| Document Title | Specification of Bulk NvData Manager |
|---|---|
| **Document Category** | Software specification (SWS) |
| **Document UID** | 949 |
| **Document Status** | published |
| **Part of AUTOSAR Standard** | Classic Platform |
| **Part of Standard Release** | R26-11 |

| Document Change History | | | |
|---|---|---|---|
| **Date** | **Release** | **Changed by** | **Description** |
| 2026-11-26 | R26-11 | AUTOSAR Release Management | • Removed extended production errors<br>• Removed BndM_GetVersionInfo API<br>• Editorial changes |
| 2025-11-27 | R25-11 | AUTOSAR Release Management | • No content changes |
| 2024-11-27 | R24-11 | AUTOSAR Release Management | • No content changes |
| 2023-11-23 | R23-11 | AUTOSAR Release Management | • No content changes |
| 2022-11-24 | R22-11 | AUTOSAR Release Management | • editorial changes |
| 2021-11-25 | R21-11 | AUTOSAR Release Management | • No content changes |
| 2020-11-30 | R20-11 | AUTOSAR Release Management | • callback functions renamed<br>• limitation added<br>• specification item prefix adapted<br>• editorial changes |
| 2019-11-28 | R19-11 | AUTOSAR Release Management | • Initial release |

**Disclaimer**

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

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

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

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

# Table of Contents

1 Introduction and functional overview 5
2 Acronyms and Abbreviations 6
3 Related documentation 7
3.1 Input documents & related standards and norms . . . 7
3.2 Related specification . . . 7
4 Constraints and assumptions 8
4.1 Limitations . . . 8
4.2 Applicability to car domains . . . 8
5 Dependencies to other modules 9
6 Requirements Tracing 10
7 Functional specification 11
7.1 Error Classification . . . 13
7.1.1 Development Errors . . . 13
7.1.2 Runtime Errors . . . 13
7.1.3 Production Errors . . . 14
8 API specification 15
8.1 Imported types . . . 15
8.2 Type definitions . . . 15
8.2.1 BndM_ConfigType . . . 15
8.2.2 BndM_BlockIdType . . . 15
8.2.3 BndM_Block<BlockId.Shortname>Type . . . 16
8.2.4 BndM_Result . . . 16
8.3 Function definitions . . . 16
8.3.1 BndM_Init . . . 16
8.3.2 BndM_GetBlockPtr . . . 17
8.3.3 BndM_WriteStart . . . 17
8.3.4 BndM_WriteBlock . . . 18
8.3.5 BndM_WriteFinalize . . . 18
8.3.6 BndM_WriteCancel . . . 19
8.3.7 Callback notifications . . . 19
8.3.7.1 BndM_JobEndNotification . . . 20
8.3.7.2 BndM_JobErrorNotification . . . 20
8.4 Scheduled functions . . . 20
8.4.1 BndM_MainFunction . . . 21
8.5 Expected interfaces . . . 21
8.5.1 Mandatory interfaces . . . 21
8.5.2 Optional interfaces . . . 21
8.5.3 Configurable interfaces . . . 22

8.5.3.1 xxx_BndMWriteStartFinish . . . 22
8.5.3.2 xxx_BndMWriteBlockFinish . . . 22
8.5.3.3 xxx_BndMWriteFinalizeFinish . . . 23
8.6 Service Interfaces . . . 23
9 Sequence diagrams 24
10 Configuration specification 25
10.1 How to read this chapter . . . 25
10.2 Containers and configuration parameters . . . 25
10.2.1 BndM . . . 25
10.2.2 BndMGeneral . . . 25
10.2.3 BndMBlockDescriptor . . . 27
10.2.4 BndMCallbackBlock . . . 28
10.2.5 BndMCallbackGeneral . . . 29
10.3 Published Information . . . 30
A Change history of AUTOSAR traceable items 31
A.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 31
A.1.1 Added Specification Items in R26-11 . . . 31
A.1.2 Changed Specification Items in R26-11 . . . 31
A.1.3 Deleted Specification Items in R26-11 . . . 31
A.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 31
A.2.1 Added Specification Items in R25-11 . . . 31
A.2.2 Changed Specification Items in R25-11 . . . 31
A.2.3 Deleted Specification Items in R25-11 . . . 32
A.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 32
A.3.1 Added Specification Items in R24-11 . . . 32
A.3.2 Changed Specification Items in R24-11 . . . 32
A.3.3 Deleted Specification Items in R24-11 . . . 32
B Not applicable requirements 33

# 1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module BulkNvDataManager.

The demand of non-volatile bulk data is increasing for use-case like variant-coding [1]. Such data is used frequently, but rarely updated. The BulkNvDataManager offers in contrast to the NvM an API to read the data directly from flash memory. In consequence a RAM mirror is avoided, but the writing of the data is more complex.

