Document Title	Specification of Basic Software Mode Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	313
Document Classification	Standard

Document Version	3.1.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
19.02.2014	3.1.0	AUTOSAR Release Management	• Items erroneously backported from 4.x corrected (BswM & Fee) • Updates to BswMDevErrorDetect, BswMSwcMode, BswMBswMode • Document formatting & consistency restored • Editorial changes
21.06.2012	3.0.0	AUTOSAR Administration	• Clarification of immediate action lists • Revised hierarchy of nested rules
29.03.2011	2.0.0	AUTOSAR Administration	• Backporting of the BswM from R4 to R3.2 • Partial Networking: Bswm handles activation of I-PDU groups and DM for PNCs • Described interaction between BswM and DCM for ECU reset and Communication control • Mode dependant routing of CAN/FR messages to LIN busses • Adapted configuration data according to 3.2.1 needs. • Legal disclaimer revised
22.10.2010	1.1.0	AUTOSAR Administration	• Include file BswMUserCallout.h added. This user defined header file contains declarations of the call out functions. • Requirement that the BswM module shall perform inter module version checks added • Information added for each configurable action which API to call • Functions

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
			BswM_TriggerSlaveRTEStop and BswM_TriggerStartUpPhase2 added to control the start and stop of the RTE on slave cores
30.11.2009	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

This specification and the material contained in it, as released by AUTOSAR is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only.
For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module BSW Mode Manager (BswM).

In R3.2 the BswM (BSW Mode Manager) is a module that implements a number of fixed use cases. Its responsibility is to arbitrate mode requests from application layer SW-Cs or other BSW modules based on simple rules, and perform actions based on the arbitration result.

2 Acronyms and abbreviations

Abbreviation / Acronym:	***Description:***
AMM	Application Mode Management
BSW	Basic Software
BswM	BSW Mode Manager
Det	Development Error Tracer
ECU	Electric Control Unit
RTE	Real Time Environment
PNC	Partial Network Cluster

Table 2-1: Table of acronyms and abbreviations

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[4] Requirements on Mode Management
AUTOSAR_SRS_ModeManagement.pdf

[5] Specification of Communication
AUTOSAR_SWS_COM.pdf

[6] Specification of FlexRay State Manager
AUTOSAR_SWS_FlexRayStateManager.pdf

[7] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[8] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[9] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[10] Specification of RTE Software
AUTOSAR_SWS_RTE.pdf

[11] Specification of LIN State Manager
AUTOSAR_SWS_LINStateManager.pdf

[12] Specification of CAN State Manager
AUTOSAR_SWS_CANStateManager.pdf

[13] Specification of Generic Network Management Interface
AUTOSAR_SWS_NetworkManagementInterface.pdf

[14] Specification of Communication Manager
AUTOSAR_SWS_COMManager.pdf

3.2 Related standards and norms

None.

4 Constraints and assumptions

4.1 Limitations

Maximum one instance of the BSW Mode Manager is used within an AUTOSAR ECU.

4.2 Applicability to car domains

The BSW Mode Manager is applicable to all car domains.

- AUTOSAR confidential -

5 Dependencies to other modules

The BSW Mode Manager has interfaces to many of the BSW Modules in the AUTOSAR architecture. The majority of these interfaces are however optional and are used based on the needs of each ECU.

5.1 RTE

The BswM connects via port interfaces to SW-Cs. Mode request are received from SW-Cs.

5.2 ComM

The ComM informs the BswM about state changes of communication channels and partial network clusters.

5.3 COM

The handling of I-PDU Groups in COM is performed by the BswM. As a part of I-PDU group start/stop it is possible to have the included signal values reset to their corresponding initialization values. BswM handles enabling and disabling of deadline monitoring of COM signals.

5.4 PduR

The BswM can enable and disable routing groups of I-PDUs in the PDU router.

5.5 CanSM

The CanSM informs the BswM about state changes of the corresponding CAN channel.

5.6 LinSM

BswM coordinates switching of LIN Schedule Tables in the LinSM with starting and stopping of the corresponding I-PDU groups in COM.

5.7 LinIF

The LIN Transport Protocol that is a part of LinIF requests modes from BswM to make sure that the correct LIN Schedule Table is active during LinTp operation.

- AUTOSAR confidential -

5.8 FrSM

The FrSM informs the BswM about mode changes of the corresponding FR channel.

5.9 DCM

The DCM performs mode requests to the BswM based on the diagnostic requests it receives. The BswM is involved solely during processing of the diagnostic requests “Communication Control” and “Reset Request”.

5.10 NM Interface

BswM will use the Nm_EnableCommunication and Nm_DisableCommunication to control the NM communication. This is required for handling of the diagnostic request “Communication Control”.

5.11 File structure

Figure 1: File structure of BSW Mode ManagerFigure 1 shows the file structure of the BSW Mode Manager and the header files of other modules it needs to include.

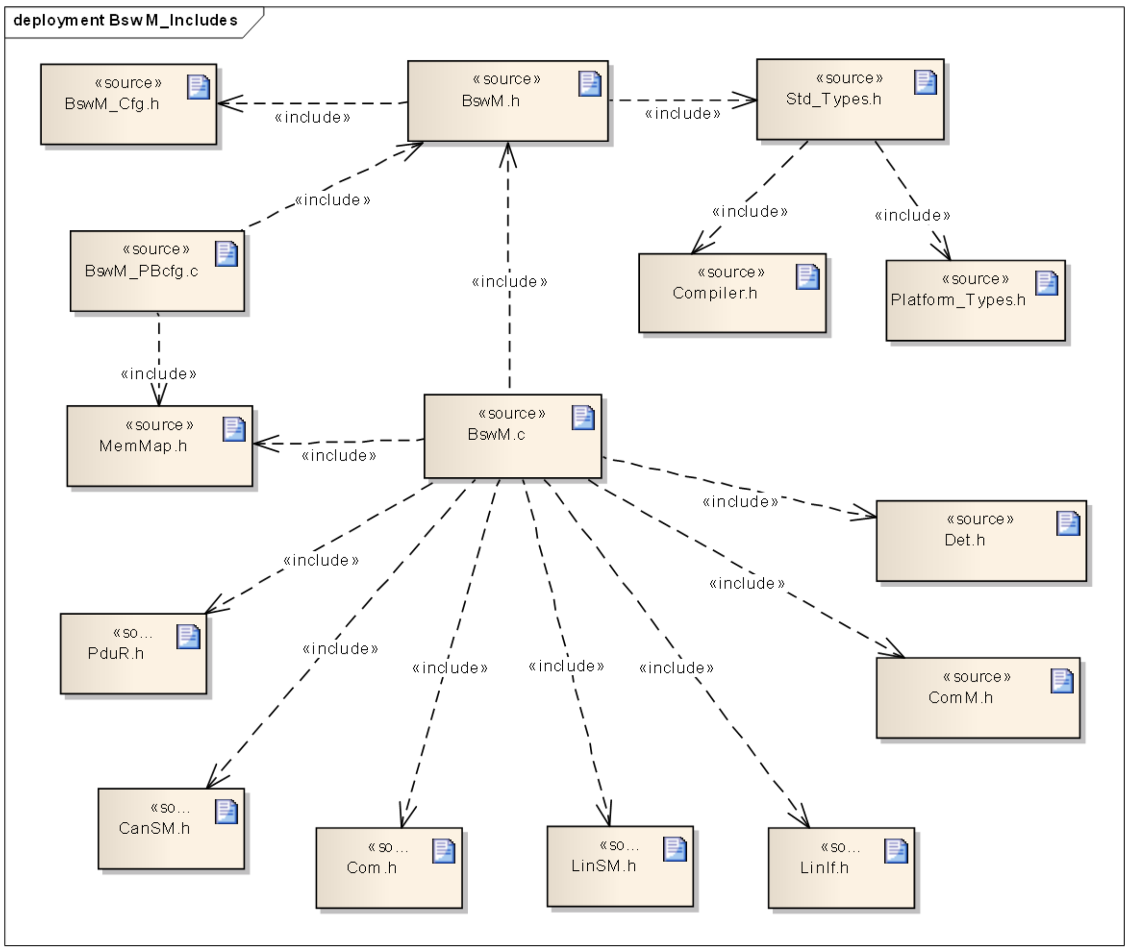


Figure 1: File structure of BSW Mode Manager

The BswM may use interfaces in AUTOSAR BSW modules that are not explicitly defined within this specification.

5.11.1 Code file structure

BswM0024:
The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- BswM_Lcfg.c – for link time configurable parameters and
- BswM_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.

5.11.2 Header file structure

BswM0025:
The BswM shall include the header files of all other BSW modules which API functions it uses.
Specifically it shall include StdTypes.h and ComStack_Types.h to avoid redefinition of types.

BswM0026:
The BswM module shall provide the following set of header files for other BSW modules to include:

1. BswM header file:
   a. BswM.h
   b. BswM_CanSM.h
   c. BswM_LinSM.h
   d. BswM_FrSM.h
   e. BswM_RTE.h
   f. BswM_COM.h
   g. BswM_ComM.h
   h. BswM_DCM.h
2. BswM configuration file:
   a. BswM_Cfg.h

6 Requirements traceability

Document: AUTOSAR requirements on Basic Software, general [3].

Requirement	***Satisfied by***
[BSW00344] Reference to link--time configuration	BswM0021
[BSW00404] Reference to post build time configuration	Chapter 5
[BSW00405] Reference to multiple configuration sets	Not applicable
[BSW00345] Configuration at Compile time	BswM0020
[BSW159] Automatic configuration	Chapter 10
[BSW167] Static configuration checking	Chapter 10
[BSW171] Configurability of optional functionality	Chapter 10
[BSW170] Data for reconfiguration of AUTOSAR SW--Components	Not applicable
[BSW00380] Separate C--Files for configuration parameters	BswM0024
[BSW00419] Separate C--Files for pre--compile time configuration parameters	BswM0024
[BSW00381] Separate configuration header file for pre--compile time parameters	BswM0026
[BSW00412] Separate H--File for configuration parameters	BswM0026
[BSW00383] List dependencies of configuration files	Chapter 5
[BSW00384] List dependencies to other modules	Chapter 5
[BSW00387] Specify the configuration class of callback function	Not applicable
[BSW00388] Introduce containers	Chapter 10
[BSW00389] Containers shall have names	Chapter 10
[BSW00390] Parameter content shall be unique within the module	Chapter 10
[BSW00391] Parameter shall have unique names	Chapter 10
[BSW00392] Parameters shall have a type	Chapter 10
[BSW00393] Parameters shall have a range	Chapter 10
[BSW00394] Specify the scope of the parameters	Chapter 10
[BSW00395] List the required parameters (per parameter)	Chapter 10

[BSW00396] Configuration classes	Chapter 10
[BSW00397] Pre--compile--time parameters	Chapter 10
[BSW00398] Link--time parameters	Chapter 10
[BSW00399] Loadable Post--build time parameters	Not applicable
[BSW00400] Selectable Post--build time parameters	Not applicable
[BSW00438] Post Build Configuration Data Structure	Chapter 10
[BSW00402] Published information	Chapter 10
[BSW101] Initialization interface	BswM0002
[BSW00406] Check module initialization	BswM0078, BswM0079, BswM0080, BswM0082, BswM0083, BswM0086
[BSW00407] Function to read out published parameters	BswM0003
[BSW00423] Usage of SW--C template to describe BSW modules with AUTOSAR Interfaces	Chapter 7.7
[BSW00336] Shutdown interface	Not Applicable
[BSW00337] Classification of errors	Chapter 7.3
[BSW00338] Detection and Reporting of development errors	Chapter 7.4
[BSW00369] Do not return development error codes via API	Chapter 8
[BSW00339] Reporting of production relevant errors and exceptions	Not Applicable
[BSW00323] API parameter checking	Chapter 8
[BSW00409] Header files for production code error IDs	Not Applicable
[BSW00385] List possible error notifications	Chapter 7.3
[BSW00386] Configuration for detecting an error	Chapter 7.4
[BSW00415] User dependent include files	BswM0026
[BSW00343] Specification and configuration of time	Chapter 10
[BSW00346] Basic set of module files	Chapter 5
[BSW158] Separation of configuration from implementation	Chapter 5
[BSW00370] Separation of callback interface from API	Chapter 8
[BSW00357] Standard API return type	Chapter 8
[BSW00377] Module specific API return types	Chapter 8
[BSW00371] Do not pass function pointers via API	Chapter 8
[BSW00358] Return type of init() functions	Chapter 8

- AUTOSAR confidential -

[BSW00414] Parameter of init function	Chapter 8
[BSW00376] Return type and parameters of main processing functions	Chapter 8
[BSW00359] Return type of callback functions	Chapter 8
[BSW00360] Parameters of callback functions	Chapter 8
[BSW00440] Function prototype for callback functions of AUTOSAR Services	Chapter 7.7
[BSW00374] Module vendor identification	Chapter 10
[BSW00379] Module identification	Chapter 10
[BSW003] Version identification	Chapter 10
[BSW00318] Format of module version numbers	Chapter 10
[BSW00321] Enumeration of module version numbers	Chapter 10

Document: AUTOSAR Requirements on Mode Management [4].

