Document Title	Specification of Basic Software Multicore Library
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	946

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2024-11-27	R24-11	AUTOSAR Release Management	• Service IDs cleaned up
2023-11-23	R23-11	AUTOSAR Release Management	• Added note for MemoryAllocation • Bugfixes
2022-11-24	R22-11	AUTOSAR Release Management	• Introduced BMC Atomic Datatypes • Reworked APIs to make use of Atomic Datatypes • Cleaned up library
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• Improved the structure of the 'error sections' of the SWS documents • CONC_643 "BSW Multicore Distribution" finalized
2019-11-28	R19-11	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration of the AUTOSAR library for atomic routines.

This library (Bmc) contains the following routines:

- flag test and set
- flag clear
- store
- load
- exchange
- compare and exchange
- fetch and add
- fetch and subtract
- fetch and or
- fetch and xor
- fetch and and
- thread fence

All routines are re-entrant and can be used by multiple runnables at the same time.

2 Acronyms and Abbreviations

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

Abbreviation/Acronym:	Description:
Bmc	Basic Software Multicore Library
DET	Default Error Tracer
s16	Mnemonic for `sint16`, specified in AUTOSAR_SWS_PlatformTypes
s32	Mnemonic for `sint32`, specified in AUTOSAR_SWS_PlatformTypes
s64	Mnemonic for`sint64`, specified in AUTOSAR_SWS_PlatformTypes
s8	Mnemonic for `sint8`, specified in AUTOSAR_SWS_PlatformTypes
u16	Mnemonic for `uint16`, specified in AUTOSAR_SWS_PlatformTypes
u32	Mnemonic for `uint32`, specified in AUTOSAR_SWS_PlatformTypes
u64	Mnemonic for `uint64`, specified in AUTOSAR_SWS_PlatformTypes
u8	Mnemonic for `uint8`, specified in AUTOSAR_SWS_PlatformTypes

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] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[4] Requirements on Libraries
AUTOSAR_CP_RS_Libraries

3.2 Related specification

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

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

[SWS_BMC_00001]

Upstream requirements: SRS_LIBS_00005

⌈The Bmc module shall provide the following files: C files `Bmc_<name>.c` used to implement the library. All C files shall be pre-fixed with 'Bmc_'. The header file `Bmc.h` provides all public function prototypes and types defined by the Bmc library specification.⌋

Implementation and grouping of routines with respect to C files is recommended as per options below and there is no restriction to follow these proposals.

Option 1: `<Name>` can be a function name providing one C file per function, e.g.: `Bmc_FlagClear.c` etc.

Option 2: `<Name>` can be a common name of a group of functions:
2.1 Group by routine family:
e.g.: `Bmc_Flag.c`, `Bmc_Fetch.c`
2.2 Group by other methods (individual grouping allowed)

Option 3: `<Name>` can be removed so that a single C file shall contain all Bmc functions, e.g.: `Bmc.c`. Using one of the above options gives certain flexibility of choosing suitable granularity with reduced number of C files. Linking only on-demand is also possible in case of some options.

6 Requirements Tracing

The following tables reference the requirements specified in [3], [4] 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
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_BMC_00015]
[SRS_BSW_00306]	AUTOSAR Basic Software Modules shall be compiler and platform independent	[SWS_BMC_00016]
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_BMC_00014]
[SRS_BSW_00374]	All Basic Software Modules shall provide a readable module vendor identification	[SWS_BMC_00044]
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_BMC_00015]
[SRS_BSW_00379]	All software modules shall provide a module identifier in the header file and in the module XML description file.	[SWS_BMC_00044]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_BMC_00044]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_BMC_00043]
[SRS_BSW_00411]	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	[SWS_BMC_00043]
[SRS_BSW_00437]	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	[SWS_BMC_00013]
[SRS_BSW_00448]	Module SWS shall not contain requirements from other modules	[SWS_BMC_00999]
[SRS_LIBS_00001]	The functional behavior of each library functions shall not be configurable	[SWS_BMC_00045]
[SRS_LIBS_00002]	A library shall be operational before all BSW modules and application SW-Cs	[SWS_BMC_00005]
[SRS_LIBS_00003]	A library shall be operational until the shutdown	[SWS_BMC_00006]
[SRS_LIBS_00004]	Using libraries shall not pass through a port interface	[SWS_BMC_00007]
[SRS_LIBS_00005]	Each library shall provide one header file with its public interface	[SWS_BMC_00001]
[SRS_LIBS_00007]	Using a library should be documented	[SWS_BMC_00008] [SWS_BMC_00012]

