	3

5.4.2.1.7. ECUM_RESET_MCU

Parameters included	
Parameter name	**Value**
EcuMResetModeId	0

5.4.2.1.8. ECUM_RESET_WDG

Parameters included	
Parameter name	**Value**
EcuMResetModeId	1

5.4.2.1.9. ECUM_RESET_IO

Parameters included	
Parameter name	**Value**
EcuMResetModeId	2

5.4.2.1.10. ECUM_STATE_SLEEP

Parameters included	
Parameter name	**Value**
EcuMShutdownTargetId	80

5.4.2.1.11. ECUM_STATE_OFF

Parameters included	
Parameter name	**Value**

Parameters included	
EcuMShutdownTargetId	128

5.4.2.1.12. ECUM_STATE_RESET

Parameters included	
Parameter name	**Value**
EcuMShutdownTargetId	144

5.4.2.1.13. EcuMCommonConfiguration

Containers included	
Container name	**Container definition**
ECUM_WKSOURCE_POWER	EcuMWakeupSource
ECUM_WKSOURCE_RESET	EcuMWakeupSource
ECUM_WKSOURCE_INTERNAL_RESET	EcuMWakeupSource
ECUM_WKSOURCE_INTERNAL_WDG	EcuMWakeupSource
ECUM_WKSOURCE_EXTERNAL_WDG	EcuMWakeupSource

Parameters included	
Parameter name	**Value**

5.4.2.1.14. ECUM_WKSOURCE_POWER

Parameters included	
Parameter name	**Value**
EcuMWakeupSourceId	0
EcuMValidationTimeout	0.0

5.4.2.1.15. ECUM_WKSOURCE_RESET

Parameters included	
Parameter name	**Value**
EcuMWakeupSourceId	1

Parameters included	
EcuMValidationTimeout	0.0

5.4.2.1.16. ECUM_WKSOURCE_INTERNAL_RESET

Parameters included	
Parameter name	**Value**
EcuMWakeupSourceId	2
EcuMValidationTimeout	0.0

5.4.2.1.17. ECUM_WKSOURCE_INTERNAL_WDG

Parameters included	
Parameter name	**Value**
EcuMWakeupSourceId	3
EcuMValidationTimeout	0.0

5.4.2.1.18. ECUM_WKSOURCE_EXTERNAL_WDG

Parameters included	
Parameter name	**Value**
EcuMWakeupSourceId	4
EcuMValidationTimeout	0.0

5.4.3. Application programming interface (API)

5.4.3.1. Type definitions

5.4.3.1.1. EcuM_ASR32_BootTargetType

Purpose	Type for boot target.

Type	`uint8`
Description	Stores the type of the next boot target. All valid values for this type are defined as preprocessor macros.

5.4.3.1.2. EcuM_ASR32_StateType

Purpose	Type for EcuM configuration.
Type	`uint8`
Description	Encodes states and sub-states of the ECU State Manager. States are encoded in the hi-nibble and sub-states are encoded in the lo-nibble. The sub-state can be determined by ANDing the state value with ECUM_SUBSTATE_MASK. All valid values for this type are defined as preprocessor macros.

5.4.3.1.3. EcuM_ASR32_UserType

Purpose	Type for a user ID of the EcuM.
Type	`uint8`
Description	A unique User ID, which must be defined for each user at system generation time.

5.4.3.1.4. EcuM_ASR40_BootTargetType

Purpose	Type for boot target.
Type	`uint8`
Description	Stores the type of the next boot target. All valid values for this type are defined as preprocessor macros.

5.4.3.1.5. EcuM_ASR40_ShutdownCauseType

Purpose	Type for the cause of shutdown.
Type	`uint8`
Description	Describes the cause for a shutdown by the ECU State Manager. It can be extended by configuration.

	All valid values for this type are defined as preprocessor macros.

5.4.3.1.6. EcuM_ASR40_StateType

Purpose	Type for EcuM configuration.
Type	`uint8`
Description	Encodes states and sub-states of the ECU State Manager. States are encoded in the hi-nibble and sub-states are encoded in the lo-nibble. The sub-state can be determined by ANDing the state value with ECUM_SUBSTATE_MASK. All valid values for this type are defined as preprocessor macros.

5.4.3.1.7. EcuM_ASR40_UserType

Purpose	Type for a user ID of the EcuM.
Type	`uint8`
Description	A unique User ID, which must be defined for each user at system generation time.

5.4.3.1.8. EcuM_BootTargetInternalType

Purpose	Type of internal 64 Bit boot target flag.
Type	`uint32[2]`
Description	Type for storing the 64 Bit representation of the boot target flag

5.4.3.1.9. EcuM_BootTargetType

Purpose	Type for boot target.
Type	`EcuM_ASR40_BootTargetType`
Description	Stores the type of the next boot target. All valid values for this type are defined as preprocessor macros.

5.4.3.1.10. EcuM_DeterminePbConfigurationRetType

Purpose	Return type of callout function EcuM_DeterminePbConfiguration().

Type	`const EcuM_ConfigType *`
Description	This type definition is used to prevent problematic nested compiler abstraction macros as in FUNC(P2CONST(), ..) in the declaration/definition of EcuM_DeterminePbConfiguration().

5.4.3.1.11. EcuM_ResetType

Purpose	Type for the Reset mechanism.
Type	`uint8`
Description	Describes the reset mechanisms supported by the ECU State Manager. It can be extended by configuration. All valid values for this type are defined as preprocessor macros.

5.4.3.1.12. EcuM_ShutdownCauseType

Purpose	EcuM Shutdown Cause Type.
Type	`EcuM_ASR40_ShutdownCauseType`
Description	Provide AUTOSAR 4.0 EcuM_ShutdownCauseType as default to other BSW modules

5.4.3.1.13. EcuM_SleepModeConfigType

Purpose	Type for the status of a wakeup source.	
Type	struct	
Members	`uint8 mcuMode`	Sleep mode of the MCU. Index for the configuration tables of the MCU driver.
	`boolean suspendCPU`	Suspend mode flag. A Flag which is set to true if the CPU is suspended, halted, or powered off during sleep mode. This flag must be set to false if the CPU keeps running while in sleep mode.
	`EcuM_WakeupSourceType wakeupSource`	Sleep period for which all set wakeup sources will be enabled.

Description	This type wraps the Mcu_ModeType. This is done to avoid types of other packages in this API specification. The system designer can define a list containing elements of this type to define the different sleep modes.

5.4.3.1.14. EcuM_StateType

Purpose	EcuM States Type.
Type	`EcuM_ASR40_StateType`
Description	Provide AUTOSAR 4.0 EcuM_StateType as default to other BSW modules

5.4.3.1.15. EcuM_SyncActionType

Purpose	
Type	`uint8`

5.4.3.1.16. EcuM_UserType

Purpose	EcuM Usere Type.
Type	`EcuM_ASR40_UserType`
Description	Provide AUTOSAR 4.0 EcuM_UserType as default to other BSW modules

5.4.3.1.17. EcuM_WakeupReactionType

Purpose	
Type	`uint8`

5.4.3.1.18. EcuM_WakeupSourceType

Purpose	Type for wakeup sources.
Type	`uint32`
Description	The bit field provides one bit for each wakeup source. In WAKEUP state, all bits cleared indicates that no wakeup source is known. In STARTUP state, all bits cleared indicates that no reason for restart or reset is known (in this case, ECUM_-WKSOURCE_RESET shall be assumed.)

	All valid values for this type are defined as preprocessor macros. Additional values may be added at configuration time.

5.4.3.1.19. EcuM_WakeupStatusType

Purpose	Type for the status of a wakeup source.
Type	`uint8`
Description	Describes the possible outcomes of the WAKEUP VALIDATION state. This type may be applied to one wakeup source or a collection of wakeup sources. All valid values for this type are defined as preprocessor macros.

5.4.3.2. Macro constants

5.4.3.2.1. ECUM_ABORT_SYNC

Purpose	Value for EcuM_SyncActionType.
Value	0x01
Description	EcuM should abort synchronization of its cores

5.4.3.2.2. ECUM_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.4.3.2.3. ECUM_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.4.3.2.4. ECUM_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.

Value	3U

5.4.3.2.5. ECUM_BOOT_TARGET_APP

Purpose	Value for EcuM_BootTargetType.
Value	0U
Description	The ECU will boot into the application.

5.4.3.2.6. ECUM_BOOT_TARGET_BOOTLOADER

Purpose	Value for EcuM_BootTargetType.
Value	1U
Description	The ECU will boot into the bootloader.

5.4.3.2.7. ECUM_BOOT_TARGET_OEM_BOOTLOADER

Purpose	Value for EcuM_BootTargetType.
Value	1U
Description	The ECU will boot into the OEM bootloader.

5.4.3.2.8. ECUM_BOOT_TARGET_OEM_BOOTLOADER_INTERNAL_0

Purpose	Value for 1st byte of EcuM_BootTargetInternalType.
Value	0x00FFA55AU
Description	The ECU will boot into the OEM bootloader. This internal value is written to memory to make reading of the values more reliable.

5.4.3.2.9. ECUM_BOOT_TARGET_OEM_BOOTLOADER_INTERNAL_1

Purpose	Value for 2nd byte of EcuM_BootTargetInternalType.
Value	0x2342EB6CU

5.4.3.2.10. ECUM_BOOT_TARGET_SYS_BOOTLOADER

Purpose	Value for EcuM_BootTargetType.
Value	2U
Description	The ECU will boot into the system supplier bootloader.

5.4.3.2.11. ECUM_BOOT_TARGET_SYS_BOOTLOADER_INTERNAL_0

Purpose	Value for 1st byte of EcuM_BootTargetInternalType.
Value	0xEB15C001U
Description	The ECU will boot into the system supplier bootloader. This internal value is written to memory to make reading of the values more reliable.

5.4.3.2.12. ECUM_BOOT_TARGET_SYS_BOOTLOADER_INTERNAL_1

Purpose	Value for 2nd byte of EcuM_BootTargetInternalType.
Value	0x1BADBABEU

5.4.3.2.13. ECUM_CAUSE_DCM

Purpose	Value for EcuM_ShutdownCauseType.
Value	3U
Description	Diagnostic Communication Manager requests a shutdown due to a service request

5.4.3.2.14. ECUM_CAUSE_ECU_STATE

Purpose	Value for EcuM_ShutdownCauseType.
Value	1U
Description	ECU state machine entered a state for shutdown

5.4.3.2.15. ECUM_CAUSE_UNKNOWN

Purpose	Value for EcuM_ShutdownCauseType.

Value	0U
Description	No cause was set.

5.4.3.2.16. ECUM_CAUSE_WDGM

Purpose	Value for EcuM_ShutdownCauseType.
Value	2U
Description	Watchdog Manager detected a failure

5.4.3.2.17. ECUM_CONTINUE_SYNC

Purpose	Value for EcuM_SyncActionType.
Value	0x00
Description	EcuM should continue synchronizing its cores before the sleep/shutdown process

5.4.3.2.18. ECUM_E_INVALID_PAR

Purpose	Development Error Code.
Value	0x13U
Description	A parameter was invalid (unspecific).

5.4.3.2.19. ECUM_E_MISMATCHED_RUN_RELEASE

Purpose	Development Error Code.
Value	0x15U
Description	RUN / POSTRUN was released though it was not requested.

5.4.3.2.20. ECUM_E_MULTIPLE_RUN_REQUESTS

Purpose	Development Error Code.
Value	0x14U

Description	RUN / POSTRUN was requested multiple times by the same user.

5.4.3.2.21. ECUM_E_NULL_POINTER

Purpose	Development Error Code.
Value	0x12U
Description	A null pointer was passed as an argument.

5.4.3.2.22. ECUM_E_PARAM_POINTER

Purpose	Development Error Code.
Value	0x03U
Description	API service called with a NULL pointer.

5.4.3.2.23. ECUM_E_RTE_SWITCH_FAILED

Purpose	Development Error Code.
Value	0x18U
Description	Failed to report EcuM mode switch to RTE.

5.4.3.2.24. ECUM_E_SERVICE_DISABLED

Purpose	Development Error Code.
Value	0x11U
Description	A service was called which was disabled by configuration.

5.4.3.2.25. ECUM_E_STATE_PAR_OUT_OF_RANGE

Purpose	Development Error Code.
Value	0x16U
Description	A state, passed as an argument to a service, was out of range (specific parameter test)

5.4.3.2.26. ECUM_E_UNINIT

Purpose	Development Error Code.
Value	0x10U
Description	A service was called prior to initialization.

5.4.3.2.27. ECUM_E_UNKNOWN_WAKEUP_SOURCE

Purpose	Development Error Code.
Value	0x17U
Description	An unknown wakeup source was passed as a parameter to an API.

5.4.3.2.28. ECUM_E_UNSUCCESSFUL_CORE_START

Purpose	Development Error Code.
Value	0x1AU
Description	The Os core has not been succesfully started.

5.4.3.2.29. ECUM_E_WRONG_CONFIG_PARAM

Purpose	Development Error Code.
Value	0x1BU
Description	The provided configuration is inconsistent.

5.4.3.2.30. ECUM_E_WRONG_CONTEXT

Purpose	Development Error Code.
Value	0x19U
Description	The API has been called from an invalid core context.

5.4.3.2.31. ECUM_MODULE_ID

Purpose	AUTOSAR module identification.

Value	10U

5.4.3.2.32. ECUM_RAM_CHECK_FAILED

Purpose	Error reason value for EcuM_ErrorHook().
Value	0x02U
Description	The RAM check routine returned a failure

5.4.3.2.33. ECUM_SID_CLEAR_WK_EV

Purpose	AUTOSAR API service ID.
Value	22U
Description	Definition of service ID for EcuM_ClearWakeupEvent().