[1] Variant coding is a vehicle specific dataset which is calculated in the production for each vehicle (and of course stored in the production).

# 2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the BulkNvDataManager that are not included in the [1, AUTOSAR Glossary].

# 3 Related documentation

## 3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

## 3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for BulkNvDataManager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for BulkNvDataManager.

# 4 Constraints and assumptions

## 4.1 Limitations

The synchronization of a potential parallel access (e.g. FlashEEPROMEmulation) to the underlying flash driver is not part of this AUTOSAR release.

Currently only PFlash writing with A/B Sector switch, present in high end microcontrollers, is supported. This limits the applicability of BndM to architectures supporting this feature.

## 4.2 Applicability to car domains

No content.

# 5 Dependencies to other modules

This module depends on the capabilities of the underlying flash driver.

# 6 Requirements Tracing

The following tables reference the requirements specified in <CITATIONS_OF_CONTRIBUTED_DOCUMENTS> and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

| Requirement | Description | Satisfied by |
|---|---|---|
| No upstream requirements specified. | | |

**Table 6.1: Requirements Tracing**

# 7 Functional specification

In general the concept how the BulkNvDataManager will manage its flash memory is vendor-specific.

The base idea is to have an A/B switching of the data blocks. This means the complete Bulk NvData will be stored in partition A. When the writing is started (BndM_WriteStart) the B partition needs to be erased. The updated blocks (BndM_WriteBlock) will be written to partition B. The finalization (BndM_WriteFinalize) will finally make partition B consistent (e.g. by coping the not updated blocks over to partition B) and switch the active partition to B (further calls to BndM_GetBlockPtr will point to the data in the partition B). Nevertheless the vendor solution could consider alternative solutions like an update through a FlashBootloader.

**[SWS_BndM_00001]** ⌈The BndM shall manage its BndM blocks (BndMBlockDescriptor) in the direct accessible memory (i.e. via pointer).⌋

**[SWS_BndM_00002]** ⌈A call of BndM_GetBlockPtr shall deliver the base pointer to the corresponding BndM block (BndMBlockDescriptor) in the currently active partition.⌋

**[SWS_BndM_00003]** ⌈A call of BndM_WriteStart shall trigger the preparation of the 2nd (free) partition.⌋

Note: Depending on the implemented strategy the preparation takes more time. This could be coordinated within the BndM_MainFunction. Note: In case of direct writing access to flash the flash-page needs to be erased.

Caveat: Depending on the hardware a parallel read and write access to code flash is not possible. In this case the overall ECU needs to be in a writing mode (e.g. FlashBootloader context or all other tasks are interrupted/stopped).

**[SWS_BndM_00007]** ⌈After preparation of the 2nd (free) partition [SWS_BndM_00003] is successfully finished (writing to the 2nd partition is possible) the callback Xxx_BndMWriteStartFinish with the result set to E_OK shall be triggered in the context of the BndM_MainFunction.⌋

**[SWS_BndM_00014]** ⌈A call of BndM_WriteStart shall be rejected with the errorCode E_NOT_OK, if the call is done within an active writing phase (phase between BndM_WriteStart and BndM_WriteFinalize).⌋

**[SWS_BndM_00004]** ⌈A call of BndM_WriteBlock shall trigger the writing of the data to the 2nd (unused) partition. The data (ImplementationDataType) shall be not modified to allow a pointer access.⌋

**[SWS_BndM_00008]** ⌈After writing of [SWS_BndM_00004] the 2nd (free) partition is finished the callback Xxx_BndMWriteBlockFinish with the result set to E_OK shall be triggered in the context of the BndM_MainFunction.⌋

**[SWS_BndM_00011]** ⌈A call of BndM_WriteBlock shall be rejected with the error-Code E_NOT_OK, if the call is done without a previous call of BndM_WriteStart. or while another writing of the same or another block is ongoing or the call is done within the finalization mode of the BndM.⌋

**[SWS_BndM_00012]** ⌈A call of BndM_WriteBlock shall be rejected with the error-Code E_NOT_OK, if the call is done while another writing of the same or another block is ongoing.⌋

**[SWS_BndM_00013]** ⌈A call of BndM_WriteBlock shall be rejected with the error-Code E_NOT_OK, if the call is done within or after the finalization mode of the BndM.⌋

**[SWS_BndM_00005]** ⌈A call of BndM_WriteFinalize shall trigger the finalization of the 2nd (unused) partition. In background the BndM shall make the 2nd (unused) partition consistent by coping all unchanged BndMBlockDescriptor to the 2nd (unused) partition. If the finalization is successful the BndM shall make the 2nd (unused) partition to the active partition and trigger the callback Xxx_BndMWriteFinalizeFinish with the result set to E_OK.⌋

Note: Further calls to BndM_GetBlockPtr will point to the data in the 2nd (now active) partition after the finalization is successful.

