Document Title	Specification of Fixed Point Interpolation Library
Document Category	Software specification (SWS)
Document UID	396
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 API Ifx_GetVersionInfo • Removed general library requirements • Corrected Service IDs of Ifx_DPSearch and Ifx_IpoCub and the description of the cuboid interpolation • Editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Fixed CheckDocumentSource errors
2023-11-23	R23-11	AUTOSAR Release Management	• Fixed CheckDocumentSource errors
2022-11-24	R22-11	AUTOSAR Release Management	• Functions updated: SWS_Ifx_00014, SWS_Ifx_00017, SWS_Ifx_00022, SWS_Ifx_00027, SWS_Ifx_00032, SWS_Ifx_00209, SWS_Ifx_00041, SWS_Ifx_00051, SWS_Ifx_00062, SWS_Ifx_00077, SWS_Ifx_00087, SWS_Ifx_00097, SWS_Ifx_00110, SWS_Ifx_00122, SWS_Ifx_00222, SWS_Ifx_00136, SWS_Ifx_00151, SWS_Ifx_00236, SWS_Ifx_00166, SWS_Ifx_00181, SWS_Ifx_00247, SWS_Ifx_00185, SWS_Ifx_00186, SWS_Ifx_00006 and SWS_Ifx_00821. • Functions added: SWS_Ifx_91002, SWS_Ifx_91003, SWS_Ifx_91004 and SWS_Ifx_91005.

△

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	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• A new requirement (SWS_Ifx_00251) has been added under Section 7.6 to provide clarity on the rounding mechanism for intermediate result calculation. • A requirement (SWS_Ifx_00250) has been removed as it is not realizable for all the scenarios
2016-11-30	4.3.0	AUTOSAR Release Management	• Added a new requirement (SWS_Ifx_00250) to provide info on symmetricity for interpolation services • A note has been added in SWS_Ifx_00016 as a suggestion to provide hardware independent solution too • Section 2 has been updated to include abbreviation for (DET) Default Error tracer • Updated IFX document to support MISRA 2012 standard. (Removed redundant statements in SWS_Ifx_00809 which already exist in SWS_BSW document and SWS_SRS document) • Modified the reference to SRS_BSW_General (SRS_BSW_00437) & (SRS_BSW_00448) for SWS_Ifx_00436 & SWS_Ifx_00999 requirements

▽

2015-07-31	4.2.2	AUTOSAR Release Management	• Added a new statement in Section 8.5 below the formula to provide more clarity to the users • Updated the “Requirements tracea-bility“ section • Updated Record layouts for distributed interpolation routines in SWS_Ifx_00185 • Updated SWS_Ifx_00001 for naming convetion under Section 5.1, File Structure
2014-10-31	4.2.1	AUTOSAR Release Management	• IFX RecordLayout Blueprint reference in section 3.1 • The usage of const is corrected in function parameters for SWS_Ifx_00004, SWS_Ifx_00014, SWS_Ifx_00015, SWS_Ifx_00017, SWS_Ifx_00020, SWS_Ifx_00022, SWS_Ifx_00025, SWS_Ifx_00027, SWS_Ifx_00030, SWS_Ifx_00032, SWS_Ifx_00205 & SWS_Ifx_00209 • Modified serial numbers in Section 3.2
2014-03-31	4.1.3	AUTOSAR Release Management	• Removed columns Element6 & Element7 in the Record Layout table of SWS_Ifx_00186
2013-10-31	4.1.2	AUTOSAR Release Management	• Corrections made for IntMap_s16u8_s8 function in Record Layout Table of SWS_Ifx_00186 • Corrected array-out-of-bounds for Ifx_IpoMap function • Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• Rounding mechanism specified for DPRatio calculation • Corrected the formula for integrated map interpolation and map interpolation • Removed unwanted Ratio calculation for integrated fix-I map look up with rounding and Integrated fix-map look up without rounding and integrated map look-up without rounding • Modified the reference to non-existant metamodel

△

			△ elementCalprmElementPrototype to Parameter-DataPrototype • Corrected for 'DependencyOnArtifact'
2011-12-22	4.0.3	AUTOSAR Administration	• Removal of rounding off feature from 'MAP lookup routines'
2010-09-30	3.1.5	AUTOSAR Administration	• DPSearch function optimised using structure pointer
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.

Table of Contents

1 Introduction and functional overview 9
2 Acronyms and Abbreviations 10
3 Related documentation 11
3.1 Input documents & related standards and norms . . . 11
3.2 Related specification . . . 11
4 Constraints and assumptions 12
4.1 Limitations . . . 12
4.2 Applicability to car domains . . . 12
5 Dependencies to other modules 13
5.1 File structure . . . 13
6 Requirements Tracing 14
7 Functional specification 15
7.1 Initialization and shutdown . . . 15
7.2 Using Library API . . . 15
7.3 Library implementation . . . 15
7.4 Interpolation routines definitions . . . 15
7.4.1 Record layouts for interpolation routines . . . 17
7.4.1.1 Record layouts for map values . . . 17
7.4.1.2 Record layout definitions . . . 18
7.5 Error Classification . . . 18
7.5.1 Development Errors . . . 18
7.5.2 Runtime Errors . . . 18
7.5.3 Production Errors . . . 19
8 API specification 20
8.1 Imported types . . . 20
8.2 Type definitions . . . 21
8.3 Function definitions . . . 21
8.3.1 Comment about routines optimization . . . 21
8.3.1.1 Optimization for routine numbers . . . 21
8.3.2 Distributed data point search and interpolation . . . 22
8.3.2.1 Data Point Search . . . 22
8.3.2.2 Curve interpolation . . . 24
8.3.2.3 Curve look-up . . . 25
8.3.2.4 Map interpolation . . . 26
8.3.2.5 Map look-up . . . 28
8.3.2.6 Map look-up without rounding . . . 30
8.3.3 Integrated data point search and interpolation . . . 31
8.3.3.1 Integrated curve interpolation . . . 32

8.3.3.2 Integrated curve look-up . . . 34
8.3.3.3 Integrated fix-curve interpolation . . . 36
8.3.3.4 Integrated fix-curve look up . . . 38
8.3.3.5 Integrated fix- I curve interpolation . . . 41
8.3.3.6 Integrated fix- I curve look up . . . 42
8.3.3.7 Integrated map interpolation . . . 45
8.3.3.8 Integrated map look-up . . . 49
8.3.3.9 Integrated map look-up without rounding . . . 51
8.3.3.10 Integrated fix- map interpolation . . . 53
8.3.3.11 Integrated fix- map look up . . . 56
8.3.3.12 Integrated fix- map look up without rounding . . . 59
8.3.3.13 Integrated fix- I map interpolation . . . 61
8.3.3.14 Integrated fix- I map look up . . . 64
8.3.3.15 Integrated fix- I map look up without rounding . . . 67
8.3.3.16 Cuboid 3D interpolation . . . 69
8.3.4 Callback notifications . . . 70
8.4 Scheduled functions . . . 70
8.5 Expected Interfaces . . . 70
8.5.1 Mandatory Interfaces . . . 70
8.5.2 Optional Interfaces . . . 71
8.5.3 Callout interfaces . . . 71
8.5.4 Configurable interfaces . . . 71
9 Sequence diagrams 72
10 Configuration specification 73
10.1 How to read this chapter . . . 73
10.2 Containers and configuration parameters . . . 73
10.3 Published Information . . . 73
A Not applicable requirements 74
B Change history of AUTOSAR traceable items 75
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 75
B.1.1 Added Specification Items in R26-11 . . . 75
B.1.2 Changed Specification Items in R26-11 . . . 75
B.1.3 Deleted Specification Items in R26-11 . . . 75
B.1.4 Added Constraints in R26-11 . . . 76
B.1.5 Changed Constraints in R26-11 . . . 77
B.1.6 Deleted Constraints in R26-11 . . . 77
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 77
B.2.1 Added Specification Items in R25-11 . . . 77
B.2.2 Changed Specification Items in R25-11 . . . 77
B.2.3 Deleted Specification Items in R25-11 . . . 77

B.3 Traceable item history of this document according to AUTOSAR Release R24-11 77
B.3.1 Added Specification Items in R24-11 77
B.3.2 Changed Specification Items in R24-11 78
B.3.3 Deleted Specification Items in R24-11 78
B.4 Traceable item history of this document according to AUTOSAR Release R23-11 78
B.4.1 Added Specification Items in R23-11 78
B.4.2 Changed Specification Items in R23-11 78
B.4.3 Deleted Specification Items in R23-11 79

1 Introduction and functional overview

Ifx routines specification specifies the functionality and API of the AUTOSAR library dedicated to interpolation routines for fixed point values.