Requirement	***Satisfied by***
[BSW09178] Support of Lists of Mode Dependant Actions	BswM0016, BswM0015
[BSW09175] Support of a configurable set of mode dependent enabled and concomitant disabled IPDU groups	BswM0038
[BSW09176] Support of a configurable set of mode dependent to be enabled IPDU groups	BswM0038
[BSW09183] Support of Mode dependent to be reset Signal Initial Values	BswM0038
[BSW09174] Support of “Disable normal Communication”	BswM0038
[BSW09180] Arbitration of Mode Requests	BswM0009, BswM0013, BswM0014
[BSW09182] Local propagation of mode change information	BswM0038
[BSW09184] Mode dependent activation and deactivation of IPDU groups	BswM0038
[BSW09228] Provision of an Interface to allow Mode Requests of BSW Modules	BswM0046, BswM0047, BswM0048, BswM0049, BswM0050, BswM0051, BswM0052
[BSW09229] Mode dependent callout	BswM0039, BswM0040
[BSW09230] All actions shall only be performed on mode change	BswM0011, BswM0023

7 Functional specification

This chapter specifies the functional behavior of the BSW Mode Manager. The operation of the BSW Mode Manager basic functionality can be described as two different tasks – Arbitration of rules and execution of actions according to the rule results.

In difference to R4.0 the BswM of R3.2 implements fixed use cases. Thus, it is not possible to freely configure any additional rules and corresponding actions. Many features of the R4.0 version are not required and therefore not part of the R3.2 BswM.

BswM0301
The BswM handles the following use cases:
- Diagnostic Reset Request handling
- Diagnostic Communication Control
- Partial Networking
- Switching of I-PDU routing group paths
- Switching of LIN schedule tables

There may be different ways of implementing this, such as generation of the complete BswM based on the configuration. However, this specification does not intend to specify any implementation details of the BSW Mode Manager. Hence, any examples stated in this document describing design details should be treated as explanatory text and not as requirements unless otherwise stated.

7.1 Use Cases

The following chapters describe the use cases that shall be supported by the R3.2 BswM.

7.1.1 DCM – Communication Control

The diagnostic service “Communication Control” involves switching of I-PDU groups and enabling or disabling the transmission of NM messages. Once the Dcm received a “Communication Control” request it calls the BswM to perform the I-PDU group switching and control of NM messages transmission.

BswM0315
If the Dcm calls the API BswM_Dcm_RequestCommunicationMode the BswM shall perform the necessary actions to fulfill the request. The actions to be executed are given by the parameters of the BswM_Dcm_RequestCommunicationMode function call.

Hint: All AUTOSAR BSW modules that may trigger transmission of PDUs provide an API to enable/disable it. To e.g. disable the whole communication in a corresponding diagnostic request, it makes sense if CDD modules (which use communication protocols) provides such an API as well. These functions may be called in the configured action list which is linked to this function. Since the R3.2 BswM implements fixed use cases calling these CDD functions must be part of the integrator code as described in BswM0343.

BswM0331
Table 7-1 describes the possible parameter values passed by the DCM to the BswM for communication control and describes the necessary actions.

Value	Action
DCM_ENABLE_RX_TX_NORM	Enable the Rx and Tx for normal communication
DCM_ENABLE_RX_DISABLE_TX_NORM	Enable the Rx and disable the Tx for normal communication
DCM_DISABLE_RX_ENABLE_TX_NORM	Disable the Rx and enable the Tx for normal communication
DCM_DISABLE_RX_TX_NORMAL	Disable Rx and Tx for normal communication
DCM_ENABLE_RX_TX_NM	Enable the Rx and Tx for network management communication
DCM_ENABLE_RX_DISABLE_TX_NM	Enable Rx and disable the Tx for network management communication
DCM_DISABLE_RX_ENABLE_TX_NM	Disable the Rx and enable the Tx for network management communication
DCM_DISABLE_RX_TX_NM	Disable Rx and Tx for network management communication
DCM_ENABLE_RX_TX_NORM_NM	Enable Rx and Tx for normal and network management communication
DCM_ENABLE_RX_DISABLE_TX_NORM_NM	Enable the Rx and disable the Tx for normal and network management communication
DCM_DISABLE_RX_ENABLE_TX_NORM_NM	Disable the Rx and enable the Tx for normal and network management communication
DCM_DISABLE_RX_TX_NORM_NM	Disable Rx and Tx for normal and network management communication

Table 7-1 Actions requested for communication control

BswM0316
When called by the DCM the BswM does no further permission checks and must not reject request.

BswM0321
If a communication control request is received to disable normal message transmission the BswM shall disable all I-PDU groups for the addressed communication channel and enable a specific I-PDU group which contains the I-PDUs allowed to be transmitted during normal communication OFF. The configuration shall reference this I-PDU group. Normal communication is controlled by switching I-PDU groups calling the COM API functions Com_IpduGroupStart and Com_IpduGroupStop.

BswM0323
If an I-PDU group needs to be enabled while normal message transmission is disabled because of a diagnostic communication control request the BswM shall not

enable the requested I-PDU group but store this I-PDU group request. The I-PDU group shall be activated when normal message transmission enabled again by means of a call to Com_IpduGroupStart – see also BswM0325.

BswM0326
If disabling of an I-PDU group is requested while normal message transmission is OFF the BswM shall set the internal state of that I-PDU group to disabled but shall not call Com_IpduGroupStop.

BswM0325
If normal message transmission is enabled again the BswM shall enable all I-PDU groups which are stored internally as active. The single I-PDU group that was enabled during disabled normal message transmission shall be disabled.

BswM0322
If requested by the communication control the BswM shall disable NM message transmission by calling the API Nm_DisableCommunication.

BswM0327
If requested by the communication control the BswM shall enable NM message transmission by calling the API Nm_EnableCommunication.

BswM0343
The BswM shall support calling an additional callout function to let the integrator implement ECU specific code which must be executed during communication control.

7.1.2 DCM – ECU Reset

If a reset request is received by the DCM the BswM is called in order to perform the actual reset action. The DCM requests the execution of a reset by calling the API function BswM_Dcm_RequestResetMode.

BswM0314
If the API BswM_Dcm_RequestResetMode is called the BswM shall call a callout function which contains the ECU specific code to perform the requested reset action. The callout must be implemented during integration. SWC for example could be killed by calling KillAllRunRequest – but the actual handling is completely up to the integrator.

BswM0313
The BswM shall not check if the requested reset action is permitted at all.

BswM0359
The application may need to be informed if it was updated by an FLASH download. In this case the DCM notifies the BswM by means of the interface BswM_Dcm_ApplicationUpdated after the ECU is restarting after the reset at the end of the FLASH download procedure. The BswM notifies the SW-C by means of a mode port in case this API function is called by the DCM.

7.1.3 Coordination and Switching of I-PDU Groups

BswM0309
The BswM shall coordinate and switch I-PDU groups upon request of the BSW modules ComM (BswM_ComM_CurrentMode, BswM_ComM_CurrentPNCMode) , CanSM (BswM_CanSM_CurrentState), FrSM(BswM_FrSM_CurrentState) and DCM (BswM_Dcm_RequestCommunicationMode).

BswM0324
The BswM shall track the states of all I-PDU groups internally in order to provide an efficient way of I-PDU group control.

BswM0310
The BswM shall coordinate I-PDU group switching according to a priority of the given I-PDU group requests. See also BswM0328.

BswM0328
The BswM shall handle I-PDU group requests from PNCs with the least priority. If normal message transmission is currently OFF, the PNC request is not serviced. This means the corresponding I-PDU groups are not enabled. However, the BswM shall remember these I-PDU groups internally as enabled. Once Communcation control is enabled again theses I-PDU groups are enabled too according to BswM0325.

BswM0329
The BswM shall enable the I-PDU groups requested by communication control just in case the corresponding State Manager (CAN or FR) is in FULL_COMMUNICATION state.

BswM0311
If the BswM cannot serve a module’s request (according to BswM0328) to switch an I-PDU group the BswM shall accept the request and switch the I-PDU group when the blocking condition disappears. This means the state of the I-PDU group needs to be stored internally in the BswM.

BswM0330
During initialization the BswM shall initialize the internal states of the I-PDU groups as disabled.

7.1.4 Handling of BusOFF and Silent Communication

BswM0363
The BswM shall deactivate the Rx deadline monitoring for the affected CAN channel in case the CanSM reports the state CANSM_BSWM_BUS_OFF or CANSM_BSWM_SILENT_COMMUNICATION via the function BswM_CanSM_CurrentState.

The BswM shall not react on any requests received from other BSW modules during the BusOff condition to enable the deadline monitoring again. These request shall be stored internally. Once the BusOff condition has disappeared the BswM shall enable

Rx deadline monitoring again. This action shall be done just in case it complies to the current requested modes, i.e. if a PNC is deactivated during BusOff condition the deadline monitoring shall be not enabled.

7.1.5 I-PDU Group Handling during Active Wakeup of a System Channel

If an active wakeup occurs for a CAN or FR channel the BswM coordinates I-PDU group switching.

An active wakeup occurs if a ComM user is requested which maps exclusively to one or more system channels and no PNC for that system channel is requested.

BswM0302
Upon an active wake up, notified by BswM_CanSm_CurrentState, but with no requested state for the PNC from ComM, the BswM shall start all configured I-PDU groups of all PNCs on that channel. The I-PDUs of those I-PDU groups shall be initialized to their default values. The BswM shall also start deadline monitoring for those I-PDU groups.

BswM0345
If the CanSM reports the state CANSM_BSWM_SILENT_COMMUNICATION to the BswM by calling the function BswM_CanSm_CurrentState and the previous reported state was CANSM_BSWM_FULL_COMMUNICATION the BswM shall call the function Com_IpduGroupStop to disable the Tx I-PDU groups for that channel. The Rx I-PDU groups remain active.

BswM0346
If the CanSM reports the state CANSM_BSWM_NO_COMMUNICATION to the BswM by calling the function BswM_CanSm_CurrentState the BswM shall call the function Com_IpduGroupStop to disable the I-PDU groups for that channel.

BswM0348
If the FrSM reports any other state than FRSM_BSWM_ONLINE to the BswM by calling the function BswM_FrSm_CurrentState and the previous state was FRSM_BSWM_ONLINE the BswM shall call the function Com_IpduGroupStop in order to disable the I-PDU groups for that channel. Deadline monitoring shall be deactivated too.

7.1.6 Partial Networking

I-PDUs of partial networks are grouped into I-PDU groups. The I-PDU groups are a means to control the receive and send behavior of the partial networks. The ComM provides the current modes of communication channels and PNCs to the BswM. Depending on the active partial networks the BswM enables/disables the corresponding I-PDU groups. The current state of the physical channel is notified via BswM_ComM_CurrentMode.