▽

△

Requirement	Description	Satisfied by
[SRS_LIBS_00015]	It shall be possible to configure the microcontroller so that the library code is shared between all callers	[SWS_BMC_00009]
[SRS_LIBS_00017]	Usage of macros should be avoided	[SWS_BMC_00010]
[SRS_LIBS_00018]	A library function may only call library functions	[SWS_BMC_00011]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Error Classification

Section "Error Handling" of the document "General Specification of Basic Software Modules" 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

There are no development errors.

7.1.2 Runtime Errors

There are no runtime errors.

7.1.3 Production Errors

There are no production errors.

7.1.4 Extended Production Errors

There are no extended production errors.

7.2 Initialization and Shutdown

[SWS_BMC_00005]

Upstream requirements: SRS_LIBS_00002

⌈The Bmc library shall not require an initialization phase. A Library function may be called at the very first step of ECU initialization, e.g. even by the OS or EcuM, thus the library shall be ready.⌋

[SWS_BMC_00006]

Upstream requirements: SRS_LIBS_00003

⌈The Bmc library shall not require a shutdown operation phase.⌋

7.3 Using Library API

[SWS_BMC_00007]

Upstream requirements: SRS_LIBS_00004

⌈The Bmc API can be directly called from BSW modules or SWCs. No port definition is required. It is a pure function call.⌋

[SWS_BMC_00008]

Upstream requirements: SRS_LIBS_00007

⌈Using a library should be documented. If a BSW module or a SWC uses a library, the developer should add an ImplementationDependencyOnArtifact in the BSW/SWC template. minVersion and maxVersion parameters correspond to the supplier version. In case of an AUTOSAR library, these parameters may be left empty because a SWC or BSW module may rely on a library behavior, not on a supplier implementation. However, the SWC or BSW modules shall be compatible with the AUTOSAR platform where they are integrated.⌋

The user has to ensure that atomic data are allocated in a memory which allows atomic access. Hence it might be useful to use a vendor specific but unique {refinement} keyword in the memory allocation keyword (MAKW) or respective software addressing methods (SwAddrMethod) which can be used to derive according MemoryAllocation options for the purpose of mapping.

7.4 Library Implementation

[SWS_BMC_00009]

Upstream requirements: SRS_LIBS_00015

⌈The Bmc library shall be implemented in a way that the code can be shared among callers in different memory partitions.⌋

[SWS_BMC_00010]

Upstream requirements: SRS_LIBS_00017

⌈Usage of macros should be avoided. The functions should be declared as functions or inline functions.⌋

[SWS_BMC_00011]

Upstream requirements: SRS_LIBS_00018

⌈A library function shall not call any BSW modules functions, e.g. the DET. A library function can call other library functions because a library function shall be re-entrant. But other BSW modules functions may not be re-entrant.⌋

[SWS_BMC_00012]

Upstream requirements: SRS_LIBS_00007

⌈The library, written in the C programming language, should conform to the MISRA C Standard. Please refer to SWS_BSW_00115 for more details.⌋

[SWS_BMC_00013]

Upstream requirements: SRS_BSW_00437

⌈Each AUTOSAR library Module implementation `<library>*.c` and `<library>*.h` shall map their code to memory sections using the AUTOSAR memory mapping mechanism.⌋