The interpolation library contains the following routines:

- Distributed data point search and interpolation
- Integrated data point search and interpolation

All routines are re-entrant.

2 Acronyms and Abbreviations

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

Abbreviation / Acronym:	Description:
Cur	Curve for Interpolation
DPSearch	Data point search
DPResult	Data point result
Ifx	Interpolation Fixed point
IpoCur	Interpolation of curve used for distributed search and interpolation
LkUpCur	Curve look-up used for distributed search and interpolation
IpoMap	Interpolation of map used for distributed search and interpolation
LkUpMap	Map look-up used for distributed search and interpolation
IntIpoCur	Integrated interpolation of curve
IntLkUpCur	Integrated curve look-up
IntIpoFixCur	Integrated interpolation of fixed curve
IntLkUpFixCur	Integrated fixed curve look-up
IntIpoFixICur	Integrated interpolation of fixed interval curve
IntLkUpFixICur	Integrated fixed interval curve look-up
IntIpoMap	Integrated interpolation of map
IntLkUpMap	Integrated map look-up
IntIpoFixMap	Integrated interpolation of fixed map
IntLkUpFixMap	Integrated fixed map look-up
IntIpoFixIMap	Integrated interpolation of fixed interval map
IntLkUpFixIMap	Integrated fixed interval map look-up
Lib	Library
Map	Map for Interpolation

Table 2.1: Acronyms and abbreviations used in the scope of this Document

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] ASAM MCD-2MC Version 1.6
http://www.asam.net

[4] IFX_RecordLayout_Blueprint
AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml

[5] Specification of Platform Types for Classic Platform
AUTOSAR_CP_SWS_PlatformTypes

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, 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

An implementation of the Ifx Library shall follow the naming rules for files outlined in [2, SWS BSW General].

6 Requirements Tracing

The following tables reference the requirements specified in [2, SWS BSW General] 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_00374]	All Basic Software Modules shall provide a readable module vendor identification	[SWS_Ifx_00814]
[SRS_BSW_00379]	All software modules shall provide a module identifier in the header file and in the module XML description file.	[SWS_Ifx_00814]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_Ifx_00814]
[SRS_BSW_00448]	Module SWS shall not contain requirements from other modules	[SWS_Ifx_00999]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Initialization and shutdown

Refer to [2, General Specification of Basic Software Modules] 7.1.20 *"Libraries"* for initialization and shutdown requirements.

7.2 Using Library API

Using a library should be documented. If a BSW module or a SW-C uses a library, the BSW/SW-C description should contain a corresponding DependencyOnArtifact element in the Bsw- or SwcImplementation.

7.3 Library implementation

[SWS_Ifx_00820] ⌈If the input value is less than the first distribution entry, then the first value of the distribution array shall be returned or used in the interpolation routines. If the input value is greater than the last distribution entry, then the last value of the distribution array shall be returned or used in the interpolation routines.⌋

[SWS_Ifx_CONSTR_04099] Monotony of axis distribution ⌈Axis distribution passed to Ifx routines shall have normal monotony sequence.⌋

[SWS_Ifx_00251] ⌈The intermediate results in interpolation calculation shall be rounded towards zero.⌋

7.4 Interpolation routines definitions

Interpolation between two given points is calculated as shown below.

where: X is the input value

x0 = data point before X

x1 = data point after X

y0 = value at x0

y1 = value at x1

Quantization error is by design and shall not be compensated in implementation.

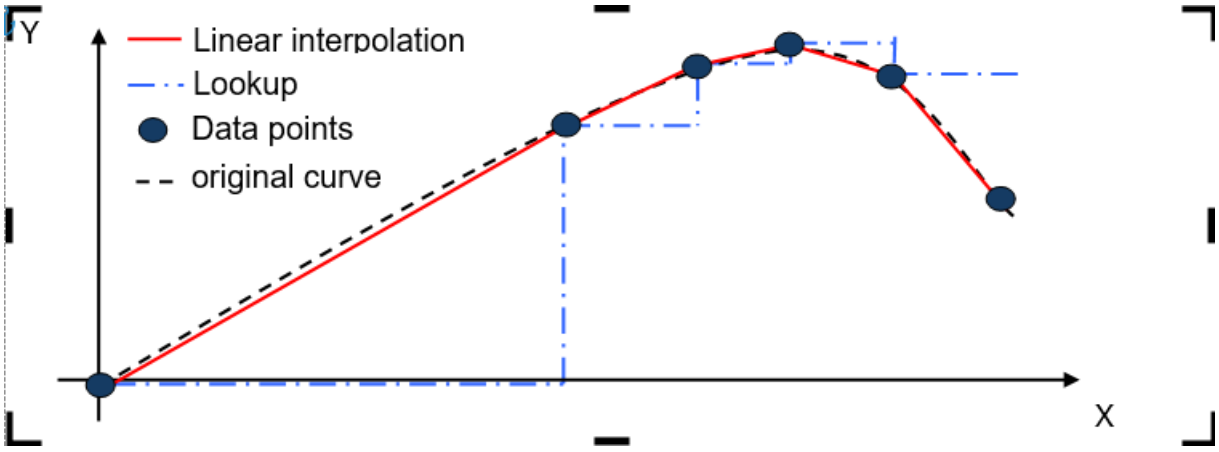


Figure 7.1: Linear and lookup interpolation

There are two interpolation methods.

- Linear interpolation
- Lookup interpolation

Above figure differentiates linear and lookup integration method. Linear method interpolates result considering two data points, whereas lookup interpolation returns entry data point.

Data points can be grouped in different ways. Examples:

```
number of samples = 5
X axis distribution = 0 to 64
Y axis distribution = 1 to 6
```

a) separate arrays for axis and curve

```
uint8 AxisCurveA_u8[5] = {0,10,26,36,64};
uint8 CurveA_u8[5] = {1,12,17,11,6};
```

b) one array containing all data (number of samples, axis, curve)

```
uint8 Curve_u8 []={5,0,10,26,36,64,1,12,17,11,6};
```

c) one structure with separate elements for size, axis, curve

```
struct { sint16 N = 5;
         uint8 X[5] ={0,10,26,36,64};
         uint8 Y[5] ={1,12,17,11,6};} Curve_u8;
```

Interpolation routines have separate arguments to support above scenarios. Routine call examples are given below for array and structure grouping respectively. Example:

```
Axis input value is 15
uint8 Ifx_IntIpoCur_u8_u8(15,5,&AxisCurveA_u8[0],&CurveA_u8[0]);
uint8 Ifx_IntIpoCur_u8_u8(15,(sint16)CurveB_u8[0],&CurveB_u8[1],&
    CurveB_u8[6]);
uint8 Ifx_IntIpoCur_u8_u8(15,CurveC_u8.N,&CurveC_u8.X[0],&CurveC_u8.Y
    [0]);
```

Interpolation can be calculated in two ways as shown below:

1. Distributed data point search and interpolation
2. Integrated data point search and interpolation

7.4.1 Record layouts for interpolation routines

Record layout specifies calibration data serialization in the ECU memory which describes the shape of the characteristics. Single record layout can be referred by multiple instances of interpolation ParameterDataPrototype. Record layouts can be nested particular values refer to the particular property of the object. With different properties of record layouts it is possible to specify complex objects.

7.4.1.1 Record layouts for map values

Due to optimization, the orientation of map values in memory is different depending on the usage of the inputs. See section 8.3.3.

1. If the "X" and "Y" inputs are not swapped then, values "Val" of maps have to be in COLUMN_DIR order.

2. If the "X" and "Y" inputs are swapped then, values "Val" of maps have to be in ROW_DIR order.

COLUMN_DIR and ROW_DIR are formats of storing map values (Val[]). More information can be found in ASAM standard [3].

The "Z" input of cuboids is the third dimension and selects the slice X / Y or Y / X - 2D maps.

Example for cuboids order:

2x2x2 cuboid representation in memory shall be COLUMN_DIR according to the ASAM standard : [1 2 3 4 5 6 7 8]

COLUMN_DIR order [z][x][y]:

Slice 1:

[1 2

3 4]

Slice 2:

[5 6

7 8]

7.4.1.2 Record layout definitions

Below table specifies record layouts supported by Ifx routines.