BswM0305

When the ComM notifies the current state of a Partial Network Cluster via the API BswM_ComM_CurrentPNCMode the BswM shall handle the I-PDU groups according to the following requirements. Since the current PNC state is notified to the BswM the previous state must be remembered in order to detect state changes.

BswM0362
If a PNC is in state PNC_NO_COMMUNICATION deadline monitoring shall be stopped. The handling of I-PDU groups is handled by two variants which shall be covered by the configuration:
Variant 1: The I-PDU groups of the PNC are started if the channel is in mode FULL_COMMUNICATION (notified by the corresponding bus state manager).
Variant 2: The I-PDU groups of a PNC are stopped.

BswM0306
If ComM notifies the sub-states PNC_REQUESTED or PNC_READY_SLEEP the BswM shall start the I-PDU groups configured for that PNC and enable deadline monitoring. If depending on the configuration (see [BswM0362]) the I-PDU groups are already started, the BswM just enables deadline monitoring.

From BswM's point of view, there is no difference between PNC_REQUESTED and PNC_READY_SLEEP.

BswM0307
If ComM notifies the sub-state PNC_PREPARE_SLEEP and the previous PNC state was PNC_NO_COMMUNICATION (passive wakeup) the BswM shall start all corresponding I-PDU groups. Deadline monitoring shall be disabled. If depending on the configuration (see [BswM0362]) the I-PDU groups are already started the BswM does nothing.

BswM0353
If ComM notifies the sub-state PNC_PREPARE_SLEEP the BswM shall enable I-PDU groups and disable deadline monitoring of I-PDU groups. An exception from this requirement is described by BswM0307 (coming from sub-state PNC_NO_COMMUNICATION).

BswM0308
If ComM notifies the sub-state PNC_NO_COMMUNICATION the BswM shall deactivate deadline monitoring. Handling of I-PDU groups shall be done according to [BswM0362].

BswM0352
The BswM configuration shall reference the PNCs and the corresponding I-PDU groups.

7.1.7 Switching of PDU Routing Path Groups

The PduR provides PDU routing path groups that may be switched at runtime. A routing path group is a group of I-PDUs that can be disabled and enabled during runtime. The group contains the destination I-PDUs and not the routing path itself. The reason is that it is desirable to enable disable I-PDUs for a specific bus and a routing path can multicast an I-PDU to several busses.

In order to provide an efficient diagnostic communication from a diagnostic tester to a LIN slave during reprogramming the BswM shall support switching of routing path groups. Once a communication control request is received which addresses a LIN channel the BswM shall change the routing path group that allows for efficient routing of the diagnostic messages to LIN. If communication control for that LIN channel is deactivated the previous routing path group shall be enabled again.

BswM0318
If the DCM calls the API BswM_Dcm_RequestCommunicationMode requesting disabling of normal message transmission for a LIN channel the BswM shall select a PDU routing path group configured which enables direct routing from the requesting CAN channel to the requested LIN channel.

BswM0319
If the DCM calls the API BswM_Dcm_RequestCommunicationMode requesting disabling of normal message transmission for a LIN channel the BswM shall switch the current LIN schedule table for that LIN channel to the corresponding Diagnostic Request LIN schedule table.

BswM0335
If the Dcm calls the function BswM_Dcm_RequestCommunicationMode in order to enable normal message transmission the BswM shall change the PDU routing path group back to the previous state.

BswM0336
The configuration of the BswM shall reference the PDU routing path groups used for a specific LIN channel.

7.1.8 Switching of LIN Schedule Tables

The BswM performs a LIN schedule table switch in case it receives corresponding requests from SWCs, the LinIF or in case a communication control request is received for a specific LIN channel. The BswM must synchronize the different LIN schedule table requests and provide a prioritisation of the requests, i.e. while a diagnostics LIN schedule table is active because a communication control request was received LIN schedule table requests from the application must not change the schedule table.

BswM0320
The BswM shall provide service ports to the SWCs for switching of LIN schedule tables.

BswM0332
If the transmission of segmented data is requested by LinTp_Transmit(), the LinIF requests a schedule table change to diagnostic request schedule by calling

BswM_LinTP_RequestMode(). The BswM shall request a change of the schedule table to diagnostic request schedule.

BswM0333
If the transmission of segmented data requested by LinTp_Transmit() is completed, the LinIF requests a schedule table change to diagnostic response schedule by calling BswM_LinTP_RequestMode(). The BswM shall request a change of the schedule table to diagnostic response schedule.

BswM0334
If the schedule table shall be changed to applicative schedule the LinIF calls the function BswM_LinTP_RequestMode(). The BswM shall request the applicative schedule table which was active before the diagnostic schedule tables were requested.

The LIN schedule table can only be changed if the LinSM is in Mode FULL_COMMUNICATION and in subMode LINSM_RUN_SCHEDULE, the schedule table request is rejected if the LinSM is not in these modes.

BswM0355
The LinSM informs the BswM about the current state by means of the API BswM_LinSM_CurrentState. The BswM shall remember the current mode and submode of the LinSM. Based on the last reported state from the LinSM the BswM shall accept a requested LIN schedule table change or reject it.

7.2 Mode Arbitration

The Mode Arbitration performed by the BswM is simple and rule based. The rules used for mode arbitration are specified in the configuration of the BSW Mode Manager module. The rules are composed of trivial Boolean expressions and the mode arbitration is thus expected to have a low runtime impact.

To know what action lists to execute the BswM is required to detect changes in mode arbitration result from previous rule evaluation. How this is done and the memory needed to store results is implementation specific and not described in this document.

The R3.2 BswM does not support rules freely configured by the integrator but only the defined use cases. However, the described arbitration mechanisms can be used to implement these use cases. The description of action lists does not imply any specific way of implementation to be used.

7.2.1 Arbitration Rules

A rule is a logical expression that is composed of a set of mode request conditions. The rules are evaluated when the input mode requests are changed or during the execution of the BswM main function. The result of the evaluation (True or False) is used to decide about execution of the corresponding mode control Action List.

Arbitration rules are given by the use cases described in 7.1. It is not possible to add new rules.

7.2.2 Mode Conditions and Logical Expressions.

The logical expression that comprises a mode arbitration rule can use different operators such as AND, OR, XOR, NAND. Each term in the expression corresponds to a mode request condition. These conditions verify if a requested or indicated mode is EQUAL or NOT_EQUAL to a certain mode. An example rule with two conditions is shown in Figure 2. The rules, and the set of available logical operations are defined as a part of the ECU configuration described in chapter 10.2.

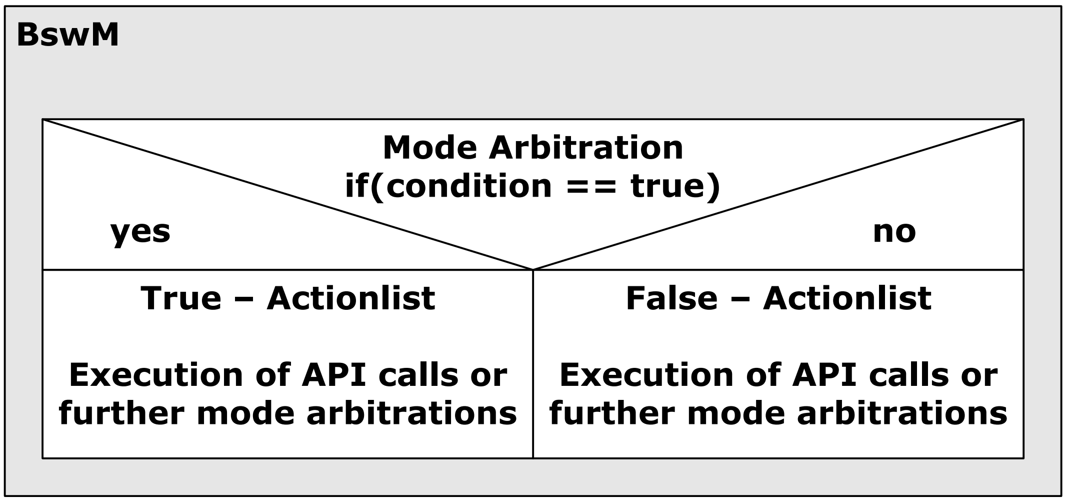

BswM
Mode Arbitration
if(condition == true)
yes
no
True – Actionlist
Execution of API calls or further mode arbitrations
False – Actionlist
Execution of API calls or further mode arbitrations

Figure 2: Rule execution in the Mode Arbitration part of the BswM.

7.2.3 Requirements on Mode Arbitration

As mentioned above, the BswM accepts mode requests as input for the mode arbitration. Mode requests normally originate from either the application SW-Cs or other BSW modules such as the DCM.

BswM0009:
The BswM shall perform mode arbitration based on incoming mode requests.

Note: All mode arbitration requests are handled in the same way by the BswM. They are configured by selection of the corresponding mode condition type in the BswMModeRequestSource configuration container.

BswM0010:
The BswM shall perform mode arbitration using predefined rules.

BswM0117:
BswM is not allowed to use results from previous mode arbitrations as input for the logical expressions. However, it is allowed to store the input values of previous mode requests for later comparison to newer requests.

BswM0344
The BswM tracks a copy of states (I-PDU groups, etc.) of other BSW modules. These state variables shall be initialized by the BswM.

7.2.3.1 Immediate and Deferred Operation

There are two different ways to schedule the processing of the mode arbitration. It is either done immediately within the context of a mode request, or deferred (cyclically) in the main function of the BswM.

If an action list has the property 'immediate', it is executed in the environment of the caller. It is the responsibility of the system integrator to ensure that execution of the action list this does not jeopardize system performance or consistency. Especially, if the caller runs (or may run) in interrupt context, the restrictions concerning usage of system functions in interrupt context apply

The difference between immediate and deferred operation is shown in the sequence diagrams in section 9.1and 9.2.

Note that the following requirements relate to arbitration rules which are not part of an action list as described in chapter 7.3 unless stated differently.

BswM0061:
A mode arbitration rule may contain any combination of immediate and deferred mode arbitration requests.

BswM0013:
It shall be possible to configure the BswM to execute the mode arbitration immediately upon a mode arbitration request. This is configured by setting the BswMRequestProcessing configuration parameter to BSWM_IMMEDIATE.

BswM0059:
Only the mode arbitration rules that use a specific immediate mode condition shall be evaluated by the BswM within the context of that specific mode request.

BswM0014:
It shall (also) be possible to defer the mode arbitration until the execution of the main function of the BswM. This is configured by setting the BswMRequestProcessing configuration parameter to BSWM_DEFERRED.

BswM0060:
All rules that use at least one deferred mode condition shall be evaluated during every execution of the main function of BswM.

BswM0068:
BswM shall delay mode arbitration requests received during the processing of its main function until it is finished.

BswM0069:

BswM shall delay mode arbitration requests received during the processing of an *immediate request* until it is finished.

BswM0358
An action list may contain a link to another arbitration rule, which has its own immediate/deferred property. In case a rule is called from within an action list the rule shall inherit the context from the topmost rule. In other words the rule's immediate/deferred property is ignored if it is called from within an action list.

Note: It is up to the implementer of the BswM to decide how to obtain the behavior required in BswM0068 and BswM0069.

7.2.4 Arbitration Behavior after Initialization

The behavior of the mode arbitration of BswM after initialization is controlled by the configuration parameters BswMModeInitValue. This parameter is configured once for each BswMModeRequestPort in the configuration. After initialization of the BswM mode requests are considered as undefined until the corresponding mode requester have made a mode arbitration request for the first time.

BswM0063:
If the optional parameter BswMModeInitValue exists the BswM shall use the corresponding mode condition until the requester requests/indicates a new mode/state.

BswM0064:
If the optional parameter BswMModeInitValue does not exist the BswM shall treat the corresponding mode condition as undefined and not use it for mode arbitration until the corresponding mode arbitration request has been updated for the first time.

BswM0065:
The BswM shall not evaluate a rule as long as any of its mode conditions have an undefined mode/state.