[SWS_BMC_00014]

Upstream requirements: SRS_BSW_00348

⌈Each AUTOSAR library Module implementation `<library>*.c` that uses AUTOSAR integer data types and/or the standard return type, shall include the header file `Std_Types.h`.⌋

[SWS_BMC_00015]

Upstream requirements: SRS_BSW_00378, SRS_BSW_00304

⌈All AUTOSAR library Modules should use the AUTOSAR data types (integers, boolean) instead of native C data types unless this library is clearly identified to be compliant only with one platform.⌋

[SWS_BMC_00016]

Upstream requirements: SRS_BSW_00306

⌈All AUTOSAR library Modules should avoid direct use of compiler and platform specific keywords unless this library is clearly identified to be compliant only with one platform.⌋

8 API specification

8.1 Imported types

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

[SWS_Bmc_91000] Definition of imported datatypes of module Bmc ⌈

Module	*Header File*	*Imported Type*
Std	Std_Types.h	Std_VersionInfoType

⌋

8.2 Type definitions

Note: Most likely the Bmc AtomicTypes will be the native datatype of the microcontroller (e.g. uint32/sint32 for a 32 bit microcontroller).

8.2.1 Bmc_AtomicUType

[SWS_Bmc_91016] Definition of datatype Bmc_AtomicUType ⌈

Name	Bmc_AtomicUType		
Kind	Type		
Derived from	uint		
Range	-	–	The Bmc_AtomicUType shall always be mapped to a platform specific type where atomic operations can be realized by the respective HW platform, to ensure that all the operations performed on this type are lock-free
Description	The type shall be used for all unsigned data items, which are used for Bmc library functions.		
Available via	Bmc.h		

⌋

8.2.2 Bmc_AtomicSType

[SWS_Bmc_91017] Definition of datatype Bmc_AtomicSType ⌈

Name	Bmc_AtomicSType		
Kind	Type		
Derived from	sint		
Range	-	–	The Bmc_AtomicSType shall always be mapped to a platform specific type where atomic operations can be realized by the respective HW platform, to ensure that all the operations performed on this type are lock-free
Description	The type shall be used for all signed data items, which are used for Bmc library functions.		
Available via	Bmc.h		

⌋

8.2.3 Bmc_AtomicFlagType

[SWS_Bmc_91018] Definition of datatype Bmc_AtomicFlagType ⌈

Name	Bmc_AtomicFlagType		
Kind	Type		
Derived from	boolean		
Range	-	–	The Bmc_AtomicFlagType shall always be mapped to a platform specific type where atomic operations can be realized by the respective HW platform, to insure that all the operations performed on this type are lock-free
Description	The type shall be used for all Flag data items, which are used for Bmc library functions.		
Available via	Bmc.h		

⌋

8.3 Macro definitions

No Macro definitions.

8.4 Function definitions

Note: All atomic operations will provide sequentially consistent ordering (see https://en.cppreference.com/w/c/atomic/memory_order#Sequentially-consistent_ordering).

8.4.1 Flag Routines

8.4.1.1 Bmc_FlagTestAndSet

[SWS_Bmc_91003] Definition of API function Bmc_FlagTestAndSet ⌈

Service Name	Bmc_FlagTestAndSet	
Syntax	`boolean Bmc_FlagTestAndSet ( volatile Bmc_AtomicFlagType* Object )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	boolean	The value pointed to by Object immediately before the effects
Description	Atomically sets the value pointed to by Object to true.	
Available via	Bmc.h	

⌋

[SWS_BMC_00019] ⌈The function `Bmc_FlagTestAndSet` atomically sets the value pointed to by Object to `TRUE`. It returns this value before the operation, i.e., `TRUE`, if it was already set and `FALSE` otherwise.⌋

8.4.1.2 Bmc_FlagClear

[SWS_Bmc_91004] Definition of API function Bmc_FlagClear ⌈

Service Name	Bmc_FlagClear
Syntax	`void Bmc_FlagClear ( volatile Bmc_AtomicFlagType* Object )`
Service ID [hex]	0x02