[SWS_Ifx_00185] ⌈

Record layout Name	Element1	Element2	Element3	Element4	Element5
Distr	<InTypeMn> N	<InTypeMn> X[]			
Cur	<OutTypeMn> Val[]				
FixIntCur	<OutTypeMn> Val[]				
IntCur	<InTypeMn> N	<InTypeMn> X[]	<OutTypeMn> Val[]		
Map	<OutTypeMn> Val[][]				
FixIntMap	<OutTypeMn> Val[][]				
IntMap	<InTypeMn> Nx	<InTypeMn> Ny	<InTypeMn> X[]	<InTypeMn> Y[]	<Out-TypeMn>Val[][]
Cub	<OutTypeMn> Val[][][]				

⌋

Remark:

The SwRecordLayouts and the InterpolationRoutineMappings to them are defined in [4]

7.5 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

For the error handling of libraries refer to [SWS_BSW_00262].

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

7.5.1 Development Errors

There are no development errors.

7.5.2 Runtime Errors

There are no runtime errors.

7.5.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

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

[SWS_Ifx_91001] Definition of imported datatypes of module Ifx ⌈

Module	*Header File*	*Imported Type*
There are no imported types.		

⌋

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 [5](AUTOSAR CP SWS PlatformTypes). The following mnemonic are used in the library routine names.

Size	Platform Type	Mnemonic
unsigned 8-Bit	boolean	NA
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: Mnemonic for Base Types

As a convention in the rest of the document:

- mnemonics will be used in the name of the routines (using <InTypeMn> means type mnemonic for an Input)
- the real type will be used in the description of the prototypes of the routines (using <InType> or <OutType>).

8.2 Type definitions

Structure definition :

[SWS_Ifx_00002] Definition of datatype Ifx_DPResultU16_Type ⌈

Name	Ifx_DPResultU16_Type	
Kind	Structure	
Elements	Index	
	Type	uint16
	Comment	Data point index
	Ratio	
	Type	uint16
	Comment	Data point ratio
Description	Structure used for data point search for index and ratio	
Available via	Ifx.h	

⌋

[SWS_Ifx_00003] ⌈Ratio shall have resolution of 2^{-16}⌋

[SWS_Ifx_00248] ⌈Ratio shall be rounded towards zero⌋

[SWS_Ifx_CONSTR_04100] Access to data point search result structure ⌈Ifx_DPResultU16_Type structure shall not be read/write/modified by the user directly. Only Ifx routines shall have access to this structure.⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Comment about routines optimization

8.3.1.1 Optimization for routine numbers

Many routines can be omitted by exchanging 'X' and 'Y' data types. With this method, reduction in total number of routines is possible in case of Map interpolation routines. This optimization of routine numbers is done based on below mentioned rules.

- Rule 1: Bigger data type of 'X' and 'Y' comes first (16 bit before 8 bit)
- Rule 2: unsigned before signed (u16 before s16)
- Order: u32, s32, u16, s16, u8, s8

In this case, below routine can be replaced as :

Ifx_IntIpoMap_s8u16_u16

With

Ifx_IntIpoMap_u16s8_u16

Note: swapped inputs need another map value order in memory, see record layout section.

8.3.2 Distributed data point search and interpolation

In this interpolation method data point search (e.g. index and ratio) is calculated using routine Ifx_DPSearch_<InTypeMn> which returns result structure Ifx_DPResultU16_Type. It contains index and ratio information. This result can be used by curve interpolation, curve look-up interpolation, map interpolation and map look-up interpolation.

8.3.2.1 Data Point Search

[SWS_Ifx_00004] Definition of API function Ifx_DPSearch_<InTypeMn> ⌈

Service Name	Ifx_DPSearch_<InTypeMn>	

Syntax	`void Ifx_DPSearch_<InTypeMn> (`
	`  Ifx_DPResultU16_Type* dpResult,`
	`  <InType> Xin,`
	`  <InType> N,`
	`  const <InType>* X_array`
	`)`	

Service ID [hex]	0x0001 to 0x0004; 0x00C1, 0x00C2	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples
	X_array	Pointer to the X axis distribution array
Parameters (inout)	None	
Parameters (out)	dpResult	Pointer to the result structure
Return value	None	
Description	Ifx_DPSearch_<InTypeMn> routine searches the position of input Xin within the given distribution array X_array, and returns index and ratio necessary for interpolation.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00006] ⌈If (X_array[0] <= Xin <= X_array[N-1]), then returned Index shall be the lowest index.

dpResult ->Index = index- dpResult ->Ratio = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])⌋

[SWS_Ifx_00008] ⌈If the input value matches with one of the distribution array values, then return the respective index and ratio = 0.

If (Xin == X_array[index]), then

dpResult ->Index = index

dpResult ->Ratio = 0⌋

[SWS_Ifx_00009] ⌈If (Xin < X_array[0]), then return first index of an array and ratio = 0

dpResult ->Index = 0

dpResult ->Ratio = 0⌋

[SWS_Ifx_00010] ⌈If (Xin > X_array[N-1]), then return last index of an array and ratio = 0

dpResult ->Index = N - 1

dpResult ->Ratio = 0⌋

[SWS_Ifx_00013] ⌈This routine returns index and ratio through the structure of type Ifx_DPResultU16_Type⌋

[SWS_Ifx_00014] ⌈Here is the list of implemented routines.⌋

Service ID[hex]	Service prototype
0x001	void Ifx_DPSearch_u8 (Ifx_DPResultU16_Type*, uint8, uint8, const uint8 *)
0x002	void Ifx_DPSearch_s8 (Ifx_DPResultU16_Type*, sint8, sint8, const sint8 *)
0x003	void Ifx_DPSearch_u16 (Ifx_DPResultU16_Type*, uint16, uint16, const uint16 *)
0x004	void Ifx_DPSearch_s16 (Ifx_DPResultU16_Type*, sint16, sint16, const sint16 *)
0x0C1	void Ifx_DPSearch_u32 (Ifx_DPResultU16_Type*, uint32, uint32, const uint32 *)
0x0C2	void Ifx_DPSearch_s32 (Ifx_DPResultU16_Type*, sint32, sint32, const sint32 *)

Table 8.2: Implemented routines for Data Point Search

8.3.2.2 Curve interpolation

[SWS_Ifx_00015] Definition of API function Ifx_IpoCur_<OutTypeMn> ⌈

Service Name	Ifx_IpoCur_<OutTypeMn>	

Syntax	`<OutType> Ifx_IpoCur_<OutTypeMn> (`
	`  const Ifx_DPResultU16_Type* dpResult,`
	`  const <InType>* Val_array`
	`)`	

Service ID [hex]	0x0005 to 0x0008; 0x00C3, 0x00C4	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResult	Data point search result
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Interpolation
Description	Based on searched index and ratio information, this routine returns the interpolation result for a curve.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00016] ⌈index = dpResult->Index

if dPResult->Ratio == 0

Result = Val_array[index]

else

Result = Val_array[index] + (Val_array[index+1] - Val_array[index]) * dpResult->Ratio

Note:

In case of missing HW support the Software solution mentioned below could also be used to avoid 64-bit arithmetic operation.

if (Val_array[index] <= Val_array[index+1]) then

Result = Val_array[index] + (Val_array[index+1] - Val_array[index]) * dpResult->Ratio

if (Val_array[index] > Val_array[index+1]) then

Result = Val_array[index] - (Val_array[index] - Val_array[index+1]) * dpResult->Ratio⌋

Note: Do not call this routine until you have searched the axis using the Ifx_DPSearch routine. Only then it is ensured that the search result (Ifx_DPResultU16_Type) contains valid data and is not used uninitialized.