7.3 Mode Control

The Mode Control part of BswM carries out all actions that should be taken based on the mode arbitration. This is done using Action Lists. An Action List is an ordered list of actions that the BswM executes when triggered by the Mode Arbitration.

The actions in an Action List can be of three types:
1. Calls to other BSW modules. A set of pre-defined actions are listed in 7.3.3.
2. Links to other action lists to be included in the execution.
3. Mode arbitration rules. These rules will be evaluated when the corresponding action list is executed. In this way a hierarchy of rules is obtained.

The BswM is not required to store or react on any BSW module specific return values on its performed actions. Due to this the different state managers in the BSW indicate their current state to the BswM to be used as input for the mode arbitration.

An action list can hold 1 to N actions. To reduce the overall number of action lists, it shall be possible to cascade them. An element of an action list can either be a concrete action or a reference to another action list, or as stated above a rule to be executed by the mode arbitration. There shall be a flag connected to every action list entry that states its type (action/reference/rule). There shall be no difference between the way a list with concrete actions and the way a list with references or even a mixed list is activated.

7.3.1 Requirements on Mode Control

BswM0015:
For each rule of the mode arbitration, BswM shall be able to execute different action lists based on if the rule evaluates to True or False.

BswM0017:
An action list comprises a set of actions that BswM shall execute in an ordered manner.

BswM0018:
An action list may contain links to other action lists that BswM shall include in the execution.

BswM0019:
An action list may also include links to mode arbitration rules that BswM shall evaluate within the scope of the execution of the current action list.

BswM0067: If a rule is included in an action list as specified in BswM0019 any action list execution resulting from that evaluation shall be executed by BswM before it continues to execute the original action list.

BswM0037:
If cascaded action lists are used (i.e. using references to other rules or action lists) the action list structure may contain up to seven (7) hierarchic levels.
Note: The purpose of this limit is to make testing of BswM implementations and generator tools possible.

BswM0062:
Action lists associated with rules evaluated in the context of the mode arbitration request shall be executed by BswM immediately when triggered by the mode arbitration, and not be deferred to the main function execution.
Rationale: This allows very short latencies on mode requests when necessary.

7.3.2 Triggered and Conditional action lists

There are two ways that an action list may be executed based on evaluation of rules. Either it is executed every time the rule is evaluated with the corresponding result, or only when the evaluation result has changed from the previous evaluation. This is called conditional and triggered execution respectively. The execution method for an action list is configured using the BswMActionListExecution parameter.

BswM0011:
If a True action list is configured for triggered execution the BswM shall only execute it when the evaluation of the corresponding rule changes from False to True.

BswM0023:
If a False action list is configured for triggered execution the BswM shall only execute it when the evaluation of the corresponding rule changes from True to False.

BswM0115:
If a True action list is configured for conditional execution the BswM shall execute it every time the corresponding rule is evaluated to True.

BswM0116:
If a False action list is configured for conditional execution the BswM shall execute it every time the corresponding rule is evaluated to False.

BswM0055:
The BswM shall abort the execution of an action list if an action returns E_NOT_OK and the corresponding BswMAbortOnFail configuration parameter is set to “true”.

7.3.3 Available Actions

The set of actions that are available to use in an action list is predefined. The reason for this is to ease ECU configuration and generation of BswM configuration code.

BswM0038:
The following actions shall be available as standard actions available for the BswM by means of selecting corresponding configuration containers and parameters.

- LinSM – Setting LIN schedule tables
- COM – Activation/Deactivation of PDU groups with or without resetting of signal initialization values.
- COM – Enabling and disabling of deadline timeout monitoring
- Network Management – Disable and Enable NM Communication
- PduR – Enable/Disable PDU Routing Path Groups, to support mode dependent PDU Gateway.

BswM0039:
The BswM shall be able to call any function in the AUTOSAR BSW as well as any user defined function.

7.3.4 Behavior of Mode Control after Initialization

The behavior of the Mode Control after initialization of the BswM is configured by the BswMRuleInitState parameter. It defines the "previous evaluation result" to be used when deciding on what action list to execute after the first evaluation of a rule after initialization. The configuration parameter BswMActionListExecution also affects the action list execution after initialization.

BswM0066:
The BswM shall act according to what is stated in Table 7-2 when a rule is evaluated for the first time after initialization.

BswMRuleInitState	***BswMActionListExecution***	***Rule evaluated to true***	***Rule evaluated to false***
BSWM_UNDEFINED	BSWM_TRIGGER	Execute "true" action list	Execute "false" action list
BSWM_TRUE	BSWM_TRIGGER	Do nothing	Execute "false" action list
BSWM_FALSE	BSWM_TRIGGER	Execute "true" action list	Do nothing
BSWM_UNDEFINED	BSWM_CONDITION	Execute "true" action list	Execute "false" action list
BSWM_TRUE	BSWM_CONDITION	Execute "true" action list	Execute "false" action list
BSWM_FALSE	BSWM_CONDITION	Execute "true" action list	Execute "false" action list

Table 7-2: Usage of the BswMRuleInitState configuration parameter

Note: The "true" and "false" action lists are optional for each rule.

7.3.5 Handling of I-PDU group switching

To perform I-PDU group switches (enable/disable) in an efficient and consistent way BswM shall perform the actual I-PDU Group Control function call at the end of processing the main function or an immediate processing request. This means that all calls to the API functions Com_IpduGroupStart and Com_IpduGroupStop for manipulation of the I-PDU groups are done one after another.

BswM0128:
BswM shall keep the states of the I-PDU groups stored in an internal variable.

BswM0129:
If any BswMPduGroupSwitch action(s) have been performed BswM shall execute the Com_IpduGroupStart and Com_IpduGroupStop commands at the end of its processing of the BswM main function or an immediate request processing.

- AUTOSAR confidential -

7.4 Error classification

BswM0029:
Development error values are of type uint8.

Type or error	*Relevance*	*Related error code*	*Value [hex]*
A service was called prior to initialization	Development	BSWM_E_NO_INIT	0x01
A null pointer was passed as an argument	Development	BSWM_E_NULL_POINTER	0x02
A parameter was invalid (unspecific)	Development	BSWM_E_PARAM_INVALID	0x03
A requesting user was out of range	Development	BSWM_E_REQ_USER_OUT_OF_RANGE	0x04
A requested mode was out of range	Development	BSWM_E_REQ_MODE_OUT_OF_RANGE	0x05
The provided configuration is inconsistent	Development	BSWM_E_PARAM_CONFIG	0x06
A parameter pointer was invalid	Development	BSWM_E_PARAM_POINTER	0x07

Table 7-3: Table of development errors

7.5 Error detection

BswM0030:
The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch BSWM*_DEV_ERROR_DETECT* (see chapter 10) shall activate or deactivate the detection of all development errors.

BswM0031:
If the *BSWM_DEV_ERROR_DETECT* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 8.

BswM0032:
The detection of production code errors cannot be switched off.

7.6 Error notification

BswM0033:
Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch *BSWM_DEV_ERROR_DETECT* is set (see chapter 10).

7.7 Version checking

BswM0136:
The BswM module shall perform Inter Module Checks to avoid integration
of incompatible files.
The imported included files shall be checked by preprocessing directives.

The following version numbers shall be verified:
- <MODULENAME>_AR_MAJOR_VERSION
- <MODULENAME>_AR_MINOR_VERSION

Where <MODULENAME> is the module short name of the other (external) modules which provide header files included by the BswM module.

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

7.8 BswM Interfaces and Ports

This chapter specifies the AUTOSAR Interfaces and Ports provided by the Basic Software Mode Manager. Note that ports on both sides of the RTE are required: The SW-C description of the Basic Software Mode Manager services will define the ports below the RTE. Each AUTOSAR SW-C, which uses the services, must contain service ports in its own SW-C description. These ports are typed with the same interfaces and have to be connected to the ports of the Basic Software Mode Manager, so that the RTE can generate the appropriate IDs and the required symbols.

SW-Cs request modes from the BSW Mode Manager. To that end, they provide a Sender Port that has a special Sender/Receiver Interface (Mode Request Interface) with one data element. The corresponding Receiver Port at the BSW Mode Manager is described in Chapter 7.8.1. The data element's type has the same values as the Mode Declarations in the Mode Declaration Group of the corresponding mode (since the ImplementationDataType of the data element is mapped to the ModeDeclaration Group).

The same SW-C that requests a mode may also be a mode user because it needs to know the arbitration result of the BSW Mode Manager. The SW-C has a Mode Switch Port, which is a R-Port with a Mode Switch Interface with one data element. This data element's type is then the Mode Declaration Group itself. In addition, other SW-Cs that do not request modes but depend on them have such a Mode Switch Port. Note that the BSW Mode Manager also needs a Mode Switch R-Port if it needs to know the current mode in addition to the requested one in its decisions.

In the context of the requesting SW-C a Mode Request Port (Sender/Receiver) is defined. The configuration of BSW Mode Manager references this port definitions. Let us assume that the SW-C defines an Application Mode `AppModeType`, a corresponding `AppModeRequestType` and an `AppModeTypeMap` that maps the two types to each other:

```
ModeDeclarationGroup AppModeType {
  { APP_MODE_A, APP_MODE_B, APP_MODE_C }
  initialMode = APP_MODE_A;
};

ImplementationDataType AppModeRequestType {
  lowerLimit = 0;
  upperLimit = 2;
};

ModeRequestTypeMap AppModeTypeMap {
  modeGroup = AppModeType;
  implementationDataType = AppModeRequestType;
};
```

In the context of the SW-C two Interfaces are defined, the `AppModeRequestInterface` of Sender/Receiver type where the SW-C is sender and the `AppModeInterface` of Mode Switch type where the SW-C can have P-Ports and R-Ports depending on the usage:

```
SenderReceiverInterface AppModeRequestInterface {
  isService = true;
  AppModeRequestType requestedMode;
};
ModeSwitchInterface AppModeInterface {
  isService = true;
  AppModeType currentMode;
};
```

Figure 3 shows how the ports of the application SW-Cs connect to the service ports of the BSW Mode Manager. The Application Mode Manager SW-C has a Mode Request Port and a Mode Switch R-port (named modeNotificationPort to distinguish it from the Mode Switch P-ports). The first port is to request changes in its application mode, the latter to receive notifications when the BswM has performed the mode change. The Mode Request Port of the Application Mode Manager (modeRequestPort0) connects to the corresponding Mode Request Port of the BSW Mode Manager. Since this is normal Sender/Receiver communication, the Application Mode Manager may connect to multiple BSW Mode Managers even on remote ECUs.

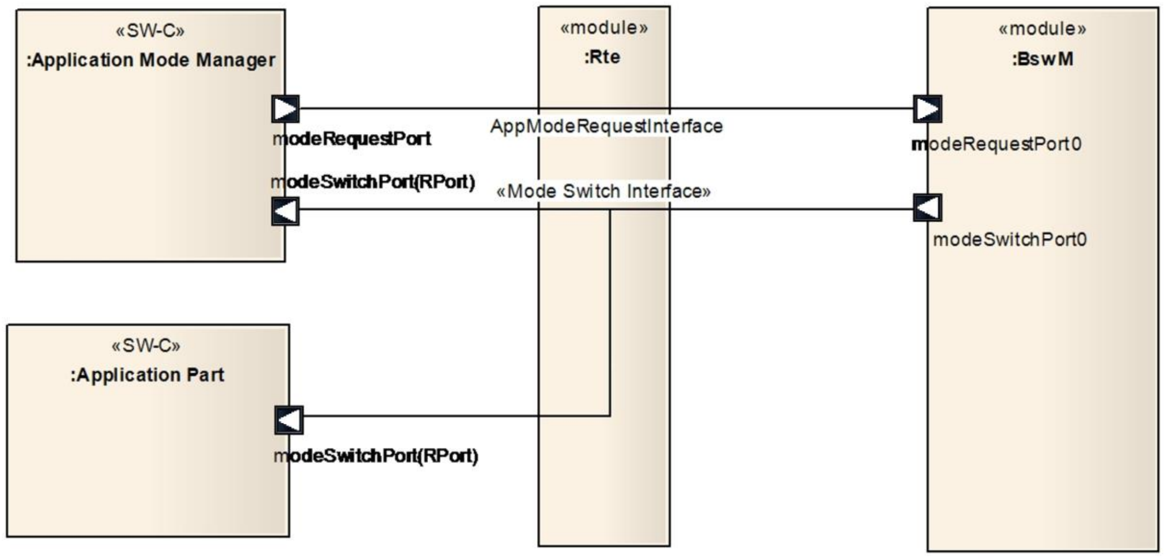


Figure 3: Connections between Application Mode Manager, Application Parts and the BSW Mode Manager

To actually switch the application mode, the BSW Mode Manager has a Mode Switch Port (modeSwitchPort0) that is implemented by the local RTE.