**[SWS_BndM_00009]** ⌈If the finalization is NOT successful (the 2nd partition is not consistent and could therefore not be used) the BndM shall keep the current active partition as the active partition and trigger the callback Xxx_BndMWriteFinalize-Finish with the result set to E_NOT_OK.⌋

**[SWS_BndM_00010]** ⌈A call of BndM_WriteFinalize without a previously called BndM_WriteStart or within the finalization mode of the BndM the DET BNDM_E_WRONG_SEQUENCE error shall be thrown.⌋

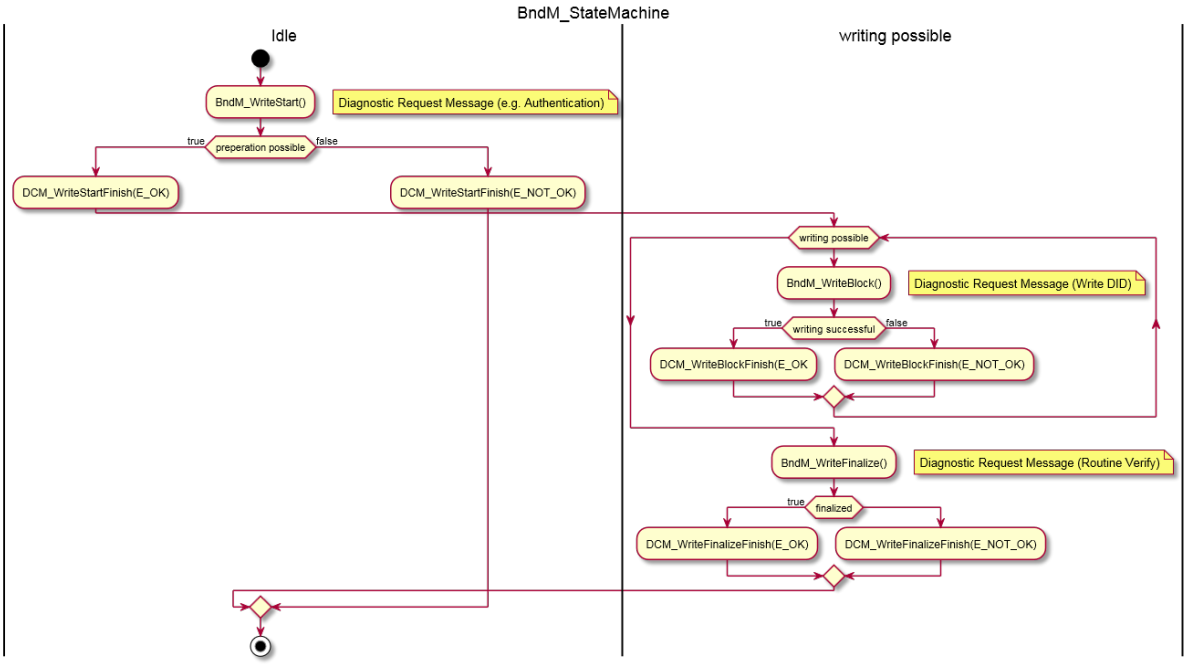



**Figure 7.1: Figure BndMStateMachine**

## 7.1 Error Classification

Chapter [2, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

### 7.1.1 Development Errors

**[SWS_BndM_00006] Definition of development errors in module BndM** ⌈

| *Type of error* | *Related error code* | *Error value* |
|---|---|---|
| API service called with wrong parameter | BNDM_E_PARAM | 0x01 |
| API called in wrong sequence | BNDM_E_WRONG_SEQUENCE | 0x02 |

⌋

### 7.1.2 Runtime Errors

There are no runtime errors.

### 7.1.3 Production Errors

There are no production errors.

