Document Title	Specification of Bit Handling Routines
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	399

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	• No content change
2023-11-23	R23-11	AUTOSAR Release Management	• No content change
2022-11-24	R22-11	AUTOSAR Release Management	• New Functions added: SWS_Bfx_91002, SWS_Bfx_00134, SWS_Bfx_00135, SWS_Bfx_91003, SWS_Bfx_00137, SWS_Bfx_91004, SWS_Bfx_00139, SWS_Bfx_91005 and SWS_Bfx_00141.
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes (only converted to LaTex) • Artifact inclusion based on ArtifactAnalysis corrected
2020-11-30	R20-11	AUTOSAR Release Management	• Chapter 7.1 Error sections updated
2019-11-28	R19-11	AUTOSAR Release Management	• Editorial changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Addition of 64bit handling requirement

△

2017-12-08	4.3.1	AUTOSAR Release Management	• Addition on mnemonic for boolean as “u8“ • Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Removal of the requirement SWS_Bfx_00204 • Updation of MISRA violation comment format • Updation of unspecified value range for BitPn, BitStartPn, BitLn and ShiftCnt • Clarifications
2015-07-31	4.2.2	AUTOSAR Release Management	• Updated SWS_Bfx_00017 for the return type of Bfx_GetBit function from 1 and 0 to TRUE and FALSE • Updated chapter 8.1 for the definition of bit addressing and updated the examples of Bfx_SetBit, Bfx_ClrBit, Bfx_GetBit, Bfx_SetBits, Bfx_CopyBit, Bfx_PutBits, Bfx_PutBit • Updated SWS_Bfx_00017 for the return type of Bfx_GetBit function from 1 and 0 to TRUE and FALSE without changing the formula • Updated SWS_Bfx_00011 and SWS_Bfx_00022 for the review comments provided for the examples • In Table 2, replaced Boolean with boolean • In SWS_Bfx_00029, in example re-placed BFX_GetBits_u16u8u8_u16 with Bfx_GetBits_u16u8u8_u16
2014-10-31	4.2.1	AUTOSAR Release Management	• Correct usage of const in function declarations • Editoral changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Editorial Changes

▽

△

2013-10-31	4.1.2	AUTOSAR Release Management	• Improve description of how to map functions to C-files • Improve the definition of error classification • Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• Change return value of Test Bit API to boolean. • Improve memory map handling • Change number of parameter in Put Bit Api.
2011-12-22	4.0.3	AUTOSAR Administration	• Requirements described with more clarity for ‘Bit Shift and Rotate‘ operations • Table correction for PutBit routines • ‘Copy Bit routine‘ interfaces corrected • Error classification support and definition removed as DET call not supported by library • Configuration parameter description / support removed for XXX_ GetVersionInfo routine • Renaming of the term DET in the abbreviation to “Default Error Trace“
2009-12-18	4.0.1	AUTOSAR Administration	• Signature for necessary Bit handling functions optimized for easy usage • Bit handling on all signed variables eliminated • Additional bit handling functions introduced
2010-02-02	3.1.4	AUTOSAR Administration	• 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.

Contents

1 Introduction and functional overview

AUTOSAR Library routines are the part of system services in AUTOSAR architecture and below figure shows position of AUTOSAR library in layered architecture.

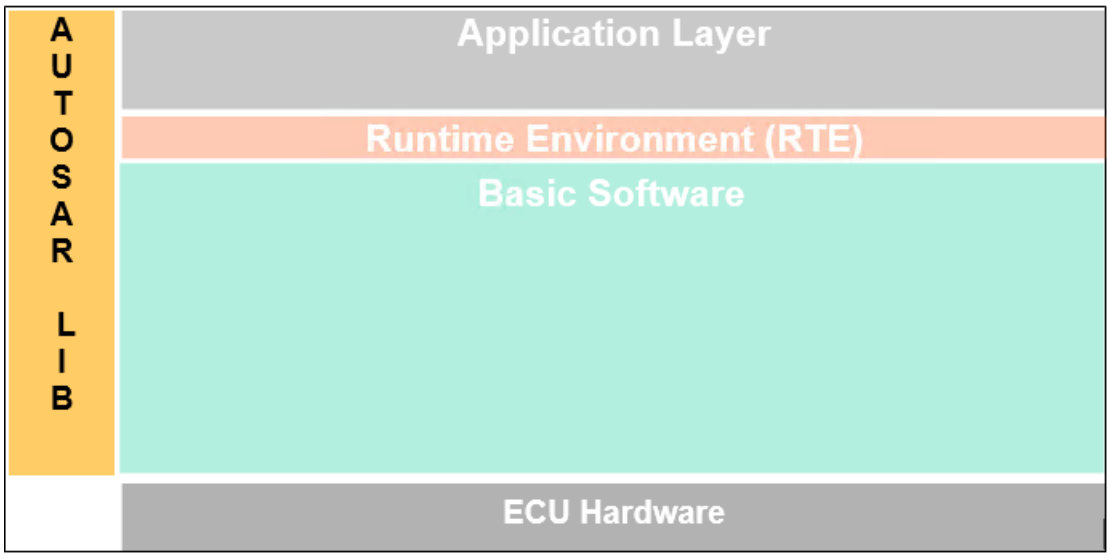

Figure 1.1: Layered Architecture

Bfx routines specification specifies the functionality, API and the configuration of the AUTOSAR library for BIT functionality dedicated to fixed-point arithmetic routines.

All bit functions are re-entrant and can handle several simultaneous requests from the application.

2 Acronyms and Abbreviations

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

Abbreviation / Acronym:	Description:
Bfx	Short name for Bitfield functions for fixed point
u8	Short name for uint8, specified in AUTOSAR_SWS_Platform Types
u16	Short name for uint16, specified in AUTOSAR_SWS_ PlatformTypes
u32	Short name for uint32, specified in AUTOSAR_SWS_ PlatformTypes
s8	Short name for sint8, specified in AUTOSAR_SWS_Platform Types
s16	Short name for sint16, specified in AUTOSAR_SWS_ PlatformTypes
s32	Short name for sint32, specified in AUTOSAR_SWS_ PlatformTypes
boolean	Boolean data type, specified in AUTOSAR_SWS_Platform Types
DET	Default Error Tracer

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] ISO/IEC 9899:1990 Programming Language - C
https://www.iso.org

[3] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[4] Requirements on Libraries
AUTOSAR_CP_RS_Libraries

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, SWS BSW General], which is also valid for IFX Library.