[SWS_Ifx_00017] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x005	sint8 Ifx_IpoCur_s8 (const Ifx_DPResultU16_Type*, const sint8 *)
0x006	sint16 Ifx_IpoCur_s16 (const Ifx_DPResultU16_Type*, const sint16 *)
0x007	uint16 Ifx_IpoCur_u16 (const Ifx_DPResultU16_Type*, const uint16 *)
0x008	uint8 Ifx_IpoCur_u8 (const Ifx_DPResultU16_Type*, const uint8 *)
0x0C3	uint32 Ifx_IpoCur_u32 (const Ifx_DPResultU16_Type*, const uint32*)
0x0C4	sint32 Ifx_IpoCur_s32 (const Ifx_DPResultU16_Type*, const sint32*)

Table 8.3: Implemented routines for Curve interpolation

8.3.2.3 Curve look-up

[SWS_Ifx_00020] Definition of API function Ifx_LkUpCur_<OutTypeMn> ⌈

Service Name	Ifx_LkUpCur_<OutTypeMn>	
Syntax	`<OutType> Ifx_LkUpCur_<OutTypeMn> ( const Ifx_DPResultU16_Type* dpResult, const <InType>* Val_array )`	
Service ID [hex]	0x000A to 0x000D; 0x00C5, 0x00C6	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResult	Data point search result
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	Based on searched index, this routine returns the corresponding value of the curve.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00021] ⌈Result = Val_array[dpResult->Index]⌋

Note: Do not call this routine until you have searched the axis using the Ifx_DPSearch routine. Only then it is ensured that the search result (Ifx_DPResultU16_Type) contains valid data and is not used uninitialized.

[SWS_Ifx_00022] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x00A	sint8 Ifx_LkUpCur_s8 (const Ifx_DPResultU16_Type*, const sint8 *)
0x00B	sint16 Ifx_LkUpCur_s16 (const Ifx_DPResultU16_Type*, const sint16 *)
0x00C	uint16 Ifx_LkUpCur_u16 (const Ifx_DPResultU16_Type*, const uint16 *)
0x00D	uint8 Ifx_LkUpCur_u8 (const Ifx_DPResultU16_Type*, const uint8 *)
0x0C5	sint32 Ifx_LkUpCur_s32 (const Ifx_DPResultU16_Type*, const sint32 *)
0x0C6	uint32 Ifx_LkUpCur_u32 (const Ifx_DPResultU16_Type*, const uint32 *)

Table 8.4: Implemented routines for Curve look-up

8.3.2.4 Map interpolation

[SWS_Ifx_00025] Definition of API function Ifx_IpoMap_<OutTypeMn> ⌈

Service Name	Ifx_IpoMap_<OutTypeMn>	
Syntax	`<OutType> Ifx_IpoMap_<OutTypeMn> ( const Ifx_DPResultU16_Type* dpResultX, const Ifx_DPResultU16_Type* dpResultY, uint16 num_value, const <InType>* Val_array )`	
Service ID [hex]	0x0010 to 0x0013; 0x00C7, 0x00C8	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResultX	Data point search result for x axis
	dpResultY	Data point search result for y axis
	num_value	Number of y axis points
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Interpolation
Description	Based on searched indices and ratios information using the relevant Ifx_DPSearch routine, this routine calculates and returns the interpolation result for a map.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00026] ⌈Based on searched indices and ratios information using the relevant Ifx_DPSearch routine, this routine calculates and returns the interpolation result for a map.

BaseIndex = dpResultX->Index * num_value + dpResultY->Index

if (dpResultX->Ratio == 0)

if (dpResultY->Ratio == 0)

Result = Val_array [BaseIndex]

else

LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

Result = LowerY + (UpperY - LowerY) * dpResultY->Ratio

else

if (dpResultY->Ratio == 0)

LowerX = Val_array[BaseIndex]

UpperX = Val_array[BaseIndex + num_value]

Result = LowerX + (UpperX - LowerX) * dpResultX->Ratio

else

LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

LowerX = LowerY + (UpperY - LowerY) * dpResultY->Ratio

LowerY = Val_array[BaseIndex + num_value]

UpperY = Val_array[BaseIndex + num_value + 1]

UpperX = LowerY + (UpperY - LowerY) * dpResultY->Ratio

Result = LowerX + (UpperX - LowerX) * dpResultX->Ratio⌋

Note: Do not call this routine until you have searched the axis using the Ifx_DPSearch routine. Only then it is ensured that the search result (Ifx_DPResultU16_Type) contains valid data and is not used uninitialized.

[SWS_Ifx_00027] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x010	uint8 Ifx_IpoMap_u8 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint8 *)

▽

△

Routine ID[hex]	Routine prototype
0x011	uint16 Ifx_IpoMap_u16 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint16 *)
0x012	sint8 Ifx_IpoMap_s8 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint8 *)
0x013	sint16 Ifx_IpoMap_s16 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint16 *)
0x0C7	sint32 Ifx_IpoMap_s32 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint32 *)
0x0C8	uint32 Ifx_IpoMap_u32 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint32 *)

Table 8.5: Implemented routines for Map interpolation

8.3.2.5 Map look-up

[SWS_Ifx_00030] Definition of API function Ifx_LkUpMap_<OutTypeMn> ⌈

Service Name	Ifx_LkUpMap_<OutTypeMn>	
Syntax	<OutType> Ifx_LkUpMap_<OutTypeMn> (const Ifx_DPResultU16_Type* dpResultX, const Ifx_DPResultU16_Type* dpResultY, uint16 num_value, const <InType>* Val_array)	
Service ID [hex]	0x0015 to 0x0018; 0x00C9, 0x00CA	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResultX	Data point search result for x axis
	dpResultY	Data point search result for y axis
	num_value	Number of y axis points
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array

▽

△

Description	Based on searched index and ratio information, this routine calculates and returns the corresponding value of the map with rounding.
Available via	Ifx.h

⌋

[SWS_Ifx_00031] ⌈BaseIndex = dpResultX->Index * num_value + dpResultY->Index⌋

[SWS_Ifx_00033] ⌈if(dpResultX->Ratio < 0.5 && dpResultY->Ratio < 0.5) then

return Val_array [BaseIndex]

if(dpResultX->Ratio ≥ 0.5 && dpResultY->Ratio < 0.5) then

return Val_array [BaseIndex + num_value]

if(dpResultX->Ratio < 0.5 && dpResultY->Ratio ≥ 0.5) then

return Val_array [BaseIndex + 1]

if(dpResultX->Ratio ≥ 0.5 && dpResultY->Ratio ≥ 0.5) then

return Val_array [BaseIndex + num_value + 1]⌋

Note: Do not call this routine until you have searched the axis to ensure the search result contains valid data and is not used uninitialized.

[SWS_Ifx_00032] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x015	uint8 Ifx_LkUpMap_u8 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint8 *)
0x016	uint16 Ifx_LkUpMap_u16 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint16 *)
0x017	sint8 Ifx_LkUpMap_s8 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint8 *)
0x018	sint16 Ifx_LkUpMap_s16 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint16 *)

▽

△

Routine ID[hex]	Routine prototype
0x0C9	sint32 Ifx_LkUpMap_s32 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint32*)
0x0CA	uint32 Ifx_LkUpMap_u32 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint32*)

Table 8.6: Implemented routines for Map look-up

8.3.2.6 Map look-up without rounding

[SWS_Ifx_00205] Definition of API function Ifx_LkUpBaseMap_<OutTypeMn> ⌈

Service Name	Ifx_LkUpBaseMap_<OutTypeMn>	

Syntax	`<OutType> Ifx_LkUpBaseMap_<OutTypeMn> (`
	`  const Ifx_DPResultU16_Type* dpResultX,`
	`  const Ifx_DPResultU16_Type* dpResultY,`
	`  uint16 num_value,`
	`  const <InType>* Val_array`
	`)`	

Service ID [hex]	0x00A5 to 0x00A8; 0x00CB, 0x00CC	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResultX	Data point search result for x axis
	dpResultY	Data point search result for y axis
	num_value	Number of y axis points
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	Based on searched index, this routine returns the corresponding value of the map.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00206] ⌈BaseIndex = dpResultX->Index * num_value + dpResultY->Index⌋

[SWS_Ifx_00207] ⌈Return Value = Val_array [BaseIndex]⌋

Note: Do not call this routine until you have searched the axis using the Ifx_DPSearch routine. Only then it is ensured that the search result (Ifx_DPResultU16_Type) contains valid data and is not used uninitialized.

[SWS_Ifx_00209] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x0A5	uint8 Ifx_LkUpBaseMap_u8 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint8 *)
0x0A6	uint16 Ifx_LkUpBaseMap_u16 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint16 *)
0x0A7	sint8 Ifx_LkUpBaseMap_s8 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint8 *)
0x0A8	sint16 Ifx_LkUpBaseMap_s16 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint16 *)
0x0CB	sint32 Ifx_LkUpBaseMap_s32 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const sint32*)
0x0CC	uint32 Ifx_LkUpBaseMap_u32 (const Ifx_DPResultU16_Type*, const Ifx_DPResultU16_Type*, uint16, const uint32*)

Table 8.7: Implemented routines for Map look-up without rounding

8.3.3 Integrated data point search and interpolation

Integrated routines perform both, data point search (e.g. index and ratio) and interpolation for a curve, map, cuboid or a look-up in a table.