# 8 API specification

## 8.1 Imported types

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

**[SWS_BndM_01018] Definition of imported datatypes of module BndM** ⌈

| *Module* | *Header File* | *Imported Type* |
|---|---|---|
| Std | Std_Types.h | Std_ReturnType |

⌋

## 8.2 Type definitions

### 8.2.1 BndM_ConfigType

**[SWS_BndM_01001] Definition of datatype BndM_ConfigType** ⌈

| | | |
|---|---|---|
| ***Name*** | BndM_ConfigType | |
| ***Kind*** | Structure | |
| ***Elements*** | implementation specific | |
| | ***Type*** | – |
| | ***Comment*** | – |
| ***Description*** | This type of the external data structure shall contain the post build initialization data for the BndM. | |
| ***Available via*** | bndm.h | |

⌋

### 8.2.2 BndM_BlockIdType

**[SWS_BndM_01002] Definition of datatype BndM_BlockIdType** ⌈

| | | | |
|---|---|---|---|
| ***Name*** | BndM_BlockIdType | | |
| ***Kind*** | Type | | |
| ***Derived from*** | uint16 | | |
| ***Range*** | 0..65535 | – | – |
| ***Description*** | Unique identification of an bulk nv block. The BndM_BlockId is assigned by the BndM. | | |
| ***Available via*** | bndm.h | | |

⌋

### 8.2.3 BndM_Block<BlockId.Shortname>Type

**[SWS_BndM_01003] Definition of datatype BndM_Block{BlockId.Shortname}Type ⌈**

| | |
|---|---|
| ***Name*** | BndM_Block{BlockId.Shortname}Type |
| ***Kind*** | Structure |
| ***Description*** | The elements of this structure data type is the C-structured representation of the configured ImplementationDataPrototype. |
| ***Available via*** | bndm_externals.h |

⌋

### 8.2.4 BndM_Result

**[SWS_BndM_01017] Definition of datatype BndM_ResultType ⌈**

| | | | |
|---|---|---|---|
| ***Name*** | BndM_ResultType | | |
| ***Kind*** | Type | | |
| ***Derived from*** | uint8 | | |
| ***Range*** | E_OK | 0x00 | Result of the asynchronous job finish notifications |
| | E_NOT_OK | 0x01 | – |
| ***Description*** | Result of the asynchronous job finish notifications | | |
| ***Available via*** | bndm.h | | |

⌋

## 8.3 Function definitions

This chapter contains all APIs provided by the module.

### 8.3.1 BndM_Init

**[SWS_BndM_01004] Definition of API function BndM_Init ⌈**

| | | |
|---|---|---|
| ***Service Name*** | BndM_Init | |
| ***Syntax*** | `void BndM_Init (`<br>`  const BndM_ConfigType* ConfigPtr`<br>`)` | |
| ***Service ID [hex]*** | 0x01 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | ConfigPtr | Pointer to the configuration set in VARIANT-POST-BUILD. |
| ***Parameters (inout)*** | None | |

▽

△

| | |
|---|---|
| ***Parameters (out)*** | None |
| ***Return value*** | None |
| ***Description*** | Initializes or reinitializes this module. |
| ***Available via*** | BndM.h |

⌋

### 8.3.2 BndM_GetBlockPtr

**[SWS_BndM_01006] Definition of API function BndM_GetBlockPtr_<BlockId.Shortname>** ⌈

| | | |
|---|---|---|
| ***Service Name*** | BndM_GetBlockPtr_<BlockId.Shortname> | |
| ***Syntax*** | `Std_ReturnType BndM_GetBlockPtr_<BlockId.Shortname> ( BndM_BlockIdType BlockId, BndM_Block{BlockId.Shortname}Type** BndM_BlockPtr )` | |
| ***Service ID [hex]*** | 0x03 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant Returns an pointer to the structure in flash | |
| ***Parameters (in)*** | BlockId | BlockId |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | BndM_BlockPtr | • BndM_BlockPtr |
| ***Return value*** | Std_ReturnType | – |
| ***Description*** | – | |
| ***Available via*** | BndM_Externals.h | |

⌋

### 8.3.3 BndM_WriteStart

**[SWS_BndM_01007] Definition of API function BndM_WriteStart** ⌈

| | | |
|---|---|---|
| ***Service Name*** | BndM_WriteStart | |
| ***Syntax*** | `Std_ReturnType BndM_WriteStart ( void )` | |
| ***Service ID [hex]*** | 0x04 | |
| ***Sync/Async*** | Asynchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | None | |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK` The preparation request is accepted.<br>`E_NOT_OK` The preparation request is rejected. |

▽

△

| | |
|---|---|
| ***Description*** | Will trigger the start of writing phase. The finish of asynchronous processing will trigger the callback xxx_BndMWriteStartFinish including the result of this operation |
| ***Available via*** | BndM.h |

⌋

Note: It is up to the stack-vendor what can run in parallel while the writing to BndM is possible or not (e.g. FEE might not work anymore).