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

4 Constraints and assumptions

4.1 Limitations

No limitations

4.2 Applicability to car domains

No restrictions

5 Dependencies to other modules

5.1 File structure

[SWS_Bfx_00220]

Upstream requirements: SRS_LIBS_00005

⌈The Bfx module shall provide the following files:

- C files, Bfx_<name>.c used to implement the library. All C files shall be prefixed with 'Bfx'.

Header file Bfx.h provides all public function prototypes and types defined by the BFX library specification⌋

[SWS_Bfx_00222] ⌈Implementation & grouping of routines with respect to C files shall be done according to one of the options described below.⌋

Option 1 : <Name> can be function name providing one C file per function,

eg.: Bfx_setbit.c etc.

Option 2 : <Name> can have common name of group of functions:

- 2.1 Group by object family:

  eg.:Bfx_set.c, Bfx_get.c

- 2.2 Group by routine family:

  eg.: Bfx_bit8.c,Bfx_bit16.c etc.

- 2.3 Group by other methods: (individual grouping allowed)

Option 3 : <Name> can be removed so that single C file shall contain all Bfx functions, eg.: Bfx.c.

Using above options gives certain flexibility of choosing suitable granularity with reduced number of C files. Depending on the tool-chain linking on demand can be possible or not.

6 Requirements Tracing

The following tables reference the requirements specified in [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_00007]	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	[SWS_Bfx_00209]
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_Bfx_00212]
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_Bfx_00213]
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_Bfx_00212]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_Bfx_00302]
[SRS_BSW_00411]	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	[SWS_Bfx_00302]
[SRS_BSW_00448]	Module SWS shall not contain requirements from other modules	[SWS_Bfx_00999]
[SRS_LIBS_00001]	The functional behavior of each library functions shall not be configurable	[SWS_Bfx_00314]
[SRS_LIBS_00002]	A library shall be operational before all BSW modules and application SW-Cs	[SWS_Bfx_00200]
[SRS_LIBS_00003]	A library shall be operational until the shutdown	[SWS_Bfx_00201]
[SRS_LIBS_00005]	Each library shall provide one header file with its public interface	[SWS_Bfx_00135] [SWS_Bfx_00137] [SWS_Bfx_00139] [SWS_Bfx_00141] [SWS_Bfx_00220]
[SRS_LIBS_00007]	Using a library should be documented	[SWS_Bfx_00205]
[SRS_LIBS_00009]	All library functions shall be re-entrant	[SWS_Bfx_00135] [SWS_Bfx_00137] [SWS_Bfx_00139] [SWS_Bfx_00141]
[SRS_LIBS_00011]	All function names and type names shall start with "Library short name_"	[SWS_Bfx_00135] [SWS_Bfx_00137] [SWS_Bfx_00139] [SWS_Bfx_00141]
[SRS_LIBS_00015]	It shall be possible to configure the microcontroller so that the library code is shared between all callers	[SWS_Bfx_00206]
[SRS_LIBS_00017]	Usage of macros should be avoided	[SWS_Bfx_00207]
[SRS_LIBS_00018]	A library function may only call library functions	[SWS_Bfx_00203] [SWS_Bfx_00208]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Error Classification

[SWS_Bfx_00223] : ⌈Section 7.1 "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_Bfx_00200]

Upstream requirements: SRS_LIBS_00002

⌈Bfx library shall not require 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_Bfx_00201]

Upstream requirements: SRS_LIBS_00003

⌈Bfx library shall not require a shutdown operation phase.⌋

7.3 Using Library API

[SWS_Bfx_00203]

Upstream requirements: SRS_LIBS_00018

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

[SWS_Bfx_00205]

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 Implementation-DependencyOnArtifact in the BSW/SWC template.

minVersion and maxVersion parameters correspond to the supplier version. In case of AUTOSAR library, these parameters may be left empty because a SWC or BSW module may rely on library behaviour, not on a supplier implementation. However, the SWC or BSW modules shall be compatible with the AUTOSAR platform where they are integrated.⌋

7.4 Library implementation

[SWS_Bfx_00206]

Upstream requirements: SRS_LIBS_00015

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

[SWS_Bfx_00207]

Upstream requirements: SRS_LIBS_00017

⌈Usage of macros must be avoided in the context of Library. The library function must be declared as function or as inline function and Macro #define should not be used.⌋

[SWS_Bfx_00208]

Upstream requirements: SRS_LIBS_00018

⌈A library function shall not call any BSW module functions, e.g. the DET. A library function can call any other library functions since all library functions are re-entrant but not BSW module functions, as they may not be re-entrant⌋

[SWS_Bfx_00209]

Upstream requirements: SRS_BSW_00007

⌈The library, written in C programming language, should confirm to the MISRA C Standard.

Please refer to SWS_BSW_00115 for more details.⌋

[SWS_Bfx_00212]

Upstream requirements: SRS_BSW_00304, SRS_BSW_00378

⌈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 a platform.⌋

[SWS_Bfx_00213]

Upstream requirements: SRS_BSW_00348

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

[SWS_Bfx_00214] ⌈All Bit Library modules shall avoid handling user faults and values outside specified range.⌋

8 API specification

8.1 Imported types

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

[SWS_Bfx_91001] Definition of imported datatypes of module Bfx ⌈

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

⌋

It is observed that since the sizes of the integer types provided by the C language are implementation-defined, the range of values that may be represented within each of the integer types will vary between implementations.

Thus, in order to improve the portability of the software, these types are defined in Platform_Types.h [6]. The following mnemonic are used in the library routine names.