5.4.3.2.34. ECUM_SID_DOWN

Purpose	AUTOSAR API service ID.
Value	31U
Description	Definition of service ID for EcuM_GoDown()

5.4.3.2.35. ECUM_SID_GET_BOOT_TAR

Purpose	AUTOSAR API service ID.
Value	19U
Description	Definition of service ID for EcuM_GetBootTarget().

5.4.3.2.36. ECUM_SID_GET_EXP_WK_EV

Purpose	AUTOSAR API service ID.
Value	25U
Description	Definition of service ID for EcuM_GetExpiredWakeupEvent().

5.4.3.2.37. ECUM_SID_GET_LAST_SHUT_TAR

Purpose	AUTOSAR API service ID.
Value	8U
Description	Definition of service ID for EcuM_GetLastShutdownTarget().

5.4.3.2.38. ECUM_SID_GET_NXT_RCNT_SHUT_CAUSE

Purpose	AUTOSAR API service ID.
Value	30U
Description	Definition of service ID for EcuM_GetNextRecentShutdown().

5.4.3.2.39. ECUM_SID_GET_PEND_WK_EV

Purpose	AUTOSAR API service ID.
Value	13U
Description	Definition of service ID for EcuM_GetPendingWakeupEvent().

5.4.3.2.40. ECUM_SID_GET_RCNT_SHUT_CAUSE

Purpose	AUTOSAR API service ID.
Value	29U
Description	Definition of service ID for EcuM_GetMostRecentShutdown().

5.4.3.2.41. ECUM_SID_GET_SHUT_CAUSE

Purpose	AUTOSAR API service ID.
Value	28U
Description	Definition of service ID for EcuM_GetShutdownCause().

5.4.3.2.42. ECUM_SID_GET_SHUT_TAR

Purpose	AUTOSAR API service ID.

Value	9U
Description	Definition of service ID for EcuM_GetShutdownTarget().

5.4.3.2.43. ECUM_SID_GET_STAT_OF_WK_SRC

Purpose	AUTOSAR API service ID.
Value	23U
Description	Definition of service ID for EcuM_GetStatusOfWakeupSource().

5.4.3.2.44. ECUM_SID_GET_VALID_WK_EV

Purpose	AUTOSAR API service ID.
Value	21U
Description	Definition of service ID for EcuM_GetValidatedWakeupEvents().

5.4.3.2.45. ECUM_SID_GET_VERSION_INFO

Purpose	AUTOSAR API service ID.
Value	0U
Description	Definition of service ID for EcuM_GetVersionInfo().

5.4.3.2.46. ECUM_SID_GO_RUN

Purpose	AUTOSAR API service ID.
Value	99U
Description	Definition of service ID for EcuM_BswMInitFinished().

5.4.3.2.47. ECUM_SID_HALT

Purpose	AUTOSAR API service ID.
Value	32U

Description	Definition of service ID for EcuM_GoHalt()

5.4.3.2.48. ECUM_SID_INIT

Purpose	AUTOSAR API service ID.
Value	1U
Description	Definition of service ID for EcuM_Init()

5.4.3.2.49. ECUM_SID_KILL_ALL_RUN_REQUESTS

Purpose	AUTOSAR API service ID.
Value	5U
Description	Definition of service ID for EcuM_KillAllRUNRequests()

5.4.3.2.50. ECUM_SID_MAIN_FUNCTION

Purpose	AUTOSAR API service ID.
Value	24U
Description	Definition of service ID for EcuM_MainFunction().

5.4.3.2.51. ECUM_SID_POLL

Purpose	AUTOSAR API service ID.
Value	33U
Description	Definition of service ID for EcuM_GoPoll()

5.4.3.2.52. ECUM_SID_RELEASE_POST_RUN

Purpose	AUTOSAR API service ID.
Value	11U
Description	Definition of service ID for EcuM_ReleasePOST_RUN()

5.4.3.2.53. ECUM_SID_RELEASE_RUN

Purpose	AUTOSAR API service ID.
Value	4U
Description	Definition of service ID for EcuM_ReleaseRUN()

5.4.3.2.54. ECUM_SID_REQUEST_POST_RUN

Purpose	AUTOSAR API service ID.
Value	10U
Description	Definition of service ID for EcuM_RequestPOST_RUN()

5.4.3.2.55. ECUM_SID_REQUEST_RUN

Purpose	AUTOSAR API service ID.
Value	3U
Description	Definition of service ID for EcuM_RequestRUN()

5.4.3.2.56. ECUM_SID_SEL_BOOT_TAR

Purpose	AUTOSAR API service ID.
Value	18U
Description	Definition of service ID for EcuM_SelectBootTarget().

5.4.3.2.57. ECUM_SID_SEL_SHUT_CAUSE

Purpose	AUTOSAR API service ID.
Value	27U
Description	Definition of service ID for EcuM_SelectShutdownCause().

5.4.3.2.58. ECUM_SID_SEL_SHUT_TAR

Purpose	AUTOSAR API service ID.

Value	6U
Description	Definition of service ID for EcuM_SelectShutdownTarget().

5.4.3.2.59. ECUM_SID_SET_WK_EV

Purpose	AUTOSAR API service ID.
Value	12U
Description	Definition of service ID for EcuM_SetWakeupEvent().

5.4.3.2.60. ECUM_SID_SHUTDOWN

Purpose	AUTOSAR API service ID.
Value	2U
Description	Definition of service ID for EcuM_Shutdown()

5.4.3.2.61. ECUM_SID_STARTUP_TWO

Purpose	AUTOSAR API service ID.
Value	26U
Description	Definition of service ID for EcuM_StartupTwo().

5.4.3.2.62. ECUM_SID_VALIDATE_WK_EV

Purpose	AUTOSAR API service ID.
Value	20U
Description	Definition of service ID for EcuM_ValidateWakeupEvent.

5.4.3.2.63. ECUM_STATE_APP_POST_RUN

Purpose	Value for EcuM_StateType.
Value	0x33U

Description	Definition of ECUM_STATE_APP_POST_RUN, substate of EcuM RUN

5.4.3.2.64. ECUM_STATE_APP_RUN

Purpose	Value for EcuM_StateType.
Value	0x32U
Description	Definition of ECUM_STATE_APP_RUN, substate of EcuM RUN.

5.4.3.2.65. ECUM_STATE_GO_OFF_ONE

Purpose	Value for EcuM_StateType.
Value	0x4dU
Description	Definition of ECUM_STATE_GO_OFF_ONE, substate of EcuM SHUTDOWN

5.4.3.2.66. ECUM_STATE_GO_OFF_TWO

Purpose	Value for EcuM_StateType.
Value	0x4eU
Description	Definition of ECUM_STATE_GO_OFF_TWO, substate of EcuM SHUTDOWN

5.4.3.2.67. ECUM_STATE_GO_SLEEP

Purpose	Value for EcuM_StateType.
Value	0x49U
Description	Definition of ECUM_STATE_GO_SLEEP, substate of EcuM SHUTDOWN

5.4.3.2.68. ECUM_STATE_OFF

Purpose	Value for EcuM_StateType.
Value	0x80U
Description	Definition of ECUM_STATE_OFF, indicates EcuM OFF state.

5.4.3.2.69. ECUM_STATE_PREP_SHUTDOWN

Purpose	Value for EcuM_StateType.
Value	0x44U
Description	Definition of ECUM_STATE_PREP_SHUTDOWN, substate of EcuM SHUTDOWN

5.4.3.2.70. ECUM_STATE_RESET

Purpose	Value for EcuM_StateType.
Value	0x90U
Description	Definition of ECUM_STATE_RESET, indicates EcuM RESET state.

5.4.3.2.71. ECUM_STATE_RUN

Purpose	Value for EcuM_StateType.
Value	0x30U
Description	Definition of ECUM_STATE_RUN, indicates EcuM RUN state.

5.4.3.2.72. ECUM_STATE_SHUTDOWN

Purpose	Value for EcuM_StateType.
Value	0x40U
Description	Definition of ECUM_STATE_SHUTDOWN, indicates EcuM SHUTDOWN state.

5.4.3.2.73. ECUM_STATE_SLEEP

Purpose	Value for EcuM_StateType.
Value	0x50U
Description	Definition of ECUM_STATE_SLEEP, indicates EcuM SLEEP state.

5.4.3.2.74. ECUM_STATE_STARTUP

Purpose	Value for EcuM_StateType.

Value	0x10U
Description	Definition of ECUM_STATE_STARTUP, indicates EcuM STARTUP state.

5.4.3.2.75. ECUM_STATE_STARTUP_ONE

Purpose	Value for EcuM_StateType.
Value	0x11U
Description	Definition of ECUM_STATE_STARTUP_ONE, substate of EcuM STARTUP.

5.4.3.2.76. ECUM_STATE_STARTUP_TWO

Purpose	Value for EcuM_StateType.
Value	0x12U
Description	Definition of ECUM_STATE_STARTUP_TWO, substate of EcuM STARTUP.

5.4.3.2.77. ECUM_STATE_WAKEUP

Purpose	Value for EcuM_StateType.
Value	0x20U
Description	Definition of ECUM_STATE_WAKEUP, indicates EcuM WAKEUP state.

5.4.3.2.78. ECUM_STATE_WAKEUP_ONE

Purpose	Value for EcuM_StateType.
Value	0x21U
Description	Definition of ECUM_STATE_WAKEUP_ONE, substate of EcuM WAKEUP.

5.4.3.2.79. ECUM_STATE_WAKEUP_REACTION

Purpose	Value for EcuM_StateType.
Value	0x23U
Description	Definition of ECUM_STATE_WAKEUP_REACTION, substate of EcuM WAKEUP.

5.4.3.2.80. ECUM_STATE_WAKEUP_TTII

Purpose	Value for EcuM_StateType.
Value	0x26U
Description	Definition of ECUM_STATE_WAKEUP_TTII, substate of EcuM WAKEUP.

5.4.3.2.81. ECUM_STATE_WAKEUP_TWO

Purpose	Value for EcuM_StateType.
Value	0x24U
Description	Definition of ECUM_STATE_WAKEUP_TWO, substate of EcuM WAKEUP.

5.4.3.2.82. ECUM_STATE_WAKEUP_VALIDATION

Purpose	Value for EcuM_StateType.
Value	0x22U
Description	Definition of ECUM_STATE_WAKEUP_VALIDATION, substate of EcuM WAKEUP.

5.4.3.2.83. ECUM_STATE_WAKEUP_WAKESLEEP

Purpose	Value for EcuM_StateType.
Value	0x25U
Description	Definition of ECUM_STATE_WAKEUP_WAKESLEEP, substate of EcuM WAKEUP.

5.4.3.2.84. ECUM_SUBSTATE_MASK

Purpose	Bitmask for sub states.
Value	0x0FU

5.4.3.2.85. ECUM_SUPERSTATE_MASK

Purpose	Bitmask for super states.

Value	0xF0U

5.4.3.2.86. ECUM_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	5U

5.4.3.2.87. ECUM_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	15U

5.4.3.2.88. ECUM_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	4U

5.4.3.2.89. ECUM_VAR_BOOTTARGET

Purpose	
Value	ECUM_VAR_CLEARED

5.4.3.2.90. ECUM_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.4.3.2.91. ECUM_WKACT_RUN

Purpose	
Value	0x00U

5.4.3.2.92. ECUM_WKACT_SHUTDOWN

Purpose	
Value	0x03U

5.4.3.2.93. ECUM_WKACT_TTII

Purpose	
Value	0x02U

5.4.3.2.94. ECUM_WKSOURCE_ALL_SOURCES

Purpose	Value for EcuM_WakeupSourceType.
Value	0xFFFFFFFFU
Description	Set all wakeup sources.

5.4.3.2.95. ECUM_WKSTATUS_DISABLED

Purpose	Value for EcuM_WakeupStatusType.
Value	4U
Description	The wakeup source is disabled and does not detect wakeup events.

5.4.3.2.96. ECUM_WKSTATUS_ENABLED

Purpose	Value for EcuM_WakeupStatusType.
Value	6U
Description	The wakeup source is enabled and does not detect wakeup events.

5.4.3.2.97. ECUM_WKSTATUS_EXPIRED

Purpose	Value for EcuM_WakeupStatusType.

Value	3U
Description	The wakeup event has not been validated and has therefore expired.

5.4.3.2.98. ECUM_WKSTATUS_NONE

Purpose	Value for EcuM_WakeupStatusType.
Value	0U
Description	No pending wakeup event was detected.

5.4.3.2.99. ECUM_WKSTATUS_PENDING

Purpose	Value for EcuM_WakeupStatusType.
Value	1U
Description	The wakeup event was detected but not yet validated.

5.4.3.2.100. ECUM_WKSTATUS_VALIDATED

Purpose	Value for EcuM_WakeupStatusType.
Value	2U
Description	The wakeup event is valid.