8.3.3.1 Integrated curve interpolation

[SWS_Ifx_00035] Definition of API function Ifx_IntIpoCur_<InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntIpoCur_<InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntIpoCur_<InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> N, const <InType>* X_array, const <InType>* Val_array )`	
Service ID [hex]	0x001A to 0x0029; 0x00CD, 0x00CE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples
	X_array	Pointer to the X axis distribution array
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Interpolation
Description	This routine returns the interpolation result of a curve at position Xin.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00036] ⌈If (X_array[0] < Xin < X_array[N -1]), then

index = lowest index for which (Xin < X_array[index + 1]).

RatioX = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])

Result = Val_array[index] + (Val_array[index+1] - Val_array[index])*RatioX⌋

[SWS_Ifx_00037] ⌈Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.

If (Xin == X_array[index]) then,

Result = Val_array[index]⌋

[SWS_Ifx_00038] ⌈If (Xin < X_array[0]) then,

Result = Val_array[0]⌋

[SWS_Ifx_00039] ⌈If (Xin > X_array[N-1]) then,

Result = Val_array[N-1]⌋

[SWS_Ifx_00041] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x01A	uint8 Ifx_IntIpoCur_u8_u8 (uint8, uint8, const uint8 *, const uint8 *)
0x01B	uint16 Ifx_IntIpoCur_u8_u16 (uint8, uint8, const uint8 *, const uint16 *)
0x01C	sint8 Ifx_IntIpoCur_u8_s8 (uint8, uint8, const uint8 *, const sint8 *)
0x01D	sint16 Ifx_IntIpoCur_u8_s16 (uint8, uint8, const uint8 *, const sint16 *)
0x01E	uint8 Ifx_IntIpoCur_u16_u8 (uint16, uint16, const uint16 *, const uint8 *)
0x01F	uint16 Ifx_IntIpoCur_u16_u16 (uint16, uint16, const uint16 *, const uint16 *)
0x020	sint8 Ifx_IntIpoCur_u16_s8 (uint16, uint16, const uint16 *, const sint8 *)
0x021	sint16 Ifx_IntIpoCur_u16_s16 (uint16, uint16, const uint16 *, const sint16 *)
0x022	uint8 Ifx_IntIpoCur_s8_u8 (sint8, sint8, const sint8 *, const uint8 *)
0x023	uint16 Ifx_IntIpoCur_s8_u16 (sint8, sint8, const sint8 *, const uint16 *)
0x024	sint8 Ifx_IntIpoCur_s8_s8 (sint8, sint8, const sint8 *, const sint8 *)
0x025	sint16 Ifx_IntIpoCur_s8_s16 (sint8, sint8, const sint8 *, const sint16 *)
0x026	uint8 Ifx_IntIpoCur_s16_u8 (sint16, sint16, const sint16 *, const uint8 *)
0x027	uint16 Ifx_IntIpoCur_s16_u16 (sint16, sint16, const sint16 *, const uint16 *)
0x028	sint8 Ifx_IntIpoCur_s16_s8 (sint16, sint16, const sint16 *, const sint8 *)
0x029	sint16 Ifx_IntIpoCur_s16_s16 (sint16, sint16, const sint16 *, const sint16 *)
0x0CD	sint32 Ifx_IntIpoCur_s32_s32 (sint32, sint32, const sint32*, const sint32*)
0x0CE	uint32 Ifx_IntIpoCur_u32_u32 (uint32, uint32, const uint32*, const uint32*)

Table 8.8: Implemented routines for Integrated curve interpolation

8.3.3.2 Integrated curve look-up

[SWS_Ifx_00045] Definition of API function Ifx_IntLkUpCur_<InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpCur_<InTypeMn>_<OutTypeMn>	

Syntax	`<OutType> Ifx_IntLkUpCur_<InTypeMn>_<OutTypeMn> (`
	`  <InType> Xin,`
	`  <InType> N,`
	`  const <InType>* X_array,`
	`  const <InType>* Val_array`
	`)`	

Service ID [hex]	0x0030 to 0x003F; 0x00CF, 0x00D0	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples
	X_array	Pointer to the X axis distribution array
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	This routine returns the value of the curve at position Xin.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00046] ⌈If (X_array[0] < Xin < X_array[N -1]), then

index = lowest index for which (Xin < X_array[index + 1]).

Result = Val_array[index]⌋

[SWS_Ifx_00047] ⌈Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.

If (Xin == X_array[index]) then,

Result = Val_array[index]⌋

[SWS_Ifx_00048] ⌈If (Xin < X_array[0]) then,

Result = Val_array[0]⌋

[SWS_Ifx_00049] ⌈If (Xin > X_array[N-1]) then,

Result = Val_array[N-1]⌋

[SWS_Ifx_00051] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x030	uint8 Ifx_IntLkUpCur_u8_u8 (uint8 , uint8, const uint8 *, const uint8 *)
0x031	uint16 Ifx_IntLkUpCur_u8_u16 (uint8 , uint8, const uint8 *, const uint16 *)
0x032	sint8 Ifx_IntLkUpCur_u8_s8 (uint8 , uint8, const uint8 *, const sint8 *)
0x033	sint16 Ifx_IntLkUpCur_u8_s16 (uint8 , uint8, const uint8 *, const sint16 *)
0x034	uint8 Ifx_IntLkUpCur_u16_u8 (uint16 , uint16, const uint16 *, const uint8 *)
0x035	uint16 Ifx_IntLkUpCur_u16_u16 (uint16 , uint16, const uint16 *, const uint16 *)
0x036	sint8 Ifx_IntLkUpCur_u16_s8 (uint16 , uint16, const uint16 *, const sint8 *)
0x037	sint16 Ifx_IntLkUpCur_u16_s16 (uint16 , uint16, const uint16 *, const sint16 *)
0x038	uint8 Ifx_IntLkUpCur_s8_u8 (sint8 , sint8, const sint8 *, const uint8 *)
0x039	uint16 Ifx_IntLkUpCur_s8_u16 (sint8 , sint8, const sint8 *, const uint16 *)
0x03A	sint8 Ifx_IntLkUpCur_s8_s8 (sint8, sint8, const sint8 *, const sint8 *)
0x03B	sint16 Ifx_IntLkUpCur_s8_s16 (sint8, sint8, const sint8 *, const sint16 *)
0x03C	uint8 Ifx_IntLkUpCur_s16_u8 (sint16, sint16, const sint16 *, const uint8 *)
0x03D	uint16 Ifx_IntLkUpCur_s16_u16 (sint16, sint16, const sint16 *, const uint16 *)
0x03E	sint8 Ifx_IntLkUpCur_s16_s8 (sint16, sint16, const sint16 *, const sint8 *)
0x03F	sint16 Ifx_IntLkUpCur_s16_s16 (sint16, sint16, const sint16 *, const sint16 *)
0x0CF	sint32 Ifx_IntLkUpCur_s32_s32 (sint32, sint32, const sint32*, const sint32*)
0x0D0	uint32 Ifx_IntLkUpCur_u32_u32 (uint32, uint32, const uint32* , const uint32*)

Table 8.9: Implemented routines for Integrated curve look-up

8.3.3.3 Integrated fix-curve interpolation

[SWS_Ifx_00055] Definition of API function Ifx_IntIpoFixCur_<InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntIpoFixCur_<InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntIpoFixCur_<InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> N, const <InType>* Val_array, <InType> Offset, <InType> Shift )`	
Service ID [hex]	0x0040 to 0x0043; 0x00D1, 0x00D2	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples
	Val_array	Pointer to the result axis distribution array
	Offset	Offset of the first sampling value for X-axis
	Shift	'Shift' is the power of 2, (2^Shift) represents X-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Interpolation
Description	This routine returns the interpolation result of a curve at position Xin based on a fix axis distribution with offset and shift.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00056] ⌈X axis distribution points shall be calculated based on Offset and Shift values.⌋

X_array [index] = Offset + index * 2^{Shift}

Example:

If Offset = 10, Shift = 2 and N = 5 then,

X_array[5] = {10, 14, 18, 22, 26}

[SWS_Ifx_00057] ⌈If (X_array[0] < Xin < X_array[N -1]), then

index = lowest index for which (Xin < X_array[index + 1]).