When the RTE performs the mode switch, it informs all connected entities (BSW Modules or SW-Cs) that are connected via Mode Switch R-Ports to the providing port. Here it is the Application Mode Manager, the other mode-dependent Application Part and the BSW Mode Manager itself (Note that it's named modeNotificationPort0 but the port type is Mode Switch Port). All of these connections are also local.

Note: To configure the BswM knowledge of what mode request ports and ECU resources are needed/available for a specific ECU is needed. Therefore the SW-C description of the BswM can only be made complete during ECU configuration time.

From now on all following interface definitions are interpreted to be in:

```
ARPackage AUTOSAR/Services/BswM
```

Note that the pseudo code presented in this chapter is not exact, but provides a hint of how the corresponding model elements needs to be defined.

7.8.1 Mode Request Ports

The BSW Mode Manager must declare a Receiver Port with the interface defined in the context of the SW-C:

```
RequirePort AppModeRequestInterface modeRequestPort0;
```

The configuration parameter `BswMSwcRequestPortRef` references this port.

To read the currently requested mode, the BSW Mode Manager implementation has to call:

```
Rte_Read_modeRequestPort0_requestedMode( &<variable> );
```

7.8.2 Mode Switch Ports

As with Mode Requests, the BSW Mode Manager only references the mode switch interfaces defined in the context of the corresponding SW-C Description in its Provide Ports for mode switches. For the above example the declaration for a mode switch is:

```
ProvidePort AppModeInterface modeSwitchPort0;
```

The configuration parameter `BswMModeSwitchInterfaceRef` references this Mode Switch interface.

To switch the currently active mode, the BSW Mode Manager implementation has to insert one of the following codes into its actions list:

```
Rte_Switch_modeSwitchPort0_currentMode( <new_mode> );

SchM_Switch_modeSwitchPort0_currentMode( <new_mode> );
```

7.8.3 Component Type and Internal Behavior

The BSW Mode Manager is a Service Component that serves Mode Requests that are local to the ECU. The ServiceComponentType for the BSW Mode Manager declares all of the above mentioned ports and some Internal Behavior.

```
ServiceComponentType BswM {
  …
  InternalBehavior {
    …
  };
};
```

The internal behavior depends on the parameter `BswMRequestProcessing` for the corresponding Mode Request Port. For `BSWM_DEFERRED` the RTE has to do nothing special, as the BSWM Mode Manager reads the request cyclically in `BswM_MainFunction`. By contrast, for `BSWM_IMMEDIATE` the RTE has to trigger mode arbitration immediately. Therefore, the BSW Mode Manager needs to register a trigger function that triggers mode arbitration. For the above example, an immediate processing of the mode request would need the following declaration in the Internal Behavior of the BSW Mode Manager:

BswM0138:
```
RunnableEntity ModeArbitrationRunnable {
  symbol = <mode_arbitration_function>;
  canBeInvokedConcurrently = TRUE;
};

DataReceiveEvent AppModeRequestEvent {
  port = modeRequestPort0;
  dataElement = requestedMode;
  startOnEvent = ModeArbitrationRunnable;
};
```

Note: To deal with Mode Requests that originates from other ECU:s another kind of service component is needed. On VFB level it looks like one global Service Component but actually it is instantiated as one Service Component that resides above the RTE for each ECU. To support that, the SW-C Template offers the ServiceProxyComponentType instead of the normal ServiceComponentType.

The specification of the Mode Management Service Proxy Component is not described within this document since it is user specific.

7.8.4 BswM Service

This chapter summarizes the Software Component Description of the BSW Mode Manager.

BswM0137:

```
ServiceComponentType BswM {

  // For each mode request
  RequirePort <request_interface> modeRequestPort<number>;

  // For each mode switch
  ProvidePort <mode_interface> modeSwitchPort<number>;

  InternalBehavior {

    RunnableEntity ModeArbitrationRunnable {
      symbol = <mode_arbitration_function>;
      canBeInvokedConcurrently = TRUE;
    };

    // For each mode request with BSWM_IMMEDIATE
    DataReceiveEvent ModeRequestEvent<number> {
      port = modeRequestPort<number>;
      dataElement = <data_element>;
      startOnEvent = ModeArbitrationRunnable;
    };

  };

};
```

8 API specification

8.1 Imported types

In this chapter, all types included from the following files are listed:

BswM0001: The BSW Mode Manager shall use only the following imported types of other modules:

Module	***Imported Type***
CanSM	CanSM_BswMCurrentStateType
Com	Com_PduGroupIdType
ComM	ComM_InhibitionStatusType
	ComM_InitStatusType
	ComM_ModeType
	ComM_PncModeType
	ComM_UserHandleType
ComStack_Types	NetworkHandleType
	PNCHandleType
Dcm	Dcm_CommunicationModeType
	Dcm_ResetModeType
FrSm	FrSM_BswM_StateType
LinIf	LinIf_SchHandleType
	LinTp_Mode
LinSM	LinSM_ModeType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

BswM0041: The following Data Types shall be used for the functions defined in this specification.

8.2.1 BswM_ConfigType

Name:	`BswM_ConfigType`	
Type:	`Structure`	
Range:	–	The contents of this structure depends on the configuration variant.
Description:	This structure contains all post-build configurable parameters of the BSW Mode Manager. A pointer to this structure is passed to the BSW Mode Manager initialization function for configuration.	

BswM0042: The structure BswM_ConfigType shall contain all post-build configurable parameters of the BSW Mode Manager. The exact content of this structure depends on the selected configuration variant

8.2.2 BswM_ModeType

Name:	`BswM_ModeType`	
Type:	`uint8, uint16`	
Range:	`0-255, 0-65535`	`--` The range of valid IDs depends on configuration and on the chosen platform type.
Description:	This type identifies the modes that can be requested by BswM Users.	

8.2.3 BswM_UserType

Name:	`BswM_UserType`	
Type:	`uint8, uint16`	
Range:	`0-255, 0-65535`	`--` The range of valid IDs depends on configuration and on the chosen platform type.
Description:	This type identifies a BswM User that makes mode requests to the BswM.	

8.3 Function definitions

8.3.1 BswM_Init

BswM0002:

Service name:	BswM_Init	
Syntax:	`void BswM_Init(` `const BswM_ConfigType * ConfigPtr` `)`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to post-build configuration data
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the BSW Mode Manager.	

BswM0043: This routine initializes the BSW Mode Manager. After execution of this routine the BSW Mode Manager is ready to arbitrate incoming mode requests.

BswM0044: This routine shall initialize all module global variables of the BSW Mode Manager.

BswM0118: BswM_Init shall only require the OS.

BswM0045: If the BswMDevErrorDetect switch is enabled, the contents of the given configuration set shall be checked for being within the allowed boundaries. If an error is detected the initialization of the BSW Mode Manager shall not be executed and the error shall be reported to the Development Error Tracer with the value BSWM_E_PARAM_CONFIG.

BswM0088:
If the BswMDevErrorDetect switch is enabled and the configuration variant is VARIANT-POST-BUILD, the function BswM_Init shall check if a NULL pointer is passed for the ConfigPtr parameter. In case of an error the remaining function BswM_Init shall not be executed and the function BswM_Init shall report development error code BSWM_E_NULL_POINTER to the Det_ReportError service of the Development Error Tracer.

8.3.2 BswM_Deinit

BswM0119:

Service name:	BswM_Deinit

Syntax:	`void BswM_Deinit(`
	
	`)`

Service ID[hex]:	0x04
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Deinitializes the BSW Mode Manager.

BswM0120:
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.

8.3.3 BswM_GetVersionInfo

BswM0003:

Service name:	BswM_GetVersionInfo	

Syntax:	`void BswM_GetVersionInfo(`
	`    Std_VersionInfoType* VersionInfo`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	VersionInfo	Pointer to where to store the version information of the module.
Return value:	None	
Description:	Returns the version information of this module.	

BswM0004:
The function BswM_GetVersionInfo shall return the version information of this module. The version information includes:
- Module Id

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

BswM0005:
The function BswM_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: BswMVersionInfoApi.

BswM0006:
If source code for caller and callee of BswM_GetVersionInfo is available, the BSW Mode Manager should realize BswM_GetVersionInfo as a macro, defined in the module's header file.

BswM0139:
If the BswMDevErrorDetect switch is enabled, the function BswM_GetVersionInfo shall check if a NULL pointer is passed for the VersionInfo parameter.
In case of an error the function BswM_GetVersionInfo shall not be executed and an error BSWM_E_PARAM_POINTER shall be reported to the Development Error Tracer.

8.3.4 BswM_ComM_CurrentMode

BswM0047:

Service name:	BswM_ComM_CurrentMode	

Syntax:	`void BswM_ComM_CurrentMode(`
	`    NetworkHandleType Network,`
	`    ComM_ModeType RequestedMode`
	`)`	

Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Network	The ComM communication channel that the indicated state corresponds to.
	RequestedMode	The current state of the ComM communication channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by ComM to indicate the current communication mode of a ComM channel.	

BswM0078:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0091:
If the BswMDevErrorDetect switch is enabled, the parameter RequestedMode shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0092:
If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.3.5 BswM_ComM_CurrentPNCMode

BswM0337

Service name:	BswM_ComM_CurrentPNCMode	
Syntax:	`void BswM_ComM_CurrentPNCMode( PNCHandleType PNC, ComM_PncModeType CurrentPncMode )`	
Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	PNC	The handle of the PNC for which the current state is reported.
	CurrentPncMode	The current mode of the PNC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by ComM to indicate the current mode of the PNC.	

BswM0338
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0339
If the BswMDevErrorDetect switch is enabled, the parameter RequestedMode shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0340

If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.3.6 BswM_Dcm_RequestCommunicationMode

BswM0048:

Service name:	BswM_Dcm_RequestCommunicationMode	
Syntax:	`void BswM_Dcm_RequestCommunicationMode( NetworkHandleType Network, Dcm_CommunicationModeType RequestedMode )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Network	The communication channel that the diagnostic mode corresponds to.
	RequestedMode	The requested diagnostic communication mode.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by DCM to request communication modes.	

BswM0079:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0093:
If the BswMDevErrorDetect switch is enabled, the parameter RequestedMode shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0094:
If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.3.7 BswM_Dcm_RequestResetMode

BswM0123

Service name:	BswM_Dcm_RequestResetMode	
Syntax:	`void BswM_Dcm_RequestResetMode( Dcm_ResetModeType RequestedMode )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	RequestedMode	The requested DCM reset mode.
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	Function called by DCM to request a reset of the ECU.

BswM0126:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0127:
If the BswMDevErrorDetect switch is enabled, the parameter RequestedMode shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

8.3.8 BswM_Dcm_ApplicationUpdated

BswM0360

Service name:	BswM_Dcm_ApplicationUpdated

Syntax:	`void BswM_Dcm_ApplicationUpdated(`
	
	 `)`

Service ID[hex]:	0x14
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This function is called by the DCM in order to report an updated application.