5.4.3.2.101. EcuM_GetBootTarget

Purpose	Wrapping macro for EcuM_GetBootTarget.
Value	EcuM_ASR40_GetBootTarget
Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.2.102. EcuM_GetLastShutdownTarget

Purpose	Wrapping macro for EcuM_GetLastShutdownTarget.
Value	EcuM_ASR40_GetLastShutdownTarget

Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.2.103. EcuM_GetMostRecentShutdown

Purpose	Wrapping macro for EcuM_GetMostRecentShutdown.
Value	EcuM_ASR40_GetMostRecentShutdown
Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.2.104. EcuM_GetNextRecentShutdown

Purpose	Wrapping macro for EcuM_GetNextRecentShutdown.
Value	EcuM_ASR40_GetNextRecentShutdown
Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.2.105. EcuM_GetShutdownCause

Purpose	Wrapping macro for EcuM_GetShutdownCause.
Value	EcuM_ASR40_GetShutdownCause
Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.2.106. EcuM_GetShutdownTarget

Purpose	Wrapping macro for EcuM_GetShutdownTarget.
Value	EcuM_ASR40_GetShutdownTarget
Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.2.107. EcuM_SelectBootTarget

Purpose	Wrapping macro for EcuM_SelectBootTarget.
Value	EcuM_ASR40_SelectBootTarget
Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.2.108. EcuM_SelectShutdownCause

Purpose	Wrapping macro for EcuM_SelectShutdownCause.
Value	EcuM_ASR40_SelectShutdownCause
Description	Provides an AUTOSAR 4.0 API as default to other BSW modules.

5.4.3.3. Objects

5.4.3.3.1. EcuM_BootTarget

Purpose	
Type	`EcuM_BootTargetInternalType`

5.4.3.4. Functions

5.4.3.4.1. EcuM_AL_DriverInitOne

Purpose	Callout in startup sequence one.
Synopsis	`void` **`EcuM_AL_DriverInitOne`** `( const EcuM_ConfigType * ConfigPtr );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This callout provides driver initialization and other hardware-related startup activities in case of a power on reset. This callout is invoked by EcuM before Gpt and OS is started.

5.4.3.4.2. EcuM_AL_DriverInitZero

Purpose	Callout in startup sequence zero.
Synopsis	`void` **`EcuM_AL_DriverInitZero`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Description	This callout provides driver initialization and other hardware-related startup activities in case of a power on reset. This callout is invoked by EcuM before Mcu is started.

5.4.3.4.3. EcuM_AL_DriverRestart

Purpose	Callout for restarting drivers.
Synopsis	`void EcuM_AL_DriverRestart ( const EcuM_ConfigType * ConfigPtr );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This callout provides driver initialization and other hardware-related startup activities in the wakeup case. This callout is invoked by EcuM early in WAKEUP 1 state.

5.4.3.4.4. EcuM_AL_Reset

Purpose	Callout for resetting the ECU.
Synopsis	`void EcuM_AL_Reset ( EcuM_ResetType reset );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This callout shall take the code for resetting the ECU.

5.4.3.4.5. EcuM_AL_SetProgrammableInterrupts

Purpose	Callout from the startup phase.
Synopsis	`void EcuM_AL_SetProgrammableInterrupts ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This callout shall set the interrupts on ECUs with programmable interrupts. This callout is invoked by EcuM before Mcu is started.

5.4.3.4.6. EcuM_AL_SwitchOff

Purpose	Callout for switching off power.
Synopsis	`void` **`EcuM_AL_SwitchOff`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This callout shuts off the power supply of the ECU. If the ECU cannot unpower itself, a reset may be an adequate reaction.

5.4.3.4.7. EcuM_ASR32_GetBootTarget

Purpose	Get boot target using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_GetBootTarget`** `( EcuM_ASR32_BootTargetType * target );`	
Service ID	19	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`target`	The currently selected boot target.
Return Value	Standard Return Code	
	`E_OK`	No Errors (Always).
Description	This function gets the currently selected boot target.	

5.4.3.4.8. EcuM_ASR32_GetLastShutdownTarget

Purpose	Get target state for shutdown sequence using AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_GetLastShutdownTarget`** `( EcuM_ASR32_StateType * shutdownTarget , uint8 * sleepMode );`	
Service ID	ECUM_SID_GET_LAST_SHUT_TAR	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`shutdownTarget`	Set to one of the following values:

		▶ ECUM_STATE_SLEEP ▶ ECUM_STATE_RESET ▶ ECUM_STATE_OFF
	`sleepMode`	If the return parameter is ECUM_STATE_SLEEP, this output parameter tells which of the configured sleep modes was actually chosen (index into EcuM_SleepMode). If the return parameter is ECUM_STATE_RESET, this out parameter tells which of the configured reset modes was actually chosen.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	The new shutdown target was not set.
Description	This function always returns the same value until the next shutdown. The return value describes the ECU state from which the last wakeup or power up occurred. Development Error Detection ▶ ECUM_E_NULL_POINTER: If a null pointer was passed as argument.	

5.4.3.4.9. EcuM_ASR32_GetShutdownTarget

Purpose	Get target state for shutdown sequence using AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_GetShutdownTarget`** `( EcuM_ASR32_StateType * shutdownTarget , uint8 * sleepMode );`	
Service ID	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`shutdownTarget`	Is set to one of the following values: ▶ ECUM_STATE_SLEEP ▶ ECUM_STATE_RESET ▶ ECUM_STATE_OFF
	`sleepMode`	If the return parameter is ECUM_STATE_SLEEP, then this output parameter identifies which of the configured sleep modes

		was actually chosen (index into EcuM_SleepMode). If the return parameter is ECUM_STATE_RESET, this out parameter tells which of the configured reset modes was actually chosen.
Return Value	Standard Return Code	
	`E_OK`	No Errors. (Always).
	`E_NOT_OK`	The service has failed (e.g. due to a Null pointer being passed).
Description	This function always returns the selected shutdown target as set by EcuM_SelectShutdownTarget. Development Error Detection ▶ ECUM_E_NULL_POINTER: If a null pointer was passed as argument.	

5.4.3.4.10. EcuM_ASR32_ReleasePOST_RUN

Purpose	Releases a POST RUN request previously done with a call to EcuM_RequestPOST_RUN using AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_ReleasePOST_RUN`** `( EcuM_ASR32_UserType user );`	
Service ID	ECUM_SID_RELEASE_POST_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`user`	ID of the entity releasing the POST RUN state.
Return Value	Standard Return Code	
	`E_OK`	The release request was accepted by EcuM.
	`E_NOT_OK`	The release request was not accepted by EcuM, a detailed error condition was sent to DET.
Description	After the last POST_RUN request has been released, EcuM indicates the SHUTDOWN mode to Rte. This mode transition can then be caught by BswM to proceed with the shutdown or sleep tasks. The service is intended for implementing AUTOSAR ports.	

5.4.3.4.11. EcuM_ASR32_ReleaseRUN

Purpose	Releases a RUN request previously done with a call to EcuM_RequestRUN using AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_ReleaseRUN`** `( EcuM_ASR32_UserType user );`	
Service ID	ECUM_SID_RELEASE_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`user`	ID of the entity releasing the RUN state.
Return Value	Standard Return Code	
	`E_OK`	The release request was accepted by EcuM.
	`E_NOT_OK`	The release request was not accepted by EcuM, a detailed error condition was sent to DET.
Description	When the last RUN request is released, EcuM leaves the RUN state. The service is intended for implementing AUTOSAR ports.	

5.4.3.4.12. EcuM_ASR32_RequestPOST_RUN

Purpose	Places a request for the POST RUN state using AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_RequestPOST_RUN`** `( EcuM_ASR32_UserType user );`	
Service ID	ECUM_SID_REQUEST_POST_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`user`	ID of the entity requesting the POST RUN state.
Return Value	Standard Return Code	
	`E_OK`	The request was accepted by EcuM.
	`E_NOT_OK`	The request was not accepted by EcuM, a detailed error condition was sent to DET.

Description	EcuM enters POST_RUN state after all RUN requests have been released and stays there for as long as POST_RUN requests are pending. Requests can be placed by every user made known to the state manager at configuration time. Requests for RUN and POST RUN must be tracked independently (in other words: two independent variables). The service is intended for implementing AUTOSAR ports

5.4.3.4.13. EcuM_ASR32_RequestRUN

Purpose	Places a request for the RUN state using AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_RequestRUN`** `( EcuM_ASR32_UserType user );`	
Service ID	ECUM_SID_REQUEST_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`user`	ID of the entity requesting the RUN state.
Return Value	Standard Return Code	
	`E_OK`	The request was accepted by EcuM.
	`E_NOT_OK`	The request was not accepted by EcuM, a detailed error condition was sent to DET.
Description	EcuM enters RUN mode after the EcuM_GoRun() callback has been called and stays there for as long as RUN requests requested by this API function are pending. Requests can be placed by every user made known to the state manager at configuration time.	

5.4.3.4.14. EcuM_ASR32_SelectBootTarget

Purpose	Select boot target using AUTOSAR 4.0 API.
Synopsis	`Std_ReturnType` **`EcuM_ASR32_SelectBootTarget`** `( EcuM_ASR32_BootTargetType target );`
Service ID	18
Sync/Async	Synchronous
Reentrancy	Reentrant

Parameters (in)	`target`	The selected boot target.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	The new boot target was not accepted by EcuM.
Description	This function sets the target which is used after reboot.	

5.4.3.4.15. EcuM_ASR32_SelectShutdownTarget

Purpose	Select target state for shutdown sequence using AUTOSAR 3.2 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR32_SelectShutdownTarget`** `( EcuM_ASR32_StateType target , uint8 mode );`	
Service ID	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`target`	The selected shutdown target. Only the following subset of the EcuM_StateType value range is accepted: ▶ ECUM_STATE_SLEEP ▶ ECUM_STATE_RESET ▶ ECUM_STATE_OFF
	`mode`	An index like value which can be dereferenced to a sleep mode (EcuM_SleepModeConfigType). Available sleep modes are defined at configuration time and are stored in the EcuM_SleepMode list. The mode parameter is only used if the target parameter equals to ECUM_STATE_SLEEP. In all other cases, it is ignored.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	The new shutdown target was not set.
Description	This function doesn't initiate any shutdown activities but only stores the target state for later use when performing a shutdown sequence. During SHUTDOWN a change of the shutdown target from and to the value ECUM_STATE_SLEEP is not allowed.	

	Development Error Detection ▶ ECUM_E_STATE_PAR_OUT_OF_RANGE: On invalid value for parameter target.

5.4.3.4.16. EcuM_ASR40_GetBootTarget

Purpose	Get boot target using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_GetBootTarget`** `( EcuM_ASR40_BootTargetType * target );`	
Service ID	19	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`target`	The currently selected boot target.
Return Value	Standard Return Code	
	`E_OK`	No Errors (Always).
Description	This function gets the currently selected boot target.	

5.4.3.4.17. EcuM_ASR40_GetLastShutdownTarget

Purpose	Get target state for shutdown sequence using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_GetLastShutdownTarget`** `( EcuM_ASR40_StateType * shutdownTarget , uint8 * sleepMode );`	
Service ID	ECUM_SID_GET_LAST_SHUT_TAR	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`shutdownTarget`	Set to one of the following values: ▶ ECUM_STATE_SLEEP ▶ ECUM_STATE_RESET ▶ ECUM_STATE_OFF
	`sleepMode`	If the return parameter is ECUM_STATE_SLEEP, this output parameter tells which of the configured sleep modes was actually chosen (index into EcuM_SleepMode). If the return parameter is ECUM_STATE_-

		RESET, this out parameter tells which of the configured reset modes was actually chosen.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	The new shutdown target was not set.
Description	This function always returns the same value until the next shutdown. The return value describes the ECU state from which the last wakeup or power up occurred. Development Error Detection ▶ ECUM_E_NULL_POINTER: If a null pointer was passed as argument.	

5.4.3.4.18. EcuM_ASR40_GetMostRecentShutdown

Purpose	Get most recent shutdown operation using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_GetMostRecentShutdown`** `( EcuM_ASR40_StateType * target , uint8 * mode , EcuM_ASR40_ShutdownCauseType * cause , uint32 * time );`	
Service ID	ECUM_SID_GET_RCNT_SHUT_CAUSE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`target`	One of these values is returned: ▶ ECUM_STATE_SLEEP ▶ ECUM_STATE_RESET ▶ ECUM_STATE_OFF
	`mode`	This parameter tells which of the configured sleep modes(target is ECUM_STATE_SLEEP) or which of the reset mechanisms(target is ECUM_STATE_RESET) was actually chosen.
	`cause`	The selected shutdown cause
	`time`	Absolute time of the shutdown if supported by hardware.
Return Value	Standard Return Code	
	`E_OK`	The service has succeeded.

	`E_NOT_OK`	The service has failed, e.g. due to NULL pointer being passed
Description	This function returns information about the most recent shutdown operation. Development Error Detection ▶ ECUM_E_NULL_POINTER: If a null pointer was passed as argument.	

5.4.3.4.19. EcuM_ASR40_GetNextRecentShutdown

Purpose	Get next most recent shutdown operation using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_GetNextRecentShutdown`** `( EcuM_ASR40_StateType * target , uint8 * mode , EcuM_ASR40_ShutdownCauseType * cause , uint32 * time );`	
Service ID	ECUM_SID_GET_NXT_RCNT_SHUT_CAUSE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`target`	One of these values is returned: ▶ ECUM_STATE_SLEEP ▶ ECUM_STATE_RESET ▶ ECUM_STATE_OFF
	`mode`	This parameter tells which of the configured sleep modes(target is ECUM_STATE_SLEEP) or which of the reset mechanisms(target is ECUM_STATE_RESET) was actually chosen.
	`cause`	The selected shutdown cause
	`time`	Absolute time of the shutdown if supported by hardware.
Return Value	Standard Return Code	
	`E_OK`	The service has succeeded.
	`E_NOT_OK`	The service has failed, e.g. due to NULL pointer being passed
Description	This function returns information about the next most recent shutdown operation. All stored shutdown information can be read by first calling EcuM_GetMostRecentShutdown and then looping over	

	Development Error Detection - ECUM_E_NULL_POINTER: If a null pointer was passed as argument.