RatioX = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])

Result = Val_array[index] + (Val_array[index+1] - Val_array[index]) * RatioX⌋

[SWS_Ifx_00058] ⌈Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.

If (Xin == X_array[index])

Result = Val_array[index]⌋

[SWS_Ifx_00059] ⌈If (Xin < X_array[0]) then,

Result = Val_array[0]⌋

[SWS_Ifx_00060] ⌈If (Xin > X_array[N-1]) then,

Result = Val_array[N-1]⌋

[SWS_Ifx_00062] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x040	uint8 Ifx_IntIpoFixCur_u8_u8 (uint8, uint8, const uint8 *, uint8, uint8)
0x041	uint16 Ifx_IntIpoFixCur_u16_u16 (uint16, uint16, const uint16 *, uint16, uint16)
0x042	sint8 Ifx_IntIpoFixCur_s8_s8 (sint8, sint8, const sint8 *, sint8, sint8)
0x043	sint16 Ifx_IntIpoFixCur_s16_s16 (sint16, sint16, const sint16 *, sint16, sint16)
0x0D1	sint32 Ifx_IntIpoFixCur_s32_s32 (sint32, sint32, const sint32 *, sint32, sint32)
0x0D2	uint32 Ifx_IntIpoFixCur_u32_u32 (uint32, uint32, const uint32 *, uint32, uint32)

Table 8.10: Implemented routines for Integrated fix-curve interpolation

8.3.3.4 Integrated fix-curve look up

[SWS_Ifx_00070] Definition of API function Ifx_IntLkUpFixCur_<InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpFixCur_<InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpFixCur_<InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> N, const <InType>* Val_array, <InType> Offset, <InType> Shift )`	
Service ID [hex]	0x0045 to 0x0048; 0x00D3, 0x00D4	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples
	Val_array	Pointer to the result axis distribution array
	Offset	Offset of the first sampling value for X-axis
	Shift	'Shift' is the power of 2, (2^Shift) represents X-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	<OutType>	Entry point of the result array
Description	This routine returns the value of a curve at position Xin based on a fix axis distribution with offset and shift.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00071] ⌈X axis distribution points shall be calculated based on Offset and Shift values.⌋

X_array [index] = Offset + index * 2^{Shift}

Example:

If Offset = 10, Shift = 2 and N = 5 then,

X_array[5] = {10, 14, 18, 22, 26}

[SWS_Ifx_00072] ⌈If (X_array[0] < Xin < X_array[N -1]), then

index = lowest index for which (Xin < X_array[index + 1]).

Result = Val_array[index]⌋

[SWS_Ifx_00073] ⌈Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.

If (Xin == X_array[index]) then,

Result = Val_array[index]⌋

[SWS_Ifx_00074] ⌈If (Xin < X_array[0]) then,

Result = Val_array[0]⌋

[SWS_Ifx_00075] ⌈If (Xin > X_array[N-1]) then,

Result = Val_array[N-1]⌋

[SWS_Ifx_00077] ⌈Here is the list of implemented routines⌋

Routine ID[hex]	Routine prototype
0x045	uint8 Ifx_IntLkUpFixCur_u8_u8 (uint8, uint8, const uint8 *, uint8, uint8)
0x046	uint16 Ifx_IntLkUpFixCur_u16_u16 (uint16, uint16, const uint16 *, uint16, uint16)
0x047	sint8 Ifx_IntLkUpFixCur_s8_s8 (sint8, sint8, const sint8 *, sint8, sint8)
0x048	sint16 Ifx_IntLkUpFixCur_s16_s16 (sint16, sint16, const sint16 *, sint16, sint16)

▽

△

Routine ID[hex]	Routine prototype
0x0D3	sint32 Ifx_IntLkUpFixCur_s32_s32 (sint32, sint32, const sint32*, sint32, sint32)
0x0D4	uint32 Ifx_IntLkUpFixCur_u32_u32 (uint32, uint32, const uint32*, uint32, uint32)

Table 8.11: Implemented routines for Integrated fix-curve look up

8.3.3.5 Integrated fix- I curve interpolation

[SWS_Ifx_00080] Definition of API function Ifx_IntIpoFixICur_<InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntIpoFixICur_<InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntIpoFixICur_<InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> N, const <InType>* Val_array, <InType> Offset, <InType> Interval )`	
Service ID [hex]	0x004A to 0x004D; 0x00D5, 0x00D6	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples
	Val_array	Pointer to the result axis distribution array
	Offset	Offset of the first sampling value for X-axis
	Interval	represents X-axis distribution point fix interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Interpolation
Description	This routine returns the interpolation result of a curve at position Xin based on a fix axis distribution with offset and interval.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00081] ⌈X axis distribution points shall be calculated based on Offset and Interval values.

X_array [index] = offset + index * Interval⌋

Example:

If Offset = 5, Interval = 12 and N = 5 then,

X_array[5] = {5, 17, 29, 41, 53}

[SWS_Ifx_00082] ⌈If (X_array[0] < Xin < X_array[N -1]), then

index = lowest index for which (Xin < X_array[index + 1]).

RatioX = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])

Result = Val_array[index] + (Val_array[index+1] - Val_array[index]) * RatioX⌋

[SWS_Ifx_00083] ⌈Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.

If (Xin == X_array[index])

Result = Val_array[index]⌋

[SWS_Ifx_00084] ⌈If (Xin < X_array[0]) then,

Result = Val_array[0]⌋

[SWS_Ifx_00085] ⌈If (Xin > X_array[N-1]) then,

Result = Val_array[N-1]⌋

[SWS_Ifx_00087] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x04A	uint8 Ifx_IntIpoFixICur_u8_u8 (uint8, uint8, const uint8 *, uint8, uint8)
0x04B	uint16 Ifx_IntIpoFixICur_u16_u16 (uint16, uint16, const uint16 *, uint16, uint16)
0x04C	sint8 Ifx_IntIpoFixICur_s8_s8 (sint8, sint8, const sint8 *, sint8, sint8)
0x04D	sint16 Ifx_IntIpoFixICur_s16_s16 (sint16, sint16, const sint16 *, sint16, sint16)
0x0D5	sint32 Ifx_IntIpoFixICur_s32_s32 (sint32, sint32, const sint32 *, sint32, sint32)
0x0D6	uint32 Ifx_IntIpoFixICur_u32_u32 (uint32, uint32, const uint32 *, uint32, uint32)

Table 8.12: Implemented routines for Integrated fix- I curve interpolation

8.3.3.6 Integrated fix- I curve look up

[SWS_Ifx_00090] Definition of API function Ifx_IntLkUpFixICur_<InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpFixICur_<InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpFixICur_<InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> N, const <InType>* Val_array, <InType> Offset, <InType> Interval )`	
Service ID [hex]	0x0050 to 0x0053	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples
	Val_array	Pointer to the result axis distribution array
	Offset	Offset of the first sampling value for X-axis
	Interval	represents X-axis distribution point fix interval
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	<OutType>	Entry point of the result array
Description	This routine returns the value of a curve at position Xin based on a fix axis distribution with offset and interval.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00091] ⌈X axis distribution points shall be calculated based on Offset and Interval values.

X_array [index] = offset + index * Interval⌋

Example:

If Offset = 5, Interval = 12 and N = 5 then,

X_array[5] = {5, 17, 29, 41, 53}

[SWS_Ifx_00092] ⌈If (X_array[0] < Xin < X_array[N -1]), then

index = lowest index for which (Xin < X_array[index + 1]).

Result = Val_array[index]⌋

[SWS_Ifx_00093] ⌈Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.