### 8.3.4 BndM_WriteBlock

**[SWS_BndM_01008] Definition of API function BndM_WriteBlock_<BlockId.Shortname>** ⌈

| | | |
|---|---|---|
| ***Service Name*** | BndM_WriteBlock_<BlockId.Shortname> | |
| ***Syntax*** | `Std_ReturnType BndM_WriteBlock_<BlockId.Shortname> (`<br>`  BndM_BlockIdType BlockId,`<br>`  const BndM_Block{BlockId.Shortname}Type* BndM_SrcPtr`<br>`)` | |
| ***Service ID [hex]*** | 0x05 | |
| ***Sync/Async*** | Asynchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | BlockId | – |
| | BndM_SrcPtr | – |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK` The write request is accepted.<br>`E_NOT_OK` The write request is rejected. |
| ***Description*** | Will persist the data in flash, that it can later directly accessed via BndM_GetBlockPtr API. The writing take a while and is finished after the successful callback xxx_BndMWriteBlockFinish | |
| ***Available via*** | BndM_Externals.h | |

⌋

Note: BndM_WriteStart needs to be called in advance

### 8.3.5 BndM_WriteFinalize

**[SWS_BndM_01009] Definition of API function BndM_WriteFinalize** ⌈

| | |
|---|---|
| ***Service Name*** | BndM_WriteFinalize |
| ***Syntax*** | `Std_ReturnType BndM_WriteFinalize (`<br>`  void`<br>`)` |
| ***Service ID [hex]*** | 0x06 |

▽

△

| | | |
|---|---|---|
| ***Sync/Async*** | Asynchronous | |
| ***Reentrancy*** | Reentrant Finalize the writing. After the successful callback xxx_BndMWriteFinalizeFinish the finalization is finished (i.e. the new stored data is available). | |
| ***Parameters (in)*** | None | |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK` The finalization request is accepted.<br>`E_NOT_OK` The finalization request is rejected. |
| ***Description*** | Will trigger the finalization of writing phase. The finish of asynchronous processing will trigger the callback xxx_BndMWriteFinalizeFinish including the result of this operation. | |
| ***Available via*** | BndM.h | |

⌋

### 8.3.6 BndM_WriteCancel

**[SWS_BndM_01010] Definition of API function BndM_WriteCancel** ⌈

| | |
|---|---|
| ***Service Name*** | BndM_WriteCancel |
| ***Syntax*** | `void BndM_WriteCancel (`<br>`  void`<br>`)` |
| ***Service ID [hex]*** | 0x07 |
| ***Sync/Async*** | Asynchronous |
| ***Reentrancy*** | Reentrant |
| ***Parameters (in)*** | None |
| ***Parameters (inout)*** | None |
| ***Parameters (out)*** | None |
| ***Return value*** | None |
| ***Description*** | Cancels the writing |
| ***Available via*** | BndM.h |

⌋

### 8.3.7 Callback notifications

This is a list of functions provided for FLS module.

#### 8.3.7.1 BndM_JobEndNotification

**[SWS_BndM_01011] Definition of callback function BndM_JobEndNotification** ⌈

| | |
|---|---|
| ***Service Name*** | BndM_JobEndNotification |
| ***Syntax*** | `void BndM_JobEndNotification ( void )` |
| ***Service ID [hex]*** | 0x08 |
| ***Sync/Async*** | Synchronous |
| ***Reentrancy*** | Non Reentrant |
| ***Parameters (in)*** | None |
| ***Parameters (inout)*** | None |
| ***Parameters (out)*** | None |
| ***Return value*** | None |
| ***Description*** | This callback function is called when a FLS job has been finished with positive result. |
| ***Available via*** | BndM.h |

⌋

#### 8.3.7.2 BndM_JobErrorNotification

**[SWS_BndM_01012] Definition of callback function BndM_JobErrorNotification** ⌈

| | |
|---|---|
| ***Service Name*** | BndM_JobErrorNotification |
| ***Syntax*** | `void BndM_JobErrorNotification ( void )` |
| ***Service ID [hex]*** | 0x09 |
| ***Sync/Async*** | Synchronous |
| ***Reentrancy*** | Non Reentrant |
| ***Parameters (in)*** | None |
| ***Parameters (inout)*** | None |
| ***Parameters (out)*** | None |
| ***Return value*** | None |
| ***Description*** | This callback function is called when a FLS job has been canceled or finished with negative result. |
| ***Available via*** | BndM.h |