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	None	
Description	Atomically sets the value pointed to by Object to false.	
Available via	Bmc.h	

⌋

[SWS_BMC_00021] ⌈The function `Bmc_FlagClear` atomically sets the value pointed to by Object to `FALSE`.⌋

8.4.2 Load and Store Routines

[SWS_BMC_00046] ⌈All load and store routines shall implicitly make use of the feature explicitly introduced by `Bmc_ThreadFence`.⌋

8.4.2.1 Bmc_Load

[SWS_Bmc_91019] Definition of API function Bmc_Load_u ⌈

Service Name	Bmc_Load_u	
Syntax	`Bmc_AtomicUType Bmc_Load_u ( const volatile Bmc_AtomicUType* Object )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	Object	–
Parameters (out)	None	
Return value	Bmc_AtomicUType	The value pointed to by Object
Description	Atomically loads the value pointed to by Object and returns it.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91020] Definition of API function Bmc_Load_s ⌈

Service Name	Bmc_Load_s	
Syntax	`Bmc_AtomicSType Bmc_Load_s ( const volatile Bmc_AtomicSType* Object )`	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	Object	–
Parameters (out)	None	
Return value	Bmc_AtomicSType	The value pointed to by Object
Description	Atomically loads the value pointed to by Object and returns it.	
Available via	Bmc.h	

⌋

8.4.2.2 Bmc_Store

[SWS_Bmc_91021] Definition of API function Bmc_Store_u ⌈

Service Name	Bmc_Store_u	
Syntax	`void Bmc_Store_u ( volatile Bmc_AtomicUType* Object, Bmc_AtomicUType Desired )`	
Service ID [hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Desired	Value to be stored
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	None	
Description	Atomically replaces the value pointed to by Object with the value of Desired.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91022] Definition of API function Bmc_Store_s ⌈

Service Name	Bmc_Store_s
Syntax	`void Bmc_Store_s ( volatile Bmc_AtomicSType* Object, Bmc_AtomicSType Desired )`

▽

△

Service ID [hex]	0x24	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Desired	Value to be stored
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	None	
Description	Atomically replaces the value pointed to by Object with the value of Desired.	
Available via	Bmc.h	

⌋

8.4.2.3 Bmc_Exchange

[SWS_Bmc_91025] Definition of API function Bmc_Exchange_u ⌈

Service Name	Bmc_Exchange_u	

Syntax	`Bmc_AtomicUType Bmc_Exchange_u (`
	`  volatile Bmc_AtomicUType* Object,`
	`  Bmc_AtomicUType Desired`
	`)`	

Service ID [hex]	0x30	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Desired	Value to be stored
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicUType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the value of Desired and returns the value pointed to by Object immediately before the effects.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91026] Definition of API function Bmc_Exchange_s ⌈

Service Name	Bmc_Exchange_s

Syntax	`Bmc_AtomicSType Bmc_Exchange_s (`
	`  volatile Bmc_AtomicSType* Object,`
	`  Bmc_AtomicSType Desired`
	`)`

Service ID [hex]	0x34
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	Desired	Value to be stored
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicSType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the value of Desired and returns the value pointed to by Object immediately before the effects.	
Available via	Bmc.h	

⌋

8.4.2.4 Bmc_CompareExchange

[SWS_Bmc_91023] Definition of API function Bmc_CompareExchange_u ⌈

Service Name	Bmc_CompareExchange_u	

Syntax	`boolean Bmc_CompareExchange_u (`
	`  volatile Bmc_AtomicUType* Object,`
	`  Bmc_AtomicUType* Expected,`
	`  Bmc_AtomicUType Desired`
	`)`	

Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Desired	Value to be stored
Parameters (inout)	Object	Object
	Expected	Value to be stored
Parameters (out)	None	
Return value	boolean	The result of the comparison
Description	Atomically compares the value pointed to by Object for equality with that in Expected, and if true, replaces the value pointed to by Object with Desired, and if false, updates the value in Expected with the value pointed to by Object.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91024] Definition of API function Bmc_CompareExchange_s ⌈

Service Name	Bmc_CompareExchange_s

Syntax	`boolean Bmc_CompareExchange_s (`
	`  volatile Bmc_AtomicSType* Object,`
	`  Bmc_AtomicSType* Expected,`
	`  Bmc_AtomicSType Desired`
	`)`

Service ID [hex]	0x44
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant	
Parameters (in)	Desired	Value to be stored
Parameters (inout)	Object	Object
	Expected	Value to be stored
Parameters (out)	None	
Return value	boolean	The result of the comparison
Description	Atomically compares the value pointed to by Object for equality with that in Expected, and if true, replaces the value pointed to by Object with Desired, and if false, updates the value in Expected with the value pointed to by Object.	
Available via	Bmc.h	

⌋

8.4.3 Fetch Routines

[SWS_BMC_00047] ⌈All fetch routines shall implicitly make use of the feature explicitly introduced by `Bmc_ThreadFence`.⌋

8.4.3.1 Bmc_FetchAdd

[SWS_Bmc_91027] Definition of API function Bmc_FetchAdd_u ⌈

Service Name	Bmc_FetchAdd_u	
Syntax	`Bmc_AtomicUType Bmc_FetchAdd_u ( volatile Bmc_AtomicUType* Object, Bmc_AtomicUType Operand )`	
Service ID [hex]	0x50	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicUType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the addition applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91028] Definition of API function Bmc_FetchAdd_s ⌈

Service Name	Bmc_FetchAdd_s	
Syntax	`Bmc_AtomicSType Bmc_FetchAdd_s ( volatile Bmc_AtomicSType* Object, Bmc_AtomicSType Operand )`	
Service ID [hex]	0x54	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicSType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the addition applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

8.4.3.2 Bmc_FetchSub

[SWS_Bmc_91033] Definition of API function Bmc_FetchSub_u ⌈

Service Name	Bmc_FetchSub_u	
Syntax	`Bmc_AtomicUType Bmc_FetchSub_u ( volatile Bmc_AtomicUType* Object, Bmc_AtomicUType Operand )`	
Service ID [hex]	0x60	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicUType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the subtraction applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91034] Definition of API function Bmc_FetchSub_s ⌈

Service Name	Bmc_FetchSub_s	
Syntax	`Bmc_AtomicSType Bmc_FetchSub_s ( volatile Bmc_AtomicSType* Object, Bmc_AtomicSType Operand )`	
Service ID [hex]	0x64	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicSType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the subtraction applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

8.4.3.3 Bmc_FetchOr

[SWS_Bmc_91031] Definition of API function Bmc_FetchOr_u ⌈

Service Name	Bmc_FetchOr_u	
Syntax	`Bmc_AtomicUType Bmc_FetchOr_u ( volatile Bmc_AtomicUType* Object, Bmc_AtomicUType Operand )`	
Service ID [hex]	0x70	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicUType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the or-operation applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91032] Definition of API function Bmc_FetchOr_s ⌈

Service Name	Bmc_FetchOr_s	
Syntax	`Bmc_AtomicSType Bmc_FetchOr_s ( volatile Bmc_AtomicSType* Object, Bmc_AtomicSType Operand )`	
Service ID [hex]	0x74	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicSType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the or-operation applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

8.4.3.4 Bmc_FetchXor

[SWS_Bmc_91035] Definition of API function Bmc_FetchXor_u ⌈

Service Name	Bmc_FetchXor_u	
Syntax	`Bmc_AtomicUType Bmc_FetchXor_u ( volatile Bmc_AtomicUType* Object, Bmc_AtomicUType Operand )`	
Service ID [hex]	0x84	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicUType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the xor-operation applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91036] Definition of API function Bmc_FetchXor_s ⌈