5.4.3.4.20. EcuM_ASR40_GetShutdownCause

Purpose	Get the cause of shutdown using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_GetShutdownCause`** `( EcuM_ASR40_ShutdownCauseType * shutdownCause );`	
Service ID	ECUM_SID_GET_SHUT_CAUSE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`shutdownCause`	The selected cause of the next shutdown.
Return Value	Standard Return Code	
	`E_OK`	The service has succeeded.
	`E_NOT_OK`	The service has failed, e.g. due to NULL pointer being passed
Description	This function returns the selected shutdown cause as set by EcuM_SelectShutdownCause. Development Error Detection - ECUM_E_NULL_POINTER: If a null pointer was passed as argument.	

5.4.3.4.21. EcuM_ASR40_GetShutdownTarget

Purpose	Get target state for shutdown sequence using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_GetShutdownTarget`** `( EcuM_ASR40_StateType * shutdownTarget , uint8 * sleepMode );`	
Service ID	ECUM_SID_GET_SHUT_TAR	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`shutdownTarget`	Is set to one of the following values: - ECUM_STATE_SLEEP - ECUM_STATE_RESET

		▶ ECUM_STATE_OFF
	`sleepMode`	If the return parameter is ECUM_STATE_SLEEP, then this output parameter identifies which of the configured sleep modes was actually chosen (index into EcuM_SleepMode). If the return parameter is ECUM_STATE_RESET, this out parameter tells which of the configured reset modes was actually chosen.
Return Value	Standard Return Code	
	`E_OK`	No Errors. (Always).
	`E_NOT_OK`	The service has failed (e.g. due to a Null pointer being passed).
Description	This function always returns the selected shutdown target as set by EcuM_SelectShutdownTarget. Development Error Detection ▶ ECUM_E_NULL_POINTER: If a null pointer was passed as argument.	

5.4.3.4.22. EcuM_ASR40_ReleasePOST_RUN

Purpose	Releases a POST RUN request previously done with a call to EcuM_RequestPOST_RUN using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_ReleasePOST_RUN`** `( EcuM_ASR40_UserType user );`	
Service ID	ECUM_SID_RELEASE_POST_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`user`	ID of the entity releasing the POST RUN state.
Return Value	Standard Return Code	
	`E_OK`	The release request was accepted by EcuM.
	`E_NOT_OK`	The release request was not accepted by EcuM, a detailed error condition was sent to DET.

Description	After the last POST_RUN request has been released, EcuM indicates the SHUTDOWN mode to Rte. This mode transition can then be caught by BswM to proceed with the shutdown or sleep tasks. The service is intended for implementing AUTOSAR ports.

5.4.3.4.23. EcuM_ASR40_ReleaseRUN

Purpose	Releases a RUN request previously done with a call to EcuM_RequestRUN using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType EcuM_ASR40_ReleaseRUN ( EcuM_ASR40_UserType user );`	
Service ID	ECUM_SID_RELEASE_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`user`	ID of the entity releasing the RUN state.
Return Value	Standard Return Code	
	`E_OK`	The release request was accepted by EcuM.
	`E_NOT_OK`	The release request was not accepted by EcuM, a detailed error condition was sent to DET.
Description	When the last RUN request is released, EcuM leaves the RUN state. The service is intended for implementing AUTOSAR ports.	

5.4.3.4.24. EcuM_ASR40_RequestPOST_RUN

Purpose	Places a request for the POST RUN state using AUTOSAR 4.0 API.
Synopsis	`Std_ReturnType EcuM_ASR40_RequestPOST_RUN ( EcuM_ASR40_UserType user );`
Service ID	ECUM_SID_REQUEST_POST_RUN
Sync/Async	Synchronous
Reentrancy	Reentrant

Parameters (in)	`user`	ID of the entity requesting the POST RUN state.
Return Value	Standard Return Code	
	`E_OK`	The request was accepted by EcuM.
	`E_NOT_OK`	The request was not accepted by EcuM, a detailed error condition was sent to DET.
Description	EcuM enters POST_RUN state after all RUN requests have been released and stays there for as long as POST_RUN requests are pending. Requests can be placed by every user made known to the state manager at configuration time. Requests for RUN and POST RUN must be tracked independently (in other words: two independent variables). The service is intended for implementing AUTOSAR ports	

5.4.3.4.25. EcuM_ASR40_RequestRUN

Purpose	Places a request for the RUN state using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_RequestRUN`** `( EcuM_ASR40_UserType user );`	
Service ID	ECUM_SID_REQUEST_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`user`	ID of the entity requesting the RUN state.
Return Value	Standard Return Code	
	`E_OK`	The request was accepted by EcuM.
	`E_NOT_OK`	The request was not accepted by EcuM, a detailed error condition was sent to DET.
Description	EcuM enters RUN mode after the EcuM_GoRun() callback has been called and stays there for as long as RUN requests requested by this API function are pending. Requests can be placed by every user made known to the state manager at configuration time.	

5.4.3.4.26. EcuM_ASR40_SelectBootTarget

Purpose	Select boot target using AUTOSAR 4.0 API.

Synopsis	`Std_ReturnType` **`EcuM_ASR40_SelectBootTarget`** `( EcuM_ASR40_BootTargetType target );`	
Service ID	18	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`target`	The selected boot target.
Return Value	Standard Return Code	
	`E_OK`	No Errors.
	`E_NOT_OK`	The new boot target was not accepted by EcuM.
Description	This function sets the target which is used after reboot.	

5.4.3.4.27. EcuM_ASR40_SelectShutdownCause

Purpose	Select cause of shutdown using AUTOSAR 4.0 API.	
Synopsis	`Std_ReturnType` **`EcuM_ASR40_SelectShutdownCause`** `( EcuM_ASR40_ShutdownCauseType target );`	
Service ID	ECUM_SID_SEL_SHUT_CAUSE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`target`	The selected shutdown cause
Return Value	Standard Return Code	
	`E_OK`	The new shutdown cause was set.
	`E_NOT_OK`	The new shutdown cause was not set.
Description	This function elects the cause for a shutdown. Development Error Detection ▶ ECUM_E_NULL_POINTER: If a null pointer was passed as argument.	

5.4.3.4.28. EcuM_CheckRamHash

Purpose	Callout for checking RAM integrity .

Synopsis	`uint8 EcuM_CheckRamHash ( void );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Return Value	Result of Check	
	`0`	RAM integrity test failed.
	`else`	RAM integrity test passed.
Description	This callout provides a RAM integrity test to ensure that RAM contents is still consistent after a long SLEEP duration. This callout is invoked by EcuM early in WAKEUP 1 state.	

5.4.3.4.29. EcuM_CheckValidation

Purpose	Callout which checks pending wakeup sources.	
Synopsis	`void EcuM_CheckValidation ( EcuM_WakeupSourceType wakeupSource );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`wakeupSource`	All pending wakeup sources.
Description	This callout is invoked in wakeup validation state. In this callout the system integrator has to check the wakeup sources for validation of a pending wakeup event.	

5.4.3.4.30. EcuM_CheckWakeup

Purpose	Callout which checks wakeup sources.	
Synopsis	`void EcuM_CheckWakeup ( EcuM_WakeupSourceType wakeupSource );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`wakeupSource`	All wakeup sources, activated in the current sleep
Parameters (in,out)	`wakeupSource`	All wakeup sources, activated in the current sleep
Description	In this callout the system integrator checks the wakeup sources for detected wakeup events. These wakeup events are reported using EcuM_SetWakeupEvent().	

	This callout is invoked during sleep state or is invoked by an ISR resulting from an wakeup event, in case the CPU is powered of during sleep.

5.4.3.4.31. EcuM_ClearWakeupEvent

Purpose	Clear wakeup events.	
Synopsis	`void EcuM_ClearWakeupEvent ( EcuM_WakeupSourceType sources );`	
Service ID	ECUM_SID_CLEAR_WK_EV	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant Non-Interruptible	
Parameters (in)	`sources`	Events of wakeup sources to be cleared.
Description	This function clears the wakeup events for a set of sources.	

5.4.3.4.32. EcuM_DefaultInitListOne

Purpose	Default implementation of the EcuMDriverInitListOne.
Synopsis	`void EcuM_DefaultInitListOne ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function provides the default implementation of the EcuMDriverInitListOne. When this function is called, the initialization of the configured EcuMDriverInitItems within the EcuM configuration container EcuMDriverInitListOne will be carried out.

5.4.3.4.33. EcuM_DefaultInitListZero

Purpose	Default implementation of the EcuMDriverInitListZero.
Synopsis	`void EcuM_DefaultInitListZero ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function provides the default implementation of the EcuMDriverInitListZero. When this function is called, the initialization of the configured EcuMDriverInitItems within the EcuM configuration container EcuMDriverInitListZero will be carried out.

5.4.3.4.34. EcuM_DefaultRestartList

Purpose	Default implementation of the EcuMDriverRestartList.
Synopsis	`void` **`EcuM_DefaultRestartList`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function provides the default implementation of the EcuMDriverRestartList. When this function is called, the initialization of the configured EcuMDriverInitItems defined within the EcuM configuration container EcuMDriverRestartList will be carried out.

5.4.3.4.35. EcuM_DeterminePbConfiguration

Purpose	Callout which determines post-build configuration.
Synopsis	`EcuM_DeterminePbConfigurationRetType` **`EcuM_DeterminePbConfiguration`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Return Value	pointer to post-build configuration
Description	In this callout the system integrator has to return the actual post-build configuration for the EcuM. This callout is invoked after EcuM_AL_DriverInitZero()

5.4.3.4.36. EcuM_DisableWakeupSources

Purpose	Callout which disables pending wakeup sources.	
Synopsis	`void` **`EcuM_DisableWakeupSources`** `( EcuM_WakeupSourceType wakeupSource );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`wakeupSource`	All pending wakeup sources.
Parameters (in,out)	`wakeupSource`	All pending wakeup sources.
Description	In this callout the system integrator can disable the functionality of pending wakeup source to set a wakeup.	

	This callout is invoked after a wakeup event is detected.

5.4.3.4.37. EcuM_EnableWakeupSources

Purpose	Callout which enables wakeup sources.	
Synopsis	`void` **`EcuM_EnableWakeupSources`** `( EcuM_WakeupSourceType wakeupSource );`	
Parameters (in)	`wakeupSource`	All wakeup sources, activated in the current sleep mode.
Parameters (in,out)	`wakeupSource`	All wakeup sources, activated in the current sleep mode.
Description	In this callout the system integrator enables the wakeup functionality of a wakeup source, which shall be activated in the current sleep mode. This callout is invoked before EcuM enter in Sleep.	

5.4.3.4.38. EcuM_ErrorHook

Purpose	Callout on errors.	
Synopsis	`void` **`EcuM_ErrorHook`** `( Std_ReturnType reason );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`reason`	Reason for calling the error hook. Currently the EcuM only reports ECUM_RAM_CHECK_FAILED with the error hook function.
Description	This function is used in unrecoverable error situations. The ECU State Manager will call the error hook. It is up to the system integrator to react accordingly (reset, halt, restart, safe state etc.).	

5.4.3.4.39. EcuM_GenerateRamHash

Purpose	Callout for writing a RAM hash.

Synopsis	`void EcuM_GenerateRamHash ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This callout is invoked by EcuM just before putting the ECU physically to sleep.

5.4.3.4.40. EcuM_GetExpiredWakeupEvents

Purpose	Get expired wakeup events.
Synopsis	`EcuM_WakeupSourceType EcuM_GetExpiredWakeupEvents ( void );`
Service ID	ECUM_SID_GET_EXP_WK_EV
Sync/Async	Synchronous
Reentrancy	Non Reentrant Non-Interruptible
Return Value	All events that have been set and for which validation has failed.
Description	This function retrieves wakeup events that have been set and for which validation has failed. Events which do not need validation are never be reported by this function.

5.4.3.4.41. EcuM_GetPendingWakeupEvents

Purpose	Gets pending wakeup events.
Synopsis	`EcuM_WakeupSourceType EcuM_GetPendingWakeupEvents ( void );`
Service ID	ECUM_SID_GET_PEND_WK_EV
Sync/Async	Synchronous
Reentrancy	Non-Reentrant Non-Interruptible
Return Value	Wakeup sources which have pending events.
Description	This function gets all wakeup events which have been set but not yet validated.

5.4.3.4.42. EcuM_GetStatusOfWakeupSource

Purpose	Get the status of wakeup sources.
Synopsis	`EcuM_WakeupStatusType EcuM_GetStatusOfWakeupSource ( EcuM_WakeupSourceType sources );`

Service ID	ECUM_SID_GET_STAT_OF_WK_SRC	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`sources`	Bit mask of the wakeup sources for which the status is requested.
Return Value	Sum status of all wakeup sources.	
	`ECUM_WKSTATUS_VALIDATED`	At least one wakeupt source is validated.
	`ECUM_WKSTATUS_PENDING`	At least one wakeupt source is pending, none is validated
	`ECUM_WKSTATUS_EXPIRED`	At least one wakeupt source is expired, none is validated or pending
	`ECUM_WKSTATUS_NONE`	No wakeup source is validated, pending or expired
Description	Returns the sum status of the requested wakeup sources	

5.4.3.4.43. EcuM_GetValidatedWakeupEvents

Purpose	Get validated wakeup events.
Synopsis	`EcuM_WakeupSourceType` **`EcuM_GetValidatedWakeupEvents`** `( void );`
Service ID	ECUM_SID_GET_VALID_WK_EV
Sync/Async	Synchronous
Reentrancy	Non-Reentrant Non-Interruptible
Return Value	Wakeup sources which have been validated events.
Description	This function retrieves wakeup events validated using EcuM_ValidateWakeupEvent.