If (Xin == X_array[index])

Result = Val_array[index]⌋

[SWS_Ifx_00094] ⌈If (Xin < X_array[0]) then,

Result = Val_array[0]⌋

[SWS_Ifx_00095] ⌈If (Xin > X_array[N-1]) then,

Result = Val_array[N-1]⌋

[SWS_Ifx_00097] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x050	uint8 Ifx_IntLkUpFixICur_u8_u8 (uint8, uint8, const uint8 *, uint8, uint8)
0x051	uint16 Ifx_IntLkUpFixICur_u16_u16 (uint16, uint16, const uint16 *, uint16, uint16)
0x052	sint8 Ifx_IntLkUpFixICur_s8_s8 (sint8, sint8, const sint8 *, sint8, sint8)
0x053	sint16 Ifx_IntLkUpFixICur_s16_s16 (sint16, sint16, const sint16 *, sint16, sint16)

▽

△

Routine ID[hex]	Routine prototype
0x0D7	sint32 Ifx_IntLkUpFixICur_s32_s32 (sint32, sint32, const sint32 *, sint32, sint32)
0x0D8	uint32 Ifx_IntLkUpFixICur_u32_u32 (uint32, uint32, const uint32 *, uint32, uint32)

Table 8.13: Implemented routines for Integrated fix- I curve look up

8.3.3.7 Integrated map interpolation

[SWS_Ifx_00098] Definition of API function Ifx_IntIpoMap_<InTypeMn><InType Mn>_<OutTypeMn> ⌈

Service Name	Ifx_IntIpoMap_<InTypeMn><InTypeMn>_<OutTypeMn>	

Syntax	`<OutType> Ifx_IntIpoMap_<InTypeMn><InTypeMn>_<OutTypeMn> (`
	`  <InType> Xin,`
	`  <InType> Yin,`
	`  <InType> Nx,`
	`  <InType> Ny,`
	`  const <InType>* X_array,`
	`  const <InType>* Y_array,`
	`  const <InType>* Val_array`
	`)`	

Service ID [hex]	0x0060 to 0x0087	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number of X axis samples
	Ny	Number of Y axis samples
	X_array	Pointer to the X axis distribution array
	Y_array	Pointer to the Y axis distribution array
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Map Interpolation
Description	This routine returns the interpolation result of a map at position Xin, Yin.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00099] ⌈Index calculation :

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00100] ⌈Ratio calculation :

RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])

RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])⌋

[SWS_Ifx_00101] ⌈LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

LowerX = LowerY + (UpperY - LowerY) * RatioY

LowerY = Val_array [BaseIndex + Ny]

UpperY = Val_array [BaseIndex + Ny + 1]

UpperX = LowerY + (UpperY - LowerY) * RatioY

Result = LowerX + (UpperX - LowerX) * RatioX⌋

[SWS_Ifx_00102] ⌈If (Xin == X_array[indexX]) and (Y_array[indexY] < Yin < Y_array[indexY+1])

Result = Val_array [BaseIndex] + (Val_array [BaseIndex+1] - Val_array[BaseIndex]) * RatioY⌋

[SWS_Ifx_00103] ⌈If (Yin == Y_array[indexY]) and (X_array[indexX] < Xin < X_array[indexX+1])

Result = Val_array [BaseIndex] + (Val_array [BaseIndex+Ny] - Val_array[BaseIndex]) * RatioX⌋

[SWS_Ifx_00104] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00105] ⌈If Xin < X_array[0], then

indexX = 0,

RatioX = 0⌋

[SWS_Ifx_00106] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1,

RatioX = 0⌋

[SWS_Ifx_00107] ⌈If Yin < Y_array[0], then

indexY = 0,

RatioY = 0⌋

[SWS_Ifx_00108] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1,

RatioY = 0⌋

[SWS_Ifx_00109] ⌈The minimum value of Nx and Ny shall be 1⌋

[SWS_Ifx_00110] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x060	uint8 Ifx_IntIpoMap_u16u8_u8 (uint16, uint8, uint16, uint16, const uint16 *, const uint8 *, const uint8 *)
0x061	uint16 Ifx_IntIpoMap_u16u8_u16 (uint16, uint8, uint16, uint16, const uint16 *, const uint8 *, const uint16 *)
0x062	sint8 Ifx_IntIpoMap_u16u8_s8 (uint16, uint8, uint16, uint16, const uint16 *, const uint8 *, const sint8 *)
0x063	sint16 Ifx_IntIpoMap_u16u8_s16 (uint16, uint8, uint16, uint16, const uint16 *, const uint8 *, const sint16 *)
0x064	uint8 Ifx_IntIpoMap_u16u16_u8 (uint16, uint16, uint16, uint16, const uint16 *, const uint16 *, const uint8 *)
0x065	uint16 Ifx_IntIpoMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, const uint16 *, const uint16 *)
0x066	sint8 Ifx_IntIpoMap_u16u16_s8 (uint16, uint16, uint16, uint16, const uint16 *, const uint16 *, const sint8 *)
0x067	sint16 Ifx_IntIpoMap_u16u16_s16 (uint16, uint16, uint16, uint16, const uint16 *, const uint16 *, const sint16 *)
0x068	uint8 Ifx_IntIpoMap_u16s8_u8 (uint16, sint8, uint16, uint16, const uint16 *, const sint8 *, const uint8 *)
0x069	uint16 Ifx_IntIpoMap_u16s8_u16 (uint16, sint8, uint16, uint16, const uint16 *, const sint8 *, const uint16 *)
0x06A	sint8 Ifx_IntIpoMap_u16s8_s8 (uint16, sint8, uint16, uint16, const uint16 *, const sint8 *, const sint8 *)
0x06B	sint16 Ifx_IntIpoMap_u16s8_s16 (uint16, sint8, uint16, uint16, const uint16 *, const sint8 *, const sint16 *)
0x06C	uint8 Ifx_IntIpoMap_u16s16_u8 (uint16, sint16, uint16, uint16, const uint16 *, const sint16 *, const uint8 *)
0x06D	uint16 Ifx_IntIpoMap_u16s16_u16 (uint16, sint16, uint16, uint16, const uint16 *, const sint16 *, const uint16 *)
0x06E	sint8 Ifx_IntIpoMap_u16s16_s8 (uint16, sint16, uint16, uint16, const uint16 *, const sint16 *, const sint8 *)
0x06F	sint16 Ifx_IntIpoMap_u16s16_s16 (uint16, sint16, uint16, uint16, const uint16 *, const sint16 *, const sint16 *)
0x070	uint8 Ifx_IntIpoMap_s16u8_u8 (sint16, uint8, sint16, sint16, const sint16 *, const uint8 *, const uint8 *)
0x071	uint16 Ifx_IntIpoMap_s16u8_u16 (sint16, uint8, sint16, sint16, const sint16 *, const uint8 *, const uint16 *)
0x072	sint8 Ifx_IntIpoMap_s16u8_s8 (sint16, uint8, sint16, sint16, const sint16 *, const uint8 *, const sint8 *)
0x073	sint16 Ifx_IntIpoMap_s16u8_s16 (sint16, uint8, sint16, sint16, const sint16 *, const uint8 *, const sint16 *)
0x074	uint8 Ifx_IntIpoMap_s16s8_u8 (sint16, sint8, sint16, sint16, const sint16 *, const sint8 *, const uint8 *)
0x075	uint16 Ifx_IntIpoMap_s16s8_u16 (sint16, sint8, sint16, sint16, const sint16 *, const sint8 *, const uint16 *)
0x076	sint8 Ifx_IntIpoMap_s16s8_s8 (sint16, sint8, sint16, sint16, const sint16 *, const sint8 *, const sint8 *)
0x077	sint16 Ifx_IntIpoMap_s16s8_s16 (sint16, sint8, sint16, sint16, const sint16 *, const sint8 *, const sint16 *)
0x078	uint8 Ifx_IntIpoMap_s16s16_u8 (sint16, sint16, sint16, sint16, const sint16 *, const sint16 *, const uint8 *)
0x079	uint16 Ifx_IntIpoMap_s16s16_u16 (sint16, sint16, sint16, sint16, const sint16 *, const sint16 *, const uint16 *)