⌋

## 8.4 Scheduled functions

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

### 8.4.1 BndM_MainFunction

**[SWS_BndM_01013] Definition of scheduled function BndM_MainFunction** ⌈

| | |
|---|---|
| ***Service Name*** | BndM_MainFunction |
| ***Syntax*** | `void BndM_MainFunction (`<br>`  void`<br>`)` |
| ***Service ID [hex]*** | 0x55 |
| ***Description*** | Schedule function for the background processing. |
| ***Available via*** | SchM_BndM.h |

⌋

## 8.5 Expected interfaces

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

### 8.5.1 Mandatory interfaces

**[SWS_BndM_01019] Definition of mandatory interfaces required by module Bnd M** ⌈

| *API Function* | *Header File* | *Description* |
|---|---|---|
| There are no mandatory interfaces. | | |

⌋

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

### 8.5.2 Optional interfaces

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

**[SWS_BndM_01020] Definition of optional interfaces requested by module Bnd M** ⌈

| *API Function* | *Header File* | *Description* |
|---|---|---|
| Det_ReportRuntimeError | Det.h | Service to report runtime errors. If a callout has been configured then this callout shall be called. |

⌋

### 8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

#### 8.5.3.1 xxx_BndMWriteStartFinish

**[SWS_BndM_01016] Definition of callout function Xxx_BndMWriteStartFinish** ⌈

| | | |
|---|---|---|
| ***Service Name*** | Xxx_BndMWriteStartFinish | |
| ***Syntax*** | `void Xxx_BndMWriteStartFinish ( BndM_BlockIdType BlockId, BndM_ResultType result )` | |
| ***Service ID [hex]*** | 0x56 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | BlockId | – |
| | result | – |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | This callback function is called when BndM_WriteStart is finished. | |
| ***Available via*** | BndM_Externals.h | |

⌋

#### 8.5.3.2 xxx_BndMWriteBlockFinish

**[SWS_BndM_01014] Definition of callout function Xxx_BndMWriteBlockFinish** ⌈

| | | |
|---|---|---|
| ***Service Name*** | Xxx_BndMWriteBlockFinish | |
| ***Syntax*** | `void Xxx_BndMWriteBlockFinish ( BndM_BlockIdType BlockId, BndM_ResultType result )` | |
| ***Service ID [hex]*** | 0x57 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | BlockId | – |
| | result | – |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | This callback function is called when BndM_WriteBlock is finished. | |
| ***Available via*** | BndM_Externals.h | |

⌋

#### 8.5.3.3 xxx_BndMWriteFinalizeFinish

**[SWS_BndM_01015] Definition of callout function Xxx_BndMWriteFinalizeFinish**
⌈

| | | |
|---|---|---|
| ***Service Name*** | Xxx_BndMWriteFinalizeFinish | |
| ***Syntax*** | `void Xxx_BndMWriteFinalizeFinish (`<br>`  BndM_BlockIdType BlockId,`<br>`  BndM_ResultType result`<br>`)` | |
| ***Service ID [hex]*** | 0x58 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | BlockId | – |
| | result | – |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | This callback function is called when BndM_WriteFinalize is finished. | |
| ***Available via*** | BndM_Externals.h | |

⌋

## 8.6 Service Interfaces

The BndM does not have service interfaces.

# 9 Sequence diagrams

No content.

# 10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module BndM.

Chapter 10.3 specifies published information of the module BndM.

## 10.1 How to read this chapter

For details refer to [2] Chapter 10.1 *"Introduction to configuration specification"*.

## 10.2 Containers and configuration parameters

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

### 10.2.1 BndM

**[ECUC_BndM_00001] Definition of EcucModuleDef BndM** ⌈

| | |
|---|---|
| **Module Name** | BndM |
| **Description** | Configuration of the BulkNvDataManager module. |
| **Post-Build Selectable Support** | false |
| **Post-Build Loadable Support** | false |

| **Included Containers** | | |
|---|---|---|
| **Container Name** | **Multiplicity** | **Description** |
| BndMBlockDescriptor | 0..* | Each container defines a Bulk NV Block which can be individually accessed. |
| BndMCallbackBlock | 0..* | This container contains the block-specific callbacks. |
| BndMCallbackGeneral | 0..1 | This container contains the general callbacks |
| BndMGeneral | 1 | Container for common configuration options. |

