MICROSAR BswM

Technical Reference

Version 1.6.0

Authors	Daniel Hof, Thomas Kuhl, Leticia Garcia Herrera
Status	Released

Document Information

History

Author	Date	Version	Remarks
Daniel Hof	2011-01-14	1.0	Creation
Daniel Hof	2011-05-27	1.1	ESCAN00051215: Added description for Nm_Fiat and some minor modifications
Daniel Hof	2011-08-03	1.2	> Added description of further Det checks > Added descriptions for automatic configuration of recommended Use Cases > Added description of multiple identities configuration > Added Passive Mode description > Added Partial Networks description
Thomas Kuhl	2011-11-03	1.2.1	> ESCAN00054154
Thomas Kuhl	2011-12-06	1.2.2	> ESCAN00055325
Thomas Kuhl	2012-01-10	1.3.0	> Add function description BswM_Dcm_ApplicationUpdated > Update Module Configuration description for Dcm service “Application Updated”
Thomas Kuhl	2012-05-07	1.4.0	> Update Template > Add chapter 5.6.1.2 Require Ports > Add chapter 6.2.4 Timer Configuration > Extend chapter 6.2.5.2.2 Adding Actions to an Action List with Timer and Mode Notification Actions
Thomas Kuhl	2012-10-02	1.5.0	> Extend chapter 6.2.2 with LinSM schedule end notification
Leticia Garcia Herrera	2013-10-12	1.6.0	> Extension of chapter 6.2.4 and modification of chapter 7.7.

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_BSWModeManager.pdf (AUTOSAR Release 3.2)	1.0.0
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	2.2.1
[3]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	3.1.0
[4]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	1.3.0
[5]	Vector	TechnicalReference_Asr_LinSM.pdf	1.12.0
[6]	Vector	TechnicalReference_ Asr_CanSM.pdf	1.15.0
[7]	Vector	TechnicalReference_ Asr_FrSM.pdf	1.11.0
[8]	Vector	TechnicalReference_ Asr_ComM.pdf	3.14.0
[9]	Vector	TechnicalReference_ Asr_LinIf.pdf	2.7.6
[10]	Vector	TechnicalReference_ Asr_PduR.pdf	3.10.0
[11]	Vector	TechnicalReference_ Asr_Dcm_<OEM>.pdf	-
[12]	Vector	TechnicalReference_Asr_Nm.pdf	2.13.0
[13]	Vector	TechnicalReference_Asr_Com.pdf	2.11.0
[14]	Vector	TechnicalReference_Asr_NmFiatB.pdf	1.0.0
[15]	Vector	TechnicalReference_Asr_NmFiatC.pdf	1.02.0

Scope of the Document

This technical reference describes the general use of the AUTOSAR Basic Software module BSW Mode Manager (BswM).

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.0	Initial version
1.1	Added support of NmFiatB
1.2	> Added automatic recommended configuration > References to other Rules can be used as an Action within an Action List > Action List Execution can be aborted if an Action fails > Added Dcm Passive Mode > Added support of multiple identities configurations > Added Mode LINSM_CURRENT_SCHEDULE > Added ComM Partial Network support
1.3	> Add BswM_Dcm_ApplicationUpdated functionality
1.4	> Add Timer handling
1.5	> Remove DCM_RESET_EXECUTION from the Dcm_ResetModeType
1.6	> Version not used
1.7	> Add Support for LinSM schedule end notification > Adapt LinTp_Modes to AUTOSAR specification
1.8	> BswMNmClassCModeRequest supported as mode request
1.9	> Implement AMD time measurement support
1.10	> Support mode notifications from NMOsek

Table 1-1 Component history

2 Introduction

Cross reference
The mandatory BSWM use cases and how to configure them are described in chapter 7. Chapter 7 is also helpful to get a basic understanding on what the BswM is doing.

This document describes the functionality, API and configuration of the AUTOSAR BSW module BswM as specified in [1].

Supported AUTOSAR Release*:	3	
Supported Configuration Variants:	pre-compile, link-time, post-build	
Vendor ID:	BswM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	BswM_MODULE_ID	042 decimal (according to ref. [4])

* For the precise AUTOSAR Release 3.x please see the release specific documentation.

The BSW Mode Manager is the module that implements the part of the Vehicle Mode Management and Application Mode Management concept that resides in the BSW. Its task is to arbitrate mode requests from BSW modules or application layer SWCs based on simple rules, and perform actions based on the arbitration result.

2.1 Architecture Overview

The following figure shows where the BswM is located in the AUTOSAR architecture.

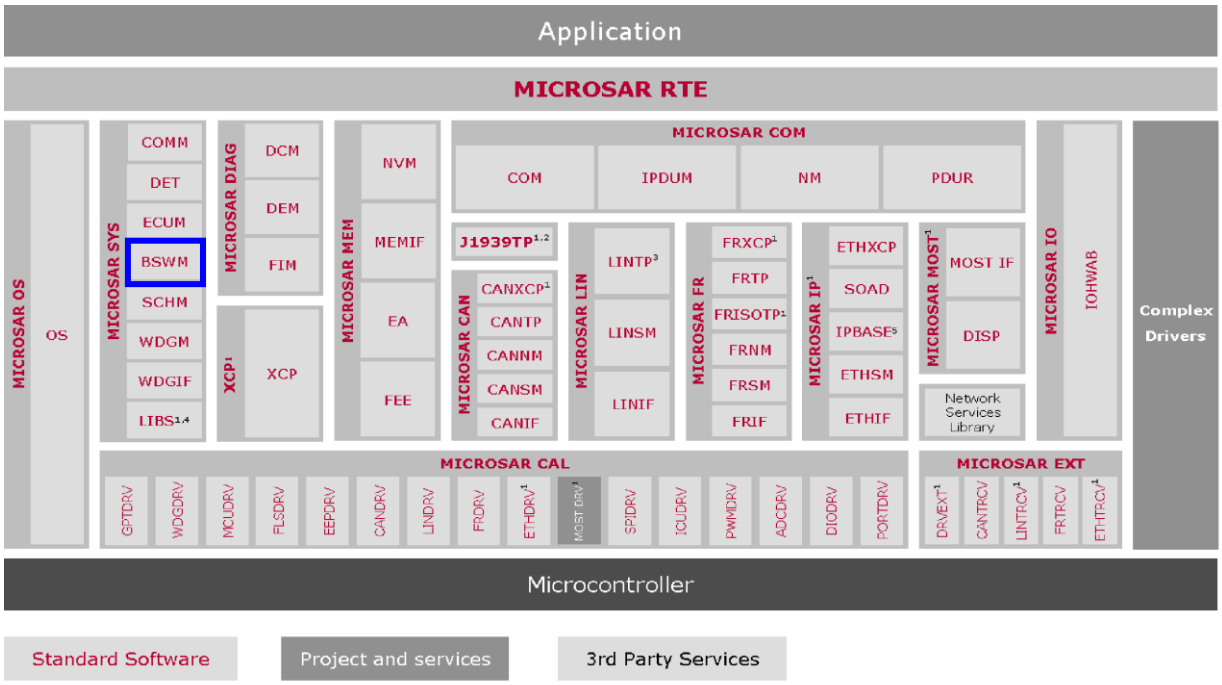

Figure 2-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the BswM. These interfaces are described in chapter 5.

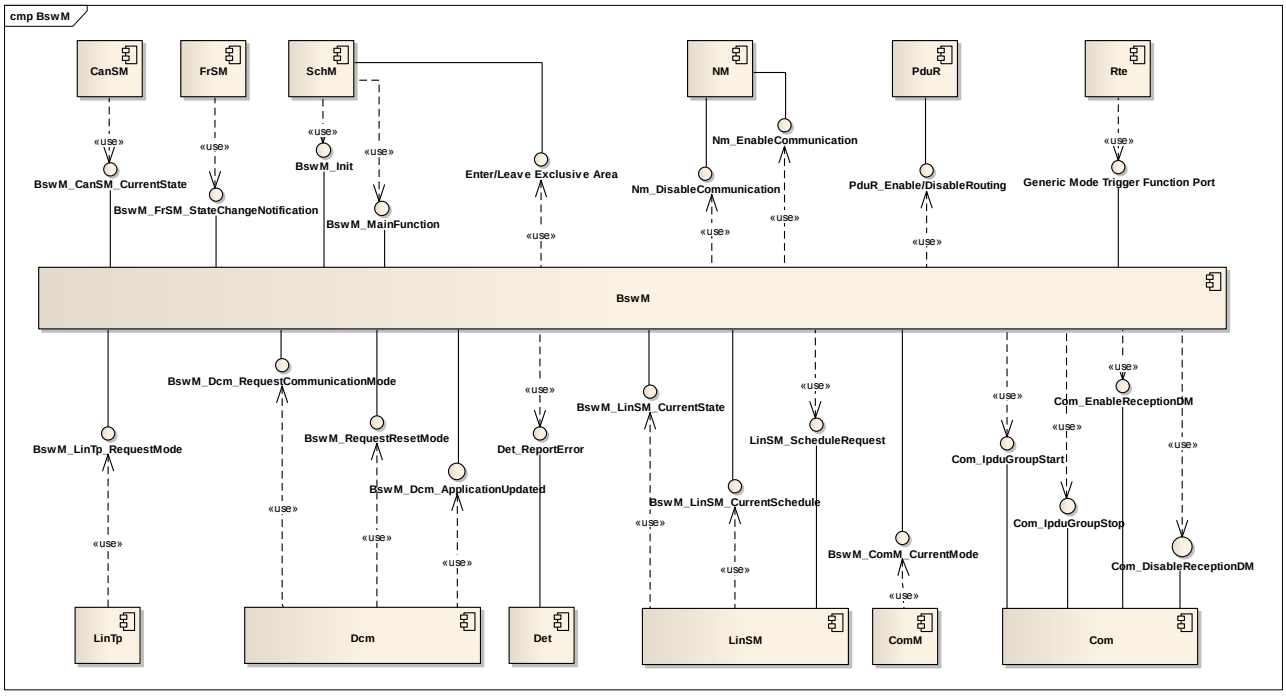

Figure 2-2 Interfaces to adjacent modules of the BswM

If a RTE is used applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. The service ports provided by the BswM are listed in chapter 5.6 and are defined in [1].

3 Functional Description

This chapter describes the general function of the BswM.

3.1 Features

The features listed in the following tables cover the complete functionality specified for the BswM.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

For further information of not supported features see also chapter 7.

Vector Informatik provides further BswM functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Mode arbitration and mode control
Configuration of rules, actions and conditions

Table 3-1 Supported AUTOSAR standard conform features

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
Cascaded rule conditions

Table 3-2 Not supported AUTOSAR standard conform features

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
-

Table 3-3 Features provided beyond the AUTOSAR standard

3.2 Initialization

The BswM is initialized via the service function `BswM_Init`(refer to chapter 5.2.2). All available modes are set to the configured initialization state, which can either be undefined or set to a specific value. If the initialization state is undefined the mode is not arbitrated until the mode request/indication function occurs for the first time. If multiple identities are

used the BswM must be initialized with the current active identity which stored into the configuration pointer. For further information refer to chapter 6.

3.3 Mode Management

The BswM manages user defined modes. A mode consists of the following parts:

> **Mode Source**: this is the trigger for the mode arbitration, a trigger can either be a SW-C indication/request function or a BSW indication/request function or the `BswM_MainFunction()`.

> **Mode Arbitration**: when the mode source trigger occurs the BswM will arbitrate a mode specific rule either immediately or deferred within the `BswM_MainFunction()`. The mode arbitration types are described in detail in chapters 3.3.1 and 3.3.2.

> **Mode Rule**: a rule is a logical expression which consists of specific conditions which use different operators. The rule is arbitrated by the BswM to be either true or false. Dependent on the evaluation result the BswM executes the configured mode action(s) (true-action(s) or false-action(s)).

> **Mode Actions**: these are either BSW service function calls or user callout function calls which are executed by the BswM after the Mode Arbitration.

3.3.1 Immediate Mode Handling

The immediate mode arbitration is done directly upon the mode request/indication function. If another mode request/indication occurs during mode arbitration the BswM queues this mode arbitration request. The mode request queue is emptied when the current mode arbitration is finished. The following sequence diagram shows this procedure:

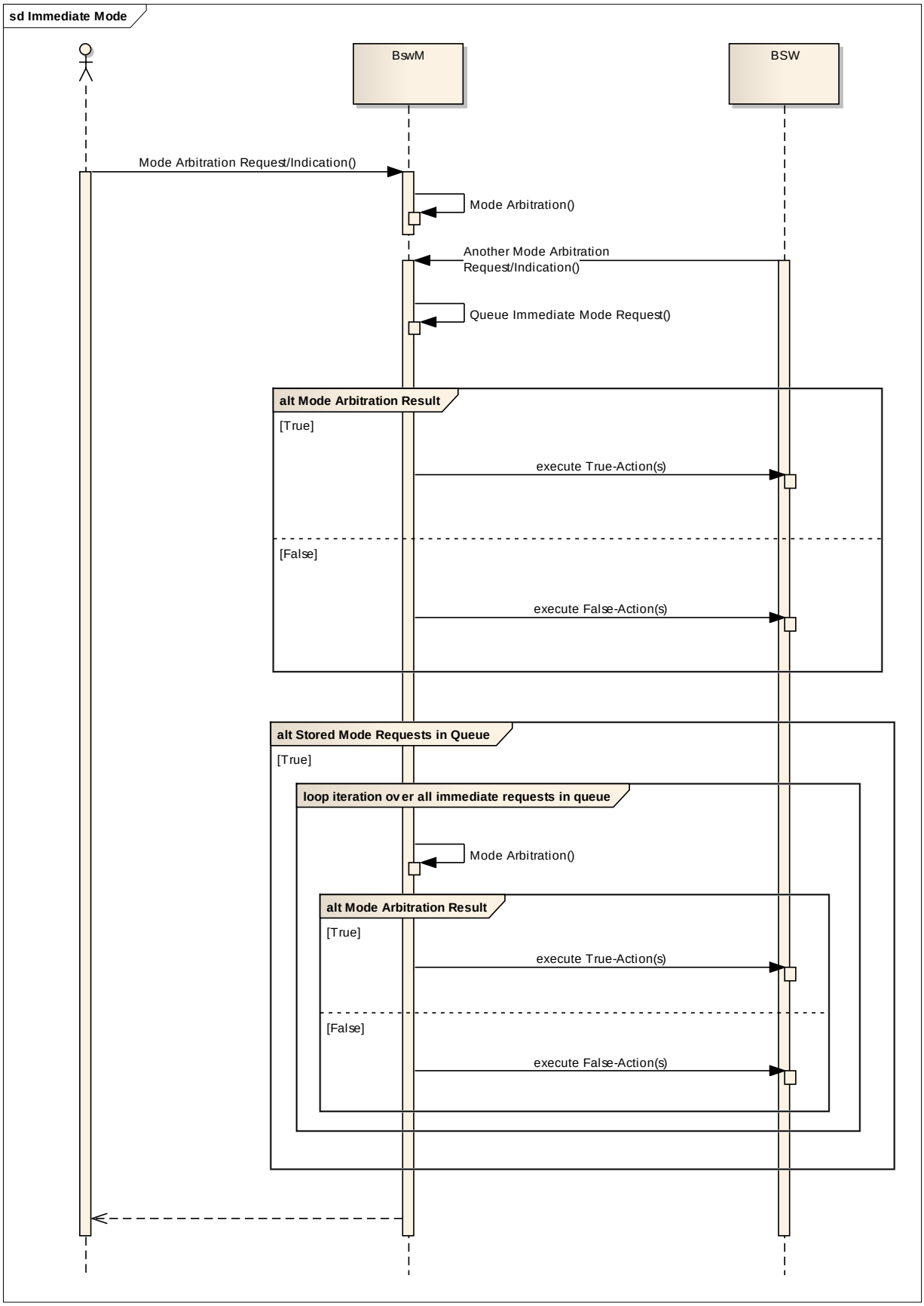

Figure 3-1 Sequence Immediate Mode

3.3.2 Deferred Mode Handling

The deferred mode arbitration is done cyclically within the execution of the `BswM_MainFunction()`. If another mode request/indication occurs during mode arbitration the BswM queues this mode arbitration request. The mode request queue is emptied at the end of the `BswM_MainFunction()`. The following sequence diagram shows this procedure:

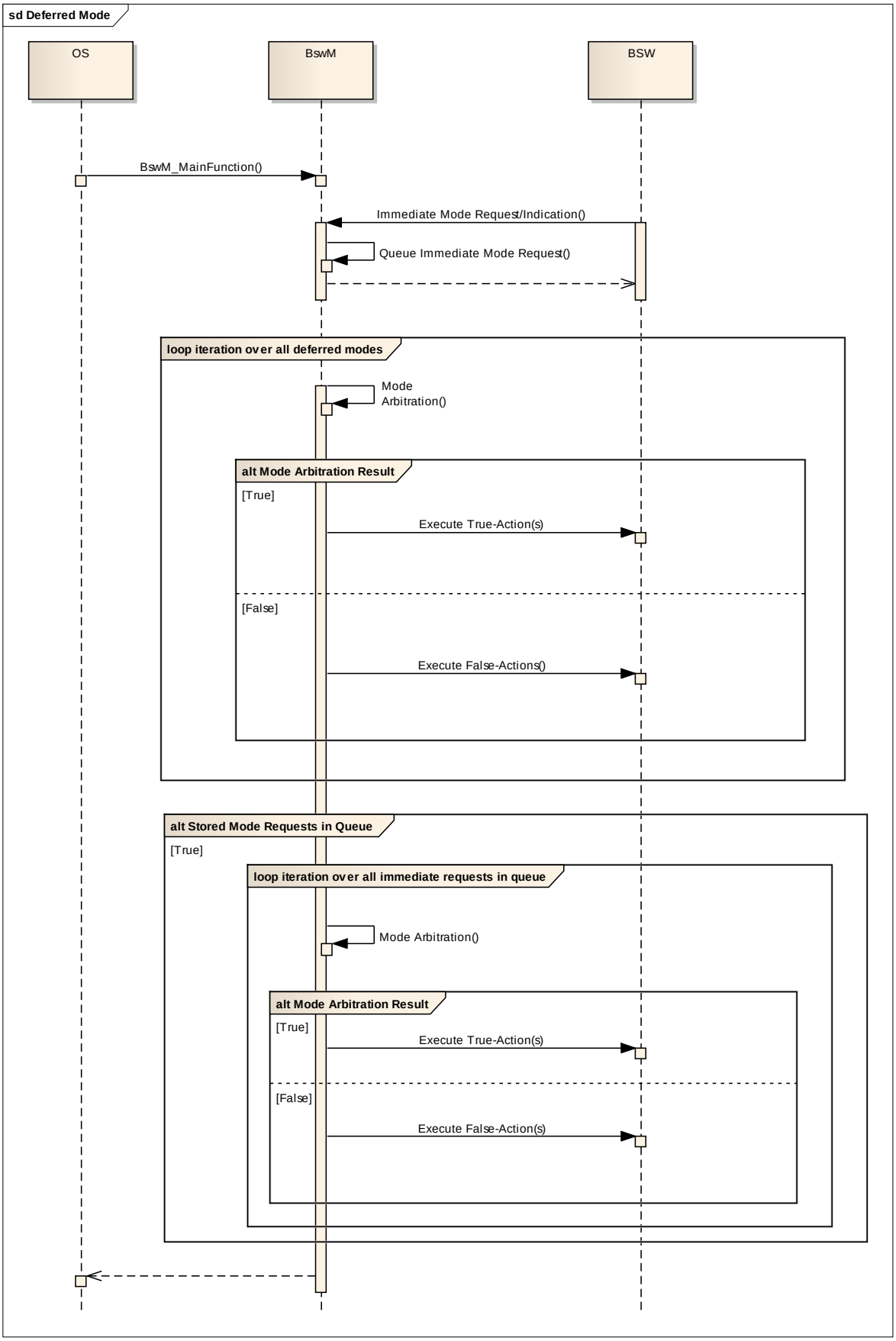

Figure 3-2 Sequence Deferred Mode

3.4 Execution of action lists

The execution of actions is done after the rule arbitration dependent on the result (true or false). 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 triggered and conditional execution. This execution type is defined via configuration, refer to chapter 6.

3.5 States

The following diagram shows the general state handling of the BswM:

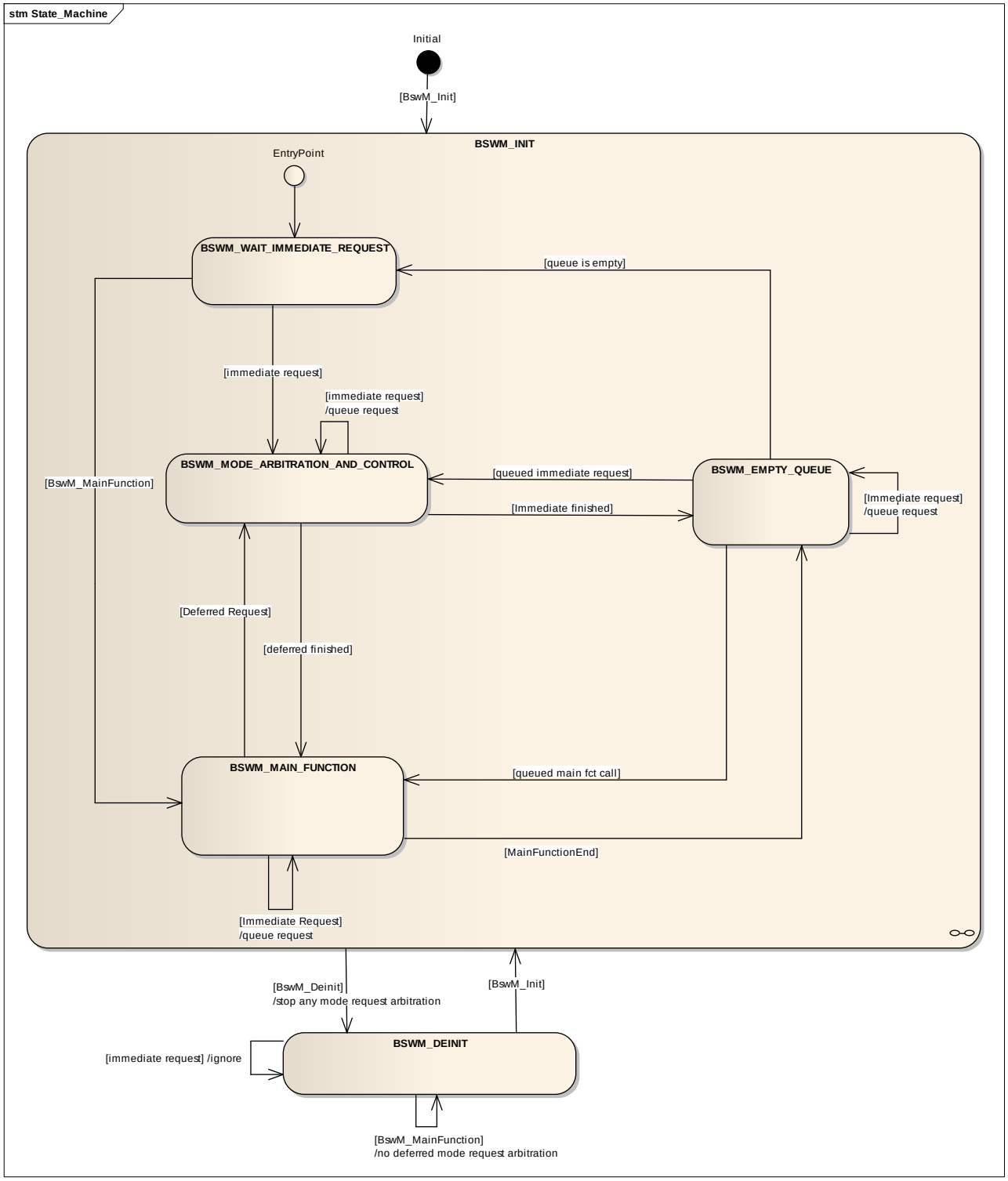

Figure 3-3 States of the BswM

> **BSWM_INIT**

The BswM is initialized and ready for immediate mode arbitration requests. Deferred mode arbitration is done within the cyclic function `BswM_MainFunction()`.