5.4.3.4.44. EcuM_GetVersionInfo

Purpose	Get version information of the EcuM.
Synopsis	`void` **`EcuM_GetVersionInfo`** `( Std_VersionInfoType *const VersionInfoPtr );`
Service ID	ECUM_SID_GET_VERSION_INFO
Sync/Async	Synchronous
Reentrancy	Non Reentrant

Parameters (out)	`VersionInfoPtr`	Pointer to where to store the version information of this module.
Description	This service returns the version information of this module. The version information includes: ▶ Module Id ▶ Vendor Id ▶ Vendor specific version numbers	

5.4.3.4.45. EcuM_GoDown

Purpose	Prepare power off or reset.	
Synopsis	`Std_ReturnType` **`EcuM_GoDown`** `( uint16 caller );`	
Service ID	ECUM_SID_DOWN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`caller`	Module ID of the calling module. Only special modules are allowed to call this function.
Return Value		
Description	This function instruct the ECU state manager module to perform a power off or a reset depending on the selected shutdown target.	

5.4.3.4.46. EcuM_GoHalt

Purpose	Prepare halt.	
Synopsis	`Std_ReturnType` **`EcuM_GoHalt`** `( void );`	
Service ID	ECUM_SID_HALT	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Return Value		
Description	Instructs the ECU State Manager module to go into a sleep mode where the microcontroller is halted, depending on the selected shutdown target.	

5.4.3.4.47. EcuM_GoPoll

Purpose	Prepare Polling sleep mode.	
Synopsis	`Std_ReturnType` **`EcuM_GoPoll`** `( void );`	
Service ID	ECUM_SID_POLL	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Return Value		
Description	Instructs the ECU State Manager module to go into a polling sleep mode depending on the selected shutdown target.	

5.4.3.4.48. EcuM_GoRun

Purpose	Enter RUN state.	
Synopsis	`Std_ReturnType` **`EcuM_GoRun`** `( void );`	
Service ID	ECUM_SID_GO_RUN	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Return Value		
Description	When EcuMFixed support is enabled, EcuM_Mainfunction() will only perform wake-up validation, and will process EcuMFixed specific functionality only after this callback has been called. This callback is supposed to be called by BswM after it has finished the initialization procedure (after both startup or wakeup), including all deferred initialization tasks.	

5.4.3.4.49. EcuM_Init

Purpose	Initialize EcuM Module.
Synopsis	`void` **`EcuM_Init`** `( void );`
Service ID	ECUM_SID_INIT
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Description	This function initializes the ECU state manager and carries out the startup procedure. The function will never return (it calls StartOS).

5.4.3.4.50. EcuM_IsValidConfig

Purpose	Validates the post-build configuration data structure used in the EcuM_Init function.	
Synopsis	`Std_ReturnType` **`EcuM_IsValidConfig`** `( const void * voidConfigPtr );`	
Parameters (in)	`voidConfigPtr`	The pointer to a EcuM post-build data structure.
Parameters (in,out)	`voidConfigPtr`	The pointer to a EcuM post-build data structure.
Return Value		
`E_OK`	When the pre-compile, link-time and platform hash values stored within the post-build structure correspond to the hash values of the compiled source files. Otherwise, E_NOT_OK will be returned.	

5.4.3.4.51. EcuM_KillAllRUNRequests

Purpose	Releases all RUN and POST_RUN requests.
Synopsis	`void` **`EcuM_KillAllRUNRequests`** `( void );`
Service ID	ECUM_SID_KILL_ALL_RUN_REQUESTS
Sync/Async	Synchronous
Reentrancy	Reentrant
Production Errors	▶ ECUM_E_ALL_RUN_REQUESTS_KILLED: thrown, if the service EcuM_KillAllRUNRequests was issued.
Description	No further RUN or POST_RUN requests are accepted until shutdown has completed. The benefit of this function over an ECU reset is that the shutdown sequence is executed, which e.g. takes care of writing back NV memory contents.

5.4.3.4.52. EcuM_MainFunction

Purpose	EcuM main function.
Synopsis	`void` **`EcuM_MainFunction`** `( void );`

Service ID	ECUM_SID_MAIN_FUNCTION
Sync/Async	Synchronous
Reentrancy	Not-Reentrant
Production Errors	▶ ECUM_E_RAM_CHECK_FAILED: thrown, if The RAM check during wakeup failed.
Description	This service implements all activities of the ECU State Manager while the OS is up and running. This function has to be called periodically by a task controlled by the BSW scheduler.

5.4.3.4.53. EcuM_OnCoreSync

Purpose	Determines if a core shall continue waiting to synchronize.
Synopsis	`EcuM_SyncActionType EcuM_OnCoreSync ( void );`
Return Value	ECUM_ABORT_SYNC when the active EcuM instance shall abort synchronization. with the other cores, otherwise ECUM_CONTINUE_SYNC.
Description	This function is repeatedly called while the EcuM on a core is blocked waiting for another core to synchronize. This callout function shall service a watchdog timer when one exists.

5.4.3.4.54. EcuM_OnEnterRun

Purpose	Callout on EnterRun.
Synopsis	`void EcuM_OnEnterRun ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the RUN state is about to be entered. This callout is invoked by EcuM on entry of RUN state.

5.4.3.4.55. EcuM_OnExitPostRun

Purpose	Callout on ExitPostRun.

Synopsis	void **EcuM_OnExitPostRun** (void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the POST_RUN state is about to be left. This callout is invoked by EcuM before leaving POST_RUN state.

5.4.3.4.56. EcuM_OnExitRun

Purpose	Callout on ExitRun.
Synopsis	void **EcuM_OnExitRun** (void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the RUN state is about to be left. This callout is invoked by EcuM before leaving RUN state.

5.4.3.4.57. EcuM_OnGoOffOne

Purpose	Callout on GO OFF 1.
Synopsis	void **EcuM_OnGoOffOne** (void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the GO OFF I state is about to be entered. This callout is invoked by EcuM on entry of GO OFF 1 state.

5.4.3.4.58. EcuM_OnGoOffTwo

Purpose	Callout on GO OFF 2.
Synopsis	void **EcuM_OnGoOffTwo** (void);

Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the GO OFF II state is about to be entered. This callout is invoked by EcuM on entry of GO OFF 2 state.

5.4.3.4.59. EcuM_OnGoSleep

Purpose	Callout on GoSleep.
Synopsis	`void EcuM_OnGoSleep ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the SLEEP state is about to be entered. This callout is invoked by EcuM on entry of SLEEP state.

5.4.3.4.60. EcuM_OnPrepShutdown

Purpose	Callout on PrepShutdown.
Synopsis	`void EcuM_OnPrepShutdown ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the PREP SHUTDOWN state is about to be entered. This callout is invoked by EcuM on entry of PREP SHUTDOWN state.

5.4.3.4.61. EcuM_OnRTEStartup

Purpose	Callout on RTEStartup.
Synopsis	`void EcuM_OnRTEStartup ( void );`
Sync/Async	Synchronous

Reentrancy	Non-Reentrant
Description	This call allows the system designer to notify that the RTE is about to be started. This callout is invoked by EcuM before starting the RTE.

5.4.3.4.62. EcuM_PostHalt

Purpose	Callout during sleep state.
Synopsis	`void EcuM_PostHalt ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	In this callout all the interrupts will be enabled which are disabled by the EcuM_PreHalt(). This will be called after a wakeup event that will return the function Mcu_SetMode() This callout is invoked by EcuM periodically in SLEEP state, but only if the CPU is not suspended (i.e. clock is reduced).

5.4.3.4.63. EcuM_PreHalt

Purpose	Callout during sleep state.
Synopsis	`void EcuM_PreHalt ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	In this callout all the execution of interrupts leading to wakeup will be disabled. This will be called before setting up the ECU in to Halt mode. This prevents the ECU from neglecting the pending wakeup events, which occurred due to the execution of ISR before the call to Mcu_SetMode() This callout is invoked by EcuM periodically in SLEEP state, but only if the CPU is not suspended (i.e. clock is reduced).

5.4.3.4.64. EcuM_SetWakeupEvent

Purpose	Set wakeup events.

Synopsis	`void EcuM_SetWakeupEvent ( EcuM_WakeupSourceType sources );`	
Service ID	ECUM_SID_SET_WK_EV	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant Non-Interruptible	
Parameters (in)	`sources`	Events of wakeup sources to be set.
Description	This function sets the wakeup events for a set of sources and starts the wakeup validation timeout timer. Development Error Detection ▶ ECUM_E_UNKNOWN_WAKEUP_SOURCE: sources contained unknown wakeup sources.	

5.4.3.4.65. EcuM_Shutdown

Purpose	Shutdown ECU.
Synopsis	`void EcuM_Shutdown ( void );`
Service ID	ECUM_SID_SHUTDOWN
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Description	Typically called from the shutdown hook, this function takes over execution control and will carry out GO OFF II activities.

5.4.3.4.66. EcuM_SleepActivity

Purpose	Callout during sleep state.
Synopsis	`void EcuM_SleepActivity ( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	It is explicitly allowed to poll wakeup sources from this callout and to call wakeup notification functions to indicate the end of the sleep state to the ECU State Manager. This callout is invoked by EcuM periodically in SLEEP state, but only if the CPU is not suspended (i.e. clock is reduced).

5.4.3.4.67. EcuM_StartWakeupSources

Purpose	Callout which starts pending wakeup sources.	
Synopsis	`void` **`EcuM_StartWakeupSources`** `( EcuM_WakeupSourceType wakeupSource );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`wakeupSource`	All pending wakeup sources.
Description	In this callout the system integrator has to set up the conditions, so that a wakeup source can validate a wakeup event. For example, set the tranceiver mode of a Can transceiver to STARTED. This callout is invoked in wakeup state.	

5.4.3.4.68. EcuM_StartupTwo

Purpose	Startup sequence two.
Synopsis	`void` **`EcuM_StartupTwo`** `( void );`
Service ID	ECUM_SID_STARTUP_TWO
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Description	This function implements the STARTUP II state.

5.4.3.4.69. EcuM_StopWakeupSources

Purpose	Callout which stops pending and expired wakeup sources.	
Synopsis	`void` **`EcuM_StopWakeupSources`** `( EcuM_WakeupSourceType wakeupSource );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`wakeupSource`	All pending/expired wakeup sources.
Description	In this callout the system integrator has to stop the wakeup sources, previously started in callout function EcuM_StartWakeupSources().	

	This callout is invoked in wakeup validation state after a failed validation of an wakeup event.

5.4.3.4.70. EcuM_ValidateWakeupEvent

Purpose	Validate wakeup events.	

Synopsis	`void` **`EcuM_ValidateWakeupEvent`**
	`( EcuM_WakeupSourceType sources );`	

Service ID	ECUM_SID_VALIDATE_WK_EV	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`sources`	Wakeup sources which events shall be validated.
Description	This function validates wakeup events. After wakeup, the ECU State Manager will stop the process during the WAKEUP VALIDATION state to wait for validation of the wakeup event. The validation is carried out with a call to this API service. Development Error Detection ▶ ECUM_E_UNKNOWN_WAKEUP_SOURCE: sources contained unknown wake-up sources.	

5.4.4. Integration notes

5.4.4.1. Exclusive areas

This section describes the exclusive areas used by the `EcuM` module.

5.4.4.1.1. SCHM_ECUM_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access. Only one exclusive area is defined for a Single-Core configuration.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB`

	tresos AutoCore Generic documentation. Refer to the section Mapping exclusive areas in the basic software modules in the Integration notes section for details.

5.4.4.1.2. SCHM_ECUM_EXCLUSIVE_AREA_{coreID}

Protected data structures	All shared data that shall be protected from mutual access in the corresponding EcuM instance. If Multi-Core support is enabled, one exclusive area is defined for each configured Os core.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the EB tresos AutoCore Generic documentation. Refer to the section Mapping exclusive areas in the basic software modules in the Integration notes section for details.

5.4.4.2. Production errors

ECUM_E_ALL_RUN_REQUESTS_KILLED	▶ EcuM_KillAllRUNRequests
ECUM_E_RAM_CHECK_FAILED	▶ EcuM_MainFunction

5.4.4.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section Memory mapping and compiler abstraction in the Integration notes section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
INTERNAL_VAR_POWER_ON_INIT_UNSPECIFIED
VAR_POWER_ON_INIT_8
VAR_CLEARED_8
VAR_CLEARED_16
CONST_8

VAR_INIT_8
CONST_16
VAR_CLEARED_UNSPECIFIED
VAR_INIT_UNSPECIFIED
CONST_32
CONST_UNSPECIFIED
CONFIG_DATA_UNSPECIFIED
CODE
ECUM_BOOTTARGET

5.4.4.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.4.4.4.1. intgr.EcuM.EB_INTREQ_EcuM_0001

Description	In order to ensure the EcuM_GetMostRecentShutdown() and EcuM_GetNextRecentShutdown() APIs return valid data, the following call chain shall be used in a shutdown or reset sequence: - EcuM_SelectShutdownTarget() - EcuM_SelectShutdownCause() - EcuM_GoDown()
Rationale	EcuM_GetMostRecentShutdown() and EcuM_GetNextRecentShutdown() return the values of the shutdown target, reset mode and shutdown cause. They will not return the expected most recent or next recent shutdown values if they are not previously set using the EcuM_SelectShutdownTarget() and EcuM_SelectShutdownCause() APIs.