⌋

### 10.2.2 BndMGeneral

**[ECUC_BndM_00002] Definition of EcucParamConfContainerDef BndMGeneral** ⌈

| Container Name | BndMGeneral |
|---|---|
| Parent Container | BndM |
| Upstream mapping defined | – |
| Description | Container for common configuration options. |
| Multiplicity | 1 |
| Pre-Compile only | true |
| Configuration Parameters | |

| Included Parameters | | |
|---|---|---|
| Parameter Name | Multiplicity | ECUC ID |
| BndMDevErrorDetect | 1 | [ECUC_BndM_00003] |
| BndMMainFunctionPeriod | 1 | [ECUC_BndM_00004] |

| No Included Containers |
|---|

⌋

### [ECUC_BndM_00003] Definition of EcucBooleanParamDef BndMDevErrorDetect ⌈

| Parameter Name | BndMDevErrorDetect |
|---|---|
| Parent Container | BndMGeneral |
| Upstream mapping defined | – |
| Description | Switches the development error detection and notification on or off.<br>• true: detection and notification is enabled.<br>• false: detection and notification is disabled. |
| Multiplicity | 1 |
| Type | EcucBooleanParamDef |
| Default value | false |
| Pre-Compile only | true |

⌋

### [ECUC_BndM_00004] Definition of EcucFloatParamDef BndMMainFunctionPeriod ⌈

| Parameter Name | BndMMainFunctionPeriod | |
|---|---|---|
| Parent Container | BndMGeneral | |
| Upstream mapping defined | – | |
| Description | The period between successive calls to the main function in seconds. | |
| Multiplicity | 1 | |
| Type | EcucFloatParamDef | |
| Range | ]0 .. INF[ | |
| Default value | – | |
| Pre-Compile only | true | |

⌋

### 10.2.3 BndMBlockDescriptor

**[ECUC_BndM_00014] Definition of EcucParamConfContainerDef BndMBlockDescriptor** ⌈

| | |
|---|---|
| **Container Name** | BndMBlockDescriptor |
| **Parent Container** | BndM |
| **Upstream mapping defined** | – |
| **Description** | Each container defines a Bulk NV Block which can be individually accessed. |
| **Multiplicity** | 0..* |
| **Pre-Compile only** | true |
| **Configuration Parameters** | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| BndMBlockIdentifier | 1 | [ECUC_BndM_00007] |
| BndMBlockDescriptor | 1 | [ECUC_BndM_00006] |
| BndMCallbackRef | 0..1 | [ECUC_BndM_00013] |

| **No Included Containers** |
|---|

⌋

**[ECUC_BndM_00007] Definition of EcucIntegerParamDef BndMBlockIdentifier** ⌈

| | | |
|---|---|---|
| **Parameter Name** | BndMBlockIdentifier | |
| **Parent Container** | BndMBlockDescriptor | |
| **Upstream mapping defined** | – | |
| **Description** | Unique identification of the block. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef (Symbolic Name generated for this parameter) | |
| **Range** | 0 .. 65535 | |
| **Default value** | – | |
| **Pre-Compile only** | true | |

⌋

**[ECUC_BndM_00006] Definition of EcucForeignReferenceDef BndMBlockDescriptor** ⌈

| | |
|---|---|
| **Parameter Name** | BndMBlockDescriptor |
| **Parent Container** | BndMBlockDescriptor |
| **Upstream mapping defined** | – |
| **Description** | This parameter defines the data structure of the block. |
| **Multiplicity** | 1 |
| **Type** | Foreign reference to IMPLEMENTATION-DATA-TYPE |
| **Pre-Compile only** | true |

⌋

### [ECUC_BndM_00013] Definition of EcucReferenceDef BndMCallbackRef ⌈

| Parameter Name | BndMCallbackRef |
|---|---|
| Parent Container | BndMBlockDescriptor |
| Upstream mapping defined | – |
| Description | Reference to the block-specific callback function. |
| Multiplicity | 0..1 |
| Type | Reference to BndMCallbackBlock |
| Pre-Compile only | true |

⌋

## 10.2.4 BndMCallbackBlock

### [ECUC_BndM_00011] Definition of EcucParamConfContainerDef BndMCallback Block ⌈

| Container Name | BndMCallbackBlock |
|---|---|
| Parent Container | BndM |
| Upstream mapping defined | – |
| Description | This container contains the block-specific callbacks. |
| Multiplicity | 0..* |
| Pre-Compile only | true |
| Configuration Parameters | |

| Included Parameters | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| BndMWriteBlockFinishFnc | 1 | [ECUC_BndM_00012] |

| No Included Containers |
|---|

⌋

### [ECUC_BndM_00012] Definition of EcucFunctionNameDef BndMWriteBlockFinishFnc ⌈

| Parameter Name | BndMWriteBlockFinishFnc |
|---|---|
| Parent Container | BndMCallbackBlock |
| Upstream mapping defined | – |
| Description | Callback function for the WriteBlockFinish callback. |
| Multiplicity | 1 |
| Type | EcucFunctionNameDef |
| Default value | – |
| Regular Expression | – |
| Pre-Compile only | true |