> **BSWM_WAIT_IMMEDIATE_REQUEST**

In this state the BswM waits for a mode arbitration request. The state is left if immediate mode arbitration is requested or when `BswM_MainFunction()` is called.

> **BSWM_MAIN_FUNCTION**

This state is entered when the `BswM_MainFunction()` is called. Within `BswM_MainFunction()` the deferred mode arbitration is done. Immediate mode arbitration requests which occur during the execution of `BswM_MainFunction()` are queued and will be executed at the end of `BswM_MainFunction()` when all deferred mode arbitration and control is finished.

> **BSWM_MODE_ARBITRATION_AND_CONTROL**

In this state the configured mode rule arbitration is done and the true-/false-action lists are executed. New mode arbitration requests are queued.

> **BSWM_EMPTY_QUEUE**

In this state the queued mode arbitration requests are executed.

> **BSWM_DEINIT**

This state is entered when the function `BswM_Deinit()` is called. No mode arbitration requests are accepted and no mode processing is done. This state can only be left when function `BswM_Init()` is called.

3.6 Error Handling

3.6.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `BSWM_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported BswM ID is 042.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
BSWM_INIT_ID (0x00)	BswM_Init()
BSWM_GETVERSIONINFO_ID (0x01)	BswM_GetVersionInfo()
BSWM_REQUESTMODE_ID (0x02)	BswM_RequestMode()
BSWM_MAINFUNCTION_ID (0x03)	BswM_MainFunction()

Service ID	Service
BSWM_DEINIT_ID (0x04)	BswM_Deinit()
BSWM_CANSM_CURRENTSTATE_ID (0x05)	BswM_CanSM_CurrentState()
BSWM_DCM_REQUESTCOMMUNICATIONMODE_ID (0x06)	BswM_Dcm_RequestCommunicationMode()
BSWM_DCM_REQUESTRESETMODE_ID (0x07)	BswM_Dcm_RequestResetMode()
BSWM_DCM_SET_PASSIVE_MODE_ID (0x10)	BswM_Dcm_SetPassiveMode()
BSWM_LINSM_CURRENTSTATE_ID (0x09)	BswM_LinSM_CurrentState()
BSWM_LINSM_CURRENTSCHEDULE_ID (0x0A)	BswM_LinSM_CurrentSchedule()
BSWM_LINTP_REQUESTMODE_ID (0x0B)	BswM_LinTp_RequestMode()
BSWM_FRSM_CURRENTSTATE_ID (0x0C)	BswM_FrSM_StateChangeNotification()
BSWM_COMM_CURRENTMODE_ID (0x0E)	BswM_ComM_CurrentMode()
BSWM_COMM_CURRENT_PNC_MODE_ID (0x15)	BswM_ComM_CurrentPNCMode()
BSWM_INITMEMORY_ID (0x80)	BswM_InitMemory()
BSWM_NM_STATE_CHANGE_ID (0x0F)	BswM_Nm_StateChangeNotification()
BSWM_DCM_APPLICATION_UPDATED_ID (0x14)	BswM_Dcm_ApplicationUpdated()

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x01	BSWM_E_NO_INIT	Service function is called while BswM is not initialized.
0x02	BSWM_E_NULL_POINTER	Service function is called with a null pointer as an argument.
0x04	BSWM_E_REQ_USER_OUT_OF_RANGE	A requesting user is out of range.
0x05	BSWM_E_REQ_MODE_OUT_OF_RANGE	A requested mode is out of range.

Table 3-5 Errors reported to DET

3.6.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled separately. The configuration of en-/disabling the checks is described in chapter 6.2.1. En-/disabling of single checks is an addition to the AUTOSAR standard which requires to en-/disable the complete parameter checking via the parameter `BSWM_DEV_ERROR_DETECT`.

The following table shows which parameter checks are performed on which services:

Service \ Check	BSWM_E_NO_INIT	BSWM_E_NULL_POINTER	BSWM_E_REQ_USER_OUT_OF_RANGE	BSWM_E_REQ_MODE_OUT_OF_RANGE
BswM_Init		■		
BswM_GetVersionInfo		■		
BswM_RequestMode	■			
BswM_MainFunction	■			
BswM_Deinit				
BswM_CanSM_CurrentState	■		■	■
BswM_Dcm_RequestCommunicationMode	■		■	■
BswM_Dcm_RequestResetMode	■			■
BswM_Dcm_SetPassiveMode	■			■
BswM_LinSM_CurrentState	■		■	■
BswM_LinSM_CurrentSchedule	■		■	■
BswM_LinSM_ScheduleEnd_Notification	■			
BswM_LinTp_RequestMode	■		■	■
BswM_FrSM_StateChangeNotification	■		■	■
BswM_ComM_CurrentMode	■		■	■
BswM_ComM_CurrentPNCMode	■			■
BswM_Nm_StateChangeNotification	■		■	■
BswM_Dcm_ApplicationUpdated	■			
BswM_InitMemory				

Table 3-6 Development Error Reporting: Assignment of checks to services

3.6.2 Production Code Error Reporting

Currently the BswM does not support any production error detection and reporting.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR BswM into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the BswM contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
BswM.c	■		This is the source file of the BswM. It contains the initialization function, the deinitialization function, the cyclic main function and all the BSW mode indication functions.
BswM.h	■		This is the header file of the BswM. It contains the interfaces to the BswM API functions.
BswM_CanSM.h	■		This header file contains the prototypes of the callback functions of the CAN State Manager.
BswM_ComM.h	■		This header file contains the prototypes of the callback functions of the Communication Manager.
BswM_Dcm.h	■		This header file contains the prototypes of the callback functions of the Diagnostic Communication Manager.
BswM_FrSM.h	■		This header file contains the prototypes of the callback functions of the FlexRay State Manager.
BswM_LinSM.h	■		This header file contains the prototypes of the callback functions of the LIN State Manager.
BswM_Nm.h	■		This header file contains the prototypes of the callback functions of the Network Management.

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool GENy.

File Name	Description
BswM_Lcfg.c	This file contains the link time configuration parameters.
BswM_Cfg.h	This header file contains precompile time configuration parameters.

Table 4-2 Generated files

4.2 Include Structure

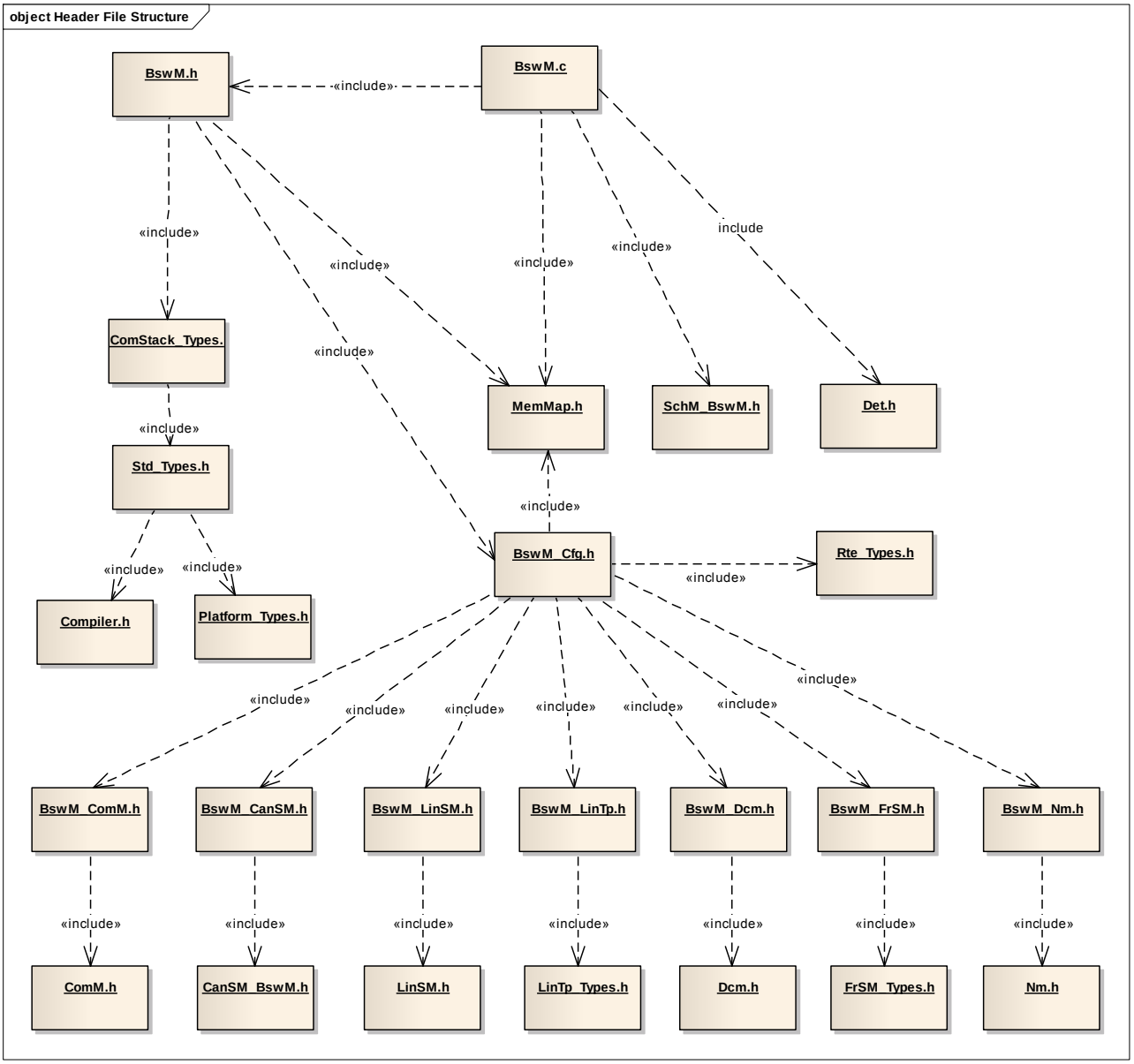

Figure 4-1 Include structure

4.3 Critical Sections

The BswM has code sections which must not be interrupted by incoming mode requests. Therefore the BswM uses one exclusive area:

BSWM_EXCLUSIVE_AREA_0

This area requires a global interrupt lock and it must be ensured that the main functions of the BSW modules which mode indication functions are used cannot interrupt each other.

4.4 Cyclic Task

The BswM has one cyclic main function `BswM_MainFunction()` which must be called cyclically. The cyclic time is up to the user but must be considered for deferred mode handling.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the BswM are described in this chapter.

Type Name	C-Type	Description	Value Range
BswM_ModeType	uint8 uint16	The range of this type depends on the configured number of modes.	`0 … 255` Used if the total number of modes is less than or equal to 255.
			`0 … 65535` Used if the total number of modes is greater than 255.
BswM_UserType	uint8 uint16	The range of this type depends on the number of user.	`0 … 255` Used if the total number of users is less than or equal to 255.
			`0 … 65535` Used if the total number of users is greater than 255.
BswM_ConfigType	uint8	Type for the configuration pointer parameter for `BswM_Init()`.	Currently the configuration pointer is only used for multiple identities configurations.

Table 5-1 Type definitions

5.2 Services provided by BswM

5.2.1 BswM_RequestMode

Prototype	
`Std_ReturnType BswM_RequestMode(BswM_UserType requesting_user, BswM_ModeType requested_mode)`	
Parameter	
`requesting_user`	Index of the user that requests the mode.
`requested_mode`	Requested mode.
Return code	
`E_OK`	Request is valid and accepted by the BswM.
`E_NOT_OK`	Request is invalid and not accepted by the BswM.
Functional Description	
General function to request modes.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is reentrant. > This function is only allowed to be used by the BswM itself, applications or SWCs must not use this function.	
Expected Caller Context	
> This function must not be called from interrupt context.	

Table 5-2 BswM_RequestMode

5.2.2 BswM_Init

Prototype	
`void BswM_Init(BswM_ConfigType ConfigPtr)`	
Parameter	
`ConfigPtr`	Pointer is only used in multiple identities configurations: ConfigPtr must contain the identity information, refer to the example below.
Return code	
–	
Functional Description	
This function initializes the BswM. All configured modes are set to the configured initial value.	
Particularities and Limitations	
> Service ID: see table 'Service IDs'. > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> This function must not be called from interrupt context.	

Table 5-3 BswM_Init

Example

Here is an example which describes the usage of BswM_Init() in a multiple identities use-case:

The include of BswM.h contains the indexes of the available identities which have the naming convention BswM_<identity-name> and are of type uint8, example:

```
CONST(uint8, BSWM_CONST) BswM_ID_1;
CONST(uint8, BSWM_CONST) BswM_ID_2;
```

Example code:

```
...
#include "BswM.h"
...
#if (BSWM_IDENTITY_MANAGER_CONFIG == STD_ON)
  BswM_Init(&BswM_ID_1);
#else
  BswM_Init(((void*)0));
#endif
```

5.2.3 BswM_InitMemory

Prototype	
`void BswM_InitMemory(void)`	
Parameter	
–	
Return code	
–	
Functional Description	
This function sets the BswM into an uninitialized state. This function must only be called if INIT variables are not initialized by the startup code.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' . > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> This function must not be called in interrupt context.	

Table 5-4 BswM_InitMemory

5.2.4 BswM_Deinit

Prototype	
`void BswM_Deinit (void)`	
Parameter	
–	
Return code	
–	
Functional Description	
This function sets the BswM into the BSWM_DEINIT state. All pending requests are cleared and no further mode requests are accepted by the BswM. This state can only be left by calling the function `BswM_Init()`.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> This function must not be called in interrupt context.	

Table 5-5 BswM_Deinit

5.2.5 BswM_GetVersionInfo

Prototype	
`void BswM_GetVersionInfo(Std_VersionInfoType* VersionInfo)`	
Parameter	
`VersionInfo`	Pointer to the address where the BswM version info shall be copied to.
Return code	
–	
Functional Description	
This function returns the version information of the BswM.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-6 BswM_GetVersionInfo

5.2.6 BswM_ComM_CurrentMode

Prototype	

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

Parameter	
`Network`	Index of the network.
`RequestedMode`	Current communication mode of the ComM.
Return code	
–	
Functional Description	
This function is called by the ComM to notify the BswM about the current communication mode of a network.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > Must only be called by the ComM.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-7 BswM_ComM_CurrentMode

5.2.7 BswM_ComM_CurrentPNCMode

Prototype	

`void BswM_ComM_CurrentPNCMode(`
`  PNCHandleType Pnc,`
`  ComM_PncModeType RequestedMode`
`)`	

Parameter	
`Pnc`	Global index of the Pnc.
`RequestedMode`	Current mode of the Pnc.
Return code	
–	
Functional Description	
This function is called by the ComM to notify the BswM about the current mode of a PNC.	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > Must only be called by the ComM.
Expected Caller Context
> This function may be called from task and interrupt context.

Table 5-8 BswM_ComM_CurrentPNCMode

5.2.8 BswM_CanSM_CurrentState

Prototype	

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

Parameter	
`Network`	Index of the network.
`CurrentState`	Current state of the CanSM.
Return code	
–	
Functional Description	
This function is called by the CanSM to notify the BswM about the current state of a specific CanSM network.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > Must only be called by the CanSM.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-9 BswM_CanSM_CurrentState

5.2.9 BswM_FrSM_StateChangeNotification

Prototype	

`void BswM_FrSM_StateChangeNotification (`
`  NetworkHandleType Network,`
`  FrSM_BswM_StateType PreviousState,`
`  FrSM_BswM_StateType CurrentState`
`)`	

Parameter	
`Network`	Index of the network.
`PreviousState`	Previous state of the FrSM
`CurrentState`	Current state of the FrSM.
Return code	
-	
Functional Description	
This function is called by the FrSM to notify the BswM about the current state of a specific FrSM network.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > Must only be called by the FrSM.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-10 BswM_FrSM_StateChangeNotification

5.2.10 BswM_LinSM_CurrentState

Prototype	

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

Parameter	
`Network`	Index of the network.
`CurrentState`	Current state of the LinSM.
Return code	
–	
Functional Description	
This function is called by the LinSM to notify the BswM about the current state of a specific LinSM network.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > Must only be called by the LinSM.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-11 BswM_LinSM_CurrentState

5.2.11 BswM_LinSM_CurrentSchedule

Prototype	

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

Parameter	
`Network`	Index of the network where the LIN schedule was changed.
`CurrentSchedule`	Index of the current active schedule table.
Return code	
–	
Functional Description	
This function is used by the LinSM to notify the BswM about a schedule change on a specific LIN network.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the LinSM.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-12 BswM_LinSM_CurrentSchedule

5.2.12 BswM_LinSM_ScheduleEnd_Notification

Prototype	

`void BswM_LinSM_SchedulEnd_Notification(`
`  NetworkHandleType Network,`
`  LinIf_SchHandleType Schedule`
`)`	

Parameter	
`Network`	Index of the network.
`Schedule`	Index of the schedule table.
Return code	
-	
Functional Description	
This function is used by the LinSM to notify the BswM when the last frame of a schedule table was transmitted.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the LinSM.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-13 BswM_LinSM_ScheduleEnd_Notification

5.2.13 BswM_LinTp_RequestMode

Prototype	

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

Parameter	
`Network`	Index of the network the LinTp request is related to.
`LinTpRequestedMode`	Requested LinTp Mode.
Return code	
-	
Functional 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.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the LinTp.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-14 BswM_LinTp_RequestMode

5.2.14 BswM_Nm_StateChangeNotification

Prototype	

`void BswM_Nm_StateChangeNotification(`
`  NetworkHandleType nmChannelHandle,`
`  Nm_StateType nmPreviousState,`
`  Nm_StateType nmCurrentState`
`)`	

Parameter	
`nmChannelHandle`	Index of the network.
`nmPreviousState`	Contains the previous state of the Nm.
`nmCurrentState`	Contains the current state of the Nm.
Return code	
-	
Functional Description	
Function called by the Nm to notify the BswM about a state change. A detailed description of the NM states can be found in [14].	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the Nm.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-15 BswM_Nm_StateChangeNotification

5.2.15 BswM_Dcm_RequestCommunicationMode

Prototype	

`void BswM_Dcm_RequestCommunicationMode(`
`  NetworkHandleType Network,`
`  Dcm_CommunicationModeType RequestedMode`
`)`	

Parameter	
`Network`	Index of the network.
`RequestedMode`	Contains the requested communication mode.
Return code	
-	
Functional Description	
Function called by the Dcm to notify the BswM about a specific communication mode.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the Dcm.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-16 BswM_Dcm_RequestCommunicationMode

5.2.16 BswM_Dcm_RequestResetMode

Prototype	

`void BswM_Dcm_RequestResetMode(`
`  Dcm_ResetModeType RequestedMode`
`)`	

Parameter	
`RequestedMode`	Contains the requested communication mode.
Return code	
-	
Functional Description	
Function called by the Dcm to notify the BswM about a specific reset mode.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the Dcm.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-17 BswM_Dcm_RequestCommunicationMode

5.2.17 BswM_Dcm_SetPassiveMode

Prototype	

`void BswM_Dcm_SetPassiveMode(`
`  boolean mode`
`)`	

Parameter	
`mode`	Contains the state of the passive mode (on/off).
Return code	
–	
Functional Description	
Function called by the Dcm to notify the BswM about the state of the passive mode. The BswM routes this request directly to the CAN state manager (CanSM) and/or to the FlexRay state manager (FrSM).	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the Dcm.	
Expected Caller Context	
> This function may be called from task and interrupt context.	

Table 5-18 BswM_Dcm_RequestCommunicationMode

5.2.18 BswM_Dcm_ApplicationUpdated

Prototype	
`void BswM_Dcm_ApplicationUpdated ( )`	
Parameter	
–	
Return code	
–	
Functional Description	
Function called by the Dcm to notify the BswM about an application is updated after flash process. The BswM stores this event and informs the application about the update by calling Appl_BswM_ApplicationUpdated() or by using the application update SWC service in the next BswM_MainFunction() call.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is asynchronous. > This function is reentrant. > The functionality must be enabled inside the BswM (“Dcm Application Updated” = TRUE) > This function must only be called by the Dcm.	

Expected Caller Context
> This function may be called from task and interrupt context.

Table 5-19 BswM_Dcm_ApplicationUpdated

5.3 Services used by BswM

In the following table services provided by other components, which are used by the BswM are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError
LinSM	LinSM_ScheduleRequest
CanSM	CanSM_SetEcuPassive
FrSM	FrSM_SetEcuPassive
Nm	Nm_EnableCommunication
Nm	Nm_DisableCommunication
Com	Com_IpduGroupStart
Com	Com_IpduGroupStop
Com	Com_EnableReceptionDM
Com	Com_DisableReceptionDM
PduR	PduR_EnableRouting
PduR	PduR_DisableRouting

Table 5-20 Services used by the BswM

5.4 Callback Functions

5.4.1 Appl_BswM_ApplicationUpdated

Prototype	
`void Appl_BswM_ApplicationUpdated ( )`	
Parameter	
–	
Return code	
–	
Functional Description	
Function called by the BswM to notify the application software about an application updated after flash process.	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > This function must only be called by the BswM. > This function is only available it is enabled in the BswM configuration and the configuration item “Provide as RteMode Switch Notification” is disabled
Expected Caller Context
> This function is called from task (BswM_MainFunction()).

Table 5-21 Appl_BswM_ApplicationUpdated

5.5 Configurable Interfaces

5.5.1 Callout Functions

A User Callout Function can be used as an item of an Action List (refer to chapter 6.2.5.2). The integrator must provide an extern declaration of the function via an application header file (refer to chapter 6.2.1). The BswM callout function declaration is described in the following table:

Prototype	
`void [Callout Function Name] ( void )`	
Parameter	
–	-
Return code	
–	-
Functional Description	
If a User Callout is configured as an item of an Action List the BswM calls this function in the context of the appropriate rule.	
Particularities and Limitations	
> -	
Call context	
> Interrupt or task context, depends on the mode/rule configuration in which the callout is used.	