Service Name	Bmc_FetchXor_s	
Syntax	`Bmc_AtomicSType Bmc_FetchXor_s ( volatile Bmc_AtomicSType* Object, Bmc_AtomicSType Operand )`	
Service ID [hex]	0x80	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Operand	Value for the operation
Parameters (inout)	Object	Object
Parameters (out)	None	
Return value	Bmc_AtomicSType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the xor-operation applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

8.4.3.5 Bmc_FetchAnd

[SWS_Bmc_91029] Definition of API function Bmc_FetchAnd_u ⌈

Service Name	Bmc_FetchAnd_u	
Syntax	`Bmc_AtomicUType Bmc_FetchAnd_u ( volatile Bmc_AtomicUType* Object, Bmc_AtomicUType Operand )`	
Service ID [hex]	0x90	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	Object	Object
	Operand	Value for the operation
Parameters (out)	None	
Return value	Bmc_AtomicUType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the and-operation applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

[SWS_Bmc_91030] Definition of API function Bmc_FetchAnd_s ⌈

Service Name	Bmc_FetchAnd_s	
Syntax	`Bmc_AtomicSType Bmc_FetchAnd_s ( volatile Bmc_AtomicSType* Object, Bmc_AtomicSType Operand )`	
Service ID [hex]	0x94	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	Object	Object
	Operand	Value for the operation
Parameters (out)	None	
Return value	Bmc_AtomicSType	The value pointed to by Object immediately before the effects
Description	Atomically replaces the value pointed to by Object with the result of the and-operation applied to the value pointed to by Object and the given Operand.	
Available via	Bmc.h	

⌋

8.4.4 Fence Routines

8.4.4.1 Bmc_ThreadFence

[SWS_Bmc_91014] Definition of API function Bmc_ThreadFence ⌈

Service Name	Bmc_ThreadFence
Syntax	`void Bmc_ThreadFence ( void )`
Service ID [hex]	0x03
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Creates a sequentially consistent acquire and release fence. An acquire and release fence instruction prevents the memory reordering of any read or write which precedes it in program order with any read or write which follows it in program order.
Available via	Bmc.h

⌋

[SWS_BMC_00041] ⌈The function `Bmc_ThreadFence` creates a sequentially consistent acquire and release fence.⌋

Note: It may also serve as a compiler barrier which stops the compiler from moving instructions across it either way for optimization purposes. Any instruction that occurs in program order before this instruction will not be reordered after this instruction. Every instruction that occurs after this instruction will not be reordered before this instruction.

8.4.5 Version API

8.4.5.1 Bmc_GetVersionInfo

[SWS_Bmc_91015] Definition of API function Bmc_GetVersionInfo ⌈

Service Name	Bmc_GetVersionInfo	
Syntax	`void Bmc_GetVersionInfo ( Std_VersionInfoType* Versioninfo )`	
Service ID [hex]	0xFF	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	Versioninfo	Pointer to where to store the version information of this module. Format according [BSW00321]
Return value	None	
Description	Returns the version information of this library.	
Available via	Bmc.h	

⌋

[SWS_BMC_00043]

Upstream requirements: SRS_BSW_00407, SRS_BSW_00411

⌈If source code for caller and callee of `Bmc_GetVersionInfo` is available, the Bmc library should realize `Bmc_GetVersionInfo` as a macro defined in the module's header file.⌋

8.5 Callback notifications

None.

8.6 Scheduled functions

The Bmc library does not have scheduled functions.

8.7 Expected Interfaces

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

8.7.1 Mandatory Interfaces

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

[SWS_Bmc_91001] Definition of mandatory interfaces required by module Bmc ⌈

API Function	Header File	Description
There are no mandatory interfaces.		

⌋

8.7.2 Optional Interfaces

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

[SWS_Bmc_91002] Definition of optional interfaces requested by module Bmc ⌈

API Function	Header File	Description
There are no optional interfaces.		