BswM0361
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

8.3.9 BswM_CanSM_CurrentState

BswM0049:

Service name:	BswM_CanSM_CurrentState

Syntax:	`void BswM_CanSM_CurrentState(`
	 `    NetworkHandleType Network,`
	 `    CanSM_BswMCurrentStateType CurrentState`
	 `)`

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Network	The CAN channel that the indicated state corresponds to.
	CurrentState	The current state of the CAN channel.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by CanSM to indicate its current state.	

BswM0080:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the state indication and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0095:
If the BswMDevErrorDetect switch is enabled, the parameter CurrentState shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the state indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0096:
If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the state indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.3.10 BswM_FrSM_CurrentState

BswM0051:

Service name:	BswM_FrSM_CurrentState	

Syntax:	`void BswM_FrSM_CurrentState(`
	`    NetworkHandleType Network,`
	`    FrSM_BswM_StateType CurrentState`
	`)`	

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Network	The FlexRay cluster that the indicated state corresponds to.
	CurrentState	The corrent state of the FlexRay cluster.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by FrSM to indicate its current state.	

BswM0082:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the state

indication and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0099:
If the BswMDevErrorDetect switch is enabled, the parameter CurrentState shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the state indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0100:
If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the state indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.3.11 BswM_LinSM_CurrentState

BswM0052:

Service name:	BswM_LinSM_CurrentState	

Syntax:	`void BswM_LinSM_CurrentState(`
	`    NetworkHandleType Network,`
	`    LinSM_ModeType CurrentState`
	`)`	

Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Network	The LIN channel that the indicated state corresponds to.
	CurrentState	The current state of the LIN channel.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by LinSM to indicate its current state.	

BswM0083:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the state indication and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0101:
If the BswMDevErrorDetect switch is enabled, the parameter CurrentState shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the state indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0102:
If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of an error, the BswM shall ignore

the state indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.3.12 BswM_LinSM_CurrentSchedule

BswM0058

Service name:	BswM_LinSM_CurrentSchedule	

Syntax:	`void BswM_LinSM_CurrentSchedule(`
	`    NetworkHandleType Network,`
	`    LinIf_SchHandleType CurrentSchedule`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Network	The LIN channel that the schedule table switch have occurred on.
	CurrentSchedule	The currently active schedule table of the LIN channel.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by LinSM to indicate the currently active schedule table for a specific LIN channel.	

BswM0086:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error the BswM shall ignore the schedule indication and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0107:
If the BswMDevErrorDetect switch is enabled, the parameter CurrentSchedule shall be checked for being in the allowed range. In case of an error the BswM shall ignore the schedule indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0108:
If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the schedule indication and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.3.13 BswM_LinTp_RequestMode

BswM0364

Service name:	BswM_LinTp_RequestMode

Syntax:	`void BswM_LinTp_RequestMode(`
	`    NetworkHandleType Network,`
	`    LinTp_Mode LinTpRequestedMode`
	`)`

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Network	The LIN channel that the LinTp mode request relates to.
	LinTpRequestedMode	The requested LIN TP mode.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Function called by LinTP to request a mode for the corresponding LIN channel. The LinTp_Mode mainly correlates to the LIN schedule table that should be used.	

BswM0112:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the error code BSWM_E_NO_INIT.

BswM0113:
If the BswMDevErrorDetect switch is enabled, the parameter LinTpRequestedMode shall be checked for being in the allowed range. In case of an error the BswM shall ignore the mode request and report the error, to the Development Error Tracer with the value BSWM_E_REQ_MODE_OUT_OF_RANGE.

BswM0114:
If the BswMDevErrorDetect switch is enabled, the parameter Network shall be checked for being in the allowed range. In case of an error, the BswM shall ignore the mode request and report the error to the Development Error Tracer with the value BSWM_E_REQ_USER_OUT_OF_RANGE.

8.4 Call-back notifications

There are no call-back notifications in the BswM.

8.5 Scheduled functions

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

8.5.1 BswM_MainFunction

BswM0053:

Service name:	BswM_MainFunction

Syntax:	`void BswM_MainFunction(`
	
	 `)`

Service ID[hex]:	0x03
Timing:	FIXED_CYCLIC
Description:	Main function of the BswM

BswM0075:
The BswM_MainFunction shall perform evaluation of all rules that uses at least one mode request with configuration parameter BswMRequestProcessing set to BSWM_DEFERRED as input.

BswM0076:
If the BswMDevErrorDetect switch is enabled, the routine shall check if the BSW Mode Manager is initialized. If the BswM_MainFunction is uninitialized called from the BSW Scheduler, then it shall return immediately without performing any action and without reporting an error.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

This chapter defines all interfaces that are required to fulfill the core functionality of the module.

BswM0007:

API function	***Description***
Com_IpduGroupStart	Starts a preconfigured I-PDU group. For example, cyclic I-PDUs will be sent out cyclically after the call of Com_IpduGroupStart(). See Chapter 7.4.5 for details. If Initialize is true all I-PDUs of the I-PDU group shall be (re-)initialized before the I-PDU group is started. That is they shall behave like after a start-up of COM, for example the old_value of the filter objects and shadow buffers of signal groups have to be (re-)initialized.
Com_IpduGroupStop	Stops a preconfigured I-PDU group. For example, cyclic I-PDUs will be stopped after the call of Com_IpduGroupStop(). See Chapter 7.4.5 for details.

8.6.2 Optional Interfaces

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

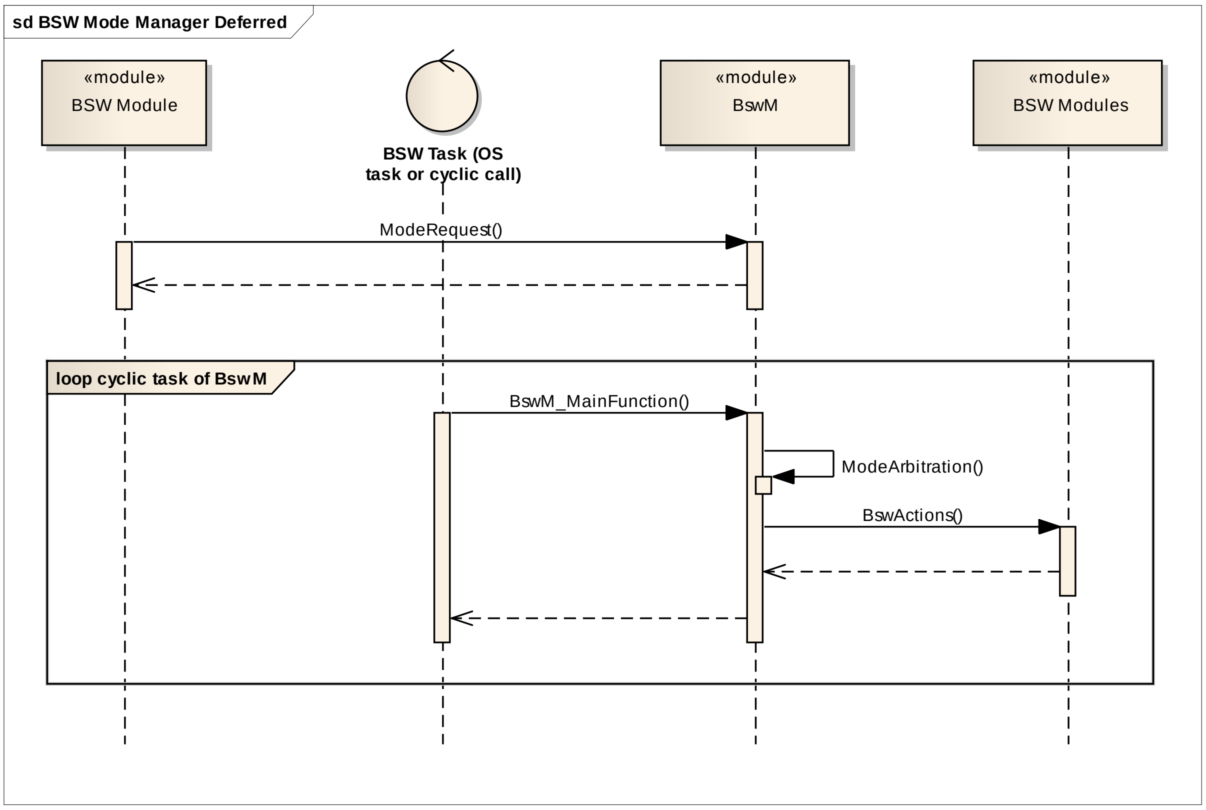
BswM0008:

API function	***Description***
ComM_GetCurrentComMode	Function to query the current Communication Mode. ComM shall use the corresponding interfaces of the Bus State Managers to get the current Communication Mode of the network. (Call to Bus State Manager API: XXXSM

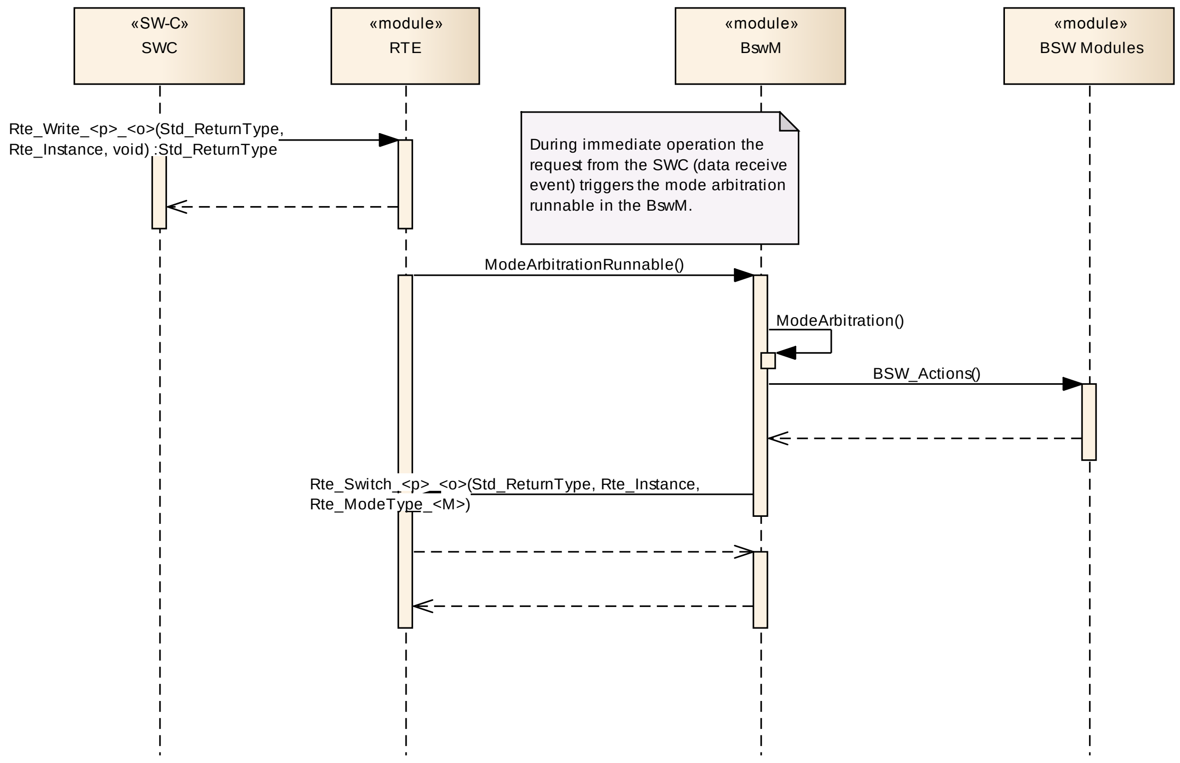
	_GetCurrentComMode(...))
ComM_GetInhibitionStatus	Returns the inhibition status of a ComM channel.
ComM_GetRequestedComMode	Function to query the currently requested Communication Mode of the corresponding user.
ComM_RequestComMode	Requesting of a Communication Mode by a user. Note: Internally mode COMM_SILENT_COMMUNICATION is not a valid request for a user, mode used for synchronization at shutdown. Valid modes are COMM_NO_COMMUNICATION and COMM_FULL_COMMUNICATION
Com_IpduGroupStart	This service starts I-PDU groups.
Com_IpduGroupStop	This service stops I-PDU groups.
Com_ReceptionDMControl	This service enables or disables I-PDU group Deadline Monitoring.
Det_ReportError	Service to report development errors.
LinSM_ScheduleRequest	The upper layer requests a schedule table to be changed on one LIN network.
Nm_DisableCommunication	Disables the NM PDU transmission ability. For that purpose <BusNm>_DisableCommunication shall be called (e.g. CanNm_DisableCommunication function is called if channel is configured as CAN).
Nm_EnableCommunication	Enables the NM PDU transmission ability. For that purpose <BusNm>_EnableCommunication shall be called (e.g. CanNm_EnableCommunication function is called if channel is configured as CAN).