Table 5-22 User Callout

5.5.2 Mode Trigger Functions

If Generic Modes are used the BswM provides a function which allows the application to indicate a specific mode state. The following table describes the mode trigger function:

Prototype
`void [Generic Mode Trigger Function Name] ( uint8 reqMode )`

Parameter	
`reqMode`	Contains the requested Mode State. Possible values are the configured Mode State Values.
Return code	
–	-
Functional Description	
If a User Callout is configured as an item of an Action List the BswM calls this function in the context of the appropriate rule.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant.	
Call context	
> Interrupt or task context.	

Table 5-23 Mode Trigger Function Prototype

5.5.3 User Condition Functions

A User Condition Function can be used in a Rule Condition (refer to chapter 6.2.5.1). The integrator must provide an extern declaration of the function via an application header file (refer to chapter 6.2.1). The BswM User Condition Function declaration is described in the following table:

Prototype	
`boolean` **`[User Condition Function Name]`** `( void )`	
Parameter	
–	-
Return code	
`boolean`	The return value must be boolean true or false.
Functional Description	
If a User Condition is configured as a condition in a Rule the BswM calls this function during the Rule Arbitration. The function returns true or false which then has effect on the Rule Arbitration result.	
Particularities and Limitations	
> -	
Call context	
> Interrupt or task context, depends on the mode/rule configuration in which the User Condition is used.	

Table 5-24 User Condition

5.6 Service Ports

5.6.1 Client Server Interface

A client server interface is related to a Provide Port at the server side and a Require Port at client side.

5.6.1.1 Provide Ports on BswM Side

At the Provide Ports of the BswM the API functions described in 5.2 are available as Runnable Entities. The Runnable Entities are invoked via Operations. The mapping from a SWC client call to an Operation is performed by the RTE. In this mapping the RTE adds Port Defined Argument Values to the client call of the SWC, if configured.

The following sub-chapters present the Provide Ports defined for the BswM and the Operations defined for the Provide Ports, the API functions related to the Operations and the Port Defined Argument Values to be added by the RTE.

5.6.1.1.1 BswM_ModeIndication_<GenericModeName>

Operation	API Function	Port Defined Argument Values
<GenericModeName>_<GenericModeTriggerFunctionName> (name depends on the configuration, see chapter 6.2.3)	<Generic Mode Trigger Function>	-

Table 5-25 Provide Port

5.6.1.1.2 BswM_DcmAppUpdate

Operation	API Function	Port Defined Argument Values
BswM_DcmAppUpdate	BswM_ApplicationUpdated	-

Table 5-26 BswM_DcmAppUpdate

5.6.1.2 Require Ports

At its Require Ports the BswM calls Operations. These Operations have to be provided by the SWCs by means of Runnable Entities. These Runnable Entities implement the callback functions expected by the BswM.

5.6.1.2.1 Mode Switch Port

Operation	Rte Interface	Mode Declaration Group
currentMode	Rte_Switch_BswM_ModeNotification_<GenericModeName>_currentMode	■ RTE_MODE_BSWM_<GenericModeName>_Mode_BSWM_<ModeStateName>

Table 5-27 Mode Switch Port

6 Configuration

This chapter describes the configuration of the BswM in GENy.

6.1 Configuration Variants

The BswM supports only the configuration variant `VARIANT-PRE-COMPILE`.

The configuration classes of the BswM parameters depend on the supported configuration variants. For their definitions please see the `BswM_bswmd.arxml` file.

6.2 Configuration with GENy

The BswM is configured with the help of the configuration tool GENy. To activate the configuration view the BswM has to be enabled in the component selection:

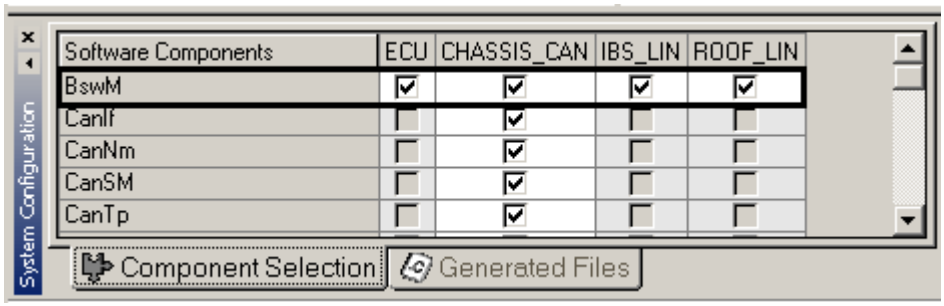

Figure 6-1 GENy Component Selection: enable BswM

The BswM is automatically enabled on all ECU channels.

6.2.1 General Configuration Options

This chapter describes the general configuration options of the BswM:

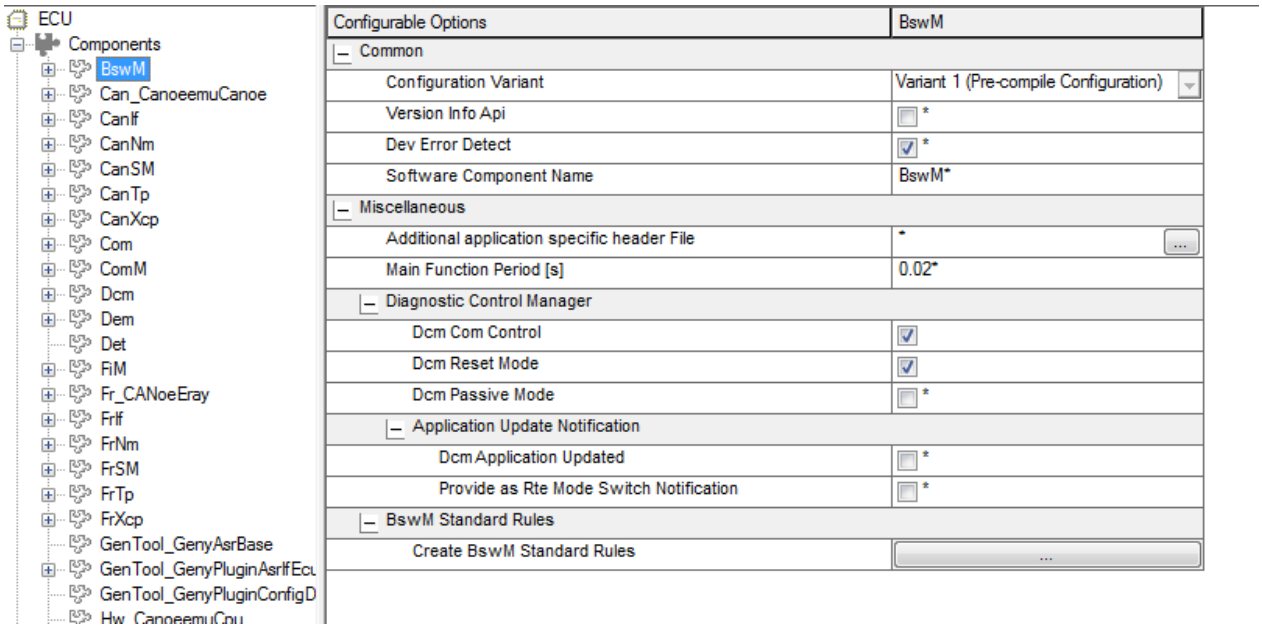

Figure 6-2 General Configuration Options

The following table describes the configuration options:

Attribute Name	Value Type	Values Default value is typed bold	Description
Configuration Variant	Enum	**Variant 1** **(Pre-compile Configuration)**	Currently only the pre-compile configuration variant is supported.
Version Info Api	Boolean	True **False**	En-/ disables the function `BswM_GetVersionInfo()` to get the major, minor and patch version information.
Dev Error Detect	Boolean	True **False**	If **Dev Error Detect** is enabled, all development errors are reported to the Development Error Tracer (DET), refer to chapter 3.6.
Software Component Name	String	**BswM**	This name is used as component name (ShortName) in the generated software component template. Change this name if you try to import the BswM components of several ECUs and have problems with name clashes.
Additional application header file	String	-	Specify the path to a header file which contains the prototypes for the user condition functions and user callout functions, if required.
Main Function Period	Float	**0.02**	This parameter configures the cycle time of the Communication Manger main function `BswM_MainFunction (in seconds)`.
Dcm Com Control	Boolean	True **False**	En-/ disables the Dcm communication control modes of the BswM. If enabled it must be ensured that the Dcm component is available and enabled and that the Dcm supports the BswM communication control modes, refer to [11] and Table 6-2.
Dcm Reset Mode	Booelan	True **False**	En-/ disables the Dcm reset modes of the BswM. If enabled it must be ensured that the Dcm component is available and enabled and that the Dcm supports the BswM reset modes, refer to [11] and Table 6-2.
Dcm Passive Mode	Boolean	True **False**	En-/ disables the Dcm passive modes of the BswM. If enabled it must be ensured that the Dcm component is available and enabled and that the Dcm supports the BswM passive modes, refer to [11] and Table 6-2.
Dcm Application Updated	Boolean	True **False**	En-/ disables the Dcm Application Updated functionality of the BswM. If enabled it must be ensured that the Dcm component is available and enabled and that the Dcm supports the Application Updated Service, refer to [11].
Provide as Rte Mode Switch Notification	Boolean	True **False**	The Dcm Application Updated notification to of the BswM is provided as Mode Port and can be used in software components.

Attribute Name	Value Type	Values Default value is typed bold	Description
Create BswM Standard Rules	-	-	Creates automatically the following recommended BswM Standard Rules: - CAN Communication Modes - Dcm Communication Control Modes - FlexRay Communication Modes - NmFiatB and NmFiatC Communication Modes - NmFiatB and NmFiatC Communication Modes

Table 6-1 General Configuration Options

6.2.2 BSW Mode Configuration

The following figure shows an example view of BSW Modes:

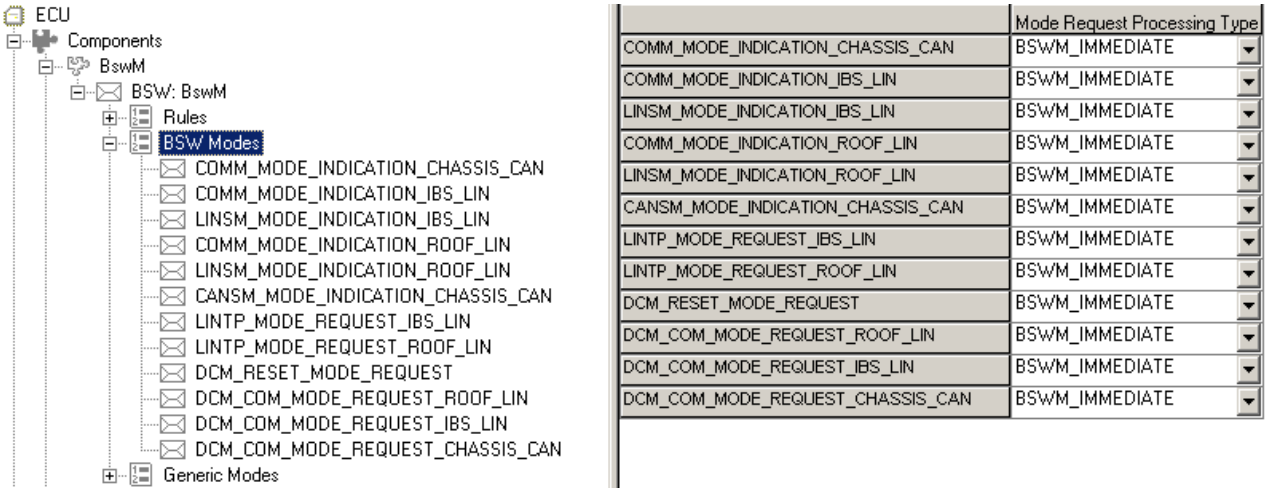

Figure 6-3 Example view of BSW Modes and their Mode Request Processing Type

Table 6-2 gives an overview of the available BSW mode indications/requests which can be used as source mode in the rule configuration:

BSW Mode Indication	Possible Modes	Description
COMM_MODE_INDICATION_ <channelName>	> COMM_NO_COMMUNICATION > COMM_SILENT_COMMUNICATION > COMM_FULL_COMMUNICATION	The ComM notifies the BswM about a change of its communication mode. Refer to [8] for further information about the ComM modes.
COMM_MODE_PNC<Pnc-index>	> COMM_PNC_NO_COMMUNICATION > COMM_PNC_PREPARE_SLEEP > COMM_PNC_READY_SLEEP > COMM_PNC_REQUESTED	The ComM notifies the BswM about a change of a PNC mode. Refer to [8] for further information about the ComM PNC modes.
CANSM_MODE_INDICATION_ <channelName>	> CANSM_BSWM_BUS_OFF > CANSM_BSWM_NO_	The CanSM notifies the BswM about a change of its

BSW Mode Indication	Possible Modes	Description
	COMMUNICATION > CANSM_BSWM_FULL_COMMUNICATION > CANSM_BSWM_SILENT_COMMUNICATION	communication mode. Refer to [6] for further information about the CanSM modes.
LINSM_MODE_INDICATION_<channelName>	> LINSM_BSWM_NO_COM > LINSM_BSWM_FULL_COM > LINSM_BSWM_RUN_SCHEDULE > LINSM_BSWM_GOTO_SLEEP	The LinSM notifies the BswM about a change of its communication mode. Refer to [5] for further information about the LinSM modes.
LINSM_CURRENT_SCHEDULE_<channelName>	> Null_Schedule > <Schedule Table Name>	The LinSM notifies the BswM about the current active LIN Schedule Table. The possible Modes depend on the available Schedule Tables in the current configuration.
LINTP_MODE_REQUEST_<channelName>	> LINTP_MODE_RELEASE > LINTP_MODE_REQUEST > LINTP_MODE_RESPONSE	The LinTp notifies the BswM about a change of its mode. Refer to [9] for further information about the LinTp modes.
DCM_RESET_MODE_REQUEST	> DCM_BOOTLOADER_RESET > DCM_HARD_RESET > DCM_KEY_ON_OFF_RESET > DCM_SOFT_RESET	The Dcm notifies the BswM about a change of its reset mode. Refer to [11] for further information about the Dcm reset modes.
DCM_COM_MODE_REQUEST_<channelName>	> DCM_DISABLE_RX_ENABLE_TX_NM > DCM_DISABLE_RX_ENABLE_TX_NORM > DCM_DISABLE_RX_ENABLE_TX_NORM_NM > DCM_DISABLE_RX_TX_NM > DCM_DISABLE_RX_TX_NORM_NM > DCM_DISABLE_RX_TX_NORMAL > DCM_ENABLE_RX_DISABLE_TX_NM > DCM_ENABLE_RX_DISABLE_TX_NORM > DCM_ENABLE_RX_TX_NM > DCM_ENABLE_RX_TX_NORM > DCM_ENABLE_RX_TX_NORM_NM	The Dcm notifies the BswM about a change of its communication mode. Refer to [11] for further information about the Dcm communication modes.
FRSM_MODE_INDICATION_<channelName>	> FRSM_BSWM_HALT_REQ > FRSM_BSWM_HALT_REQ_ECU_PASSIVE > FRSM_BSWM_KEYSLOT_ONLY > FRSM_BSWM_KEYSLOT_ONLY_ECU_PASSIVE > FRSM_BSWM_ONLINE	The FrSM notifies the BswM about a change of its mode. Refer to [7] for further information about the FrSM modes.

BSW Mode Indication	Possible Modes	Description
	> FRSM_BSWM_ONLINE_ECU_ PASSIVE > FRSM_BSWM_ONLINE_PASSIVE > FRSM_BSWM_ONLINE_PASSIVE _ECU_PASSIVE > FRSM_BSWM_READY > FRSM_BSWM_READY_ECU_PASSIVE > FRSM_BSWM_STARTUP > FRSM_BSWM_STARTUP_ECU_ PASSIVE > FRSM_BSWM_WAKEUP > FRSM_BSWM_WAKEUP_ECU_ PASSIVE	
NM_MODE_INDICATION_ <channelName>	> NM_STATE_BUS_SLEEP > NM_STATE_NORMAL_OPERATION > NM_STATE_PREPARE_BUS_SLEEP > NM_STATE_READY_SLEEP > NM_STATE_WAIT_CHECK_ACTIVATION > NM_STATE_WAIT_NETWORK_STARTUP > NM_STATE_REPEAT_MESSAGE > NM_STATE_SYNCHRONIZE > NM_STATE_BUS_OFF	The NmOsek, NmFiatB or NmFiatC notifies the BswM about a change of its mode. Refer to [14] for further information about the NmFiatB modes. Refer to [15] for further information about the NmFiatC modes.
LINSM_SCHEDULE_END_<channelName>	> schedule table identifier	The LinSM notifies the BswM about the end of the schedule table.

Table 6-2 Overview BSW Mode Indications

Caution
The LinSM_SCHEDULE_END_<channelName> notifications can only be used if the schedule table end notification is enabled inside the LinIf module configuration and the LinSM schedule table configuration.

Attribute Name	Value Type	Values Default value is typed bold	Description
Mode Request Processing Type	Enum	**BSWM_IMMEDIATE** BSWM_DEFERRED	Specify Mode Request Processing Type: BSWM_IMMEDIATE: all rules which contain this BSW Mode in at least one condition will be arbitrated immediately upon the

Attribute Name	Value Type	Values Default value is typed bold	Description
			occurrence of the appropriate Generic Mode Trigger Function. The requested Mode State will be considered immediately. BSWM_DEFERRED: all rules which contain this mode in at least one condition will be arbitrated during cyclically during the execution of the `BswM_MainFunction()`.

Table 6-3 Configuration parameter for BSW Modes

6.2.3 Generic Mode Configuration

A Generic Mode will be created by right mouse click on and **Add Generic Mode**:

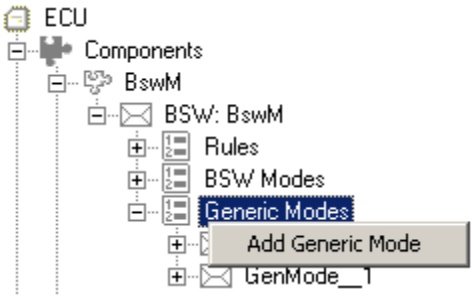

Figure 6-4 Addition of a Generic Mode

A Generic Mode has the following general configuration parameter:

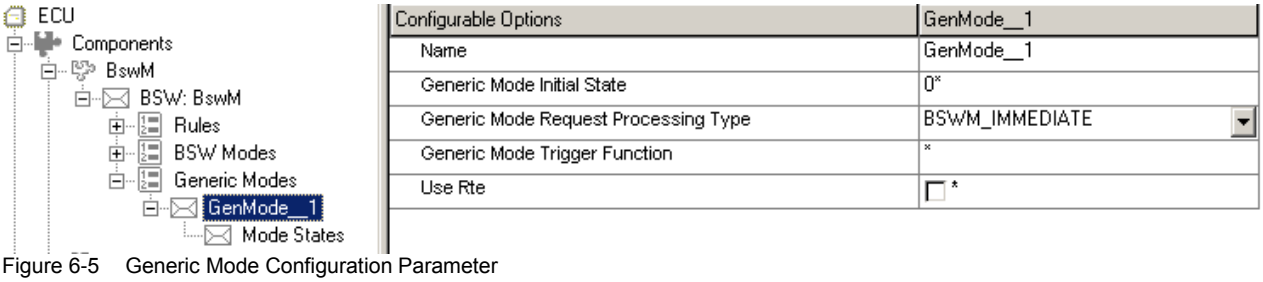

Figure 6-5 Generic Mode Configuration Parameter

Attribute Name	Value Type	Values Default value is typed bold	Description
Name	String	**GenMode__<index>**	Optional: specify a name for the mode.
Generic Mode Initial State	Integer	**0**	Specify the Initial State.
Generic Mode Request Processing Type	Enum	**BSWM_IMMEDIATE** BSWM_DEFERRED	Specify Mode Request Processing Type: BSWM_IMMEDIATE: all rules which contain this Generic Mode in at least one condition will be arbitrated immediately upon the occurrence of the appropriate Generic Mode Trigger Function. The requested Mode State will be considered immediately. BSWM_DEFERRED: all rules which contain this mode in at least one condition will be

Attribute Name	Value Type	Values Default value is typed bold	Description
			arbitrated during cyclically during the execution of the `BswM_MainFunction()`.
Generic Mode Trigger Function	String	*	Specify the Generic Mode Trigger Function Name: Rte is used: the appropriate interfaces are provided by the Rte. Rte is not used: the file `BswM_Cfg.h` contains the appropriate extern declaration of this function.
Use Rte	Boolean	**FALSE** TRUE	If **Use Rte** is enabled the BswM generates a SWC template file which contains the provided C/S Port interface description to change the Generic Mode State.

Table 6-4 Generic Mode general configuration parameter

6.2.4 Timer Configuration

The BswM provides the possibility to configure timers as AUTOSAR extension. These timers may be used within the rules as conditions (see chapter 6.2.5.1) and as actions (see chapter 6.2.5.2).

An action that a timer shall be started or stopped can be added to a rule (see chapter 6.2.5.2.2 “Action Timer Control” for further details).

A rule can have conditions that may evaluate the state (expired, started or stopped) of the timer. A timer has three possible states:

> BSWM_TMR_EXPIRED

> BSWM_TMR_STARTED

> BSWM_TMR_STOPPED

A timer is added to the configuration of the BswM by right mouse click on Timer in the BswM configuration view:

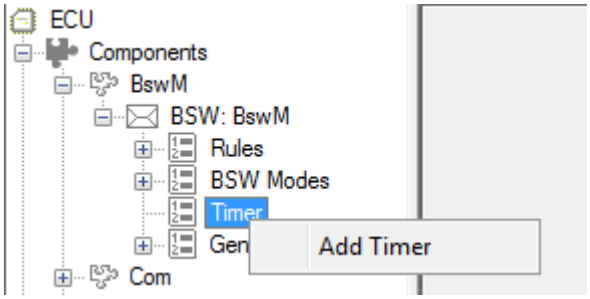

Figure 6-6 Adding a Timer to the configuration of the BswM

Figure 6-6 shows the general configuration options of the BswM timers. A timer has a unique name which either is the default name **Timer_<number>** or the name the user has configured.

Furthermore, the timer value in milliseconds can be defined. This parameter specifies when does a timer expires changing its state to “BSWM_TMR_EXPIRED” and has thus influence in the actions that are executed.