⌋

### 10.2.5 BndMCallbackGeneral

**[ECUC_BndM_00015] Definition of EcucParamConfContainerDef BndMCallback General** ⌈

| | |
|---|---|
| **Container Name** | BndMCallbackGeneral |
| **Parent Container** | BndM |
| **Upstream mapping defined** | – |
| **Description** | This container contains the general callbacks |
| **Multiplicity** | 0..1 |
| **Pre-Compile only** | true |
| **Configuration Parameters** | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| BndMWriteFinalizeFinishFnc | 1 | [ECUC_BndM_00010] |
| BndMWriteStartFinishFnc | 1 | [ECUC_BndM_00009] |

| **No Included Containers** |
|---|

⌋

**[ECUC_BndM_00010] Definition of EcucFunctionNameDef BndMWriteFinalize FinishFnc** ⌈

| | |
|---|---|
| **Parameter Name** | BndMWriteFinalizeFinishFnc |
| **Parent Container** | BndMCallbackGeneral |
| **Upstream mapping defined** | – |
| **Description** | Callback function for the WriteFinalizeFinish callback. |
| **Multiplicity** | 1 |
| **Type** | EcucFunctionNameDef |
| **Default value** | – |
| **Regular Expression** | – |
| **Pre-Compile only** | true |

⌋

**[ECUC_BndM_00009] Definition of EcucFunctionNameDef BndMWriteStartFinishFnc** ⌈

| | |
|---|---|
| **Parameter Name** | BndMWriteStartFinishFnc |
| **Parent Container** | BndMCallbackGeneral |
| **Upstream mapping defined** | – |
| **Description** | Callback function for the WriteStartFinish callback. |
| **Multiplicity** | 1 |
| **Type** | EcucFunctionNameDef |
| **Default value** | – |
| **Regular Expression** | – |
| **Pre-Compile only** | true |

⌋

## 10.3 Published Information

For details refer to [2] Chapter 10.3 *“Published Information”*.

# A Change history of AUTOSAR traceable items

## A.1 Traceable item history of this document according to AUTOSAR Release R26-11

### A.1.1 Added Specification Items in R26-11

none

### A.1.2 Changed Specification Items in R26-11

| Number | Heading |
|---|---|
| [ECUC_BndM_00002] | Definition of EcucParamConfContainerDef BndMGeneral |
| [SWS_BndM_01018] | Definition of imported datatypes of module BndM |

**Table A.1: Changed Specification Items in R26-11**

### A.1.3 Deleted Specification Items in R26-11

| Number | Heading |
|---|---|
| [ECUC_BndM_00005] | Definition of EcucBooleanParamDef BndMVersionInfoApi |
| [SWS_BndM_01005] | Definition of API function BndM_GetVersionInfo |

**Table A.2: Deleted Specification Items in R26-11**

## A.2 Traceable item history of this document according to AUTOSAR Release R25-11

### A.2.1 Added Specification Items in R25-11

none

### A.2.2 Changed Specification Items in R25-11

| Number | Heading |
|---|---|
| [ECUC_BndM_00014] | Definition of EcucParamConfContainerDef BndMBlockDescriptor |
| [SWS_BndM_01018] | Definition of imported datatypes of module BndM |

▽

△

| Number | Heading |
|---|---|
| [SWS_BndM_01020] | Definition of optional interfaces requested by module BndM |

**Table A.3: Changed Specification Items in R25-11**

### A.2.3 Deleted Specification Items in R25-11

| Number | Heading |
|---|---|
| [ECUC_BndM_00008] | Definition of EcucReferenceDef BndMDeviceIndex |

**Table A.4: Deleted Specification Items in R25-11**

## A.3 Traceable item history of this document according to AUTOSAR Release R24-11

### A.3.1 Added Specification Items in R24-11

none

### A.3.2 Changed Specification Items in R24-11

none

### A.3.3 Deleted Specification Items in R24-11

none

# B Not applicable requirements

**[SWS_BndM_00999]** ⌈These requirements are not applicable to this specification.⌋