9 Sequence diagrams

9.1 Deferred operation of BswM



9.2 Immediate operation of BswM


«SW-C»
SWC
«module»
RTE
«module»
BswM
«module»
BSW Modules
Rte_Write_<p>_<o>(Std_ReturnType, Rte_Instance, void) :Std_ReturnType
During immediate operation the request from the SWC (data receive event) triggers the mode arbitration runnable in the BswM.
ModeArbitrationRunnable()
ModeArbitration()
BSW_Actions()
Rte_Switch_<p>_<o>(Std_ReturnType, Rte_Instance, Rte_ModeType_<M>)

10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module BSW Mode Manager.

Chapter 10.3 specifies published information of the module BSW Mode Manager.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:
- AUTOSAR Layered Software Architecture [2].
- AUTOSAR ECU Configuration Specification [8]. This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

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

10.1.1 Configuration and configuration parameters

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

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

10.1.2 Variants

The BSW Mode Manager provides two configuration variants.

BswM0020:
Variant 1 – VARIANT-PRE-COMPILE: In this configuration variant all parameters need to be configured pre compile time.

BswM0021:
Variant 2 - VARIANT-LINK-TIME: This configuration variant contains a mix of pre compile time and link time parameters. The configuration class of each parameter is defined in chapter 10.2.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:
- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:
- the general section
- the configuration parameter section
- the section of included/referenced containers

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
<Reference a valid (sub)container by its name, e.g.,CanController>	*<Specifies the possible number of instances of the referenced container and its contained configuration parameters. Possible values: <multiplicity> <min_multiplicity..max_multiplicity>>*	*<Describe the scope of the referenced sub-container if known or mark it as "- -". The scope describes the impact of the configuration parameter: Does the setting affect only one instance of the module (instance), all instances of this module (module), the ECU or a network. Possible values of scope : instance, module, ECU, network> <Describe the dependencies with respect to the scope if known ot mark it as "- -".>*

Pre-compile time	-	specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
X	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time	-	specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
X	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build	-	specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***

Label	***Description***
X	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

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

10.2.1 BswM

Module Name	*BswM*
Module Description	Configuration of the BswM (Basic SW Mode Manager) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR BswM module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
BswMGeneral	1	General configuration parameters of the Basic SW Mode Manager.

10.2.2 BswMGeneral

SWS Item	**BswM0800_Conf :**
Container Name	BswMGeneral
Description	General configuration parameters of the Basic SW Mode Manager.
Configuration Parameters	

SWS Item	**BswM0811_Conf :**		
Name	BswMDevErrorDetect {BSWM_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification ON or OFF. true: Enabled false: Disabled		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**BswM0813_Conf :**		
Name	BswMMainFunctionPeriod		
Description	The cycle time of the periodic main function of BswM. Defined in seconds .		
Multiplicity	0..1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE

	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**BswM0812_Conf :**		
Name	BswMVersionInfoApi		
Description	Switches the possibility to read the version information with the service BswM_GetVersionInfo(). true: Enabled false: Disabled		
Multiplicity	1		
Type	BooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.3 BswMConfig

SWS Item	**BswM0895_Conf :**
Container Name	BswMConfig
Description	This container contains the configuration parameters and sub containers of the AUTOSAR BswM module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMArbitration	1	This container includes all configuration sub-containers and parameters related to the mode arbitration functionality of the BswM.
BswMModeControl	1	This container includes all configuration sub-containers and parameters related to the mode control functionality of the BswM.

10.2.4 BswMArbitration

SWS Item	**BswM0801_Conf :**
Container Name	BswMArbitration
Description	This container includes all configuration sub-containers and parameters related to the mode arbitration functionality of the BswM.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMLogicalExpression	0..*	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. There may be an unlimited number of arguments in each logical expression. Note that the order of evaluation of the expressions is not defined. Note that for the NAND logical expression only two operands are supported.
BswMModeCondition	0..*	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.
BswMModeRequestPort	0..*	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: 1. Mode requests from the SW-C:s 2. Mode Requests from other BSW modules such as the DCM. 3. State/mode indications from the RTE or other BSW modules such as the bus specific State Managers. Note that the BswM treats all request and indications in the exact same way.
BswMRule	0..*	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. This container also references the action lists that shall be invoked when the rule is evaluated to True or False.

10.2.5 BswMRule

SWS Item	**BswM0806_Conf :**
Container Name	BswMRule
Description	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. This container also references the action lists that shall be invoked when the rule is evaluated to True or False.
Configuration Parameters	

SWS Item	**BswM00935_Conf :**		
Name	BswMNestedExecutionOnly		
Description	This parameter defines for its related Rule if the Rule is an Independent rule or a Subordinate rule; false: an Independent rule, i.e. to be evaluated each time applicable (both as standalone Rule driven by its own BswMModeRequestSource and when referenced by another Rule). true: a Subordinated rule, to be evaluated ONLY as a result of being referenced in one or more Action Lists.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**BswM0888_Conf :**		
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_UNDEFINED, the evaluation result is always treated as changed at the first evaluation of the rule after initialization. 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.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	BSWM_FALSE		
	BSWM_TRUE		
	BSWM_UNDEFINED		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**BswM0819_Conf :**		
Name	BswMRuleExpressionRef		
Description	This choice reference either references the mode condition or the logical expression that is evaluated for each rule.		
Multiplicity	1		
Type	Reference to [BswMLogicalExpression]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**BswM0818_Conf :**		
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 to [BswMActionList]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**BswM0817_Conf :**		
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 to [BswMActionList]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME

	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.6 BswMModeRequestPort

SWS Item	**BswM0805_Conf :**
Container Name	BswMModeRequestPort
Description	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: 1. Mode requests from the SW-C:s 2. Mode Requests from other BSW modules such as the DCM. 3. State/mode indications from the RTE or other BSW modules such as the bus specific State Managers. Note that the BswM treats all request and indications in the exact same way.
Configuration Parameters	

SWS Item	**BswM0889_Conf :**		
Name	BswMModeInitValue		
Description	This parameter defines the initial mode value that is used by BswM for the corresponding mode request after initialization. This parameter is optional and shall only be used for Mode Requests that does not already have an initial value. When this parameter is not present the requested mode is undefined after initialization of BswM. The requested mode will remain undefined until the requester performs a request. Mode arbitration rules shall not be evaluated as long as any of the used mode requests is undefined.		
Multiplicity	0..1		
Type	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	dependency: If the parameter BswMReqModeUndefAfterInit is set to true for a mode request it overrides this parameter.		

SWS Item	**BswM0822_Conf :**		
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		
Type	EnumerationParamDef		
Range	BSWM_DEFERRED		
	BSWM_IMMEDIATE		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	

Scope / Dependency	scope: Module

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMModeRequestSource	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.

10.2.7 BswMModeRequestSource

SWS Item	**BswM0856_Conf :**
Choice container Name	BswMModeRequestSource
Description	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.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMCanSMIndication	0..1	This is an indication of the current state of the CAN State Manager.
BswMComMIndication	0..1	This is an indication of the current communication mode of a channel in the Communication Manager.
BswMComMPncRequest	0..1	This is a request of the current communication mode of a Partial Network Cluster in the Communication Manager.
BswMDcmApplicationUpdatedIndication	0..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.
BswMDcmCommunicationCtrlModeRequest	0..1	The source of the mode request is the Diagnostic Communication Manager.
BswMDcmResetModeRequest	0..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.
BswMFrSMIndication	0..1	This is an indication of the current state of the FlexRay State Manager.
BswMLinTpModeRequest	0..1	This is a LinTp mode request from the LinIf. This port corresponds to a call of the BswM_LinTp_RequestMode API.
BswMSwcModeNotification	0..1	This is a mode switch notification associated with a RTE switch interface.
BswMSwcModeRequest	0..1	The source of the mode request is a SW Component.

10.2.8 BswMCanSMIndication

SWS Item	**BswM0857_Conf :**

Container Name	BswMCanSMIndication
Description	This is an indication of the current state of the CAN State Manager.
Configuration Parameters	

SWS Item	**BswM0870_Conf :**		
Name	BswMCanSMChannelRef		
Description	This is a reference to the CAN channel handle that the mode request corresponds to.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.9 BswMDcmCommunicationCtrlModeRequest

SWS Item	**BswM0863_Conf :**
Container Name	BswMDcmCommunicationCtrlModeRequest
Description	The source of the mode request is the Diagnostic Communication Manager.
Configuration Parameters	

SWS Item	**BswM0876_Conf :**		
Name	BswMDcmComMNetwork		
Description	This parameter specifies the network the request relates to.		
Multiplicity	1		
Type	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.10 BswMDcmResetModeRequest

SWS Item	**BswM0897_Conf :**
Container Name	BswMDcmResetModeRequest

Description	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.
Configuration Parameters	

No Included Containers

10.2.11 BswMDcmApplicationUpdatedIndication

SWS Item	**BswM0898_Conf :**
Container Name	BswMDcmApplicationUpdatedIndication
Description	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.
Configuration Parameters	

No Included Containers

10.2.12 BswMFrSMIndication

SWS Item	**BswM0858_Conf :**
Container Name	BswMFrSMIndication
Description	This is an indication of the current state of the FlexRay State Manager.
Configuration Parameters	

SWS Item	**BswM0872_Conf :**		
Name	BswMFrSMChannelRef		
Description	This is a reference to the FlexRay Cluster handle that the mode request corresponds to.		
Multiplicity	1		
Type	Reference to [ComMChannel]		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.13 BswMComMIndication

SWS Item	**BswM0880_Conf :**
Container Name	BswMComMIndication
Description	This is an indication of the current communication mode of a channel in the Communication Manager.

Configuration Parameters

SWS Item	**BswM0883_Conf :**	
Name	BswMComMChannelRef	
Description	This is a reference to the Communication Manager channel handle that the indication corresponds to.	
Multiplicity	1	
Type	Reference to [ComMChannel]	

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.14 BswMLinTpModeRequest

SWS Item	**BswM0914_Conf :**
Container Name	BswMLinTpModeRequest
Description	This is a LinTp mode request from the LinIf. This port corresponds to a call of the BswM_LinTp_RequestMode API.
Configuration Parameters	

SWS Item	**BswM0915_Conf :**
Name	BswMLinTpChannelRef
Description	This is a reference to the LIN Interface Channel that the mode request corresponds to.
Multiplicity	1
Type	Reference to [LinIfChannel]

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.15 BswMSwcModeNotification

SWS Item	**BswM0892_Conf :**
Container Name	BswMSwcModeNotification
Description	This is a mode switch notification associated with a RTE switch interface.
Configuration Parameters	

SWS Item	**BswM0893_Conf :**
Name	BswMSwcModeNotificationIfcRef
Description	This is a instance reference to the ModeDeclarationGroupPrototype of the

	BswServiceComponent that the Mode Switch Notification corresponds to. The destinationContext element is deprecated and will be removed in future.		
Multiplicity	1		
Type	Instance reference to [ModeDeclarationGroupPrototype context: ComponentPrototype* PortPrototype]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.16 BswMSwcModeRequest

SWS Item	**BswM0862_Conf :**
Container Name	BswMSwcModeRequest
Description	The source of the mode request is a SW Component.
Configuration Parameters	

SWS Item	**BswM0878_Conf :**		
Name	BswMSwcModeRequestPortRef		
Description	This is a instance reference to the ModeDeclarationGroupPrototype of the BswServiceComponent. The ModeDeclarationGroupPrototype PortPrototype shall be connected to the requesting SoftwareComponentPrototype's PortPrototype. The destinationContext element is deprecated and will be removed in future.		
Multiplicity	1		
Type	Instance reference to [ModeDeclarationGroupPrototype context: ComponentPrototype* PortPrototype]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.17 BswMModeCondition

SWS Item	**BswM0807_Conf :**
Container Name	BswMModeCondition
Description	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.
Configuration Parameters	

SWS Item	**BswM0815_Conf :**
Name	BswMConditionType

Description	This parameter specifies what kind of comparison that is made for the evaluation of the mode condition.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	BSWM_EQUALS		
	BSWM_EQUALS_NOT		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**BswM0821_Conf :**		
Name	BswMConditionMode		
Description	This parameter references the mode request port that is used for the condition.		
Multiplicity	1		
Type	Reference to [BswMModeRequestPort]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Instance		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMConditionValue	1	This container holds the parameters and references necessary to identify the mode type and the value that the mode request is compared to.