Note:

The naming convention for the api's with boolean return type/parameter type is given as _u8 which shall be interpreted as _b. (Boolean)

If there is no boolean data type present in the return type/parameter type then _u8 shall be interpreted as _u8 only.

Size	Platform Type	Mnemonic
unsigned 8-Bit	boolean	u8
signed 8-Bit	sint8	s8
signed 16-Bit	sint16	s16
signed 32-Bit	sint32	s32
unsigned 8-Bit	uint8	u8
unsigned 16-Bit	uint16	u16
unsigned 32-Bit	uint32	u32

Table 8.1: Base Types

As described in [6], the ranges for each of the base types are shown in Table 2.

Base Type	Range
boolean	[TRUE,FALSE]
uint8	[0, 255]
sint8	[-128, 127]
uint16	[0, 65535]

▽

△

sint16	[-32768, 32767]
uint32	[0, 4294967295]
sint32	[-2147483648, 2147483647]

Table 8.2: Ranges for Base Types

As a convention in the rest of the document:

- Mnemonics will be used in the name of the routines (using <InTypeMn1> that means Type Mnemonic for Input 1)
- The real type will be used in the description of the prototypes of the routines (using <InType> or <OutType>).

The bit addressing for the document is

- The bit position of the lowest significant bit is defined as 0(zero)
- The bit field length is defined as the number of bits.

8.2 Type definitions

None

8.3 Comment about functions optimized for target

The functions described in this library may be realized as regular functions or as a , inline functions

8.4 Bit functions definitions

8.4.1 Bfx_SetBit

[SWS_Bfx_00001] Definition of API function Bfx_SetBit_<TypeMn>u8 ⌈

Service Name	Bfx_SetBit_<TypeMn>u8	
Syntax	`void Bfx_SetBit_<TypeMn>u8 ( <Type>* Data, uint8 BitPn )`	
Service ID [hex]	0x01 to 0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BitPn	Bit position
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall set the logical status of input data as '1' at the requested bit position.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00002] ⌈

Expected functionality:

*Data = *Data | (0x01 << BitPn)

For Example:

Data = 10001010b

Bfx_SetBit_u8u8(&Data, 2)

The Data will be updated to 10001110b⌋

[SWS_Bfx_00008] ⌈List of implemented functions⌋

Function ID[hex]	Function prototype	Maximum value of BitPn
0x001	void Bfx_SetBit_u8u8(uint8*, uint8)	7
0x002	void Bfx_SetBit_u16u8(uint16*, uint8)	15
0x003	void Bfx_SetBit_u32u8(uint32*, uint8)	31
0x004	void Bfx_SetBit_u64u8(uint64*, uint8)	63

8.4.2 Bfx_ClrBit

[SWS_Bfx_00010] Definition of API function Bfx_ClrBit_<TypeMn>u8 ⌈

Service Name	Bfx_ClrBit_<TypeMn>u8	

Syntax	`void Bfx_ClrBit_<TypeMn>u8 (`
	`  <Type>* Data,`
	`  uint8 BitPn`
	`)`	

Service ID [hex]	0x06 to 0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BitPn	Bit position
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall clear the logical status of the input data to '0' at the requested bit position.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00011] ⌈

Expected functionality:

*Data = (*Data & ~(0x01 << BitPn))

For Example:

Data = 10001010b

Bfx_ClrBit_u8u8(&Data, 1)

The Data will be updated to 10001000b⌋

[SWS_Bfx_00015] ⌈List of implemented functions⌋

Function ID[hex]	Function prototype	Maximum value of BitPn
0x006	void Bfx_ClrBit_u8u8(uint8*, uint8)	7
0x007	void Bfx_ClrBit_u16u8(uint16*, uint8)	15
0x008	void Bfx_ClrBit_u32u8(uint32*, uint8)	31
0x009	void Bfx_ClrBit_u64u8(uint64*, uint8)	63

8.4.3 Bfx_GetBit

[SWS_Bfx_00016] Definition of API function Bfx_GetBit_<InTypeMn>u8_u8 ⌈

Service Name	Bfx_GetBit_<InTypeMn>u8_u8	

Syntax	`boolean Bfx_GetBit_<InTypeMn>u8_u8 (`
	`  <InType> Data,`
	`  uint8 BitPn`
	`)`	

Service ID [hex]	0x0a to 0x0c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input data
	BitPn	Bit position
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	Bit Status
Description	This function shall return the logical status of the input data for the requested bit position.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00017] ⌈

Result = TRUE, ((Data & (0x01 << BitPn)) != 0)

Result = FALSE, else

For Example:

Bfx_GetBit_u8u8(10001010b, 1)

returns TRUE⌋

[SWS_Bfx_00020] ⌈List of implemented functions⌋

Function ID[hex]	Function prototype	maximum value of BitPn
0x00A	boolean Bfx_GetBit_u8u8_ u8(uint8,uint8)	7
0x00B	boolean Bfx_GetBit_u16u8_ u8(uint16,uint8)	15
0x00C	boolean Bfx_GetBit_u32u8_ u8(uint32,uint8)	31
0x00D	boolean Bfx_GetBit_u64u8_ u8(uint64,uint8)	63

8.4.4 Bfx_SetBits

[SWS_Bfx_00021] Definition of API function Bfx_SetBits_<TypeMn>u8u8u8 ⌈

Service Name	Bfx_SetBits_<TypeMn>u8u8u8	

Syntax	`void Bfx_SetBits_<TypeMn>u8u8u8 (`
	`  <Type>* Data,`
	`  uint8 BitStartPn,`
	`  uint8 BitLn,`
	`  uint8 Status`
	`)`	

Service ID [hex]	0x20 to 0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BitStartPn	Start bit position
	BitLn	Bit field length
	Status	Status value
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall set the input data as '1' or '0' as per 'Status' value starting from 'BitStartPn' for the length 'BitLn'.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00022] ⌈

15	14	13	12	11	10	9	-	2	1	0
1	1	1	0	1	0	0	0	0	0	0
						<Bit Start Pn>				
		< BitLn >								