5.5. Nm

5.5.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
NmChannelConfig	1..n	Container for the configuration (parameters) of a Nm channel. The channel parameter shall be harmonized within the whole communication stack.
NmGlobalConfig	1..1	This container contains all global configuration parameters of the Nm Interface.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT	
Label	Config Variant	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	VariantPreCompile	
Range	VariantPreCompile	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.5.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1

Parameters included	
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	3
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	12
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID

Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	29
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.5.1.2. NmChannelConfig

Containers included		
Container name	**Multiplicity**	**Description**
NmBusType	1..1	**Label:** Nm Bus Type Defines the buss type of Nm channel. ▶ Select `NmGenericBusNmPrefix` for generic modules which the prefix identifies its Bus <BusNm>

Containers included		
		▶ Select `NmStandardBusNmConfig` for standard AUTOSAR <BusNm>s

Parameters included	
Parameter name	**Multiplicity**
NmActiveCoordinator	0..1
NmChannelId	1..1
NmChannelSleepMaster	1..1
NmCoordClusterIndex	0..1
NmNodeDetectionEnabled	1..1
NmNodeIdEnabled	1..1
NmPassiveModeEnabled	1..1
NmRepeatMsgIndEnabled	1..1
NmShutdownDelayTimer	1..1
NmStateReportEnabled	1..1
NmSynchronizingNetwork	1..1
NmComMChannelRef	1..1
NmStateReportSignalRef	0..1
NmWaitForShutdownTime	0..1

Parameter Name	NmActiveCoordinator
Label	Nm Active Gateway Enable
Description	*The functionality related to this parameter is not supported by the current implementation.* Defines whether a NM Coordinator is an active gateway (NmActiveCoordinator = TRUE) or a passive. Scope: ▶ Channel Dependency on parameter(s): ▶ This feature is available only if the parameter `NmBusSynchronizationEnabled` is set to true.
Multiplicity	0..1

Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmChannelId**	
Label	Nm Channel Id	
Description	Defines channel index value. Implementation Type: NetworkHandleType Range: ▶ 0..254 Dependency on parameter(s): ▶ The Channel id has to be unique ▶ NmChannelId shall be the same as the ComMChannelId of the ComMChannel referenced by NmComMChannelRef	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmChannelSleepMaster**
Label	Nm Channel SleepMaster
Description	Defines if the sleep of this network can be absolutely decided by the local node only and that no other nodes can oppose that decision. If this parameter is set to TRUE, the Nm shall assume that the channel is always ready to go to sleep and that no callouts to Nm_RemoteSleepIndication () or Nm_RemoteSleepCancellation() will be made from the <BusNm> representing this channel. If this parameter is set to FALSE, the Nm shall not assume that the network is ready to sleep until a callout has been made to Nm_RemoteSleepCancellation(). Scope: ▶ Channel Dependency on parameter(s): ▶ If the parameter NmCoordClusterIndex is not defined, this parameter is not valid.

Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmCoordClusterIndex**
Label	Nm Coordination Cluster Index
Description	Defines the posibility that the channel to be part of a coordinator cluster The configuration parameter `NmCoordClusterIndex` is used for specifying to which coordination cluster a channel belongs. If this parameter is undefined for a channel, the corresponding channel does not belong to an NM coordination cluster. The parameter is optional; by enabling it and entering a number "x", the channel is assigned to the Nm coordination cluster "x". Range: ▶ 0..255 Dependency on parameter(s): ▶ There should be at least two channels in a cluster. ▶ Cluster indices must span a dense, zero-based number domain. ▶ Passive mode shall be disabled if channel belongs to a co-ordinated cluster of networks
Multiplicity	0..1
Type	INTEGER
Default value	0
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmNodeDetectionEnabled**
Label	Nm Node Detection
Description	Enables the Node Detection feature. Enables the handling *Repeat Message Request Bit* in the *Control Bit Vector*.

	If the *Request Message Bit* in the NM message set the nodes receiving the message start sending NM messages. For setting the *Repeat Message Request Bit* in NM messages following API function is provided: ▶ `Nm_RepeatMessageRequest()` Dependency on parameter(s): ▶ Support for Node Identifiers must be enabled ▶ Passive Mode must be disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmNodeIdEnabled**	
Label	Nm Node Identifier	
Description	Enables transmission of the source node identifier in NM messages. Enable support for sending and receiving of Node Ids in NM messages and provide functions for retrieving the node identifier from the most recently received NM PDU or the local node identifier. The following API functions are provided if the feature is enabled: ▶ `Nm_GetNodeIdentifier()` ▶ `Nm_GetLocalNodeIdentifier()`	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmPassiveModeEnabled**
Label	Nm Passive Mode
Description	Enables support for 'Passive Mode' of BusNms.

	In passive mode, the BusNm modules will not be able to wake up the bus and will not send NM messages. They will only listen to the NM messages and silently monitor the bus. Dependency on parameter(s): ▶ Passive mode shall not be enabled if the channel belongs to a cordinated cluster of networks ▶ Each channel shall have the same NmPassiveModeEnabled setting as the global NmPassiveModeEnabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmRepeatMsgIndEnabled**
Label	Nm Repeat Messgage Indication Enabled
Description	Enables the Repeat Message Bit Indication.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmShutdownDelayTimer**
Label	Nm Shutdown Delay Timer
Description	Defines the time in seconds which the NM Coordination algorithm shall delay the release of this channel with. This time must be elapsed after starting synchronous shutdown, before an NM channel starts "fall asleep". Range: ▶ 0..65535
Multiplicity	1..1
Type	FLOAT

Default value	0.0
Configuration class	**VariantPreCompile:** | VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmStateReportEnabled**
Label	Nm State Reporting
Description	Enables whether the NMS shall be set for the corresponding network. ▶ False: No NMS shall be set ▶ True: The NMS shall be set Scope: ▶ Module Dependency on parameter(s):: ▶ If State Reporting is enabled parameters 'NmStateChangeIndEnabled' and 'NmComUserDataSupport' shall be enabled. ▶ If State Reporting is enabled a Com Signal Reference (NmStateReportSignalRef) shall be configured.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** | VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmSynchronizingNetwork**
Label	Nm Synchronizing Network
Description	Defines if this network (if set to true) is a synchronizing network for the NM coordination cluster which it belongs to. The network is expected to call Nm_SynchronizationPoint() at regular intervals Scope: ▶ Channel Dependency on parameter(s): ▶ If the parameter NmCoordClusterIndex is not defined, this parameter is not valid.

Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmComMChannelRef**
Label	ComM Channel Reference
Description	Reference to the corresponding network (ComM channel). Dependency on parameter(s): ▶ Bus type of NM channel must match bus type of referenced network channel (ComM). ▶ The Passive Mode of the Nm channel must be disabled if the referenced network channel (ComM) expects the Nm variant 'FULL'.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmStateReportSignalRef**
Label	ComSignal Reference for NMS
Description	Reference to the signal for setting the NMS by calling Com_SendSignal for the respective channel Dependency on parameter(s): ▶ Signal must be configured in COM. Only available if NmStateReportEnabled == true
Multiplicity	0..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmWaitForShutdownTime**
Label	Nm Wait For Shutdown Time

Description	The time is defined in seconds In case parameter NmWaitForShutdownTime is enabled for a channel and the BusNm stays more is Synchronize State then the configurred value (NmWaitForShutdownTime) the channel shall be considered inactive. In case parameter NmWaitForShutdownTime is enabled for a channel and the BusNm is in Buss Sleep State (because of Buss Off) and channel has NmSynchronizingNetwork parameter set to TRUE, the Coordination cluster shall wait for a Synchronization Point the number of mainfunctions. The parameter is optional; It shall be only avaialble if NmCoordClusterIndex is specified for the channel. Range: ▶ 0..65535
Multiplicity	0..1
Type	FLOAT
Default value	0.0
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.5.1.3. NmBusType

Containers included		
Container name	**Multiplicity**	**Description**
NmGenericBusNmConfig	1..1	
NmStandardBusNmConfig	1..1	

5.5.1.4. NmGenericBusNmConfig

Parameters included	
Parameter name	**Multiplicity**
NmGenericBusNmPrefix	1..1

Parameter Name	**NmGenericBusNmPrefix**

Description	Defines the prefix which identifies the generic <BusNm>. This will be used to determine the API name to be called by Nm for the provided interfaces of the <BusNm>. This string will used for the module prefix before the "_-" character in the API call name There should be a header file available for this generic Nm as `NmGenericBusNmPrefix.h` Dependency on parameter(s): - Generic Bus Nm cannot be an empty string. - Generic Nm prefix must start with a letter. It can contain alphabets, numbers, '_-'
Multiplicity	1..1
Type	STRING
Configuration class	**VariantPreCompile:** | VariantPreCompile
Origin	AUTOSAR_ECUC

5.5.1.5. NmStandardBusNmConfig

Parameters included	
Parameter name	**Multiplicity**
NmStandardBusType	1..1

Parameter Name	**NmStandardBusType**
Description	Defines the bus type of the channel for standard AUTOSAR <BusNm>s and is used to determine which set of API calls to be called by Nm for the <BusNm>s. Note: The Ethernet bus' NM is UdpNm ! LinNm is not supported as standard bus type Range: - NM_BUSNM_CANNM - NM_BUSNM_FRNM - NM_BUSNM_LINNM - NM_BUSNM_UDPNM
Multiplicity	1..1
Type	ENUMERATION

Range	NM_BUSNM_CANNM NM_BUSNM_FRNM NM_BUSNM_LINNM NM_BUSNM_UDPNM	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.5.1.6. NmGlobalConfig

Containers included		
Container name	**Multiplicity**	**Description**
NmGlobalConstants	1..1	
NmGlobalFeatures	1..1	
NmGlobalProperties	1..1	

5.5.1.7. NmGlobalConstants

Parameters included	
Parameter name	**Multiplicity**
NmNumberOfChannels	1..1

Parameter Name	**NmNumberOfChannels**
Label	Nm Number Of Channels
Description	Defines the number of NM channels allowed within one ECU In this Nm implementation this number must be equal to the number of entries in the list of NM channels. Range: ▶ 1..255 Dependency on parameter(s): ▶ Number of channels must match to the number of entries in the channel list.
Multiplicity	1..1

Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.5.1.8. NmGlobalFeatures

Parameters included	
Parameter name	**Multiplicity**
NmMultiCoreSupport	1..1
NmBusSynchronizationEnabled	1..1
NmCarWakeUpCallback	0..1
NmCarWakeUpRxEnabled	1..1
NmComControlEnabled	1..1
NmComUserDataSupport	1..1
NmCoordinatorSupportEnabled	1..1
NmCoordinatorSyncSupport	0..1
NmGlobalCoordinatorTime	0..1
NmPassiveModeEnabled	0..1
NmPduRxIndicationEnabled	1..1
NmProvideRemoteSleepCallbacks	1..1
NmRemoteSleepIndicationCallback	1..1
NmRemoteSleepCancellationCallback	1..1
NmRemoteSleepIndEnabled	1..1
NmStateChangeIndEnabled	1..1
NmStateChangeNotificationCallout	0..1
NmUserDataEnabled	1..1

Parameter Name	**NmMultiCoreSupport**
Label	Nm multicore support
Description	Enables MultiCoreSupport.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**NmBusSynchronizationEnabled**	
Label	Nm Bus Synchronization Enabled	
Description	Enables buss synchronization support of the <BusNm>s. This feature is required for NM Coordinator nodes only.It triggers the synchronization of a CanNm bus after a remote sleep indication has been notified. Dependency on parameter(s): ▶ Passive Mode must be disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmCarWakeUpCallback**	
Label	Nm Car WakeUp Callback Func	
Description	Name of the callback function to be called if `Nm_CarWakeUpIndication()` is called. Dependencies: ▶ This parameter is only available if `NmCarWakeUpRxEnabled == TRUE`	
Multiplicity	0..1	
Type	FUNCTION-NAME	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmCarWakeUpRxEnabled**
Label	Nm CarWakeUp Rx Enable
Description	Enables or disables CWU detection. ▶ FALSE - CarWakeUp not supported ▶ TRUE - CarWakeUp supported

Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmComControlEnabled**
Label	Nm Communication Control
Description	Enables the Communication Control support. The following API functions are provided if the feature is enabled: ▶ `Nm_EnableCommunication()` ▶ `Nm_DisableCommunication()` Dependency on parameter(s): ▶ At least one NM channel must have bus type *NM_BUSNM_CANNM* or *NM_BUSNM_FRNM* or *NM_BUSNM_UDPNM*. ▶ Passive Mode must be disabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmComUserDataSupport**
Label	Nm Com User Data Support
Description	Enables setting of NMUserData via SW-C. If NmComUserDataSupport is enabled the API `Nm_SetUserData` shall not be available. Dependency on parameter(s): ▶ If buss type is FrNm, FrNmComUserDataSupport should have the same value as NmComUserDataSupport ▶ If buss type is CanNm, CanNmComUserDataSupport should have the same value as NmComUserDataSupport ▶ If buss type is UdpNm, UdpNmComUserDataSupport should have the same value as NmComUserDataSupport

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmCoordinatorSupportEnabled**	
Label	Enable Nm coordinator	
Description	Enables NM Coordinator support. Dependency: Only valid if NmRemoteSleepIndEnabled set to TRUE and NmPassiveModeEnabled set to FALSE. Also there must exist more than one Nm channel. Dependency on parameter(s): ▶ If `NmCoordinatorSupportEnabled` is enabled, there need to be at least two channels being assigned to the same cluster. ▶ Passive Mode must not be enabled for all channels ▶ NmBusSynchronizationEnabled must be enabled ▶ NmRemoteSleepIndEnabled must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmCoordinatorSyncSupport**
Label	Nm Coordinator Ready To Sleep Indication
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter control activation of interface used by BusNM to notify receptioon of "NM Coordinator Sleep Ready" bit. Pre-processor switch for enabling NM coordinator synchronization support. The following API function is provided if the feature is enabled: ▶ `Nm_CoordReadyToSleepIndication()`