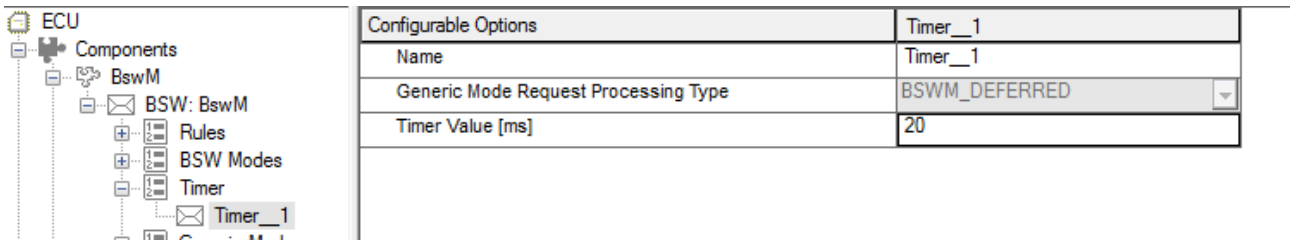

Figure 6-7 Configuration of a Timer

6.2.5 Rule Configuration

For a proper BswM configuration at least one rule must be created. A rule is added to the BswM configuration by right mouse click on **Rules** in the BswM configuration view:

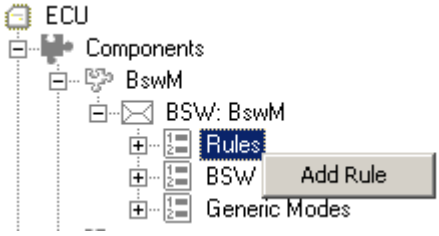

Figure 6-8 Adding a rule to the BswM configuration

Figure 6-9 shows the general configuration options of the BswM rules. A rule has a unique name which either is the default name **Rule_<number>** or the name the user has configured.

Furthermore it can be defined which state a rule has after (re-)initialization. This parameter specifies how a rule is treated when it is evaluated the first time after (re-)initialization and has thus influence on which actions are executed.

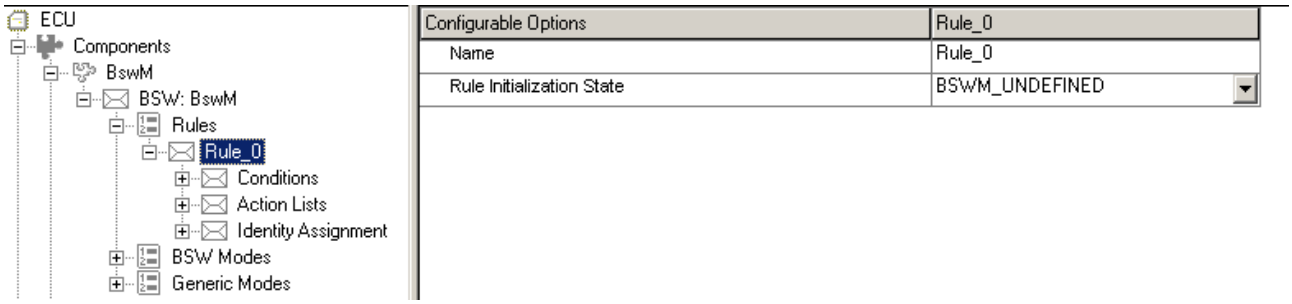

Figure 6-9 Rule Naming and initial state

Note
The **Rule Initialization State** parameter has only influence if an action list of this rule uses **ActionListExecution** type **BSWM_TRIGGER**, please refer to chapter Action List Execution Type for details.

Attribute Name	Value Type	Values Default value is typed bold	Description
Name	String	**Rule<number>** user defined string	The user can define a meaningful name for each rule. The name must be unique for each rule and must not contain spaces or characters which do not belong to the ANSI C Standard.
Rule Initialization State	Enum	**BSWM_UNDEFINED** BSWM_TRUE BSWM_FALSE	Specifies the rule state after (re-) initialization.

Table 6-5 General Rule Configuration Options

6.2.5.1 Configuration of Rule Conditions

A rule needs at least one condition which is added to the rule configuration by right mouse click on the rule:

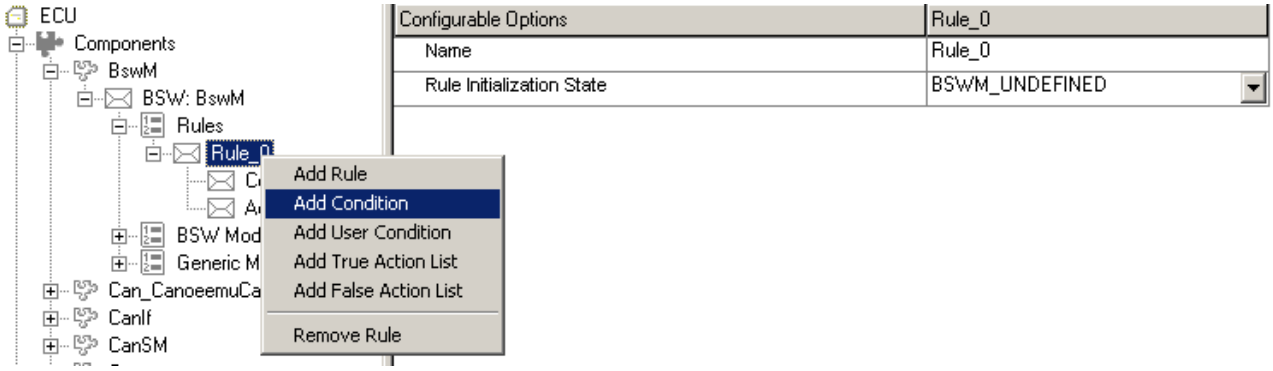

Figure 6-10 Adding a condition to a rule

Figure 6-11 shows the configuration view of a condition which consists of the source mode and the requested mode and the operator of this condition:

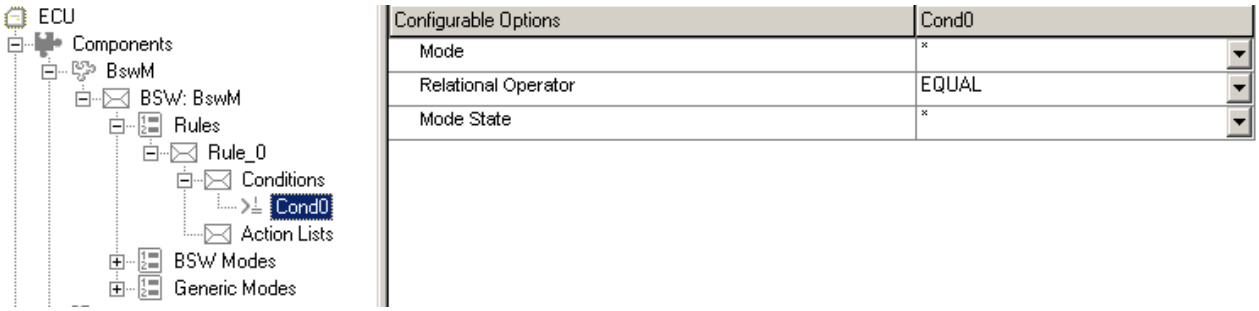

Figure 6-11 Configuration view of a rule condition

Description of the condition configuration parameter:

Attribute Name	Value Type	Values Default value is typed bold	Description
Mode	Enum	* < BSW Mode name > < Generic Mode name > < Previous Mode name >	Specifies the BSW Mode, the Generic Mode or the Previous Mode which shall be used for comparison in this condition.
Relational Operator	Enum	**EQUAL** NOT_EQUAL	Specifies the operator which is used for comparison of SourceMode to RequestedMode.
Mode State	Enum	* **< Possible Mode States the Mode can have >**	Specifies the Mode State which shall be used for comparison in this condition.

Table 6-6 Condition Configuration Parameter

Caution
A Rule must contain at least one condition which uses either a BSW Mode or a Generic Mode. A Rule which has only one condition which uses a previous Mode (e.g. FRSM_PREVIOUS_MODE) is not allowed.

Additionally a user condition can be added:

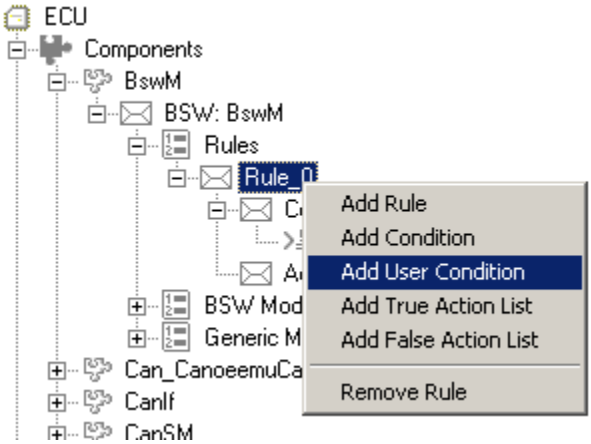

Figure 6-12 Adding a User Condition

The user condition configuration view looks like as follows:

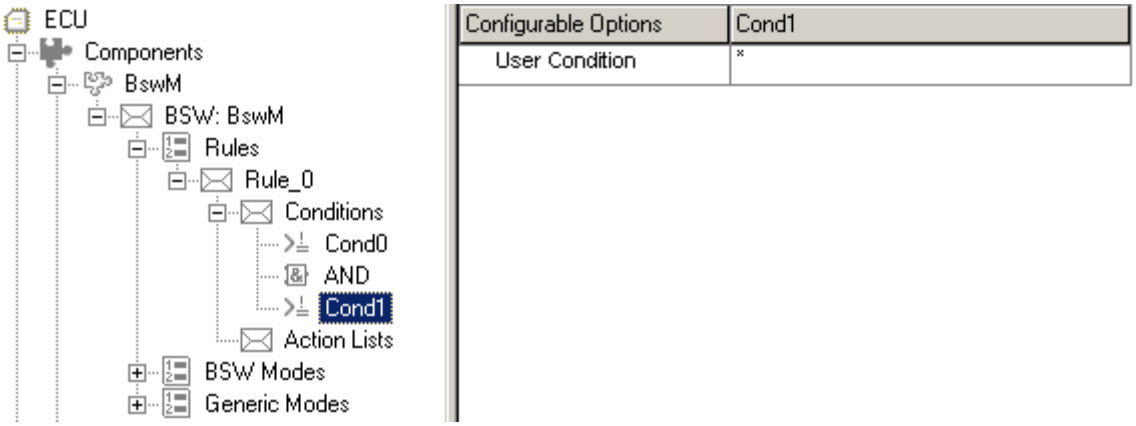

Figure 6-13 Configuration view of a user condition

Attribute Name	Value Type	Values Default value is typed bold	Description
User Condition	String	*	Specify the string of the function or variable which shall be used in this condition. The extern declaration of the used function/variable must be provided via an Application Header File, refer to chapter 6.2.1.

Table 6-7 User Condition Configuration Parameter

If a rule has multiple conditions the operator can be configured for the operation of these conditions:

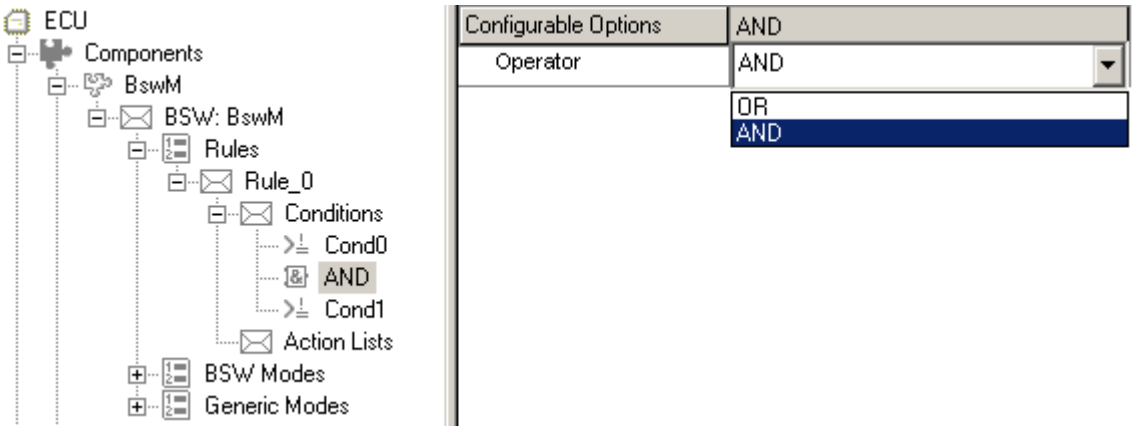

Figure 6-14 Configuration of the Operator of multiple conditions

Attribute Name	Value Type	Values Default value is typed bold	Description
Operator	Enum	**AND** OR	Specifies the operator which shall be used for evaluation of the rule conditions.

Table 6-8 Condition Operation Parameter

6.2.5.2 Configuration of Rule Action Lists

A rule has either the evaluation result **true** or **false**, for each possible result value an action list can created by right mouse click on the rule, the so called **Action List True** or **Action List False**:

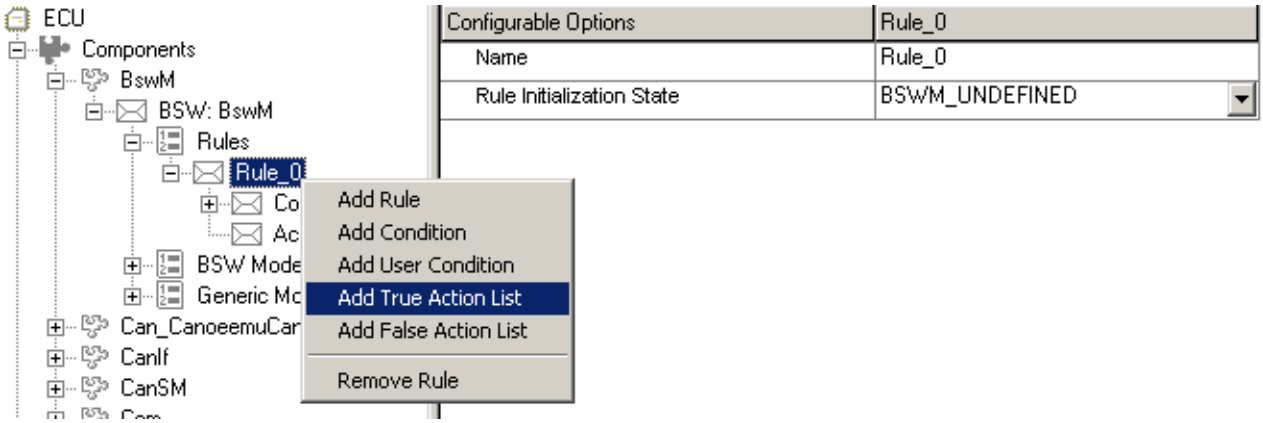

Figure 6-15 Adding Action Lists to a Rule

6.2.5.2.1 Action List Execution Type

An Action List has the configuration parameter **ActionListExecution** which specifies if the action list will be executed every time when the rule arbitration has the appropriate result or only when the arbitration result is not equal to the result of the last rule arbitration:

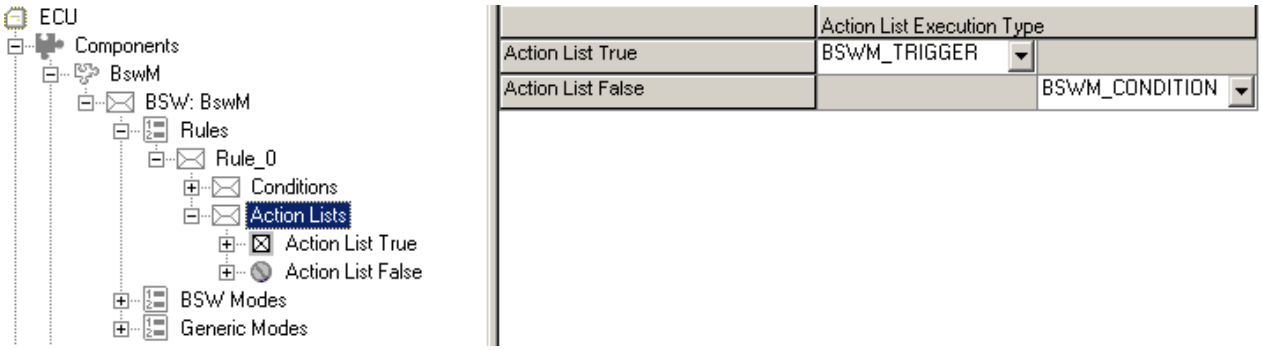

Figure 6-16 Action List Execution Type

Attribute Name	Value Type	Values Default value is typed bold	Description
Action List Execution Type	Enum	**BSWM_TRIGGER** BSWM_CONDITION	Specifies the execution of an action list: BSWM_TRIGGER: action list will only be executed if the rule arbitration result changes. BSWM_CONDITION: action list will be executed every time the rule arbitration has the appropriate result.

Table 6-9 Action List Execution Type Configuration Parameter

6.2.5.2.2 Adding Actions to an Action List

By right mouse click on an action list an action can be added:

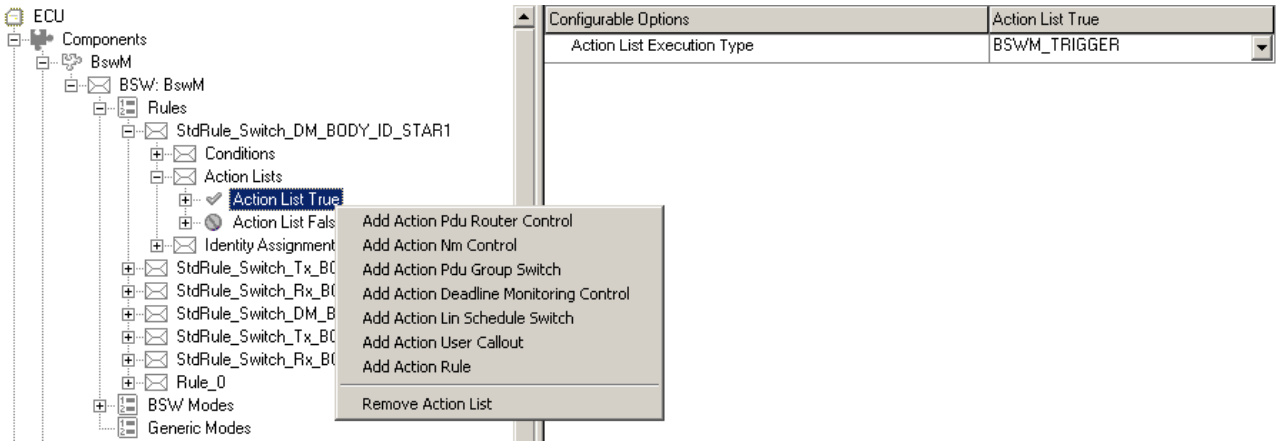

Figure 6-17 Adding Actions to an Action List

> Action Pdu Router Control

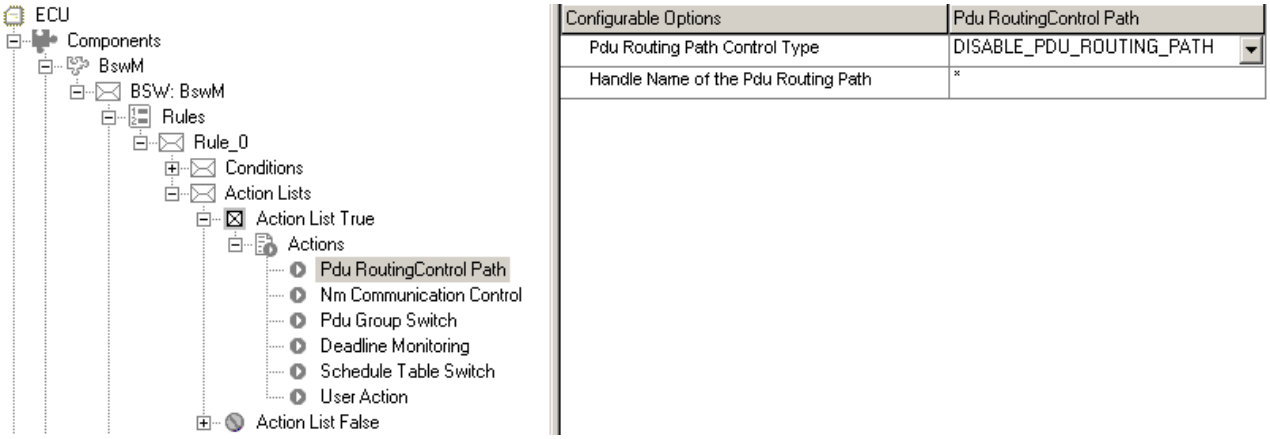

Figure 6-18 Action Pdu Router Control

Attribute Name	Value Type	Values Default value is typed bold	Description

Pdu Routing Path Control Type	Enum	**DISABLE_PDU_ROUTING_PATH**
		ENABLE_PDU_ROUTING_PATH
			This action enables/disables a PduR Routing Path.
			Used API:
			`PduR_EnableRouting()`
			`PduR_DisableRouting()`
			For further information refer to [10].

Handle Name of the Pdu Routing Path	String	*	Specify the Handle for the Pdu Routing Path which shall be enabled/disabled.

Table 6-10 Action Pdu Router Control Configuration Parameter

> Action Nm Communication Control

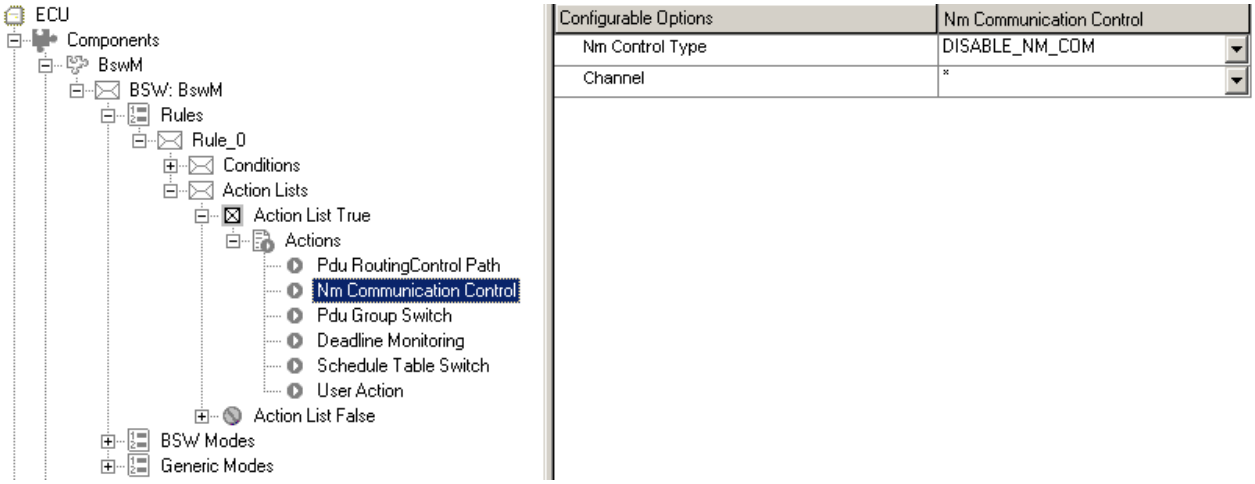

Figure 6-19 Action Nm Communication Control

Attribute Name	Value Type	Values Default value is typed bold	Description

Nm Control Type	Enum	**DISABLE_NM_COM**
		ENABLE_NM_COM
			This action enables/disables Nm Communication.
			Used API:
			`Nm_EnableCommunication()`
			`Nm_DisableCommunication()`
			For further information refer to [12].