⌋

For Example:

Data = 1110100000000111b

Bfx_SetBits_u16u8u8u8(&Data, 5, 5, 1)

The Data will be updated to 1110101111100111b

[SWS_Bfx_00025] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of Bit Ln	Maximum value of Bit StartPn	Maximum value for BitStartPn + BitLn
0x020	void Bfx_SetBits_ u8u8u8u8(uint8*, uint8, uint8, uint8)	8	7	8
0x021	void Bfx_SetBits_ u16u8u8u8(uint16*, uint8, uint8, uint8)	16	15	16
0x022	void Bfx_SetBits_ u32u8u8u8(uint32*, uint8, uint8, uint8)	32	31	32
0x023	void Bfx_SetBits_ u64u8u8u8(uint64*, uint8, uint8, uint8)	64	63	64

8.4.5 Bfx_GetBits

[SWS_Bfx_00028] Definition of API function Bfx_GetBits_<TypeMn>u8u8_<TypeMn> ⌈

Service Name	Bfx_GetBits_<TypeMn>u8u8_<TypeMn>	
Syntax	`<Type> Bfx_GetBits_<TypeMn>u8u8_<TypeMn> ( <Type> Data, uint8 BitStartPn, uint8 BitLn )`	
Service ID [hex]	0x26 to 0x28	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input data
	BitStartPn	Start bit position
	BitLn	Bit field length
Parameters (inout)	None	
Parameters (out)	None	
Return value	<Type>	Bit field sequence
Description	This function shall return the Bits of the input data starting from 'BitStartPn' for the length of 'Bit Ln'.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00029] ⌈

15	14	13	12	11	10	9	-	2	1	0
1	1	1	0	1	0	0	0	1	1	1
						BitStart Pn				
		< BitLn >								

⌋

For Example:

Bfx_GetBits_u16u8u8_u16(1110100000000111b, 9, 5)

returns 0000000000010100b

[SWS_Bfx_00034] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of Bit Ln	Maximum value of Bit StartPn	Maximum value for BitStartPn + BitLn
0x026	uint8 Bfx_GetBits_ u8u8u8_ u8(uint8,uint8,uint8)	8	7	8
0x027	uint16 Bfx_GetBits_ u16u8u8_ u16(uint16,uint8,uint8)	16	15	16
0x028	uint32 Bfx_GetBits_ u32u8u8_ u32(uint32,uint8,uint8)	32	31	32
0x029	uint64 Bfx_GetBits_ u64u8u8_ u64(uint64,uint8,uint8)	64	63	64

8.4.6 Bfx_SetBitMask

[SWS_Bfx_00035] Definition of API function Bfx_SetBitMask_<TypeMn><TypeMn> ⌈

Service Name	Bfx_SetBitMask_<TypeMn><TypeMn>	
Syntax	`void Bfx_SetBitMask_<TypeMn><TypeMn> ( <Type>* Data, <Type> Mask )`	
Service ID [hex]	0x2a to 0x2c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Mask	Mask used to set bits

▽

△

Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall set the data to logical status '1' as per the corresponding Mask bits when set to value 1 and remaining bits will retain their original values.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00036] ⌈

Expected functionality:

*Data = *Data | Mask

⌋

[SWS_Bfx_00038] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0X02A	void Bfx_SetBitMask_u8u8(uint8*, uint8)
0X02B	void Bfx_SetBitMask_u16u16(uint16*, uint16)
0X02C	void Bfx_SetBitMask_u32u32(uint32*, uint32)
0x02D	void Bfx_SetBitMask_u64u64(uint64*, uint64)

8.4.7 Bfx_ClrBitMask

[SWS_Bfx_00039] Definition of API function Bfx_ClrBitMask_<TypeMn><TypeMn> ⌈

Service Name	Bfx_ClrBitMask_<TypeMn><TypeMn>	

Syntax	`void Bfx_ClrBitMask_<TypeMn><TypeMn> (`
	`  <Type>* Data,`
	`  <Type> Mask`
	`)`	

Service ID [hex]	0x30 to 0x32	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Mask	Mask value
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	

▽

△

Description	This function shall clear the logical status to '0' for the input data for all the bit positions as per the mask.
Available via	Bfx.h

⌋

[SWS_Bfx_00040] ⌈

This function shall clear the data to logical status '0' as per the corresponding mask bits value when set to 1. The remaining bits shall retain their original values.

Expected functionality:

*Data = *Data & ~Mask

⌋

[SWS_Bfx_00045] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0x030	void Bfx_ClrBitMask_u8u8(uint8*, uint8)
0x031	void Bfx_ClrBitMask_u16u16(uint16*, uint16)
0x032	void Bfx_ClrBitMask_u32u32(uint32*, uint32)
0x033	void Bfx_ClrBitMask_u64u64(uint64*, uint64)

8.4.8 Bfx_TstBitMask

[SWS_Bfx_00046] Definition of API function Bfx_TstBitMask_<InTypeMn><InTypeMn>_u8 ⌈

Service Name	Bfx_TstBitMask_<InTypeMn><InTypeMn>_u8	

Syntax	`boolean Bfx_TstBitMask_<InTypeMn><InTypeMn>_u8 (`
	`  <InType> Data,`
	`  <InType> Mask`
	`)`	

Service ID [hex]	0x36 to 0x38	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input data
	Mask	Mask value
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	Value

▽

△

Description	This function shall return TRUE, if all bits defined in Mask value are set in the input Data value. In all other cases this function shall return FALSE.
Available via	Bfx.h

⌋

[SWS_Bfx_00047] ⌈

Result = TRUE, ((Data & Mask) == Mask)

Result = FALSE, all other case

⌋

For example:

Bfx_TstBitMask_u8u8_u8(10010011b,10010000b) returns TRUE.