⌋

8.7.3 Configurable interfaces

None.

9 Sequence diagrams

Not applicable.

10 Configuration specification

10.1 Published Information

[SWS_BMC_00044]

Upstream requirements: SRS_BSW_00402, SRS_BSW_00374, SRS_BSW_00379

⌈The standardized common published parameters as required by SRS_BSW_00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules⌋

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

10.2 Configuration Option

[SWS_BMC_00045]

Upstream requirements: SRS_LIBS_00001

⌈The Bmc library shall not have any configuration options that may affect the functional behavior of the routines. I.e. for a given set of input parameters, the outputs shall be always the same. For example, the returned value in case of error shall not be configurable.⌋

However, a library vendor is allowed to add specific configuration options concerning library implementation, e.g. for resource consumption optimization.

A History of Constraints and Specification Items

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

A.1 Constraint and Specification Item History of this Document According to AUTOSAR Release R22-11

A.1.1 Added Specification Items in R22-11

[SWS_BMC_00046] [SWS_BMC_00047] [SWS_Bmc_91016] [SWS_Bmc_91017] [SWS_Bmc_91018] [SWS_Bmc_91019] [SWS_Bmc_91020] [SWS_Bmc_91021] [SWS_Bmc_91022] [SWS_Bmc_91023] [SWS_Bmc_91024] [SWS_Bmc_91025] [SWS_Bmc_91026] [SWS_Bmc_91027] [SWS_Bmc_91028] [SWS_Bmc_91029] [SWS_Bmc_91030] [SWS_Bmc_91031] [SWS_Bmc_91032] [SWS_Bmc_91033] [SWS_Bmc_91034] [SWS_Bmc_91035] [SWS_Bmc_91036]

A.1.2 Changed Specification Items in R22-11

[SWS_BMC_00013] [SWS_BMC_00014] [SWS_BMC_00016] [SWS_Bmc_91003] [SWS_Bmc_91004] [SWS_Bmc_91014] [SWS_Bmc_91015]

A.1.3 Deleted Specification Items in R22-11

[SWS_BMC_00017] [SWS_BMC_00023] [SWS_BMC_00025] [SWS_BMC_00027] [SWS_BMC_00029] [SWS_BMC_00031] [SWS_BMC_00033] [SWS_BMC_00035] [SWS_BMC_00037] [SWS_BMC_00039] [SWS_Bmc_91005] [SWS_Bmc_91006] [SWS_Bmc_91007] [SWS_Bmc_91008] [SWS_Bmc_91009] [SWS_Bmc_91010] [SWS_Bmc_91011] [SWS_Bmc_91012] [SWS_Bmc_91013]

A.2 Constraint and Specification Item History of this Document According to AUTOSAR Release R23-11

A.2.1 Added Specification Items in R23-11

[SWS_Bmc_91000] [SWS_Bmc_91001] [SWS_Bmc_91002]

A.2.2 Changed Specification Items in R23-11

[SWS_Bmc_91025] [SWS_Bmc_91026]

A.2.3 Deleted Specification Items in R23-11

none

A.3 Constraint and Specification 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

[SWS_Bmc_91019] [SWS_Bmc_91020] [SWS_Bmc_91021] [SWS_Bmc_91022] [SWS_Bmc_91023] [SWS_Bmc_91024] [SWS_Bmc_91025] [SWS_Bmc_91026] [SWS_Bmc_91027] [SWS_Bmc_91028] [SWS_Bmc_91029] [SWS_Bmc_91030] [SWS_Bmc_91031] [SWS_Bmc_91032] [SWS_Bmc_91033] [SWS_Bmc_91034] [SWS_Bmc_91035] [SWS_Bmc_91036]

A.3.3 Deleted Specification Items in R24-11

none

B Not applicable requirements

[SWS_BMC_00999]

Upstream requirements: SRS_BSW_00448

⌈These requirements are not applicable to this specification.⌋