10.2.18 BswMConditionValue

SWS Item	**BswM0816_Conf :**
Choice container Name	BswMConditionValue
Description	This container holds the parameters and references necessary to identify the mode type and the value that the mode request is compared to.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMBswMode	0..1	This container defines the value and type of a mode in the BSW.
BswMSwcMode	0..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.

10.2.19 BswMBswMode

SWS Item	**BswM0869_Conf :**

Container Name	BswMBswMode
Description	This container defines the value and type of a mode in the BSW.
Configuration Parameters	

SWS Item	**BswM0867_Conf :**		
Name	BswMBswModeSourceType		
Description	This element is deprecated and will be removed in future. This parameter specifies the type of mode request.		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	BSWM_CANSM_INDICATION		
	BSWM_COMM_INDICATION		
	BSWM_COMM_PNC_REQUEST		
	BSWM_DCM_APPLICATION_UPDATED_INDICATION		
	BSWM_DCM_COMMUNICATION_CTRL_REQUEST		
	BSWM_DCM_RESET_REQUEST		
	BSWM_FRSM_INDICATION		
	BSWM_LINTP_MODE_REQUEST		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**BswM0866_Conf :**		
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		
Type	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.20 BswMSwcMode

SWS Item	**BswM0868_Conf :**
Container Name	BswMSwcMode
Description	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.
Configuration Parameters	

SWS Item	**BswM0865_Conf :**

Name	BswMSwcConditionType		
Description	This element is deprecated and will be removed in future. This parameter defines if the condition relates to a mode request or a mode indication.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	BSWM_SWC_MODE_NOTIFICATION		
	BSWM_SWC_MODE_REQUEST		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**BswM0864_Conf :**		
Name	BswMSwcModeDeclRef		
Description	This is a foreign reference to the Mode Declaration used for the mode requests corresponding to this condition.		
Multiplicity	1		
Type	Foreign reference to [ModeDeclaration]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.21 BswMLogicalExpression

SWS Item	**BswM0808_Conf :**
Container Name	BswMLogicalExpression
Description	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. There may be an unlimited number of arguments in each logical expression. Note that the order of evaluation of the expressions is not defined. Note that for the NAND logical expression only two operands are supported.
Configuration Parameters	

SWS Item	**BswM0814_Conf :**	
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	EnumerationParamDef	
Range	BSWM_AND	
	BSWM_NAND	
	BSWM_OR	
	BSWM_XOR	

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**BswM0820_Conf :**		
Name	BswMArgumentRef		
Description	This is a choice reference either to a mode condition or a sub-expression.		
Multiplicity	1..*		
Type	Choice reference to [BswMLogicalExpression , BswMModeCondition]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.22 BswMModeControl

SWS Item	**BswM0802_Conf :**
Container Name	BswMModeControl
Description	This container includes all configuration sub-containers and parameters related to the mode control functionality of the BswM.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMAction	0..*	Each container of this type defines an action. These actions can be part of one or several action lists.
BswMActionList	0..*	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.

10.2.23 BswMAction

SWS Item	**BswM0810_Conf :**
Container Name	BswMAction
Description	Each container of this type defines an action. These actions can be part of one or several action lists.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMAvailableActions	1	Choice container including the available actions to be used in the action lists.

10.2.24 BswMActionList

SWS Item	**BswM0809_Conf :**
Container Name	BswMActionList
Description	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.
Configuration Parameters	

SWS Item	**BswM0894_Conf :**		
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.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	BSWM_CONDITION		
	BSWM_TRIGGER		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Instance		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMActionListItem	1..*	This container defines an item in an action list.

10.2.25 BswMAvailableActions

SWS Item	**BswM0826_Conf :**
Choice container Name	BswMAvailableActions
Description	Choice container including the available actions to be used in the action lists.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
BswMDeadlineMonitoringControl	0..1	This container includes all parameters related to enabling and disabling of deadline monitoring for one or several PDUs in COM.
BswMLinScheduleSwitch	0..1	This container includes all parameters related to a switch of LIN schedule table. LinSM_ScheduleRequest is called when this action is configured.
BswMNMControl	0..1	This container includes all parameters related to enabling and disabling of Network Management communication. Disabling of NM communication can be requested by DCM. Nm_EnableCommunication or Nm_DisableCommunication is called when this action is configured.
BswMPduGroupSwitch	0..1	This container includes references to the PDU groups that shall be enabled and disabled.
BswMPduRouterControl	0..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.
BswMRteSwitch	0..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. It is expected that the Software Component Description of the BswM contains a PPortPrototype for each instance of this container. The provided port shall be typed by the BswMRteSwitchInterfaceRef. The shortName of the PPortPrototype shall be equal to the shortName of this container.
BswMUserCallout	0..1	This container includes all details needed for a user defined function call.

10.2.26 BswMDeadlineMonitoringControl

SWS Item	**BswM0830_Conf :**
Container Name	BswMDeadlineMonitoringControl
Description	This container includes all parameters related to enabling and disabling of deadline monitoring for one or several PDUs in COM.
Configuration Parameters	

SWS Item	**BswM0852_Conf :**		
Name	BswMDisabledDMPduGroupRef		
Description	This is a reference to a PDU Group for which the Deadline Monitoring should be disabled. BswM shall call Com_DisableReceptionDM for each referenced IPduGroup.		
Multiplicity	0..*		
Type	Reference to [ComIPduGroup]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	dependency: BswMEnabledDMPduGroupRef		

SWS Item	**BswM0851_Conf :**		
Name	BswMEnabledDMPduGroupRef		
Description	This is a reference to a PDU Group for which the Deadline Monitoring should be enabled. BswM shall call Com_EnableReceptionDM for each referenced IPduGroup.		
Multiplicity	0..*		
Type	Reference to [ComIPduGroup]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.27 BswMNMControl

SWS Item	**BswM0837_Conf :**
Container Name	BswMNMControl
Description	This container includes all parameters related to enabling and disabling of Network Management communication. Disabling of NM communication can be requested by DCM. Nm_EnableCommunication or Nm_DisableCommunication is called when this action is configured.
Configuration Parameters	

SWS Item	**BswM0838_Conf :**		
Name	BswMNMAction		
Description	This parameter specifies if the communication of the corresponding NM channel should be enabled or disabled.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	BSWM_NM_DISABLE		
	BSWM_NM_ENABLE		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**BswM0839_Conf :**		
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		
Type	Reference to [ComMChannel]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.28 BswMLinScheduleSwitch

SWS Item	**BswM0827_Conf :**
Container Name	BswMLinScheduleSwitch
Description	This container includes all parameters related to a switch of LIN schedule table. LinSM_ScheduleRequest is called when this action is configured.
Configuration Parameters	

SWS Item	**BswM0842_Conf :**		
Name	BswMLinScheduleRef		
Description	This is a reference to the LIN schedule table that the LIN SM shall change to.		
Multiplicity	1		
Type	Reference to [LinIfScheduleTable]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE

	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.29 BswMPduGroupSwitch

SWS Item	**BswM0828_Conf :**
Container Name	BswMPduGroupSwitch
Description	This container includes references to the PDU groups that shall be enabled and disabled.
Configuration Parameters	

SWS Item	**BswM0913_Conf :**		
Name	BswMPduGroupSwitchReinit		
Description	This parameter defines if the the values for the parameters like periodical timer, minimum delay timer etc is retainer or reinitialized during a PDU Group Switch.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**BswM0850_Conf :**		
Name	BswMDisabledPduGroupRef		
Description	This is a reference to a PDU Group that should be disabled. BswM shall call Com_IpduGroupStop for each referenced IPduGroup. BswMDisabledPduGroupRef and BswMEnabledPduGroupRef shall not reference the same IPduGroup in a single action.		
Multiplicity	0..*		
Type	Reference to [ComIPduGroup]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**BswM0849_Conf :**		
Name	BswMEnabledPduGroupRef		
Description	This is a reference to a PDU Group that should be enabled. BswM shall call Com_IpduGroupStart for each referenced IPduGroup. BswMDisabledPduGroupRef and BswMEnabledPduGroupRef shall not reference the same IPduGroup in a single action.		
Multiplicity	0..*		
Type	Reference to [ComIPduGroup]		
ConfigurationClass	***Pre-compile***	X	VARIANT-PRE-COMPILE

	time		
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	dependency: BswMDisabledDMPduGroupRef		

No Included Containers

10.2.30 BswMPduRouterControl

SWS Item	**BswM0853_Conf :**
Container Name	BswMPduRouterControl
Description	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.
Configuration Parameters	

SWS Item	**BswM0923_Conf :**		
Name	BswMDisabledPduRoutingPathGroupRef		
Description	This is a reference to the PDU Routing Path Group for which the routing in the PDU Router should be disabled. BswM shall call PduR_DisableRouting for each referenced PduRRoutingPathGroup.		
Multiplicity	0..*		
Type	Reference to [PduRRoutingPathGroup]		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**BswM0855_Conf :**		
Name	BswMEnabledPduRoutingPathGroupRef		
Description	This is a reference to the PDU Routing Path Group for which the routing in the PDU Router should be enabled. BswM shall call PduR_EnableRouting for each referenced PduRRoutingPathGroup.		
Multiplicity	0..*		
Type	Reference to [PduRRoutingPathGroup]		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.31 BswMUserCallout

SWS Item	**BswM0834_Conf :**

Container Name	BswMUserCallout
Description	This container includes all details needed for a user defined function call.
Configuration Parameters	

SWS Item	**BswM0843_Conf :**		
Name	BswMUserCalloutFunction		
Description	This parameter specifies the complete function call including all parameters. The parameters are specified during configuration time, and cannot be changed during run time. Any return values passed by the callout will be ignored.		
Multiplicity	1		
Type	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.32 BswMRteSwitch

SWS Item	**BswM0803_Conf :**
Container Name	BswMRteSwitch
Description	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. It is expected that the Software Component Description of the BswM contains a PPortPrototype for each instance of this container. The provided port shall be typed by the BswMRteSwitchInterfaceRef. The shortName of the PPortPrototype shall be equal to the shortName of this container.
Configuration Parameters	

SWS Item	**BswM0877_Conf :**		
Name	BswMRteSwitchInterfaceRef		
Description	This is a foreign reference to the SenderReceiverInterface which types the PPortPrototoype that is used for the indication.		
Multiplicity	1		
Type	Foreign reference to [SenderReceiverInterface]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

SWS Item	**BswM0896_Conf :**
Name	BswMSwitchedMode
Description	This parameter contains the integer value that corresponds to a certain mode in a Mode Declaration Group.

Multiplicity	1		
Type	Foreign reference to [ModeDeclaration]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.3 Published Information

[BSWM001_PI] The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].

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