[SWS_Bfx_00050] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0x036	boolean Bfx_TstBitMask_u8u8_u8(uint8,uint8)
0x037	boolean Bfx_TstBitMask_u16u16_u8(uint16,uint16)
0x038	boolean Bfx_TstBitMask_u32u32_u8(uint32,uint32)
0x039	boolean Bfx_TstBitMask_u64u64_u8(uint64,uint64)

8.4.9 Bfx_TstBitLnMask

[SWS_Bfx_00051] Definition of API function Bfx_TstBitLnMask_<InTypeMn><InTypeMn>_u8 ⌈

Service Name	Bfx_TstBitLnMask_<InTypeMn><InTypeMn>_u8	

Syntax	`boolean Bfx_TstBitLnMask_<InTypeMn><InTypeMn>_u8 (`
	`  <InType> Data,`
	`  <InType> Mask`
	`)`	

Service ID [hex]	0x3a to 0x3c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input data
	Mask	Mask value
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	Data
Description	This function makes a test on the input data and if at least one bit is set as per the mask, then the function shall return TRUE, otherwise it shall return FALSE.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00055] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0x03A	boolean Bfx_TstBitLnMask_u8u8_u8(uint8,uint8)
0x03B	boolean Bfx_TstBitLnMask_u16u16_u8(uint16,uint16)
0x03C	boolean Bfx_TstBitLnMask_u32u32_u8(uint32,uint32)
0x03D	boolean Bfx_TstBitLnMask_u64u64_u8(uint64,uint64)

8.4.10 Bfx_TstParityEven

[SWS_Bfx_00056] Definition of API function Bfx_TstParityEven_<InTypeMn>_u8 ⌈

Service Name	Bfx_TstParityEven_<InTypeMn>_u8	

Syntax	`boolean Bfx_TstParityEven_<InTypeMn>_u8 (`
	`  <InTypeMn> Data`
	`)`	

Service ID [hex]	0x40 to 0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input Data
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	Status
Description	This function tests the number of bits set to 1. If this number is even, it shall return TRUE, otherwise it returns FALSE.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00060] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0x040	boolean Bfx_TstParityEven_u8_u8(uint8)
0x041	boolean Bfx_TstParityEven_u16_u8(uint16)
0x042	boolean Bfx_TstParityEven_u32_u8(uint32)
0x043	boolean Bfx_TstParityEven_u64_u8(uint64)

8.4.11 Bfx_ToggleBits

[SWS_Bfx_00061] Definition of API function Bfx_ToggleBits_<TypeMn> ⌈

Service Name	Bfx_ToggleBits_<TypeMn>	

Syntax	`void Bfx_ToggleBits_<TypeMn> (`
	`  <Type>* Data`
	`)`	

Service ID [hex]	0x46 to 0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	Data	Pointer to input data

▽

△

Parameters (out)	None
Return value	None
Description	This function toggles all the bits of data (1's Complement Data).
Available via	Bfx.h

⌋

[SWS_Bfx_00065] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0x046	void Bfx_ToggleBits_u8(uint8*)
0x047	void Bfx_ToggleBits_u16(uint16*)
0x048	void Bfx_ToggleBits_u32(uint32*)
0x049	void Bfx_ToggleBits_u64(uint64*)

8.4.12 Bfx_ToggleBitMask

[SWS_Bfx_00066] Definition of API function Bfx_ToggleBitMask_<TypeMn><TypeMn> ⌈

Service Name	Bfx_ToggleBitMask_<TypeMn><TypeMn>	

Syntax	`void Bfx_ToggleBitMask_<TypeMn><TypeMn> (`
	`  <Type>* Data,`
	`  <Type> Mask`
	`)`	

Service ID [hex]	0x4a to 0x4c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Mask	Mask
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function toggles the bits of data when the corresponding bit of the mask is enabled and set to 1.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00069] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0x04A	void Bfx_ToggleBitMask_u8u8(uint8*, uint8)
0x04B	void Bfx_ToggleBitMask_u16u16(uint16*, uint16)
0x04C	void Bfx_ToggleBitMask_u32u32(uint32*, uint32)
0x04D	void Bfx_ToggleBitMask_u64u64(uint64*, uint64)

8.4.13 Bfx_ShiftBitRt

[SWS_Bfx_00070] Definition of API function Bfx_ShiftBitRt_<TypeMn>u8 ⌈

Service Name	Bfx_ShiftBitRt_<TypeMn>u8	

Syntax	`void Bfx_ShiftBitRt_<TypeMn>u8 (`
	`  <Type>* Data,`
	`  uint8 ShiftCnt`
	`)`	

Service ID [hex]	0x50 to 0x52	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ShiftCnt	Shift right count
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall shift data to the right by ShiftCnt. The most significant bit (left-most bit) is replaced by a '0' bit and the least significant bit (right-most bit) is discarded for every single bit shift cycle.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00075] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of ShiftCnt
0X050	void Bfx_ShiftBitRt_u8u8(uint8*, uint8)	7
0X051	void Bfx_ShiftBitRt_u16u8(uint16*, uint8)	15
0X052	void Bfx_ShiftBitRt_u32u8(uint32*, uint8)	31
0x053	void Bfx_ShiftBitRt_u64u8(uint64*, uint8)	63

8.4.14 Bfx_ShiftBitLt

[SWS_Bfx_00076] Definition of API function Bfx_ShiftBitLt_<TypeMn>u8 ⌈

Service Name	Bfx_ShiftBitLt_<TypeMn>u8	

Syntax	`void Bfx_ShiftBitLt_<TypeMn>u8 (`
	`  <Type>* Data,`
	`  uint8 ShiftCnt`
	`)`	

Service ID [hex]	0x56 to 0x58	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ShiftCnt	Shift left count
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall shift data to the left by ShiftCnt. The least significant bit (right-most bit) is replaced by a '0' bit and the most significant bit (left-most bit) is discarded for every single bit shift cycle.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00080] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of ShiftCnt
0X056	void Bfx_ShiftBitLt_u8u8(uint8*, uint8)	7
0X057	void Bfx_ShiftBitLt_u16u8(uint16*, uint8)	15
0X058	void Bfx_ShiftBitLt_u32u8(uint32*, uint8)	31
0x059	void Bfx_ShiftBitLt_u64u8(uint64*, uint8)	63