Multiplicity	0..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmGlobalCoordinatorTime**
Label	Nm Global Coordinator Time
Description	*The functionality related to this parameter is not supported by the current implementation.* This parameter defines the maximum shutdown time of a connected and coordinated NM-Cluster Note: This includes nested connections. Dependencies: This feature is available only if `NmBusSynchronizationEnabled` set to 'TRUE' Range: ▶ 0..Infinity
Multiplicity	0..1
Type	FLOAT
Default value	0
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmPassiveModeEnabled**
Label	Nm Passive Mode Enabled
Description	Enables Global switch which determines if all channels have the same passive mode value.
Multiplicity	0..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**NmPduRxIndicationEnabled**
Label	PduR Rx Indication
Description	Enables the PDU Rx Indication Provide API to BusNm modules for indication of a received NM message: ▶ `Nm_PduRxIndication()` If this parameter is enabled, the Nm module will generate a declaration for the function `Nm_PduRxIndication()` as described in the SWS. It is the task of the integrator/user to provide the definition of the function!
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmProvideRemoteSleepCallbacks**
Label	Nm Provide Remote Sleep Callbacks
Description	If this parameter is set to true, user can provide callback functions for Nm_RemoteSleepCancellation and Nm_RemoteSleepIndication
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**NmRemoteSleepIndicationCallback**
Label	Nm Remote Sleep Indication Callback
Description	Callback function name for Nm_RemoteSleepIndication
Multiplicity	1..1
Type	FUNCTION-NAME
Default value	
Configuration class	**PreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**NmRemoteSleepCancellationCallback**
Label	Nm Remote Sleep Cancellation Callback
Description	Callback function name for Nm_RemoteSleepCancellation
Multiplicity	1..1
Type	FUNCTION-NAME
Default value	
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**NmRemoteSleepIndEnabled**
Label	Nm Remote Sleep Indication Enabled
Description	Enables Remote Sleep Indication support. This feature is required for NM Coordinator nodes only. The Remote Sleep Indication allows the BusNm modules to detect a situation where all nodes in the cluster are ready to sleep apart from one node which still keeps the bus awake. The following API functions are provided if the feature is enabled: ▶ `Nm_CheckRemoteSleepIndication()` Dependency on parameter(s): ▶ Passive Mode must be disabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmStateChangeIndEnabled**
Label	Nm State Change Indication
Description	Enables the Network Management state change notification. Provide API to BusNm modules for indication of a state change:

	▶ `Nm_StateChangeNotification()` If this parameter is enabled, the Nm module will provide the callback function `Nm_StateChangeNotification()` as described in the SWS.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**NmStateChangeNotificationCallout**
Label	Provide State Change Notification Callout
Description	Enables the callout function `Nm_StateChangeNotificationCallout()` to be called within the callback function `Nm_StateChangeNotification()`. Dependency on parameter(s): ▶ This parameter is only available if `NmStateChangeIndEnabled == TRUE`
Multiplicity	0..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**NmUserDataEnabled**
Label	Nm User Data
Description	Enables User Data support. The following API functions are provided if the feature is enabled: ▶ `Nm_GetUserData()` ▶ `Nm_SetUserData()` *(Only if Passive Mode is disabled for at least one channel and NmComUserDataSupport is disabled)*
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile

Origin	AUTOSAR_ECUC

5.5.1.9. NmGlobalProperties

Parameters included	
Parameter name	**Multiplicity**
NmCycletimeMainFunction	1..1
NmDevErrorDetect	1..1
NmVersionInfoApi	1..1

Parameter Name	**NmCycletimeMainFunction**	
Label	Main Function Cycle Time [s]	
Description	Defines the period between successive calls to the Main Function of the NM Interface in seconds. Calling the main function is only required if Nm coordinator functionality is enabled.	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.02	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmDevErrorDetect**	
Label	Enable Development Error Detection	
Description	Enables development error detection and notification.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**NmVersionInfoApi**
Label	Enable Version Info API
Description	Enables vesrion information support

	▶ `Nm_GetVersionInfo()`
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

5.5.1.10. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the Nm can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.5.2. Application programming interface (API)

5.5.2.1. Type definitions

5.5.2.1.1. Nm_BusNmFpType

Purpose	Type holding functions pointers to BusNm functions.
Type	struct

Members	`Nm_TS_BusNm_StdFPtrType PassiveStartUp`	Points to the BusNm_PassiveStartup() function of the lower layer modules.
	`Nm_TS_BusNm_GetStateFPtrType GetState`	Points to the BusNm_GetState() function of the lower layer modules.
	`Nm_TS_BusNm_StdFPtrType NetworkRequest`	Points to the BusNm_NetworkRequest() function of the lower layer modules.
	`Nm_TS_BusNm_StdFPtrType NetworkRelease`	Points to the BusNm_NetworkRelease() function of the lower layer modules.
	`Nm_TS_BusNm_GetInfoFPtrType GetUserData`	Points to the BusNm_GetUserData() function of the lower layer modules.
	`Nm_TS_BusNm_SetUserDataFPtrType SetUserData`	Points to the BusNm_SetState() function of the lower layer modules.
	`Nm_TS_BusNm_GetInfoFPtrType GetNodeIdentifier`	Points to the BusNm_GetNodeIdentifier() function of the lower layer modules.
	`Nm_TS_BusNm_GetInfoFPtrType GetLocalNodeIdentifier`	Points to the BusNm_GetLocalNodeIdentifier() function of the lower layer modules.
	`Nm_TS_BusNm_StdFPtrType RepeatMessageRequest`	Points to the BusNm_RepeatMessageRequest() function of the lower layer modules.
	`Nm_TS_BusNm_GetInfoFPtrType GetPduData`	Points to the BusNm_GetPduData() function of the lower layer modules.
	`Nm_TS_BusNm_StdFPtrType RequestBusSynchronization`	Points to the BusNm_RequestBusSynchronization() function of the lower layer modules.
	`Nm_TS_BusNm_ChkRemoteSleepIndFPtrType CheckRemoteSleepIndication`	Points to the BusNm_CheckRemoteSleepIndication() function of the lower layer modules.
	`Nm_TS_BusNm_StdFPtrType DisableCommunication`	Points to the BusNm_DisableCommunication() function of the lower layer modules.
	`Nm_TS_BusNm_StdFPtrType EnableCommunication`	Points to the BusNm_EnableCommunication() function of the lower layer modules.
Description	This type is only used internally in the Nm but must be published in the public module API to enable optimisations.	

5.5.2.1.2. Nm_BusNmType

Purpose	BusNm types.

Type	`uint8`

5.5.2.1.3. Nm_ModeType

Purpose	Operational modes of the network management.
Type	`uint8`

5.5.2.1.4. Nm_StateType

Purpose	States of the network management state machine.
Type	`uint8`

5.5.2.1.5. Nm_TS_BusNm_ChkRemoteSleepIndFPtrType

Purpose	Function Pointer for Nm functions with return type Std_ReturnType and 2 parameters as for BusNm_CheckRemoteSleepIndication() .
Type	`Std_ReturnType(*)(NetworkHandleType nmChannelHandle, boolean *nmRemoteSleepIndPtr)`

5.5.2.1.6. Nm_TS_BusNm_GetInfoFPtrType

Purpose	Function Pointer for Nm functions with return type Std_ReturnType and 2 parameters as for BusNm_GetUserData() .
Type	`Std_ReturnType(*)(NetworkHandleType nmChannelHandle, uint8 *nmPduDataPtr)`

5.5.2.1.7. Nm_TS_BusNm_GetStateFPtrType

Purpose	Function Pointer for Nm functions with return type Std_ReturnType and 3 parameters as for BusNm_GetState() .
Type	`Std_ReturnType(*)(NetworkHandleType nmChannelHandle, Nm_StateType *nmStatePtr, Nm_ModeType *nmModePtr)`

5.5.2.1.8. Nm_TS_BusNm_SetUserDataFPtrType

Purpose	Function Pointer for Nm functions with return type Std_ReturnType and 2 parameters as for BusNm_SetUserData() .
Type	`Std_ReturnType(*)(NetworkHandleType nmChannelHandle, const uint8 *nmPduDataPtr)`

5.5.2.1.9. Nm_TS_BusNm_StdCbkFPtrType

Purpose	Function pointer for Nm callback functions with return type void and parameter of NetworkHandleType.
Type	`void(*)(NetworkHandleType nmChannelHandle)`

5.5.2.1.10. Nm_TS_BusNm_StdFPtrType

Purpose	Function Pointer for standard Nm functions with return type Std_ReturnType and parameter of NetworkHandleType.
Type	`Std_ReturnType(*)(NetworkHandleType nmChannelHandle)`

5.5.2.2. Macro constants

5.5.2.2.1. NM_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.5.2.2.2. NM_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.5.2.2.3. NM_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.

Value	3U

5.5.2.2.4. NM_BUSNM_CANNM

Purpose	CAN NM type.
Value	0U

5.5.2.2.5. NM_BUSNM_FRNM

Purpose	FR NM type.
Value	1U

5.5.2.2.6. NM_BUSNM_GENERICNM

Purpose	Generic NM type.
Value	4U

5.5.2.2.7. NM_BUSNM_LINNM

Purpose	LIN NM type.
Value	2U

5.5.2.2.8. NM_BUSNM_UDPNM

Purpose	UDP NM type.
Value	3U

5.5.2.2.9. NM_BUSNM_UNDEF

Purpose	NM type undefined.
Value	0xFFU

5.5.2.2.10. NM_CC_STATE_PREPARE_SHUTDOWN

Purpose	Timers are initialized for the shutdown.
Value	2U

5.5.2.2.11. NM_CC_STATE_SHUTDOWN_ABORTED

Purpose	A shudown was aborted and the conditions are rechecked in the next main function call.
Value	4U

5.5.2.2.12. NM_CC_STATE_SHUTDOWN_INITIATED

Purpose	The shutdown sequence is currently running.
Value	3U

5.5.2.2.13. NM_CC_STATE_WAIT_FOR_SHUTDOWN

Purpose	Wait for the shutdown conditions to become true. This is checked in all relevant Nm functions syncronously.
Value	0U

5.5.2.2.14. NM_CC_STATE_WAIT_SYNCHRONIZATION

Purpose	Wait for synchronization if a synchronizing network is part of the cluster.
Value	1U

5.5.2.2.15. NM_E_HANDLE_UNDEF

Purpose	Development Error Code.
Value	1U
Description	This error is reported if an API function is called with invalid parameters; e.g. an invalid channel ID

5.5.2.2.16. NM_E_NETWORKRELEASE

Purpose	Error id reported in case SchM call fails for NetworkRelease.
Value	248U

5.5.2.2.17. NM_E_NETWORKREQUEST

Purpose	Error id reported in case SchM call fails for NetworkRequest.
Value	249U

5.5.2.2.18. NM_E_PARAM_POINTER

Purpose	Development Error Code.
Value	2U
Description	This error is reported if an API function is called with NULL pointer; e.g. NULL_PTR

5.5.2.2.19. NM_E_PASSIVESTARTUP

Purpose	Error id reported in case SchM call fails for PassiveStartup.
Value	250U

5.5.2.2.20. NM_E_REQUESTBUSSYNCHRONIZATION

Purpose	Error id reported in case SchM call fails for RequestBusSynchronization.
Value	247U

5.5.2.2.21. NM_E_UNINIT

Purpose	Development Error Code.
Value	0U
Description	This error is reported if an API function is called prior to initialization of the Nm module.

5.5.2.2.22. NM_MODE_BUS_SLEEP

Purpose	Bus-Sleep Mode.
Value	0U

5.5.2.2.23. NM_MODE_NETWORK

Purpose	Network Mode.
Value	3U

5.5.2.2.24. NM_MODE_PREPARE_BUS_SLEEP

Purpose	Prepare-Bus Sleep Mode.
Value	1U

5.5.2.2.25. NM_MODE_SYNCHRONIZE

Purpose	Synchronize Mode.
Value	2U

5.5.2.2.26. NM_MODULE_ID

Purpose	AUTOSAR module identification.
Value	29U

5.5.2.2.27. NM_STATE_BUS_SLEEP

Purpose	
Value	1U

5.5.2.2.28. NM_STATE_NORMAL_OPERATION

Purpose	
Value	4U

5.5.2.2.29. NM_STATE_PREPARE_BUS_SLEEP

Purpose	
Value	2U

5.5.2.2.30. NM_STATE_READY_SLEEP

Purpose	
Value	3U

5.5.2.2.31. NM_STATE_REPEAT_MESSAGE

Purpose	
Value	5U

5.5.2.2.32. NM_STATE_SYNCHRONIZE

Purpose	
Value	6U

5.5.2.2.33. NM_STATE_UNINIT

Purpose	
Value	0U

5.5.2.2.34. NM_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	5U

5.5.2.2.35. NM_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.

Value	12U

5.5.2.2.36. NM_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	5U

5.5.2.2.37. NM_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.5.2.2.38. Nm_RemoteSleepCancelation

Purpose	Work around for typos in the AUTOSAR SWS documents.
Value	Nm_RemoteSleepCancellation(nmNetworkHandle)

5.5.2.3. Objects

5.5.2.3.1. Nm_BusNmFp

Purpose	Array holding function pointers to BusNm modules.
Type	`const Nm_BusNmFpType`
Description	This array is only used internally in the Nm but must be published in the public module API to enable optimisations.