Channel	Object	* < Channel Name >	Specify the channel where the Nm communication shall be enabled/disabled.

Table 6-11 Action Nm Communication Control Configuration Parameter

> Action Pdu Group Switch

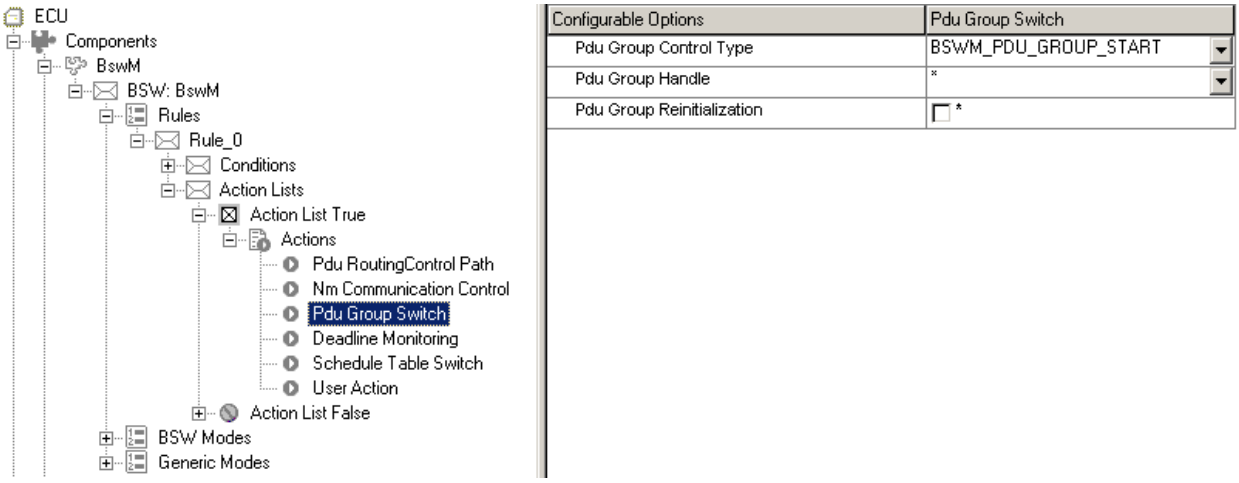

Figure 6-20 Action Pdu Group Switch

Attribute Name	Value Type	Values Default value is typed bold	Description

Pdu Group Control Type	Enum	**BSWM_PDU_GROUP_START**
		BSWM_PDU_GROUP_STOP
			Specify if the Pdu Group shall be started/stopped.
			Used API:
			`Com_IpduGroupStart ()`
			`Com_IpduGroupStop ()`
			For further information refer to [13].

Pdu Group Handle	Object	***** < Pdu Group Handle Name >	Specify the Pdu Group which shall be started/stopped.
Pdu Group Reinitialization	Boolean	Enabled **Disabled**	If enabled the Pdu Group signal values are initialized upon start. This parameter has only influence if BswMPduGroupType is **BSWM_PDU_GROUP_START**.

Table 6-12 Action Pdu Group Switch Configuration Parameter

Caution

This Action shall only be used for the bus types CAN and FlexRay, furthermore it must be ensured that the Tx and Rx Pdu Group are disabled in the CanSM and FrSM configuration.

For bus type LIN the LinSM still handles the Pdu Groups.

> Action Deadline Monitoring

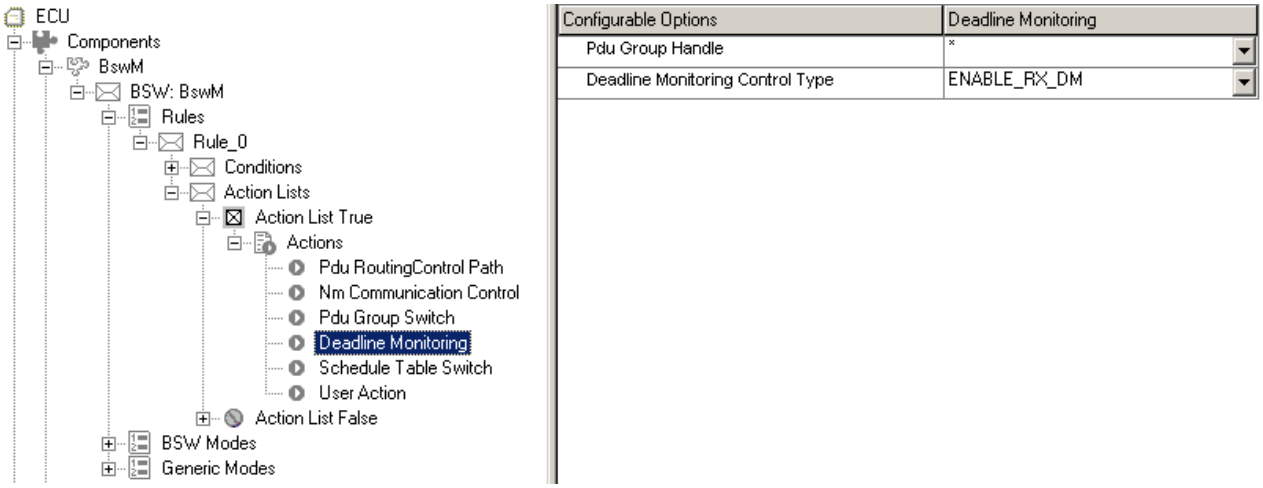

Figure 6-21 Action Deadline Monitoring

Attribute Name	Value Type	Values Default value is typed bold	Description
Pdu Group Handle	Object	* < Pdu Group Handle Name >	Specify the Pdu Group for which the deadline monitoring shall be enabled/disabled.

Deadline Monitoring Control Type	Boolean	**ENABLE_RX_DM**
		DISABLE_RX_DM
			Specify if the deadline monitoring shall be enabled/disabled.
			Used API:
			`Com_EnableReceptionDM ()`
			`Com_DisableReceptionDM ()`
			For further information refer to [13].

Table 6-13 Action Deadline Monitoring Configuration Parameter

> Action Schedule Table Switch

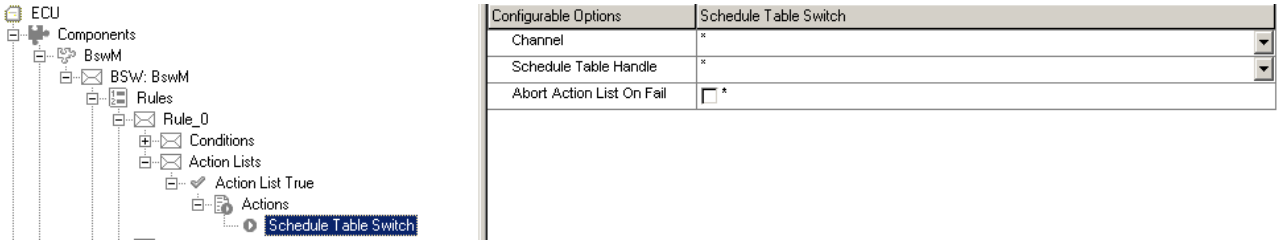

Figure 6-22 Action Schedule Table Switch

Attribute Name	Value Type	Values Default value is typed bold	Description
Channel	Object	* < Channel Name >	Specify the channel where the schedule table shall be changed.
Schedule Table Handle	Object	* < Schedule Handle Name >	Specify the schedule table which shall be requested. Used API: `LinSM_ScheduleRequest()` For further information refer to [5].
Abort Action List On Fail	Boolean	**false** true	This parameter specifies if the Action List Execution shall be abort or not if `LinSM_ScheduleRequest()` returns E_NOT_OK.

Table 6-14 Action Schedule Table Switch Configuration Parameter

> Action User Callout

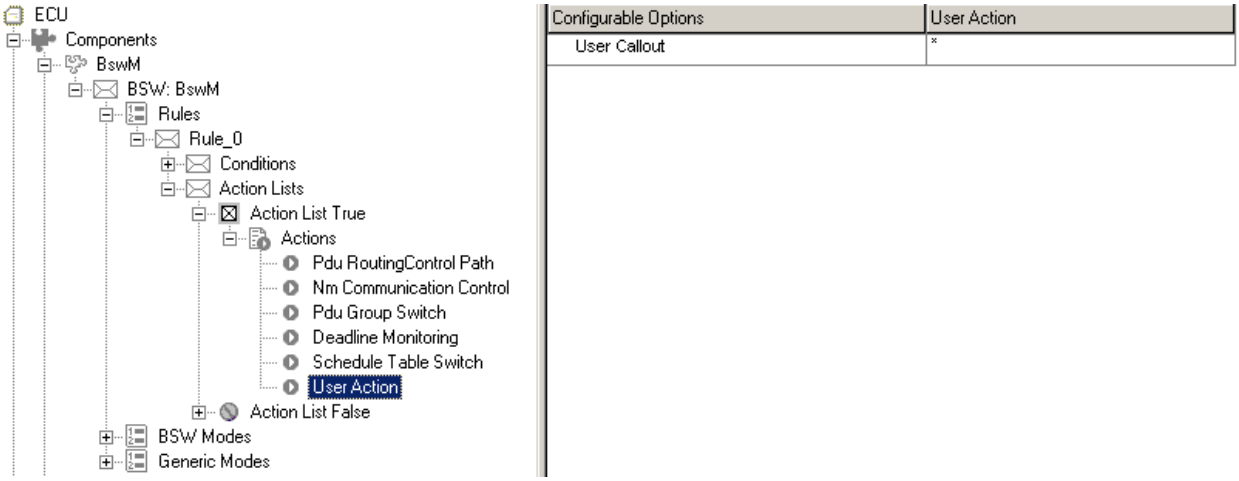

Figure 6-23 Action User Callout

Attribute Name	Value Type	Values Default value is typed bold	Description
ActionUserCallout	String	*	Specify the name of the user callout function. An external declaration of this function must be provided via the application header file on the main configuration page (refer to chapter 6.2.1).

Table 6-15 Action Schedule Table Switch Configuration Parameter

> Action Rule

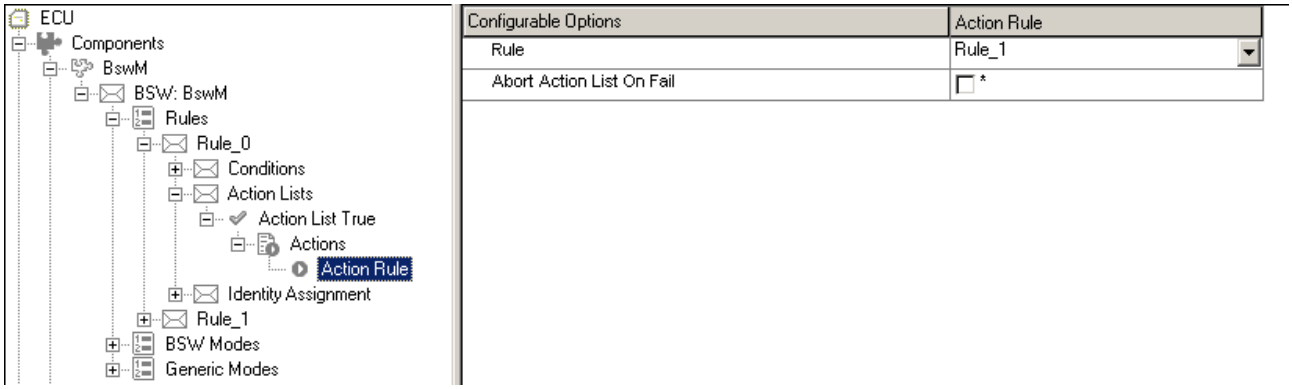

Figure 6-24 Action Rule

Attribute Name	Value Type	Values Default value is typed bold	Description
Rule	Object	* < Rule Name >	Specify the reference to the Rule which shall be arbitrated within this Action List.
Abort Action List On Fail	Boolean	**false** true	If the configured Rule fails (means an Action List of this Rule is aborted because an Action failed) the Action List execution is aborted.

Table 6-16 Action Schedule Table Switch Configuration Parameter

> Action Timer Control

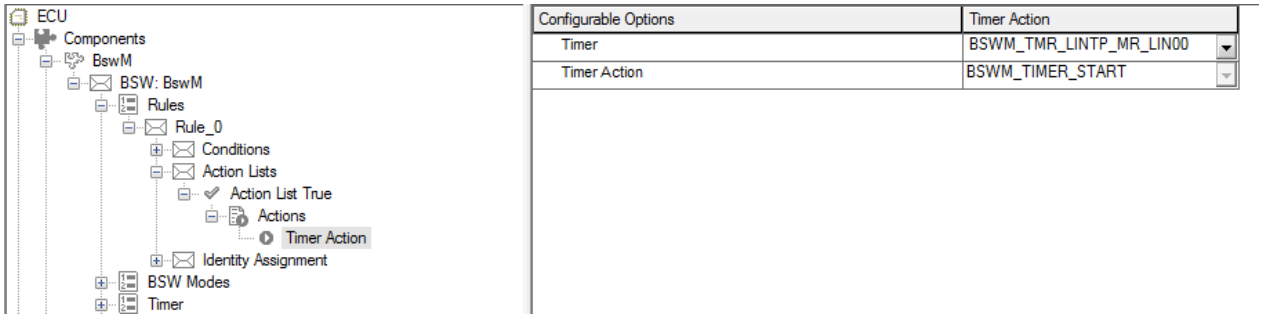

Figure 6-25 Action Timer Control

Attribute Name	Value Type	Values Default value is typed bold	Description
Timer	Object	* < Timer Name >	Specify the Timer for which the timer action shall be performed.
Timer Action	Enum	**BSWM_TIMER_STARTED** BSWM_TIMER_STOPPED	Specify if the timer shall be started or stopped.

Table 6-17 Action Timer Control Configuration Parameter

> Action Mode Notification

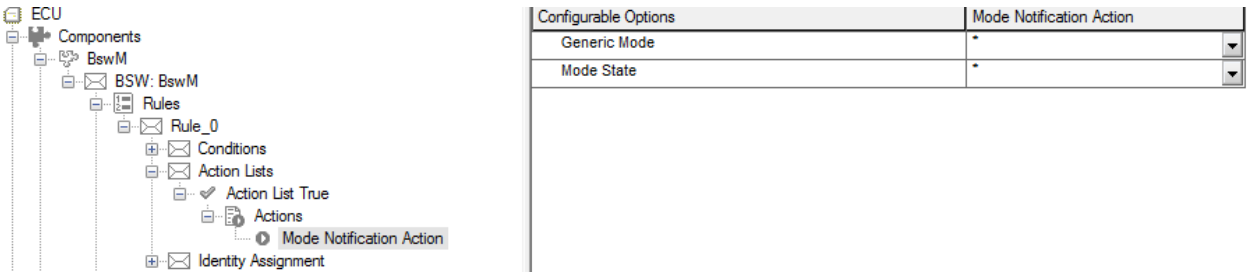

Figure 6-26 Action Mode Notification

Attribute Name	Value Type	Values Default value is typed bold	Description
Generic Mode	Object	* < Generic Mode Name >	Specify the Generic Mode for which the notification shall be called.
Mode State	Object	* < Mode Sate Name >	Specify if the Mode State which shall be notified.

Table 6-18 Action Mode Notification Configuration Parameter

Caution
Mode Notifications are only supported for Generic Modes which handled via the Rte API. (Refer to 6.2.3 Generic Mode Configuration for details)

6.2.5.3 Configuration of the Identity Assignment

For multiple identities use-cases the BswM allows the user to configure relevance of a Rule for the available identities:

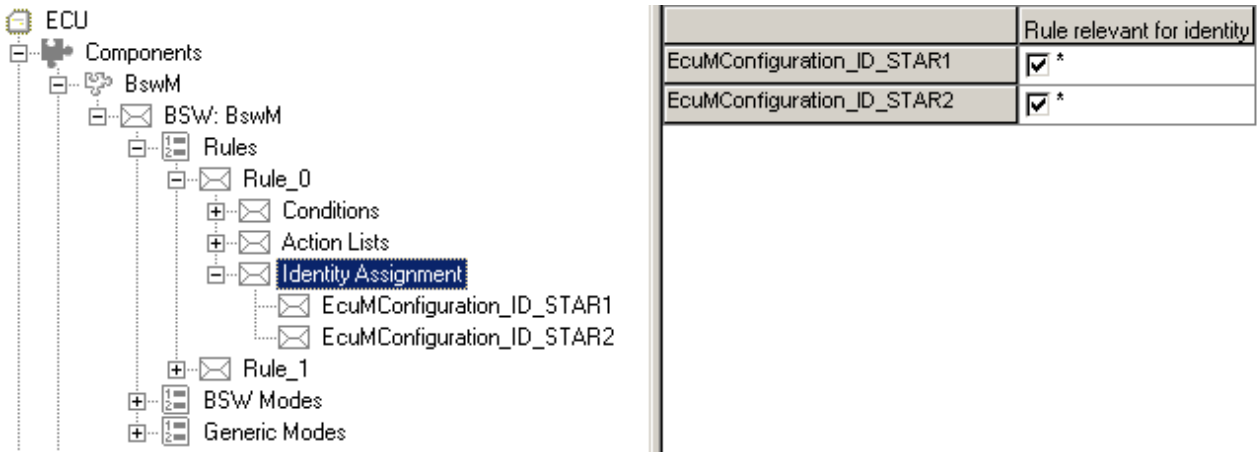

Figure 6-27 Configuration of the Identity Assignment

The BswM automatically detects all available identities and displays them for each Rule in the Identity Assignment view. If a Rule is created it is assigned to all identities per default.

If a Rule is only relevant for specific identities the user can uncheck the appropriate checkbox.

Example

Here is an example to show the usage of the Identity Assignment:

Configuration:

> Two channels

> A BswM Rule “Rule_Example” which uses BSW Source Modes of both channels in its conditions

> Two identities which have the following channel assignment:

	Identity 0	Identity 1
Channel 0	■	■
Channel 1	■	

Table 6-19 Example Identity-Channel Assignment

If “Identity 0” is active there is no problem to arbitrate “Rule_Example” because both channels are active and therefore the used BSW Source Modes in the Rule conditions have correct values.

If “Identity 1” is active it makes no sense to arbitrate “Rule_Example” because only Channel 0 is active and the used Source Modes of Channel 1 don’t have correct values because this channel is not active. So the arbitration result will never be correct.

To prevent the arbitration of “Rule_Example” when “Identity 1” is active the user must uncheck the checkbox for this Identity in the Rule Configuration Identity Assignment.

7 Use Cases of BSWM

This chapter describes use cases of the BswM.

Note
The configuration of the BSWM use cases often depends on your system and application. Therefore the following descriptions are rather a guide. Consider that most of the recommended Use Cases can be configured automatically via the “Create BswM Standard Rules” Button, refer to Table 6-1.

The following Use Cases are described in this chapter

> **CAN Communication Modes** more
(mandatory, can be created automatically via the “Create BswM Standard Rules” Button, refer to Table 6-1)

> **DCM Communication Control Modes** more
(mandatory if you have to support Communication Control service, can be created automatically via the “Create BswM Standard Rules” Button, refer to Table 6-1)

> **DCM Reset Modes** more
(mandatory if you have to support DCM Reset service)

> **LIN Communication Modes** more
(depends on your application)

> **LIN TP Modes** more
(depends on your application)

> **FlexRay Communication Modes** more
(mandatory if you use FlexRay, can be created automatically via the “Create BswM Standard Rules” Button, refer to Table 6-1)

> **NmFiatB Communication Modes** more
(mandatory if you use NmFiatB, can be created automatically via the “Create BswM Standard Rules” Button, refer to Table 6-1).

> **NmFiatC Communication Modes** more
(mandatory if you use NmFiatC, can be created automatically via the “Create BswM Standard Rules” Button, refer to Table 6-1).

> **Partial Network Cluster Modes** more
(depends on the availability of the Partial Network Feature).

7.1 CAN Communication Modes

The CanSM notifies the BswM about its current state via the mode indication function **CANSM_MODE_INDICATION_<channel-name>** which can be used as a trigger for mode arbitration and mode control. Additionally the BswM provides the value of the previous

CanSM state which helps to design rules for specific state transitions. Figure 7-1 shows the possible CanSM states:

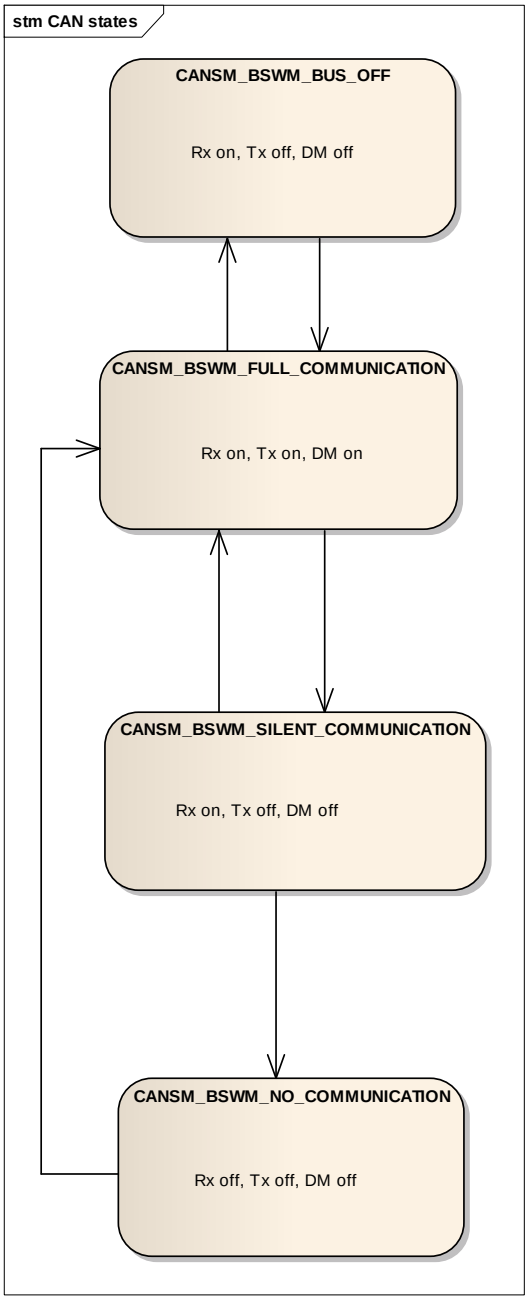

Figure 7-1 CanSM communication modes

The arrows in Figure 7-1 mark the possible CanSM mode indications which can be used as a condition for a specific rule. So it is possible to design rules for any state transition by using the condition **CANSM_MODE_INDICATION_<channel-name>** as source mode and the appropriate CanSM state as requested mode. Furthermore another condition can be added which checks the previous CanSM state, the source mode parameter is **CANSM_PREVIOUS_MODE_<channel-name>**. Figure 7-1 contains the proposed

communication states (Rx on/off, Tx on/off, DM on/off) for each CanSM state, according to this proposal the specific action lists for each transition rule can be created.