8.4.15 Bfx_RotBitRt

[SWS_Bfx_00086] Definition of API function Bfx_RotBitRt_<TypeMn>u8 ⌈

Service Name	Bfx_RotBitRt_<TypeMn>u8	

Syntax	`void Bfx_RotBitRt_<TypeMn>u8 (`
	`  <Type>* Data,`
	`  uint8 ShiftCnt`
	`)`	

Service ID [hex]	0x5a to 0x5c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ShiftCnt	Shift count
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall rotate data to the right by ShiftCnt. The least significant bit is rotated to the most significant bit location for every single bit shift cycle.	
Available via	Bfx.h	

⌋

For example:

If ShiftCnt = 1 then,

uint8 Data = 0001 0111 (before rotate right)

Data = 1000 1011 (after rotate right)

If ShiftCnt = 3 then,

uint8 Data = 0001 0111 (before rotate right)

Data = 1110 0010 (after rotate right)

[SWS_Bfx_00090] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of ShiftCnt
0X05A	void Bfx_RotBitRt_u8u8(uint8*, uint8)	7
0X05B	void Bfx_RotBitRt_u16u8(uint16*, uint8)	15
0X05C	void Bfx_RotBitRt_u32u8(uint32*, uint8)	31
0x05D	void Bfx_RotBitRt_u64u8(uint64*, uint8)	63

8.4.16 Bfx_RotBitLt

[SWS_Bfx_00095] Definition of API function Bfx_RotBitLt_<TypeMn>u8 ⌈

Service Name	Bfx_RotBitLt_<TypeMn>u8	
Syntax	`void Bfx_RotBitLt_<TypeMn>u8 ( <Type>* Data, uint8 ShiftCnt )`	
Service ID [hex]	0x60 to 0x62	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ShiftCnt	Shift count
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall rotate data to the left by ShiftCnt. The most significant bit is rotated to the least significant bit location for every single bit shift cycle.	
Available via	Bfx.h	

⌋

For example:

If ShiftCnt = 1 then,

uint8 Data = 1011 0111 (before rotate left)

Data = 0110 1111 (after rotate left)

If ShiftCnt = 3 then,

uint8 Data = 1011 0111 (before rotate left)

Data = 1011 1101 (after rotate left)

[SWS_Bfx_00098] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of ShiftCnt
0X060	void Bfx_RotBitLt_u8u8(uint8*, uint8)	7
0X061	void Bfx_RotBitLt_u16u8(uint16*, uint8)	15
0X062	void Bfx_RotBitLt_u32u8(uint32*, uint8)	31
0x063	void Bfx_RotBitLt_u64u8(uint64*, uint8)	63

8.4.17 Bfx_CopyBit

[SWS_Bfx_00101] Definition of API function Bfx_CopyBit_<TypeMn>u8<TypeMn>u8 ⌈

Service Name	Bfx_CopyBit_<TypeMn>u8<TypeMn>u8	

Syntax	`void Bfx_CopyBit_<TypeMn>u8<TypeMn>u8 (`
	`  <Type>* DestinationData,`
	`  uint8 DestinationPosition,`
	`  <Type> SourceData,`
	`  uint8 SourcePosition`
	`)`	

Service ID [hex]	0x66 to 0x68	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	DestinationPosition	Destination position
	SourceData	Source data
	SourcePosition	Source position
Parameters (inout)	DestinationData	Pointer to destination data
Parameters (out)	None	
Return value	None	
Description	This function shall copy a bit from source data from bit position to destination data at bit position.	
Available via	Bfx.h	

⌋

For Example:

DestinationData = 10100001b

BFX_CopyBit_u8u8u8u8(&DestinationData, 6, 11011010, 1)

The DestinationData will have 11100001b

[SWS_Bfx_00108] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value for SourcePosition and DestinationPosition
0X066	void Bfx_CopyBit_u8u8u8u8(uint8*, uint8, uint8, uint8)	7
0X067	void Bfx_CopyBit_u16u8u16u8(uint16*, uint8, uint16, uint8)	15
0X068	void Bfx_CopyBit_u32u8u32u8(uint32*, uint8, uint32, uint8)	31
0x069	void Bfx_CopyBit_u64u8u64u8(uint64*, uint8, uint64, uint8)	63

8.4.18 Bfx_PutBits

[SWS_Bfx_00110] Definition of API function Bfx_PutBits_<TypeMn>u8u8<TypeMn> ⌈

Service Name	Bfx_PutBits_<TypeMn>u8u8<TypeMn>	
Syntax	`void Bfx_PutBits_<TypeMn>u8u8<TypeMn> ( <Type>* Data, uint8 BitStartPn, uint8 BitLn, <Type> Pattern )`	
Service ID [hex]	0x70 to 0x72	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BitStartPn	Start bit position
	BitLn	Bit field length
	Pattern	Pattern to be set
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall put bits as mentioned in Pattern to the input Data from the specified bit position.	
Available via	Bfx.h	

⌋

For Example:

Data = 11110000b

Bfx_PutBits_u8u8u8u8(&Data, 1, 3, 00000011b)

The Data will have 11110110b

[SWS_Bfx_00112] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of Bit Ln	Maximum value of Bit StartPn	Maximum value for BitStartPn + BitLn
0x070	void Bfx_PutBits_ u8u8u8u8(uint8*, uint8, uint8, uint8)	8	7	8
0x071	void Bfx_PutBits_ u16u8u8u16(uint16*, uint8, uint8, uint16)	16	15	16
0x072	void Bfx_PutBits_ u32u8u8u32(uint32*, uint8, uint8, uint32)	32	31	32
0x073	void Bfx_PutBits_ u64u8u8u64(uint64*, uint8, uint8, uint64)	64	63	64

8.4.19 Bfx_PutBitsMask

[SWS_Bfx_00120] Definition of API function Bfx_PutBitsMask_<TypeMn><TypeMn><TypeMn> ⌈