5.5.2.4. Functions

5.5.2.4.1. Nm_BusSleepMode

Purpose	Notifies that Bus Sleep Mode has been entered.
Synopsis	`void Nm_BusSleepMode ( NetworkHandleType nmNetworkHandle );`

Service ID	32	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel.
Description	This function provides a notification that the network management has entered Bus-Sleep Mode. Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.2. Nm_CheckRemoteSleepIndication

Purpose	Implementation of Nm_CheckRemoteSleepIndication.	
Synopsis	`Std_ReturnType` **`Nm_CheckRemoteSleepIndication`** `( NetworkHandleType nmNetworkHandle , boolean * nmRemoteSleepIndPtr );`	
Service ID	13	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel.
Parameters (out)	`nmRemoteSleepIndPtr`	Pointer where check result of remote sleep indication shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No error.
	`E_NOT_OK`	Checking of remote sleep indication bits has failed. NetworkHandle does not exist (development only). Module not yet initialized (development only).
Description	This function checks if remote sleep indication takes place or not. This in turn calls the BusNm_CheckRemoteSleepIndication() for the bus specific NM layer (e.g. CanNm_CheckRemoteSleepIndication() function is called if channel is configured as CAN). Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.3. Nm_CoordReadyToSleepIndication

Purpose	Indicate Ready to Sleep.	
Synopsis	`void` **`Nm_CoordReadyToSleepIndication`** `( NetworkHandleType nmChannelHandle );`	
Parameters (in)	`nmChannelHandle`	Identification of the NM-channel
Description	Sets an indication, when the NM Coordinator Sleep Ready bit in the Control Bit Vector is set The functionality needs to be implemented by the customer, as it is not specified by AUTOSAR.	

5.5.2.4.4. Nm_DisableCommunication

Purpose	Disable NM-PDU transmissions.	
Synopsis	`Std_ReturnType` **`Nm_DisableCommunication`** `( NetworkHandleType NetworkHandle );`	
Service ID	4	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Disabling of NM PDU transmission ability has failed. NetworkHandle does not exist (development only). The module is not yet initialized (development only).
Description	This function calls the respective function at CanNm module, to disable the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service. This functionality is not provided by FrNm module. Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.5. Nm_EnableCommunication

Purpose	Enable NM-PDU transmissions.	
Synopsis	`Std_ReturnType` **`Nm_EnableCommunication`** `( NetworkHandleType NetworkHandle );`	
Service ID	5	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Enabling of NM PDU transmission ability has failed. NetworkHandle does not exist (development only). The module is not yet initialized (development only).
Description	This function calls the respective function at CanNm module to enable the NM PDU transmission ability due to a ISO14229 Communication Control (28hex) service. This functionality is not provided by FrNm module. Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.6. Nm_GetCoordinatorState

Purpose	Get the State of the Nm Coordinator.	
Synopsis	`Std_ReturnType` **`Nm_GetCoordinatorState`** `( uint8 nmCoordinatorId , uint8 * nmStatePtr );`	
Parameters (in)	`nmCoordinatorId`	Identification of the coordinator.
	`nmStatePtr`	Pointer where state of the Coordinator shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of NM state has failed.
Description	This function returns the state of the Nm coordinator.	

	Preconditions: ▶ The coordinator id should be valid and the module should have been initialized

5.5.2.4.7. Nm_GetLocalNodeIdentifier

Purpose	Configure the get node identifier for the local node.	
Synopsis	Std_ReturnType **Nm_GetLocalNodeIdentifier** (NetworkHandleType NetworkHandle , uint8 * nmNodeIdPtr);	
Service ID	11	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise.	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
Parameters (out)	`nmNodeIdPtr`	Pointer where node identifier of the local node shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No error.
	`E_NOT_OK`	Getting of the node identifier of the local node has failed. NetworkHandle does not exist (development only). Module not yet initialized (development only).
Description	The function configures the Get node identifier for the local node. For that purpose BusNm_GetLocalNodeIdentifier shall be called (e.g. CanNm_GetLocalNodeIdentifier function is called if channel is configured as CAN). Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.8. Nm_GetNodeIdentifier

Purpose	Get the Node Identifier.
Synopsis	Std_ReturnType **Nm_GetNodeIdentifier** (NetworkHandleType NetworkHandle , uint8 * nmNodeIdPtr);

Service ID	10	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
Parameters (out)	`nmNodeIdPtr`	Pointer where node identifier out of the last successfully received NM-message shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of the node identifier out of the last received NM-message has failed. NetworkHandle does not exist (development only). Module not yet initialized (development only).
Description	This function gets the node identifier out of the last successfully received NM-message. Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.9. Nm_GetPduData

Purpose	Get Pdu Data.	
Synopsis	`Std_ReturnType` **`Nm_GetPduData`** `( NetworkHandleType NetworkHandle , uint8 * nmPduData );`	
Service ID	8	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
	`nmPduData`	Pointer where NM PDU shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of NM PDU data has failed. NetworkHandle does not exist (development

		only) module not yet initialized (development only).
Description	This function gets the whole PDU data out of the most recently received NM message. Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.10. Nm_GetState

Purpose	Get the State of the network management module.	
Synopsis	`Std_ReturnType` **`Nm_GetState`** `( NetworkHandleType nmNetworkHandle , Nm_StateType * nmStatePtr , Nm_ModeType * nmModePtr );`	
Service ID	14	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel.
	`nmStatePtr`	Pointer where state of the network management shall be copied to.
	`nmModePtr`	Pointer to the location where the mode of the network management shall be copied to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of NM state has failed. NetworkHandle does not exist (development only). Module not yet initialized (development only).
Description	This function returns the state of the network management. This function in turn calls the BusNm_GetState function (e.g. CanNm_GetState function is called if channel is configured as CAN). Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.11. Nm_GetUserData

Purpose	Get user data out of the last received NM message.	
Synopsis	`Std_ReturnType` **`Nm_GetUserData`** `( NetworkHandleType NetworkHandle , uint8 * nmUserDataPtr );`	
Service ID	7	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Channel Identification of the NM-channel.
Parameters (out)	`nmUserDataPtr`	Pointer to a memory area where the extracted data shall be written to.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Getting of user data has failed. NetworkHandle does not exist (development only). Module not yet initialized (development only).
Description	This function gets user data out of the last successfully received NM message. For that purpose BusNm_GetUserData shall be called (e.g. CanNm_GetUserData function is called if channel is configured as CAN). Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.12. Nm_Init

Purpose	Initializes the NM Interface.
Synopsis	`void` **`Nm_Init`** `( void );`
Service ID	0
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function initializes the Nm Interface (Nm).

5.5.2.4.13. Nm_MainFunction

Purpose	This function is supplied for the NM coordinator functionality.	
Synopsis	`void Nm_MainFunction ( void );`	
Service ID	16	
Description	This function implements the processes of the NM Interface, which need a fix cyclic scheduling. This function is supplied for the NM coordinator functionality (Nm020). However, specific implementation may not need it (Nm093). Preconditions: ▶ NM Interface must be initialized before	

5.5.2.4.14. Nm_NetworkMode

Purpose	Notifies that the network management has entered Network Mode.	
Synopsis	`void Nm_NetworkMode ( NetworkHandleType nmNetworkHandle );`	
Service ID	30	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel.
Description	This function gives a notification that the network management has entered Network Mode. The callback function enables transmission of application messages. Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.15. Nm_NetworkRelease

Purpose	Releases the Bus communication.
Synopsis	`Std_ReturnType Nm_NetworkRelease ( NetworkHandleType NetworkHandle );`
Service ID	3
Sync/Async	Synchronous
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise

Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Releasing of bus communication has failed. NetworkHandle does not exist (development only). The module is not yet initialized (development only).
Description	This function calls the BusNm_NetworkRelease() bus specific function (e.g. CanNm_NetworkRelease() function is called if channel is configured as CAN). Preconditions: - The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.16. Nm_NetworkRequest

Purpose	Requests bus communication.	
Synopsis	`Std_ReturnType **Nm_NetworkRequest** ( NetworkHandleType NetworkHandle );`	
Service ID	2	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	NM channel for which the network shall be requested.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Requesting of bus communication has failed. NetworkHandle does not exist (development only). The module is not yet initialized (development only).
Description	This function calls the function BusNm_NetworkRequest() (e.g. CanNm_NetworkRequest() function is called if channel is configured as CAN). Preconditions: - The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.17. Nm_NetworkStartIndication

Purpose	Notifies that Network Mode has been entered.	
Synopsis	`void` **`Nm_NetworkStartIndication`** `( NetworkHandleType nmNetworkHandle );`	
Service ID	33	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel.
Description	This function provides a notification that a NM-message has been received in the Bus-Sleep Mode, which indicates that some nodes in the network have already entered the Network Mode. The callback function starts the network management state machine. Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.18. Nm_PassiveStartUp

Purpose	Passive start of Network Management.	
Synopsis	`Std_ReturnType` **`Nm_PassiveStartUp`** `( NetworkHandleType NetworkHandle );`	
Service ID	1	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Passive start of network management has failed. NetworkHandle does not exist (development only) module not yet initialized development only).

Description	This function calls the BusNm_PassiveStartUp function (e.g. CanNm_PassiveStartUp function is called if channel is configured as CAN). Preconditions: - The channel handle should be valid and the module should have been initialized for this channel (checked).

5.5.2.4.19. Nm_PrepareBusSleepMode

Purpose	Notification that the network management has entered Prepare Bus-Sleep Mode.	
Synopsis	`void Nm_PrepareBusSleepMode ( NetworkHandleType nmNetworkHandle );`	
Service ID	31	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel.
Description	This function provides a notification that the network management has entered Prepare Bus-Sleep Mode. The callback function disables transmission of application messages. Preconditions: - The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.20. Nm_RemoteSleepCancellation

Purpose	Notification that not all other nodes are ready to sleep.	
Synopsis	`void Nm_RemoteSleepCancellation ( NetworkHandleType nmNetworkHandle );`	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel
Description	Notification that the network management has detected that not all other nodes on the network are longer ready to enter Bus-Sleep Mode. If the Nm coordinator support is disabled this callback function is implemented as function-like macro.	

	This function is not specified in AUTOSAR R3.x Nm SWS but in the R4.0 draft SWS.

5.5.2.4.21. Nm_RemoteSleepIndication

Purpose	Notification that all other nodes are ready to sleep.	
Synopsis	`void` **`Nm_RemoteSleepIndication`** `( NetworkHandleType nmNetworkHandle );`	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel
Description	This function provides a notification that the network management has detected that all other nodes are ready to sleep. The NM gateway checks if the Bus is still required. If the Nm coordinator support is disabled this API function is implemented as empty function-like macro, as it provides no functionality.	

5.5.2.4.22. Nm_RepeatMessageRequest

Purpose	Set Repeat Message Request bit.	
Synopsis	`Std_ReturnType` **`Nm_RepeatMessageRequest`** `( NetworkHandleType NetworkHandle );`	
Service ID	9	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
Return Value	Standard Return Code	
	`E_OK`	No error.
	`E_NOT_OK`	Setting of Repeat Message Request Bit has failed. NetworkHandle does not exist (development only). Module not yet initialized (development only).
Description	This function sets the Repeat Message Request Bit for NM messages transmitted next on the bus. For that purpose BusNm_RepeatMessageRequest() shall be called (e.g. CanNm_RepeatMessageRequest() function is called if channel is configured as CAN) Preconditions:	

	▶ The channel handle should be valid and the module should have been initialized for this channel (checked).

5.5.2.4.23. Nm_SetUserData

Purpose	Set user data.	
Synopsis	`Std_ReturnType` **`Nm_SetUserData`** `( NetworkHandleType NetworkHandle , const uint8 * nmUserDataPtr );`	
Service ID	6	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant for the same NetworkHandle. Reentrant otherwise	
Parameters (in)	`NetworkHandle`	Identification of the NM-channel.
	`nmUserDataPtr`	User data for the next transmitted NM message.
Return Value	Standard Return Code	
	`E_OK`	No Error.
	`E_NOT_OK`	Setting of user data has failed. NetworkHandle does not exist (development only). The module is not yet initialized (development only).
Description	This function sets the user data for NM messages transmitted next on the bus. For that purpose BusNm_SetUserData shall be called (e.g. CanNm_SetUserData function is called if channel is configured as CAN). Preconditions: ▶ The channel handle should be valid and the module should have been initialized for this channel (checked).	

5.5.2.4.24. Nm_SynchronizationPoint

Purpose	Notification that this is a suitable point in time to initiate the coordination algorithm on.	
Synopsis	`void` **`Nm_SynchronizationPoint`** `( NetworkHandleType nmNetworkHandle );`	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel

Description	Notification NM Coordinator functionality that this is a suitable point in time to initiate the coordination algorithm on If the Nm coordinator support is disabled this callback function is implemented as function-like macro. This function is not specified in AUTOSAR R3.x Nm SWS but in the R4.0 draft SWS.

5.5.2.4.25. Nm_TxTimeoutException

Purpose	Notification service to indicate that an attempt to send a NM message failed.	
Synopsis	`void` **`Nm_TxTimeoutException`** `( NetworkHandleType nmNetworkHandle );`	
Parameters (in)	`nmNetworkHandle`	Identification of the NM-channel
Description	This function provides an indication that an attempt to send an NM message failed.	

5.5.3. Integration notes

5.5.3.1. Exclusive areas

This section describes the exclusive areas used by the `Nm` module.

5.5.3.1.1. SCHM_NM_EXCLUSIVE_AREA_0

Protected data structures	All shared data that shall be protected from mutual access.
Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Refer to the section `Mapping exclusive areas in the basic software modules` in the `Integration notes` section for details.

5.5.3.2. Production errors

Production errors are not reported by the `Nm` module.

5.5.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
CONST_UNSPECIFIED
CONST_8
VAR_CLEARED_UNSPECIFIED
VAR_CLEARED_16
VAR_INIT_8
VAR_CLEARED_8

5.5.3.4. Integration requirements

WARNING ⚠

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the Nm module.