As Figure 7-1 shows Tx and DM are in the same state synchronously: the state is **on** in CanSM mode CANSM_BSWM_FULL_COMMUNICATION and **off** in all other CanSM modes. So the following rule can be defined for Tx and DM:

> **STEP 1**: Chose the mode request processing type **BSWM_IMMEDIATE** for the CanSM mode indication, this ensures that the appropriate actions are executed immediately upon change of the CanSM mode:

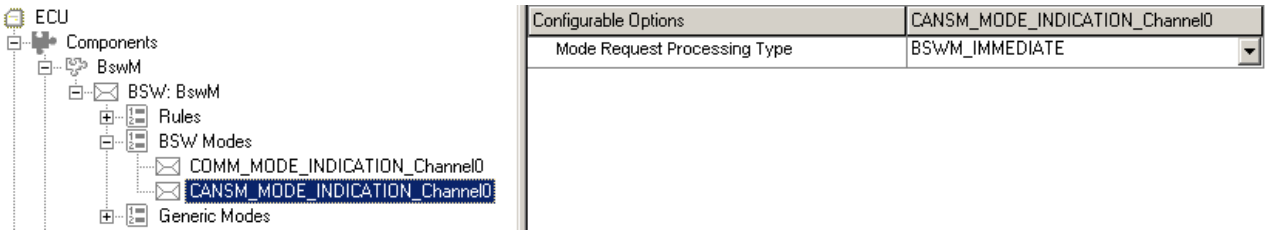

Figure 7-2 Example mode request processing type “BSWM_IMMEDIATE”

> **STEP 2**: Create the rules **Switch_Tx_DM** and **Switch_Rx** and chose the value **BSWM_FALSE** for the **Rule Initialization State** parameter. This ensures that the rule actions are only executed when the rule is evaluated the first time to result true:

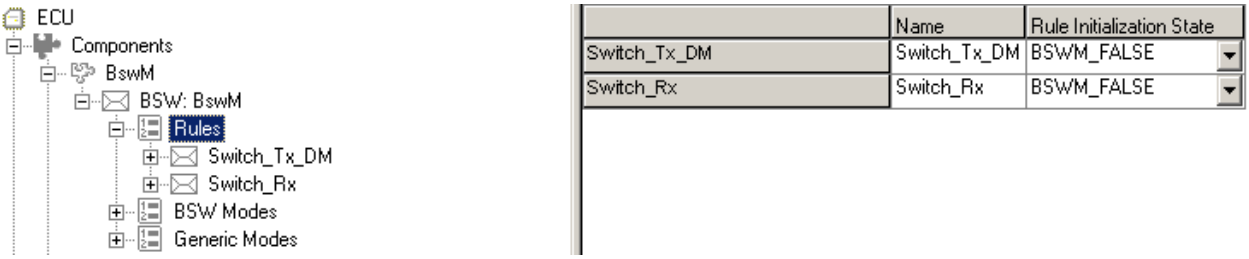

Figure 7-3 Example rule initial state

> **STEP 3**: Configure the rule conditions:

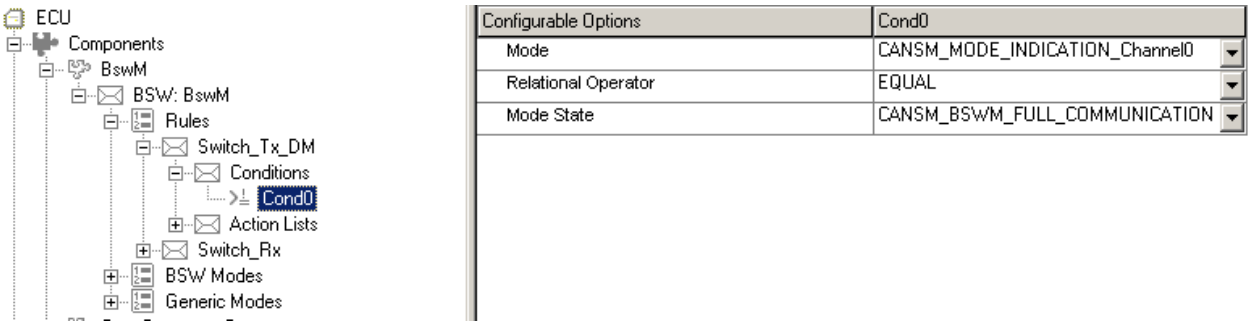

Figure 7-4 Example rule conditions for Tx and DM

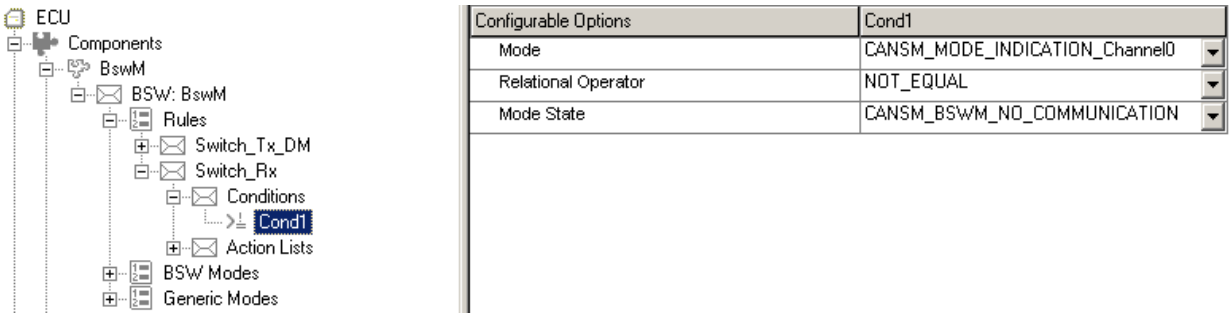

Figure 7-5 Example rule conditions for Rx

The conditions of rule **Switch_Tx_DM** shown in Figure 7-4 shall have the following effect:

> Tx and DM are only allowed to be switched on if the CanSM is in state **CANSM_BSWM_FULL_COMMUNICATION**

> If the evaluation result of this rule is true Tx and DM are switched on, if it is false Tx and DM are switched off

The conditions of rule **Switch_Rx** shown in Figure 7-5 shall have the following effect:

> Rx is only allowed to be switched on if the CanSM is not in state **CANSM_BSWM_NO_COMMUNICATION**

> If the evaluation result of this rule is true Rx is switched on, if it is false Rx is switched off

> **STEP 4**: Configure the action list execution type to **BSWM_TRIGGER** which has the effect that the action list is only executed when the evaluation result of the rule has changed and not every time the rule is evaluated which prevents redundant execution of the actions:

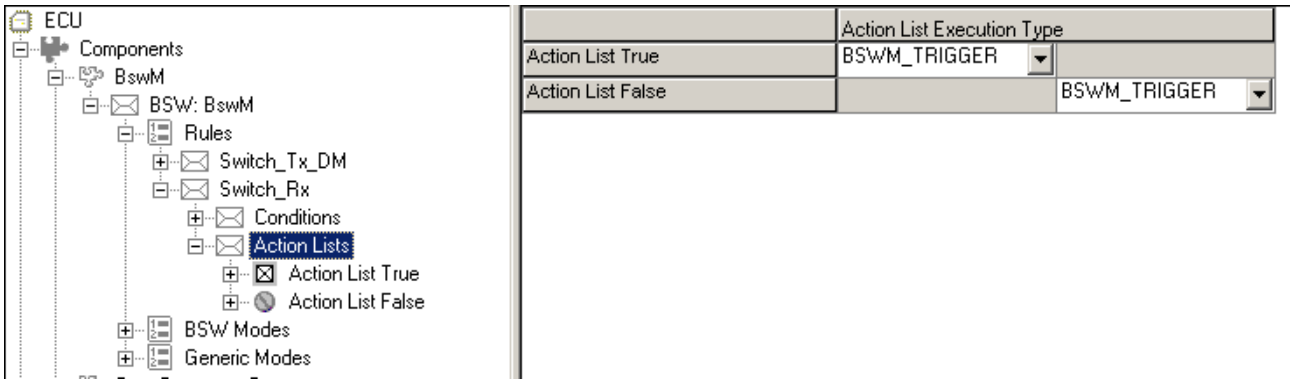

Figure 7-6 Example Action List Execution Type

> **STEP 5**: specify the action lists:

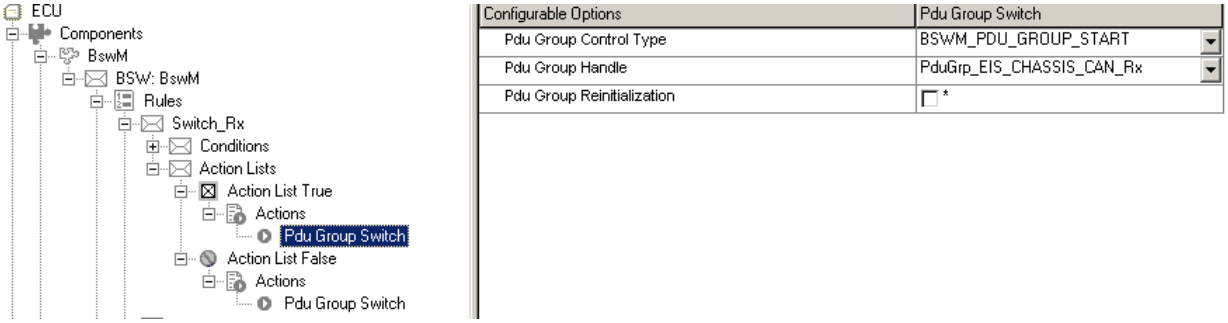

Figure 7-7 Action List True for rule Switch_Rx

Figure 7-7 shows the Action List True for rule **Switch_Rx** which contains the action **Pdu Group Switch** with type **BSWM_PDU_GROUP_START** and the Pdu group handle for the Rx Pdu Group. The BswMPduGroupSwitchReinit parameter defines if the signal values shall be initialized or not.

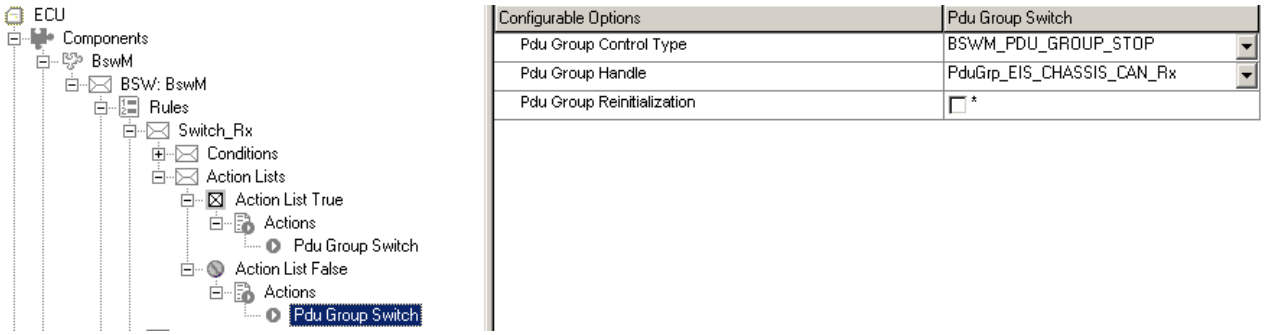

Figure 7-8 Action List False for rule “Switch_Rx”

Figure 7-8 shows the Action List False for rule **Switch_Rx** which contains the action **Pdu Group Switch** with type **BSWM_PDU_GROUP_STOP** and the Pdu group handle for the Rx Pdu Group. The BswMPduGroupSwitchReinit parameter has no influence in case of stopping the Pdu group.

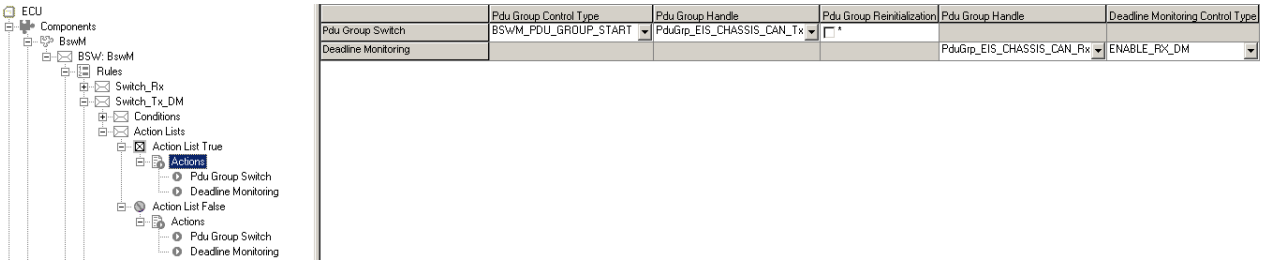

Figure 7-9 Action List True for rule “Switch_Tx_DM”

Figure 7-9 shows Action List True for rule **Switch_Tx_DM** which contains the action **Pdu Group Switch** with type **BSWM_PDU_GROUP_START** and the Pdu group handle for the Tx Pdu Group. The second action is to enable the **Deadline Monitoring** for the Rx Pdu Group.

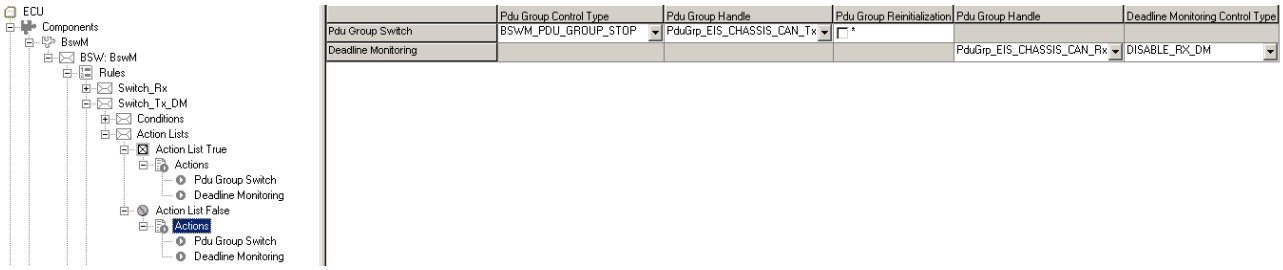

Figure 7-10 Action List False for rule “Switch_Tx_DM”

Figure 7-10 shows the Action List False for rule **Switch_Tx_DM**, the Tx Pdu Group is stopped and the Rx Deadline Monitoring is disabled.

If Dcm communication control is used the rules have to be extended as described in the following tables:

7.1.1 Rule Switch_Tx_DM

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Tx_DM	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: CANSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: CANSM_BSWM_FULL_COMMUNICATION	Tx and DM are only allowed to be on in state CANSM_BSWM_FULL_ COMMUNICATION.
Operator	AND	All conditions must be fulfilled.
Condition 2	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORM_NM	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 3	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This Dcm Mode State must not forbid Tx.

Attribute Name	Values	Description
Operator	AND	All conditions must be fulfilled.
Condition 4	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM_NM	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 5	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM	This Dcm Mode State must not forbid Tx.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Tx Pdu Group > Enable Deadline Monitoring for Rx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Tx Pdu Group > Disable Deadline Monitoring for Rx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-1 Configuration of rule for starting and stopping Tx and DM for CAN with Dcm Com Control

7.1.2 Rule Switch_Rx

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Rx	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: CANSM_MODE_INDICATION_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: CANSM_BSWM_NO_COMMUNICATION	Rx is not allowed in state CANSM_BSWM_NO_ COMMUNICATION.
Operator	AND	All conditions must be fulfilled.
Condition 2	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 3	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM_NM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 4	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORM_NM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 5	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This Dcm Mode State must not forbid Rx.
Action List True	**Action List Execution Type**:	The Execution Type is

Attribute Name	Values	Description
	BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Rx Pdu Group	**BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Rx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-2 Configuration of rule for starting and stopping Rx and DM for FlexRay with Dcm Com Control

7.2 Dcm Communication Control Modes

The BswM supports the DCM communication control modes which are indicated by the mode source **DCM_COM_MODE_REQUEST_<channel-name>**. Dependent on the requested communication control modes the user can configure the specific rules, conditions and actions:

Requested Mode	Actions
DCM_ENABLE_RX_TX_NORM	> Pdu Group Switch: start Tx IPdu-Group(s) > Pdu Group Switch: start Rx IPdu-Group(s) > Deadline Monitoring Control: enable the deadline monitoring for the Rx IPdu-Group(s)
DCM_ENABLE_RX_DISABLE_TX_NORM	> Pdu Group Switch: stop Tx IPdu-Group(s) > Pdu Group Switch: start Rx IPdu-Group(s) > Deadline Monitoring Control: enable the deadline monitoring for the Rx IPdu-Group(s)
DCM_DISABLE_RX_ENABLE_TX_NORM	> Pdu Group Switch: start Tx IPdu-Group(s) > Pdu Group Switch: stop Rx IPdu-Group(s) > Deadline Monitoring Control: disable the deadline monitoring for the Rx IPdu-Group(s)
DCM_DISABLE_RX_TX_NORMAL	> Pdu Group Switch: stop Tx IPdu-Group(s) > Pdu Group Switch: stop Rx IPdu-Group(s) > Deadline Monitoring Control: disable the deadline monitoring for the Rx IPdu-Group(s)
DCM_ENABLE_RX_TX_NM	> Nm Control: Enable Nm communication
DCM_ENABLE_RX_DISABLE_TX_NM	> Nm Control: Disable Nm communication
DCM_DISABLE_RX_ENABLE_TX_NM	> Nm Control: Enable Nm communication
DCM_DISABLE_RX_TX_NM	> Nm Control: Disable Nm communication
DCM_ENABLE_RX_TX_NORM_NM	> Nm Control: Enable Nm communication > Pdu Group Switch: stop Tx IPdu-Group(s)

Requested Mode	Actions
	> Pdu Group Switch: start Rx IPdu-Group(s) > Deadline Monitoring Control: enable the deadline monitoring for the Rx IPdu-Group(s)
DCM_ENABLE_RX_DISABLE_TX_NORM_NM	> Nm Control: Disable Nm communication > Pdu Group Switch: stop Tx IPdu-Group(s) > Pdu Group Switch: start Rx IPdu-Group(s) > Deadline Monitoring Control: enable the deadline monitoring for the Rx IPdu-Group(s)
DCM_DISABLE_RX_TX_NORM_NM	> Nm Control: Disable Nm communication > Pdu Group Switch: stop Tx IPdu-Group(s) > Pdu Group Switch: start Rx IPdu-Group(s) > Deadline Monitoring Control: disable the deadline monitoring for the Rx IPdu-Group(s)

Table 7-3 Recommended communication control modes

Caution
Consider the communication states of the Bus State Managers when you use the Dcm communication control modes. Refer to the examples for bus type CAN and FlexRay (chapters 0 and 7.6). For bus type LIN there is a different handling which is described in chapter 7.5.

7.3 Dcm Reset Modes

The BswM supports specific DCM Reset Modes which are indicated by the mode source **DCM_RESET_MODE_REQUEST**. Dependent on the requested reset mode the user can configure the specific rules, conditions and actions. The BswM supports the following Dcm Reset Modes:

> DCM_BOOTLOADER_RESET

> DCM_HARD_RESET

> DCM_KEY_ON_OFF_RESET

> DCM_SOFT_RESET

7.4 LIN Communication Modes

The LIN communication modes are indicated by the LinSM mode **LINSM_MODE_INDICATION_<channel-name>**. The possible requested mode values are shown in the following figure:

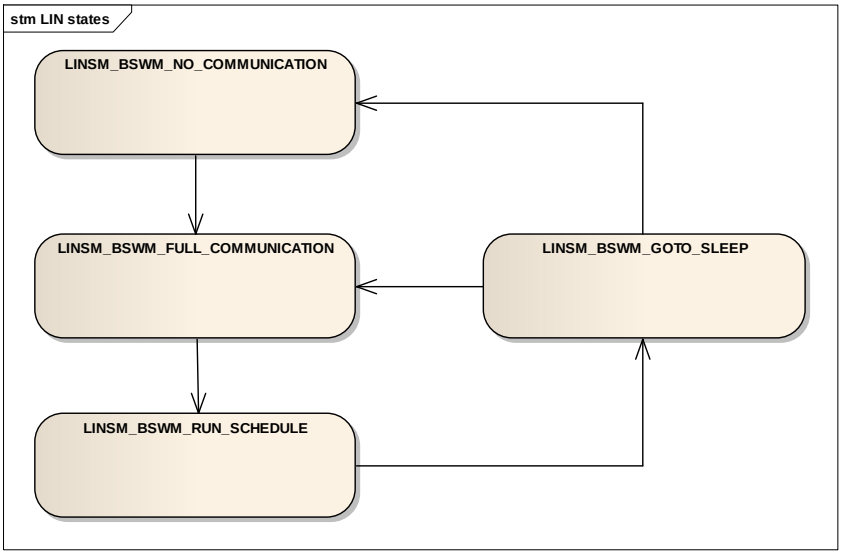

Figure 7-11 LIN communication states

The LinSM only accepts schedule requests in the mode state LINSM_BSWM_FULL_COMMUNICATION, so for example a rule can be defined which requests a schedule table every time this state is entered. Furthermore a User Condition can be used to specify which schedule table shall be requested.

Example:

There are two schedule tables, one which shall be requested when clamp 15 is off and one when clamp 15 is on. To ensure LIN communication without delay the appropriate schedule table shall be requested directly upon entering the state LINSM_BSWM_RUN_SCHEDULE. To fulfill these requirements the following Modes and Rules are necessary:

> one Generic Mode which allows the user to notify the BswM about the clamp 15 mode state

> one Rule which checks the condition if the LinSM is in the state LINSM_BSWM_RUN_SCHEDULE and if state of clamp 15 mode is on, the Action List True requests the schedule table for clamp 15 on

> one Rule which checks the condition if the LinSM is in the state LINSM_BSWM_RUN_SCHEDULE and if state of clamp 15 mode is off, the Action List True requests the schedule table for clamp 15 off

The following sub-chapters describe the configuration parameter for this example.

7.4.1 Generic Mode for clamp 15

Attribute Name	Values	Description
Name	Clamp_15	Name of the Generic Mode.
Generic Mode Initial State	0	Value after initialization is Clamp_15_On.
Generic Mode Request Processing Type	BSWM_IMMEDIATE	Rules which use this mode shall be arbitrated immediately.

Attribute Name	Values	Description
Generic Mode Trigger Function	SetClamp15State	This function is used by the application to set the mode state to either Clamp_15_On or Clamp_15_Off.
Use Rte	-	If you use the Rte enable this option.
Mode State Clamp_15_On	0	Add a Mode State Clamp_15_On which has value 0.
Mode State Clamp_15_Off	1	Add a Mode State Clamp_15_Off which has value 1.

Table 7-4 Recommended configuration for Generic Mode clamp 15

7.4.2 Rule clamp 15 on

Attribute Name	Values	Description
Name	RuleClamp15On	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: LINSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: LINSM_BSWM_RUN_SCHEDULE	To request schedule tables the LinSM must be in state LINSM_BSWM_ RUN_SCHEDULE.
Operator	AND	Both conditions are considered.
Condition 2	**Mode**: Clamp_15 **Relational Operator**: EQUAL **Mode State**: Clamp_15_On	Generic Mode **Clamp_15** must be on.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Schedule Table Switch for clamp 15 on	The Action List True requests the schedule table for clamp 15 on.