Service Name	Bfx_PutBitsMask_<TypeMn><TypeMn><TypeMn>	

Syntax	`void Bfx_PutBitsMask_<TypeMn><TypeMn><TypeMn> (`
	`  <Type>* Data,`
	`  <Type> Pattern,`
	`  <Type> Mask`
	`)`	

Service ID [hex]	0x80 to 0x82	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Pattern	Pattern to be set
	Mask	Mask value
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall put all bits defined in Pattern and for which the corresponding Mask bit is set to 1 in the input Data.	
Available via	Bfx.h	

⌋

For Example:

Bfx_PutBitsMask_u8u8u8(11100000b, 11001101b, 00001111b)

results in *Data = 11101101b

[SWS_Bfx_00124] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype
0x080	void Bfx_PutBitsMask_u8u8u8(uint8*, uint8, uint8)
0x081	void Bfx_PutBitsMask_u16u16u16(uint16*, uint16, uint16)
0x082	void Bfx_PutBitsMask_u32u32u32(uint32*, uint32, uint32)
0x083	void Bfx_PutBitsMask_u64u64u64(uint64*, uint64, uint64)

8.4.20 Bfx_PutBit

[SWS_Bfx_00130] Definition of API function Bfx_PutBit_<TypeMn>u8u8 ⌈

Service Name	Bfx_PutBit_<TypeMn>u8u8	

Syntax	`void Bfx_PutBit_<TypeMn>u8u8 (`
	`  <Type>* Data,`
	`  uint8 BitPn,`
	`  boolean Status`
	`)`	

Service ID [hex]	0x85 to 0x87	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BitPn	Bit position
	Status	Status value
Parameters (inout)	Data	Pointer to input data
Parameters (out)	None	
Return value	None	
Description	This function shall update the bit specified by BitPn of input data as '1' or '0' as per 'Status' value.	
Available via	Bfx.h	

⌋

For Example:

uint8 InputData = 11100111b;

Bfx_PutBit_u8u8u8(&InputData, 4, TRUE);

results in InputData = 11110111b

[SWS_Bfx_00132] ⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum value of BitPn
0x085	void Bfx_PutBit_u8u8u8(uint8*, uint8, boolean)	7
0x086	void Bfx_PutBit_u16u8u8(uint16*, uint8, boolean)	15
0x087	void Bfx_PutBit_u32u8u8(uint32*, uint8, boolean)	31
0x088	void Bfx_PutBit_u64u8u8(uint64*, uint8, boolean)	63

8.4.21 Bfx_ShiftBitSat

[SWS_Bfx_91002] Definition of API function Bfx_ShiftBitSat_<TypeMn>s8 ⌈

Service Name	Bfx_ShiftBitSat_<TypeMn>s8	

Syntax	`<Type> Bfx_ShiftBitSat_<TypeMn>s8 (`
	`  sint8 ShiftCnt,`
	`  <Type> Data`
	`)`	

Service ID [hex]	0x100	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ShiftCnt	Shift count (-<MaxShiftRight> ... -1: right, 1 ... <MaxShiftLeft>: left)
Parameters (inout)	Data	Input value
Parameters (out)	None	
Return value	<Type>	Shifted and saturated bit pattern.
Description	Arithmetic shift with saturation	
Available via	Bfx.h	

⌋

[SWS_Bfx_00134] ⌈

If the shift count is greater than or equal to zero, then shift the value in Data by the amount specified by shift count to left.

For signed data an arithmetic shift is performed. The vacated bits are filled with zeros and the result is saturated if its sign bit differs from the sign bits that are shifted out.

For unsigned data a logical shift is performed. In this case the result is saturated, if the leading one bit is shifted out.

If the shift count is less than zero, right-shift the value in Data by the absolute value of the shift count. The vacated bits are filled with the sign-bit (the most significant bit) and bits shifted out are discarded.

Note that a shift right by the word width leaves all zeros or all ones in the result, depending on the sign-bit.

Example of 32 bit signed integer: The range for shift count is -32 to +31, allowing a shift left up to 31 bit positions and a shift right up to 32 bit positions (a shift right by 32 bits leaves all zeros or all ones in the result, depending on the sign bit).⌋

[SWS_Bfx_00135]

Upstream requirements: SRS_LIBS_00005, SRS_LIBS_00009, SRS_LIBS_00011

⌈List of Implemented Functions:⌋

Function ID[hex]	Function prototype	Max Shift Left	Max Shift Right
0x0F1	Bfx_ShiftBitSat_s8s8_s8	8	8
0X0F2	Bfx_ShiftBitSat_u8s8_u8	8	8
0X0F3	Bfx_ShiftBitSat_s16s8_s16	16	16
0X0F4	Bfx_ShiftBitSat_u16s8_u16	16	16
0X0F5	Bfx_ShiftBitSat_s32s8_s32	31	32
0X0F6	Bfx_ShiftBitSat_u32s8_u32	31	32
0X0F7	Bfx_ShiftBitSat_s64s8_s64	63	64
0X0F8	Bfx_ShiftBitSat_u64s8_u64	63	64

8.4.22 Bfx_CountLeadingOnes

[SWS_Bfx_91003] Definition of API function Bfx_CountLeadingOnes_<TypeMn>
⌈

Service Name	Bfx_CountLeadingOnes_<TypeMn>	

Syntax	`uint8 Bfx_CountLeadingOnes_<TypeMn> (`
	`  <Type> Data`
	`)`	

Service ID [hex]	0x101	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input data
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	Bit position
Description	Count the number of consecutive ones in Data starting with the most significant bit and return the result.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00137]

Upstream requirements: SRS_LIBS_00005, SRS_LIBS_00009, SRS_LIBS_00011

⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum return value
0xF0	uint8 Bfx_CountLeadingOnes_u8 (uint8)	8
0xF1	uint8 Bfx_CountLeadingOnes_u16 (uint16)	16
0xF2	uint8 Bfx_CountLeadingOnes_u32 (uint32)	32
0xF3	uint8 Bfx_CountLeadingOnes_u64 (uint64)	64

8.4.23 Bfx_CountLeadingSigns

[SWS_Bfx_91004] Definition of API function Bfx_CountLeadingSigns_<TypeMn>
⌈

Service Name	Bfx_CountLeadingSigns_<TypeMn>	

Syntax	`uint8 Bfx_CountLeadingSigns_<TypeMn> (`
	`  <Type> Data`
	`)`	

Service ID [hex]	0x102	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input data
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	Bit position
Description	Count the number of consecutive bits which have the same value as most significant bit in Data, starting with bit at position msb minus one. Put the result in Data. It is the number of leading sign bits minus one, giving the number of redundant sign bits in Data.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00139]

Upstream requirements: SRS_LIBS_00005, SRS_LIBS_00009, SRS_LIBS_00011

⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum return value
0xF4	uint8 Bfx_CountLeadingSigns_s8 (sint8)	8
0xF5	uint8 Bfx_CountLeadingSigns_s16 (sint16)	16
0xF6	uint8 Bfx_CountLeadingSigns_s32 (sint32)	32
0xF7	uint8 Bfx_CountLeadingSigns_s64 (sint64)	64

8.4.24 Bfx_CountLeadingZeros

[SWS_Bfx_91005] Definition of API function Bfx_CountLeadingZeros_<TypeMn>
⌈

Service Name	Bfx_CountLeadingZeros_<TypeMn>	

Syntax	`uint8 Bfx_CountLeadingZeros_<TypeMn> (`
	`  <Type> Data`
	`)`	

Service ID [hex]	0x103	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Data	Input data
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint8	Bit position
Description	Count the number of consecutive zeros in Data starting with the most significant bit and return the result.	
Available via	Bfx.h	

⌋

[SWS_Bfx_00141]

Upstream requirements: SRS_LIBS_00005, SRS_LIBS_00009, SRS_LIBS_00011

⌈List of implemented functions:⌋

Function ID[hex]	Function prototype	Maximum return value
0xF8	uint8 Bfx_CountLeadingZeros_u8 (uint8)	8
0xF9	uint8 Bfx_CountLeadingZeros_u16 (uint16)	16
0xFA	uint8 Bfx_CountLeadingZeros_u32 (uint32)	32
0xFB	uint8 Bfx_CountLeadingZeros_u64 (uint64)	64

8.5 Callback notifications

None

8.6 Scheduled functions

The Bfx library does not have scheduled functions.

8.7 Expected interfaces

None

8.7.1 Mandatory interfaces

None

8.7.2 Optional interfaces

None

8.7.3 Configurable interfaces

None

8.8 Version API

8.8.1 Bfx_GetVersionInfo

[SWS_Bfx_00301] Definition of API function Bfx_GetVersionInfo ⌈

Service Name	Bfx_GetVersionInfo	
Syntax	`void Bfx_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	Bfx.h	

⌋

The version information of a BSW module generally contains:

Module Id

Vendor Id

Vendor specific version numbers

[SWS_Bfx_00302]

Upstream requirements: SRS_BSW_00407, SRS_BSW_00411

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

9 Sequence diagrams

Not applicable

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 Bfx.

Chapter 10.3 specifies published information of the module Bfx.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral.

10.2 Containers and configuration parameters

[SWS_Bfx_00314]

Upstream requirements: SRS_LIBS_00001

⌈The Bfx 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 resources consumption optimization.

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral.

A Not applicable requirements

[SWS_Bfx_00999]

Upstream requirements: SRS_BSW_00448

⌈These requirements are not applicable to this specification.⌋

B History of Specification Items

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

B.1 Specification Item History of this document compared to AUTOSAR R23-11

B.1.1 Added Specification Items in R24-11

none

B.1.2 Changed Specification Items in R24-11

none

B.1.3 Deleted Specification Items in R24-11

none

B.2 Specification Item History of this document compared to AUTOSAR R22-11

B.2.1 Added Specification Items in R23-11

none

B.2.2 Changed Specification Items in R23-11

Number	Heading
[SWS_Bfx_00008]	
[SWS_Bfx_00015]	
[SWS_Bfx_00016]	Definition of API function Bfx_GetBit_<InTypeMn>u8_u8
[SWS_Bfx_00020]	
[SWS_Bfx_00025]	

▽

△

Number	Heading
[SWS_Bfx_00034]	
[SWS_Bfx_00038]	
[SWS_Bfx_00045]	
[SWS_Bfx_00046]	Definition of API function Bfx_TstBitMask_<InTypeMn><InTypeMn>_u8
[SWS_Bfx_00050]	
[SWS_Bfx_00051]	Definition of API function Bfx_TstBitLnMask_<InTypeMn><InTypeMn>_u8
[SWS_Bfx_00055]	
[SWS_Bfx_00056]	Definition of API function Bfx_TstParityEven_<InTypeMn>_u8
[SWS_Bfx_00060]	
[SWS_Bfx_00065]	
[SWS_Bfx_00069]	
[SWS_Bfx_00075]	
[SWS_Bfx_00080]	
[SWS_Bfx_00090]	
[SWS_Bfx_00098]	
[SWS_Bfx_00108]	
[SWS_Bfx_00112]	
[SWS_Bfx_00124]	
[SWS_Bfx_00132]	
[SWS_Bfx_00135]	
[SWS_Bfx_00137]	
[SWS_Bfx_00139]	
[SWS_Bfx_00141]	
[SWS_Bfx_91001]	Definition of imported datatypes of module Bfx
[SWS_Bfx_91003]	Definition of API function Bfx_CountLeadingOnes_<TypeMn>
[SWS_Bfx_91004]	Definition of API function Bfx_CountLeadingSigns_<TypeMn>
[SWS_Bfx_91005]	Definition of API function Bfx_CountLeadingZeros_<TypeMn>

Table B.1: Changed Specification Items in R23-11

B.2.3 Deleted Specification Items in R23-11

none