Table 7-5 Configuration of example rule clamp 15 on

7.4.3 Rule clamp 15 off

Attribute Name	Values	Description
Name	RuleClamp15Off	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: LINSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: LINSM_BSWM_RUN_SCHEDULE	To request schedule tables the LinSM must be in state LINSM_BSWM_ RUN_SCHEDULE.
Operator	AND	Both conditions are considered.
Condition 2	**Mode**: Clamp_15 **Relational Operator**: EQUAL **Mode State**: Clamp_15_Off	Generic Mode **Clamp_15** must be off.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Schedule Table Switch for clamp 15 off	The Action List True requests the schedule table for clamp 15 off.

Table 7-6 Configuration of example rule clamp 15 off

7.5 LIN TP Modes

The LinTp notifies the BswM about its mode states which can have the following values:

> **LINTP_MODE_REQUEST**: the LinTp requests the schedule table which contains the master request frame

> **LINTP_MODE_RESPONSE**: the LinTp requests the schedule table which contains the slave response frame

> **LINTP_MODE_RELEASE**: the LinTp diagnostic mode is finished, an applicative schedule table can be requested

Furthermore it can be differentiated between the standard mode and the flash mode:

> **LIN_STANDARD**: an applicative schedule table is active, diagnostic communication is handled via run-once master request/slave response schedule tables. For example the applicative schedule table is a run-continuous schedule table which is interrupted only for a short time by the LinTp mode request to transmit once the master request or the slave response frame via run-once schedule tables.

> **LIN_FLASH**: will be entered if the Dcm module requests a communication mode which does not allow applicative LIN traffic because the whole band width is needed for flashing. In this mode no applicative schedule tables shall be active. To speed up the flash process run-continuous schedule tables are used for the master request and the slave response frame.

Figure 7-12 shows the state diagram of these modes:

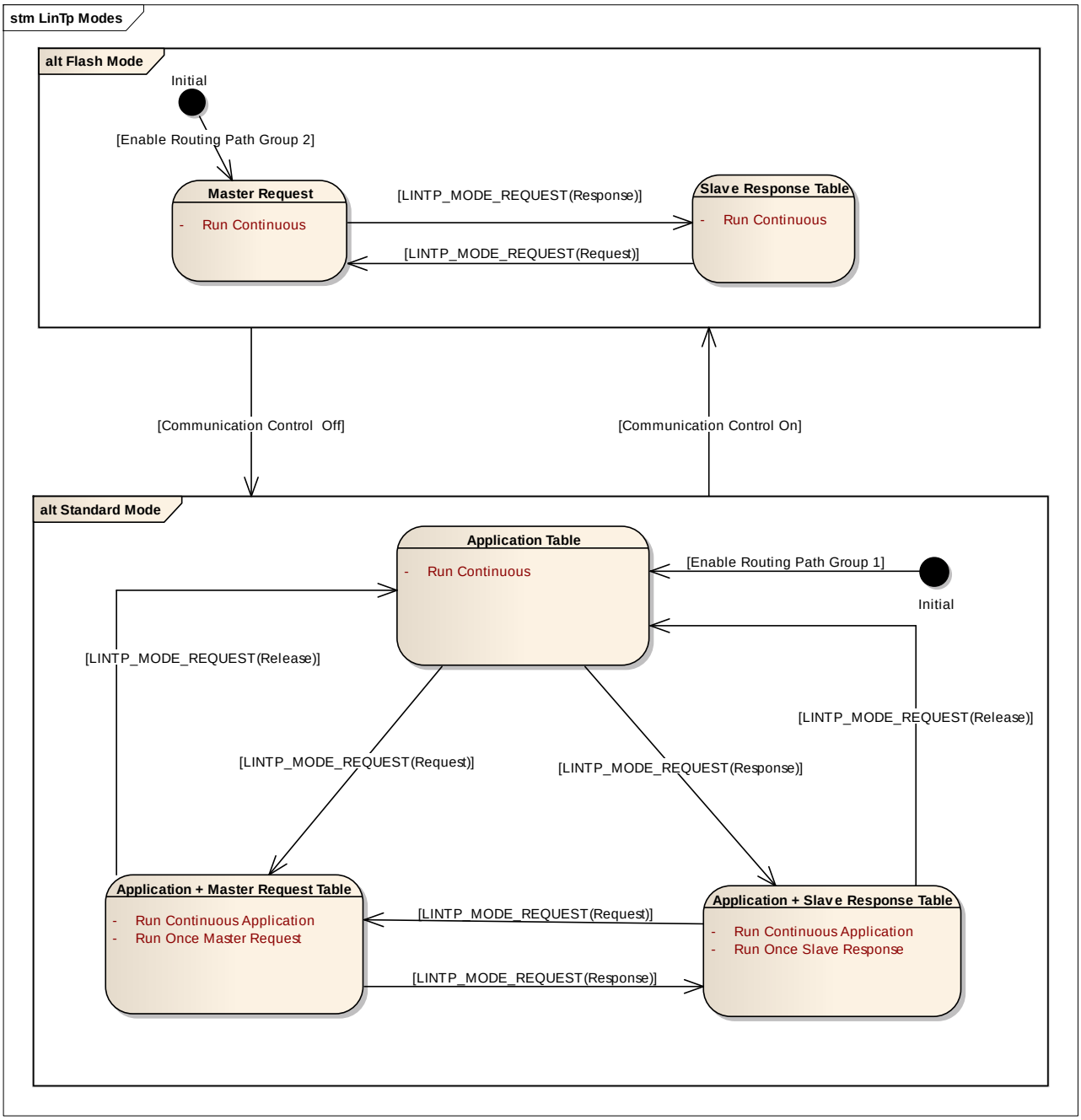

Figure 7-12 LinTp Modes

To fulfill these requirements the following BswM modes and rules can be defined:

> **Generic Mode LIN Diagnostic Mode**: this mode has two possible states: LIN_STANDARD and LIN_FLASH, default value is LIN_STANDARD, mode request processing is BSWM_DEFERRED. This mode will be used to cyclically check the LIN Diagnostic Mode and thus for cyclic run-once schedule table request of the master request/slave response schedule table in LIN_STANDARD mode.

- **Rule Switch LIN Diagnostic Mode**: this rule checks the if the Dcm communication control mode forbids standard communication. The Action List True disables the routing path 1, enables routing path 2, requests the master request run-continuous and calls a user callout function which switches the Generic Mode State to LIN_FLASH.
- **Rule Application Schedule in Standard Mode**: this rule checks if the LinSM changes its mode state to LINSM_BSWM_RUN_SCHEDULE, the Action List True requests the application schedule table and enables the appropriate routing path.
- **Rule Master Request in Standard Mode**: this rule checks if the requested LinTp Mode State changes its mode state to LINTP_MODE_REQUEST and if the Generic Mode LIN Diagnostic Mode is LIN_STANDARD. Since the Generic Mode is a deferred Mode, this rule is arbitrated cyclically within the `BswM_MainFunction()`, so as long as the LinTp Mode State does not change the master request run-once schedule table is requested cyclically.
- **Slave Response in Standard Mode**: this rule checks if the requested LinTp Mode State changes its mode state to LINTP_MODE_RESPONSE and if the Generic Mode LIN Diagnostic Mode is LIN_STANDARD. Since the Generic Mode is a deferred Mode, this rule is arbitrated cyclically within the `BswM_MainFunction()`, so as long as the LinTp Mode State does not change the slave response run-once schedule table is requested cyclically.
- **Rule Master Request in Flash Mode**: this rule checks if the requested LinTp Mode State is LINTP_MODE_REQUEST and if the Generic Mode LIN Diagnostic Mode is LIN_FLASH. The Action List True requests the master request run-continuous schedule table.
- **Rule Slave Response in Flash Mode**: this rule checks if the requested LinTp Mode State is LINTP_MODE_RESPONSE and if the Generic Mode LIN Diagnostic Mode is LIN_FLASH. The Action List True requests the slave response run-continuous schedule table.

Refer to the following sub-chapters for configuration details.

7.5.1 Generic Mode Configuration

The following table contains the recommended configuration parameter values for the Generic Mode:

Attribute Name	Values	Description
Name	LinDiagMode	Name of the Generic Mode.
Generic Mode Initial State	0	Value after initialization is LIN_STANDARD.
Generic Mode Request Processing Type	BSWM_DEFERRED	Rules which use this mode shall be arbitrated cyclically because in LIN_STANDARD mode the master request/Slave response schedule tables must be triggered cyclically.
Generic Mode Trigger Function	TriggerLinDiagMode	This function is used to set the Mode State to either LIN_FLASH or LIN_STANDARD triggered by the Dcm communication control mode. Refer to chapter 7.5.3 for usage of TriggerLinDiagMode.
Use Rte	-	If you use the Rte enabled this option.
Mode State LIN_STANDARD	0	Add a Mode State LIN_STANDARD which has value 0.
Mode State LIN_FLASH	1	Add a Mode State LIN_FLASH which has value 1.

Table 7-7 Recommended configuration for Generic LIN Mode

7.5.2 Rule Configuration

7.5.2.1 Rule for application schedule

The following table contains the recommended configuration parameter values for the application schedule:

Attribute Name	Values	Description
Name	LinRunSchedule	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition	**Mode**: LINSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: LINSM_BSWM_RUN_SCHEDULE	The rule checks if the LinSM mode of the appropriate LIN channel is equal to LINSM_RUN_SCHEDULE .
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Schedule Table Switch application schedule > Pdu Routing Control Path (Disable Group 2) > Pdu Routing Control Path (Enable Group 1)	

Table 7-8 Configuration of rule for application schedule

7.5.2.2 Rule for switching the LIN Diagnostic Mode

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	SwitchLinDiagMode	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM	This condition checks if communication control is active.
Operator	OR	The **OR**-Operator is used to check if at least one Dcm communication control mode is used which forbids standard communication.
Condition 2	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM_NM	This condition checks if communication control is active.
Operator	OR	The **OR**-Operator is used to check if at least one Dcm communication control mode is used which forbids standard communication.
Condition 3	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORM_NM	This condition checks if communication control is active.
Operator	OR	The **OR**-Operator is used to check if at least one Dcm communication control mode is used which forbids standard communication.
Condition 4	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This condition checks if communication control is active.
Operator	OR	The **OR**-Operator is used to check if at least one Dcm communication control mode is used

Attribute Name	Values	Description
		which forbids standard communication.
Condition 5	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM	This condition checks if communication control is active.
Operator	OR	The **OR**-Operator is used to check if at least one Dcm communication control mode is used which forbids standard communication.
Condition 6	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM_NM	This condition checks if communication control is active.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > User Callout **SetLinDiagFlashMode** > Schedule Table Switch master request schedule run-continuous > Pdu Routing Control Path (Disable Group 1) > Pdu Routing Control Path (Enable Group 2)	The Action List True executes all necessary actions for the LIN_FLASH Mode.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > User Callout **SetLinDiagStandardMode** > Schedule Table Switch application schedule > Pdu Routing Control Path (Disable Group 2) > Pdu Routing Control Path (Enable Group 1)	The Action List True executes all necessary actions for the LIN_STANDARD Mode.

Table 7-9 Configuration of rule for switching the LIN Diagnostic Mode

7.5.2.3 Rule for master request in LIN_STANDARD Mode

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	LinTpMasterRequestStandard	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: LINTP_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: LINTP_MODE_REQUEST	This condition checks if the LinTp requests the master request mode.
Operator	AND	The **AND**-Operator is used because both conditions must be true.
Condition 2	**Mode**: LinDiagMode **Relational Operator**: EQUAL **Mode State**: LIN_STANDARD	This condition checks if we are in LIN_STANDARD.
Action List True	**Action List Execution Type**: BSWM_CONDITION **Actions**: > Schedule Table Switch master request schedule run-once	The Execution Type is **BSWM_CONDITION** because in LIN_STANDARD the schedule table must be requested cyclically as long as the rule result is true.

Table 7-10 Configuration of rule for switching the master request schedule in standard mode

7.5.2.4 Rule for slave response in LIN_STANDARD Mode

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	LinTpSlaveResponseStandard	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: LINTP_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: LINTP_MODE_RESPONSE	This condition checks if the LinTp requests the slave response mode.
Operator	AND	The **AND**-Operator is used because both conditions must be true.
Condition 2	**Mode**: LinDiagMode **Relational Operator**: EQUAL **Mode State**: LIN_STANDARD	This condition checks if we are in LIN_STANDARD.
Action List True	**Action List Execution Type**: BSWM_CONDITION **Actions**: > Schedule Table Switch slave response schedule run-once	The Execution Type is **BSWM_CONDITION** because in LIN_STANDARD the schedule table must be requested cyclically.

Table 7-11 Configuration of rule for switching the slave response schedule in standard mode

7.5.2.5 Rule for master request in LIN_FLASH Mode

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	LinTpMasterRequestFlash	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: LINTP_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: LINTP_MODE_REQUEST	This condition checks if the LinTp requests the master request mode.
Operator	AND	The **AND**-Operator is used because both conditions must be true.
Condition 2	**Mode**: LinDiagMode **Relational Operator**: EQUAL **Mode State**: LIN_FLASH	This condition checks if we are in LIN_FLASH.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Schedule Table Switch master request schedule run-continuous	The Execution Type is **BSWM_TRIGGER** because in LIN_FLASH the schedule table must only be requested once upon result change from false to true.

Table 7-12 Configuration of rule for switching the master request schedule in flash mode

7.5.2.6 Rule for slave response in LIN_FLASH Mode

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	LinTpSlaveResponseFlash	Name of the Rule.
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: LINTP_MODE_REQUEST_<channelName> **Relational Operator**: EQUAL **Mode State**: LINTP_MODE_RESPONSE	This condition checks if the LinTp requests the slave response mode.
Operator	AND	The **AND**-Operator is used because both conditions must be true.
Condition 2	**Mode**: LinDiagMode **Relational Operator**: EQUAL **Mode State**: LIN_FLASH	This condition checks if we are in LIN_STANDARD.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Schedule Table Switch slave response schedule run-continuous	The Execution Type is **BSWM_TRIGGER** because in LIN_FLASH the schedule table must only be requested once upon result change from false to true.

Table 7-13 Configuration of rule for switching the slave response schedule in standard mode

7.5.3 User Callouts

As described in chapter 7.5.3 two user callouts are used in the Action Lists:

> SetLinDiagFlashMode

> SetLinDiagStandardMode

These functions are called by the BswM when the Dcm requests LIN_FLASH or LIN_STANDARD mode. The task of this function is to set the Generic Mode State of **LinDiagMode** (refer to chapter 7.5.1), refer to the following examples:

Example

```
void SetLinDiagFlashMode(void)
{
  /* Call the trigger function of Generic Mode "LinDiagMode"
     and set the Mode State to LIN_FLASH */
  TriggerLinDiagMode(BSWM_LinDiagMode_LIN_FLASH);
}

void SetLinDiagStandardMode (void)
{
  /* Call the trigger function of Generic Mode "LinDiagMode"
     and set the Mode State to LIN_STANDARD */
  TriggerLinDiagMode(BSWM_LinDiagMode_LIN_STANDARD);
}
```

7.6 FlexRay Communication Modes

The FrSM notifies the BswM about its state changes. Figure 7-13 shows the possible states:

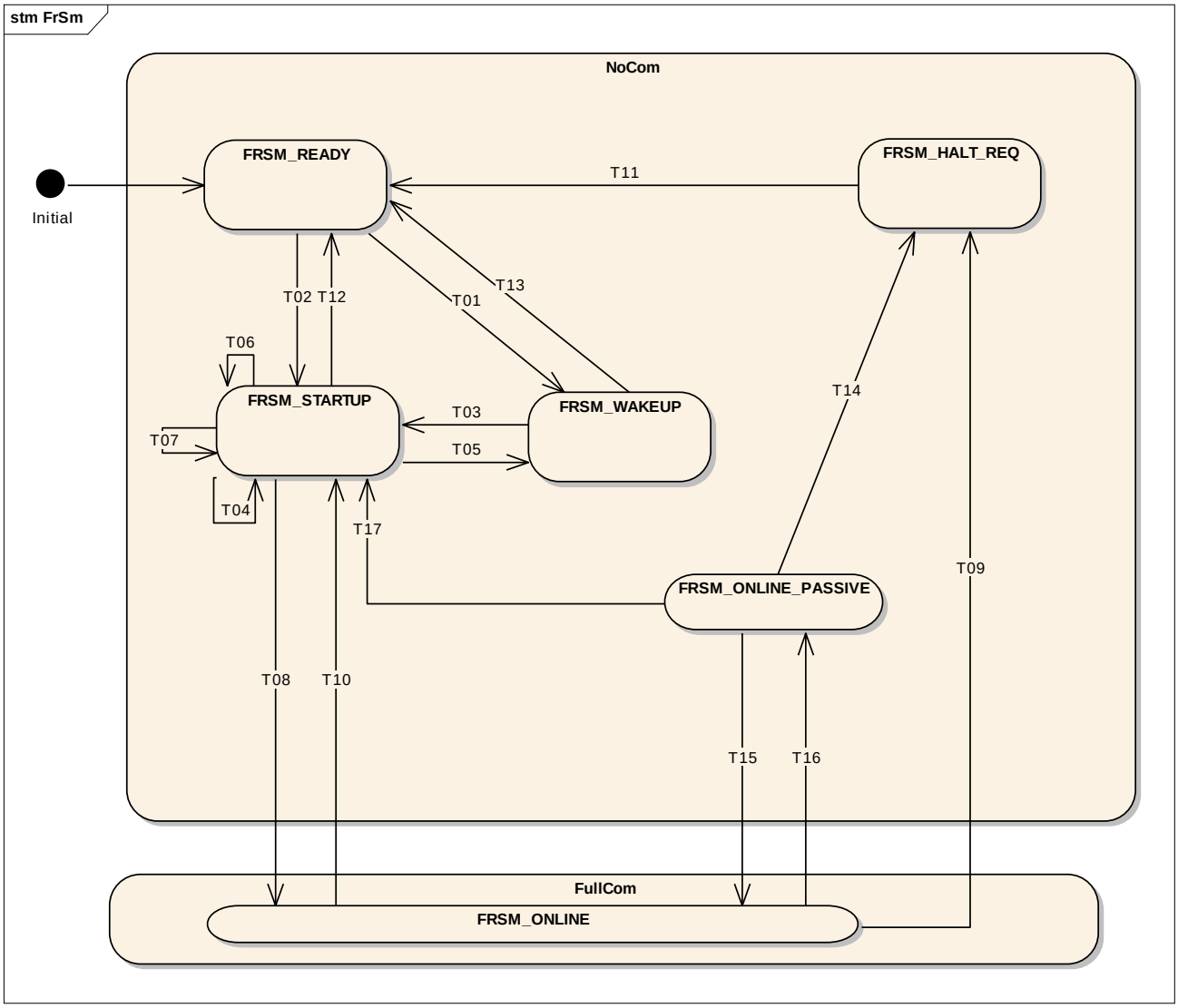

Figure 7-13 FlexRay States

During transitions T08 and T15 the communication has to be started, during transitions T10, T16 and T09 the communication has to be stopped. To fulfill these requirements a rule is configured which uses the FrSM Mode FRSM_MODE_INDICATION and the Mode State FRSM_BSWM_ONLINE in its condition to identify whether the communication has to be started (means FrSM enters state FRSM_ONLINE) or the communication has to be stopped (means FrSM leaves state FRSM_ONLINE):

7.6.1 Rule Switch_Rx_Tx_DM_FlexRay

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Rx_Tx_DM_FlexRay	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the state FRSM_ONLINE is entered for the first time.
Condition 1	**Mode**: FRSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: FRSM_BSWM_ONLINE	This condition checks if the FrSM is in state FRSM_ONLINE.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Rx Pdu Group > Pdu Group Switch, start Tx Pdu Group > Deadline Monitoring, enable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Rx Pdu Group > Pdu Group Switch, stop Tx Pdu Group > Deadline Monitoring, disable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-14 Configuration of rule for starting and stopping the communication for FlexRay

Screenshot of the example rule configuration:

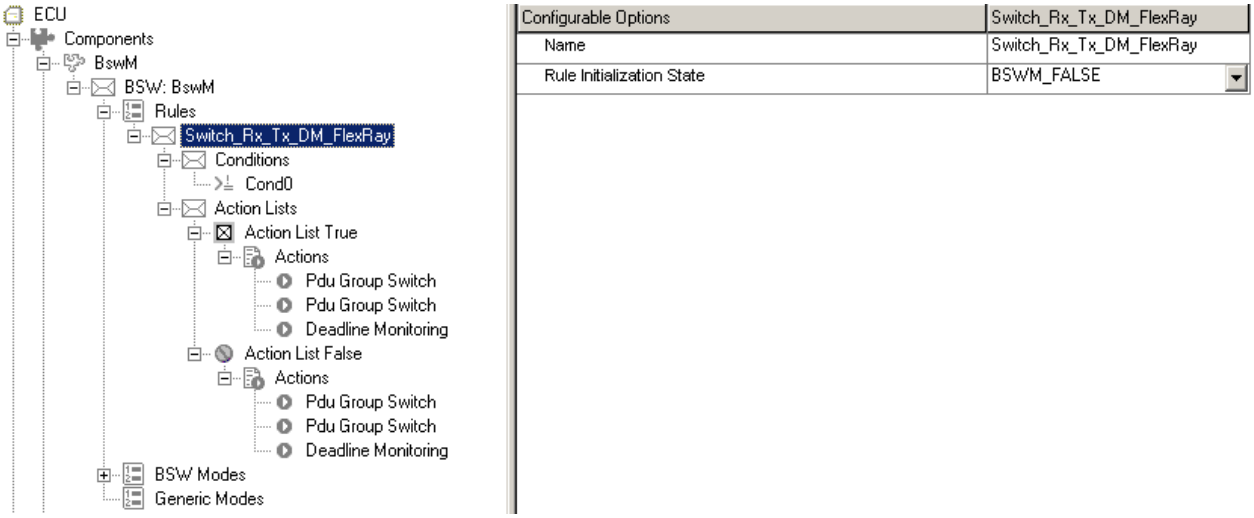

Figure 7-14 Screenshot FlexRay example rule

If Dcm communication control is used, the Dcm Mode DCM_COM_MODE_REQUEST must be considered in the rule configuration. It is not possible to switch Rx, Tx and DM at once as described in rule **Switch_Rx_Tx_DM_Flexray**, there must be a rule for Rx/DM and another rule for Tx, because the Dcm communication Modes control each communication state separately. So two rules are created for Tx and Rx/DM:

7.6.2 Rule Switch_Tx_FlexRay

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Tx_FlexRay	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: FRSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: FRSM_BSWM_ONLINE	This condition checks if the FrSM is in state FRSM_ONLINE.
Operator	AND	All conditions must be fulfilled.
Condition 2	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**:	This Dcm Mode State must not forbid Tx.

Attribute Name	Values	Description
	DCM_DISABLE_RX_TX_NORM_NM	
Operator	AND	All conditions must be fulfilled.
Condition 3	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 4	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM_NM	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 5	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM	This Dcm Mode State must not forbid Tx.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Tx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Tx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-15 Configuration of rule for starting and stopping Tx for FlexRay with Dcm Com Control

7.6.3 Rule Switch_Rx_DM_FlexRay

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Rx_DM_FlexRay	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure

Attribute Name	Values	Description
		that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: FRSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: FRSM_BSWM_ONLINE	This condition checks if the FrSM is in state FRSM_ONLINE.
Operator	AND	All conditions must be fulfilled.
Condition 2	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 3	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator:** NOT EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM_NM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 4	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORM_NM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 5	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This Dcm Mode State must not forbid Rx.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Rx Pdu Group > Enable the Deadline Monitoring for the Rx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**:	The Execution Type is

Attribute Name	Values	Description
	BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Rx Pdu Group > Disable the Deadline Monitoring for the Rx Pdu Group	**BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-16 Configuration of rule for starting and stopping Rx and DM for FlexRay with Dcm Com Control

7.7 NmFiatB and NmFiatC Communication Modes

The NmFiatB and NmFiatC notify the BswM about their state changes via the callback function BswM_Nm_StateChangeNotification. Figure 7-15 shows the different communication states of Rx, Tx and DM in the appropriate Nm states:

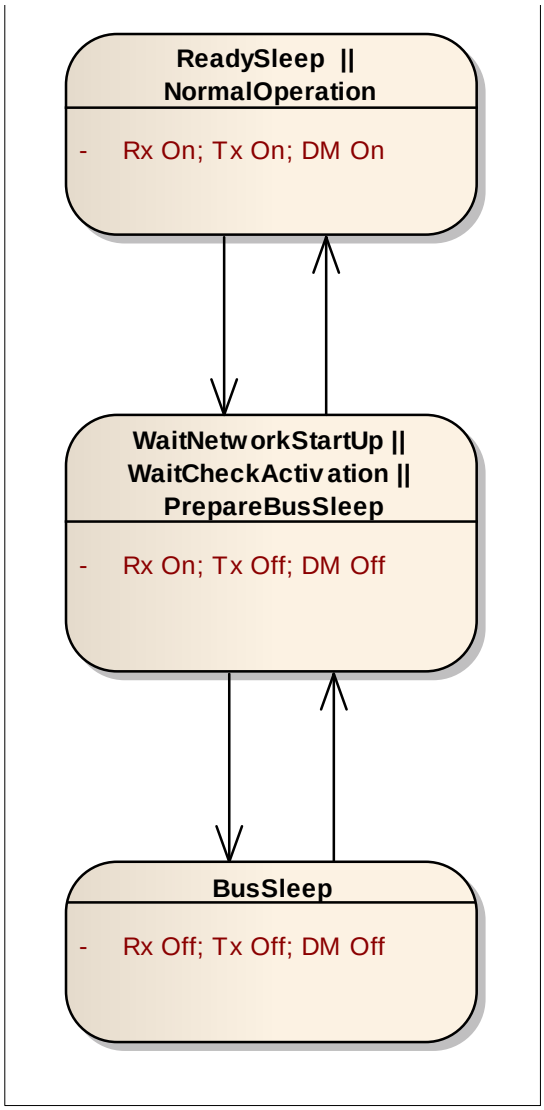

Figure 7-15 NmFiatB and NmFiatC state transitions

The following sub-chapters describe example rules which fulfill the recommended communication states shown in Figure 7-15.

Caution
If NmFiatB or NmFiatC are used, the CAN communication modes as described in chapter 7.1 must not be used.

The rules for NmFiatB and NmFiatC must be configured manually. The standard rules button cannot be used in this case.

7.7.1 Rule Switch_Tx_DM

This rule starts/stops the Tx communication path and the Rx deadline monitoring. The following table contains the recommended configuration parameter:

Rule Switch_Tx_DM for NMFiat Class B:

Attribute Name	Values	Description
Name	Switch_Tx_DM	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: NM_STATE_NORMAL_OPERATION	This condition checks if the Nm is in state NM_STATE_NORMAL_ OPERATION.
Operator	**OR**	Condition 1 OR Condition 2 must be true.
Condition 2	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: NM_STATE_READY_SLEEP	This condition checks if the Nm is in state NM_STATE_READY_ SLEEP.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Tx Pdu Group > Deadline Monitoring, enable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**:	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the

Attribute Name	Values	Description
	> Pdu Group Switch, stop Tx Pdu Group > Deadline Monitoring, disable the Rx DM	result changes from true to false.

Table 7-17 Configuration of rule for starting and stopping the Tx communication for NmFiat B

Rule Switch_Tx_DM for NMFiat Class C:

Attribute Name	Values	Description
Name	Switch_Tx_DM	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: CANSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: CANSM_BSWM_FULL_COMMUNICATION	This condition checks if the CanSm is in state CANSM_BSWM_FULL_COMMUNICATION.
Operator	**AND**	All conditions must be fulfilled.
Condition 2	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: NM_STATE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_BUS_SLEEP.
Operator	**AND**	All conditions must be fulfilled.
Condition 3	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: NM_STATE_PREPARE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_PREPARE_BUS_SLEEP.
Operator	**AND**	All conditions must be fulfilled.
Condition 4	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT EQUAL	This condition checks if the Nm is not in state NM_STATE_WAIT_CHECK_ACTIVATION.

Attribute Name	Values	Description
	Mode State: NM_STATE_WAIT_CHECK_ACTIVATION	
Operator	**AND**	All conditions must be fulfilled.
Condition 5	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: NM_STATE_WAIT_NETWORK_STARTUP	This condition checks if the Nm is not in state NM_STATE_WAIT_NETWORK_STARTUP.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Tx Pdu Group > Deadline Monitoring, enable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Tx Pdu Group > Deadline Monitoring, disable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-18 Configuration of rule for starting and stopping the Tx communication for NmFiat C

Screenshot of the example rule configuration:

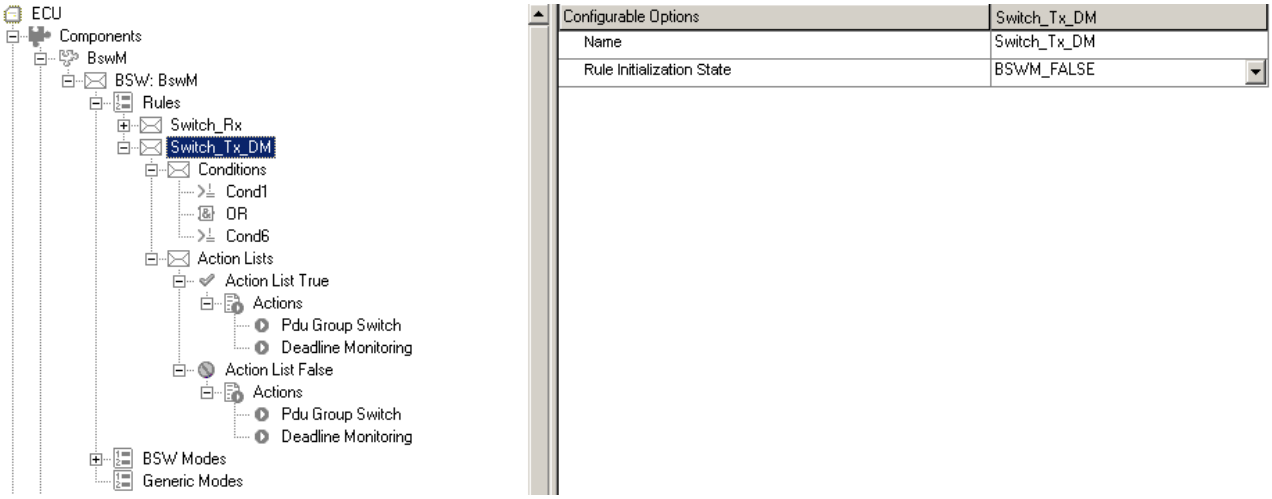

Figure 7-16 Screenshot example rule for Tx communication with NmFiatB and NmFiatC

7.7.2 Rule Switch_Rx

This rule starts/stops the Rx communication path. The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Rx	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_BUS_SLEEP.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Rx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Rx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-19 Configuration of rule for starting and stopping the Rx communication for NmFiatB and NmFiatC

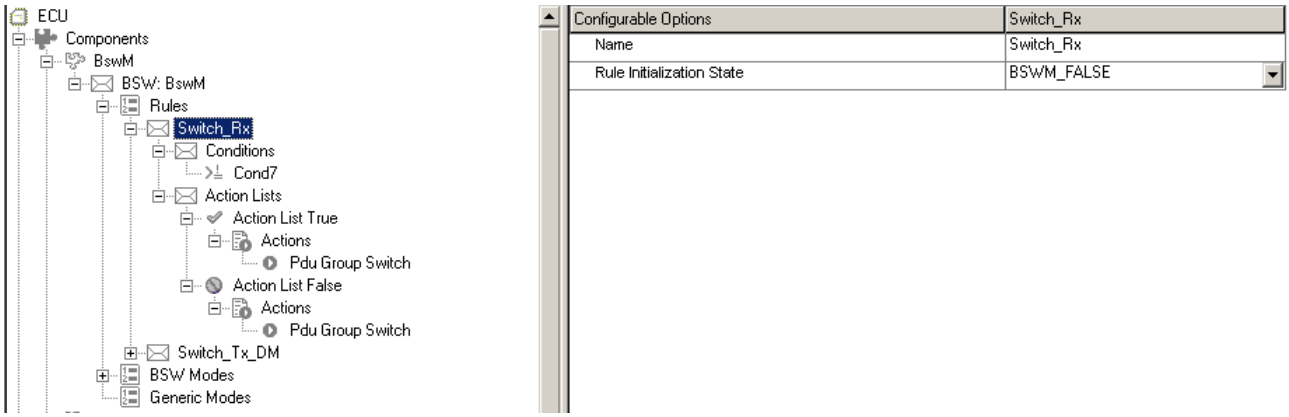

Figure 7-17 Screenshot example rule for Rx communication with NmFiatB and NmFiatC

7.7.3 Rule Switch_Tx_DM with communication control

The following table contains the recommended configuration parameter:

Rule Switch_Tx_DM for Fiat Class B:

Attribute Name	Values	Description
Name	Switch_Tx_DM	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: CANSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: CANSM_BSWM_FULL_COMMUNICATION	This condition checks if the CanSM is in state CANSM_BSWM_FULL_COMMUNICATION.
Operator	AND	All conditions must be fulfilled.
Condition 2	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_BUS_SLEEP.
Operator	AND	All conditions must be fulfilled.
Condition 3	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_PREPARE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_PREPARE_BUS_SLEEP.
Operator	AND	All conditions must be fulfilled.
Condition 4	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_WAIT_CHECK_ACTIVATION	This condition checks if the Nm is not in state NM_STATE_WAIT_CHECK_ACTIVATION.
Operator	AND	All conditions must be fulfilled.
Condition 5	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL	This condition checks if the Nm is not in state NM_STATE_WAIT_NETWORK_STARTUP.

Attribute Name	Values	Description
	Mode State: NM_STATE_WAIT_NETWORK_STARTUP	
Operator	AND	All conditions must be fulfilled.
Condition 6	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORM_NM	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 7	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 8	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM_NM	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 9	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM	This Dcm Mode State must not forbid Tx.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Tx Pdu Group > Deadline Monitoring, enable Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Tx Pdu Group > Deadline Monitoring, disable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-20 Configuration of rule for starting and stopping Tx for NmFiatB with Dcm Com Control

Rule Switch_Tx_DM for Fiat Class C:

Attribute Name	Values	Description
Name	Switch_Tx_DM	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: CANSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: CANSM_BSWM_FULL_COMMUNICATION	This condition checks if the CanSm is in state CANSM_BSWM_FULL_COMMUNICATION.
Operator	AND	All conditions must be fulfilled.
Condition 2	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_BUS_SLEEP.
Operator	AND	All conditions must be fulfilled.
Condition 3	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_PREPARE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_PREPARE_BUS_SLEEP.
Operator	AND	All conditions must be fulfilled.
Condition 4	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_WAIT_CHECK_ACTIVATION	This condition checks if the Nm is not in state NM_STATE_WAIT_CHECK_ACTIVATION.
Operator	AND	All conditions must be fulfilled.
Condition 5	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_WAIT_NETWORK_STARTUP	This condition checks if the Nm is not in state NM_STATE_WAIT_NETWORK_STARTUP.
Operator	AND	All conditions must be fulfilled.

Attribute Name	Values	Description
Condition 6	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORM_NM	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 7	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 8	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM_NM	This Dcm Mode State must not forbid Tx.
Operator	AND	All conditions must be fulfilled.
Condition 9	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_ENABLE_RX_DISABLE_TX_NORM	This Dcm Mode State must not forbid Tx.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, start Tx Pdu Group > Deadline Monitoring, enable Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Tx Pdu Group > Deadline Monitoring, disable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-21 Configuration of rule for starting and stopping Tx for NmFiatC with Dcm Com Control

7.7.4 Rule Switch_Rx with communication control

The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Rx	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE, this and Action List Execution Type **BSWM_TRIGGER** ensure that the Action List True is not executed until the evaluation of the conditions, results in BSWM_TRUE.
Condition 1	**Mode**: NMFIAT_MODE_INDICATION_<channelName> **Relational Operator**: NOT_EQUAL **Mode State**: NM_STATE_BUS_SLEEP	This condition checks if the Nm is not in state NM_STATE_BUS_SLEEP.
Operator	AND	All conditions must be fulfilled.
Condition 2	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 3	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_ENABLE_TX_NORM_NM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 4	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORM_NM	This Dcm Mode State must not forbid Rx.
Operator	AND	All conditions must be fulfilled.
Condition 5	**Mode**: DCM_COM_MODE_REQUEST_<channelName> **Relational Operator**: NOT EQUAL **Mode State**: DCM_DISABLE_RX_TX_NORMAL	This Dcm Mode State must not forbid Rx.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions**:	The Execution Type is **BSWM_TRIGGER** because the actions shall

Attribute Name	Values	Description
	> Pdu Group Switch, start Rx Pdu Group > Enable the Deadline Monitoring for the Rx Pdu Group	only be executed when the result changes from false to true.
Action List False	**Action List Execution Type**: BSWM_TRIGGER **Actions**: > Pdu Group Switch, stop Rx Pdu Group > Disable the Deadline Monitoring for the Rx Pdu Group	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from true to false.

Table 7-22 Configuration of rule for starting and stopping Rx for NmFiatB and NmFiatC with Dcm Com Control

7.8 Partial Network Cluster Modes

If the PNC support is enabled in the ComM configuration the BswM will provide a BSW Mode for each available PNC. A simple use-case for the BswM is to enable/disable the Com deadline monitoring according to the requested ComM PNC Mode:

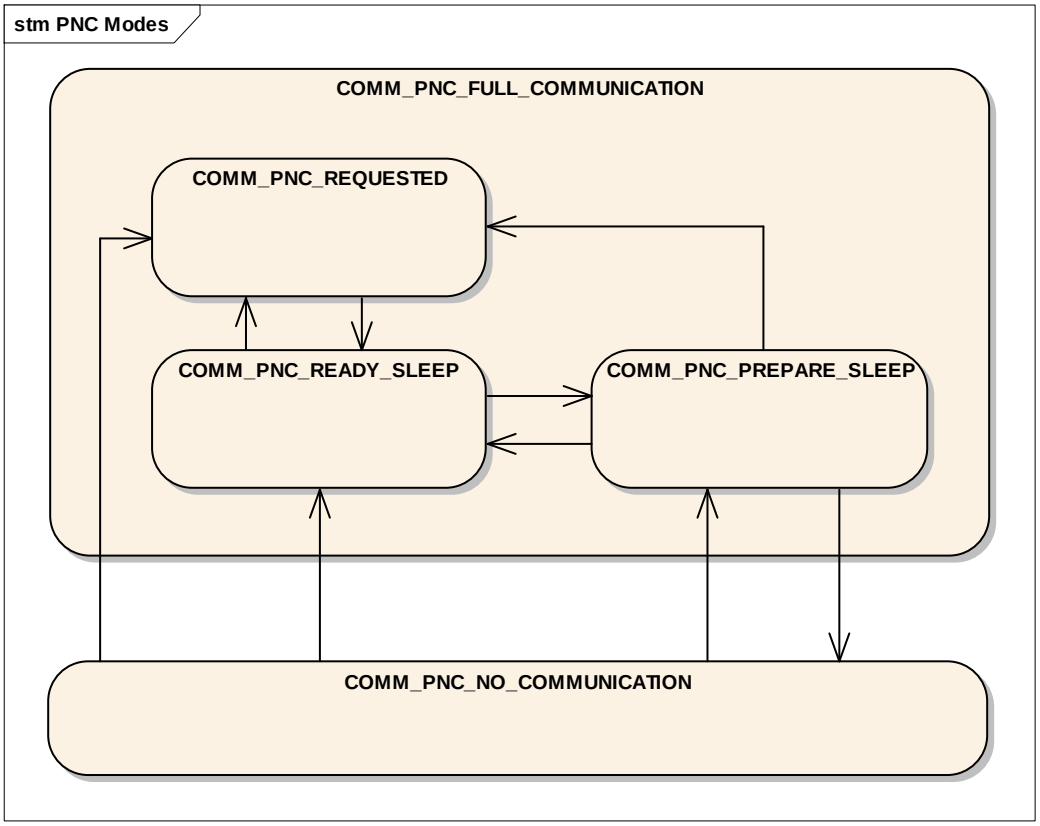

Figure 7-18 ComM PNC Modes

The Tx and Rx IPDU groups are started/stopped for the communication channel dependent on the bus type as described in chapter 7. For the PNCs the Com Rx deadline monitoring is handled separately dependent on the PNC Mode:

> **COMM_PNC_FULL_COMMUNICATION**: the Rx deadline monitoring is enabled for the appropriate IPDU group.

> **COMM_PNC_NO_COMMUNICATION**: the Rx deadline monitoring is disabled for the appropriate IPDU group.

The communication modes of a specific channel enable the Rx deadline monitoring for all Rx IPDU groups which belong to that channel, every time this happens the deadline monitoring for Rx IPDU groups which belong to a PNC which is in COMM_PNC_NO_COMMUNICATION mode must be disabled. As a result the following BswM configuration is recommended:

> For each PNC: a Rule which enables the deadline monitoring, refer to chapter 7.8.1.

> For each PNC: a Rule which disables the deadline monitoring, refer to chapter 7.8.2.

> For each communication channel: extension of the Rule which enables the deadline monitoring: disable the deadline monitoring afterwards again for PNCs which are in COMM_PNC_NO_COMMUNICATION, refer to 7.8.3.

7.8.1 Rule Switch_On_PNC

This rule enables Rx deadline monitoring for a specific PNC. The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_On_PNC<index>	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE.
Condition 1	**Mode**: COMM_MODE_PNC<index> **Relational Operator**: NOT_EQUAL **Mode State**: COMM_PNC_NO_COMMUNICATION	This condition checks if the PNC mode is not equal to COMM_PNC_NO_ COMMUNICATION (means the PNC is in COMM_PNC_FULL_ COMMUNICATION)
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions:** > Deadline Monitoring, enable the Rx DM	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the result changes from false to true.

Table 7-23 Rule Switch_On_PNC

7.8.2 Rule Switch_Off_PNC

This rule disables Rx deadline monitoring for a specific PNC. The following table contains the recommended configuration parameter:

Attribute Name	Values	Description
Name	Switch_Off_PNC<index>	Name of the Rule
Rule Initialization State	BSWM_TRUE	Value after initialization is BSWM_TRUE because the PNC is state COMM_PNC_NO_ COMMUNICATION after initialization.
Condition 1	**Mode**: COMM_MODE_PNC<index> **Relational Operator**: EQUAL **Mode State**: COMM_PNC_NO_COMMUNICATION	This condition checks if the PNC mode is not equal to COMM_PNC_NO_ COMMUNICATION (means the PNC is in COMM_PNC_FULL_ COMMUNICATION)
Action List True	**Action List Execution Type**: BSWM_CONDITION **Actions:** > Deadline Monitoring, disable the Rx DM	The Execution Type is **BSWM_CONDITION** because the actions shall be executed every time the Rule result is TRUE.

Table 7-24 Rule Switch_Off_PNC

7.8.3 Extension for communication start

For example a CAN channel enters state CANSM_BSWM_FULL_COMMUNICATION the IPDU groups are started and the Rx deadline monitoring is enabled. Afterwards the deadline monitoring for Rx IPDU groups of PNCs which are in COMM_PNC_NO_COMMUNICATION mode must be disabled again, so the standard Rule for switching the deadline monitoring is extended:

Attribute Name	Values	Description
Name	StdRule_Switch_DM_<channelName>	Name of the Rule
Rule Initialization State	BSWM_FALSE	Value after initialization is BSWM_FALSE because the state is not equal to CANSM_BSWM_FULL_COMMUNICATION.
Condition 1	**Mode**: CANSM_MODE_INDICATION_<channelName> **Relational Operator**: EQUAL **Mode State**: CANSM_BSWM_FULL_COMMUNICATION	This condition checks if the CanSM mode is equal to CANSM_BSWM_FULL_COMMUNICATION.
Action List True	**Action List Execution Type**: BSWM_TRIGGER **Actions:** > Deadline Monitoring, enable the Rx DM, repeat this action for all Rx IPDU groups > Action Rule: Switch_Off_PNC<index_0> > Action Rule: Switch_Off_PNC<index_1> > … **Caution** Consider the order of the Actions: the Switch_Off_Pnc-Rules must be executed after enabling the deadline monitoring.	The Execution Type is **BSWM_TRIGGER** because the actions shall only be executed when the Rule result changes. Execute all Switch_Off_PNC Rules (refer to chapter 7.8.2) which check if the appropriate PNC is in state COMM_PNC_NO_COMMUNICATION, if true the deadline monitoring for this PNC is disabled again.

Table 7-25 Extension of Rule StdRule_Switch_DM

8 AUTOSAR Standard Compliance

8.1 Deviations

-

8.2 Additions/ Extensions

The BswM supports NmFiat Class B, NmFiat Class C and NmOsekmode arbitrations and timers.

8.3 Limitations

8.3.1 Mode arbitration and mode control

The BswM supports more than one condition per rule but supports only the operators **AND** and **OR** for comparison between conditions. Action lists can only contain simple actions or rules, links to other action lists are not supported.

8.3.2 BSW modes

The following BSW mode indication functions do not implement any functionality at the moment:

> BswM_LinSM_ScheduleEnd_Notification

9 Glossary and Abbreviations

9.1 Glossary

Term	Description
GENy	Generation tool for CANbedded and MICROSAR components

Table 9-1 Glossary

9.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DM	Deadline Monitoring
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PNC	Partial Network Cluster
PPort	Provide Port
RPort	Require Port
RTE	Runtime Environment
Rx	Reception
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification
Tx	Transmission

Table 9-2 Abbreviations

10 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com