△

Routine ID[hex]	Routine prototype
0x07A	sint8 Ifx_IntIpoMap_s16s16_s8 (sint16, sint16, sint16, sint16, const sint16 *, const sint16 *, const sint8 *)
0x07B	sint16 Ifx_IntIpoMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, const sint16 *, const sint16 *)
0x07C	uint8 Ifx_IntIpoMap_u8u8_u8 (uint8, uint8, uint8, uint8, const uint8 *, const uint8 *, const uint8 *)
0x07D	uint16 Ifx_IntIpoMap_u8u8_u16 (uint8, uint8, uint8, uint8, const uint8 *, const uint8 *, const uint16 *)
0x07E	sint8 Ifx_IntIpoMap_u8u8_s8 (uint8, uint8, uint8, uint8, const uint8 *, const uint8 *, const sint8 *)
0x07F	sint16 Ifx_IntIpoMap_u8u8_s16 (uint8, uint8, uint8, uint8, const uint8 *, const uint8 *, const sint16 *)
0x080	uint8 Ifx_IntIpoMap_u8s8_u8 (uint8, sint8, uint8, uint8, const uint8 *, const sint8 *, const uint8 *)
0x081	uint16 Ifx_IntIpoMap_u8s8_u16 (uint8, sint8, uint8, uint8, const uint8 *, const sint8 *, const uint16 *)
0x082	sint8 Ifx_IntIpoMap_u8s8_s8 (uint8, sint8, uint8, uint8, const uint8 *, const sint8 *, const sint8 *)
0x083	sint16 Ifx_IntIpoMap_u8s8_s16 (uint8, sint8, uint8, uint8, const uint8 *, const sint8 *, const sint16 *)
0x084	uint8 Ifx_IntIpoMap_s8s8_u8 (sint8, sint8, sint8, sint8, const sint8 *, const sint8 *, const uint8 *)
0x085	uint16 Ifx_IntIpoMap_s8s8_u16 (sint8, sint8, sint8, sint8, const sint8 *, const sint8 *, const uint16 *)
0x086	sint8 Ifx_IntIpoMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, const sint8 *, const sint8 *)
0x087	sint16 Ifx_IntIpoMap_s8s8_s16 (sint8, sint8, sint8, sint8, const sint8 *, const sint8 *, const sint16 *)
0x0D9	sint32 Ifx_IntIpoMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, const sint32 *, const sint32 *)
0x0DA	uint32 Ifx_IntIpoMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, const uint32 *, const uint32 *)

Table 8.14: Implemented routines for Integrated map interpolation

8.3.3.8 Integrated map look-up

[SWS_Ifx_00111] Definition of API function Ifx_IntLkUpMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <InType> Ny, const <InType>* X_array, const <InType>* Y_array, const <InType>* Val_array )`	
Service ID [hex]	0x008A to 0x008D	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number of X axis samples
	Ny	Number of Y axis samples
	X_array	Pointer to the X axis distribution array
	Y_array	Pointer to the Y axis distribution array
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	This routine returns the value a map at position Xin, Yin with rounding.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00112] ⌈Index calculation:

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00113] ⌈Ratio calculation:

if (indexX < (Nx - 1))

RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])

else

RatioX = 0

if (indexY < (Ny - 1))

RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])

else

RatioY = 0⌋

[SWS_Ifx_00114] ⌈if(RatioX < 0.5 && RatioY < 0.5) then

Result = Val_array [BaseIndex]

if(RatioX ≥ 0.5 && RatioY < 0.5) then

Result = Val_array [BaseIndex + Ny]

if(RatioX < 0.5 && RatioY ≥ 0.5) then

Result = Val_array [BaseIndex + 1]

if(RatioX ≥ 0.5 && RatioY ≥ 0.5) then

Result = Val_array [BaseIndex + Ny + 1]⌋

[SWS_Ifx_00116] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00117] ⌈If Xin < X_array[0], then

indexX = 0⌋

[SWS_Ifx_00118] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1⌋

[SWS_Ifx_00119] ⌈If Yin < Y_array[0], then

indexY = 0⌋

[SWS_Ifx_00120] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1⌋

[SWS_Ifx_00122] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x08A	uint8 Ifx_IntLkUpMap_u8u8_u8(uint8, uint8, uint8, uint8, const uint8 *, const uint8 *, const uint8 *)
0x08B	sint8 Ifx_IntLkUpMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, const sint8 *, const sint8 *)
0x08C	uint16 Ifx_IntLkUpMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, const uint16 *, const uint16 *)
0x08D	sint16 Ifx_IntLkUpMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, const sint16 *, const sint16 *)
0x0DB	sint32 Ifx_IntLkUpMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, const sint32 *, const sint32 *)
0x0DC	uint32 Ifx_IntLkUpMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, const uint32 *, const uint32 *)

Table 8.15: Implemented routines for Integrated map look-up

8.3.3.9 Integrated map look-up without rounding

[SWS_Ifx_00211] Definition of API function Ifx_IntLkUpBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <InType> Ny, const <InType>* X_array, const <InType>* Y_array, const <InType>* Val_array )`	
Service ID [hex]	0x00AA to 0x00AD; 0x00DD, 0x00DE	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number of X axis samples
	Ny	Number of Y axis samples
	X_array	Pointer to the X axis distribution array
	Y_array	Pointer to the Y axis distribution array
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	This routine returns the value a map at position Xin, Yin without rounding.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00212] ⌈Index calculation:

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00214] ⌈Return Value = Val_array [BaseIndex]⌋

[SWS_Ifx_00216] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00217] ⌈If Xin < X_array[0], then

indexX = 0⌋

[SWS_Ifx_00218] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1⌋

[SWS_Ifx_00219] ⌈If Yin < Y_array[0], then

indexY = 0⌋

[SWS_Ifx_00220] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1⌋

[SWS_Ifx_00222] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x0AA	uint8 Ifx_IntLkUpBaseMap_u8u8_u8(uint8, uint8, uint8, uint8, const uint8 *, const uint8 *, const uint8 *)
0x0AB	sint8 Ifx_IntLkUpBaseMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, const sint8 *, const sint8 *)
0x0AC	uint16 Ifx_IntLkUpBaseMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, const uint16 *, const uint16 *)
0x0AD	sint16 Ifx_IntLkUpBaseMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, const sint16 *, const sint16 *)
0x0DD	sint32 Ifx_IntLkUpBaseMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, const sint32 *, const sint32 *)
0x0DE	uint32 Ifx_IntLkUpBaseMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, const uint32 *, const uint32 *)

Table 8.16: Implemented routines for Integrated map look-up without rounding

8.3.3.10 Integrated fix- map interpolation

[SWS_Ifx_00123] Definition of API function Ifx_IntIpoFixMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntIpoFixMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntIpoFixMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <inType> Ny, const <InType>* Val_array, <InType> OffsetX, <InType> ShiftX, <InType> OffsetY, <InType> ShiftY )`	
Service ID [hex]	0x0090 to 0x0093; 0x00DF, 0x00E0	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number to X axis samples
	Ny	Number to Y axis samples
	Val_array	Pointer to the result axis distribution array
	OffsetX	Offset of the first sampling value for X-axis
	ShiftX	'Shift' is the power of 2, (2^ShiftX) represents X-axis distribution point interval
	OffsetY	Offset of the first sampling value for Y-axis
	ShiftY	'Shift' is the power of 2, (2^ShiftY) represents Y-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Interpolation
Description	This routine returns the interpolation result of a map at position Xin, Yin based on a fix axes distribution with offset and shift.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00124] ⌈X and Y axis distribution points shall be calculated based on Offset and Shift values.

X_array[index] = OffsetX + index * 2^{ShiftX}

Y_array[index] = OffsetY + index * 2^{ShiftY}

⌋

Example:

If Offset = 10, Shift = 2 and N = 5 then,

axis = {10, 14, 18, 22, 26} (applicable to X and Y axis)

[SWS_Ifx_00125] ⌈Index calculation :

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00126] ⌈Ratio calculation :

RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])

RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])⌋

[SWS_Ifx_00127] ⌈LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

LowerX = LowerY + (UpperY - LowerY) * RatioY

LowerY = Val_array [BaseIndex + Ny]

UpperY = Val_array [BaseIndex + Ny + 1]

UpperX = LowerY + (UpperY - LowerY) * RatioY

Result = LowerX + (UpperX - LowerX) * RatioX⌋

[SWS_Ifx_00128] ⌈If (Xin == X_array[indexX]) and (Y_array[indexY] < Yin < Y_array[indexY+1])

Result = Val_array [BaseIndex] + (Val_array [BaseIndex+1] - Val_array[BaseIndex]) * RatioY⌋

[SWS_Ifx_00129] ⌈If (Yin == Y_array[indexY]) and (X_array[indexX] < Xin < X_array[indexX+1])

Result = Val_array [BaseIndex] + (Val_array [BaseIndex+Ny] - Val_array[BaseIndex]) * RatioX⌋

[SWS_Ifx_00130] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00131] ⌈If Xin < X_array[0], then

indexX = 0,

RatioX = 0⌋

[SWS_Ifx_00132] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1,

RatioX = 0⌋

[SWS_Ifx_00133] ⌈If Yin < Y_array[0], then

indexY = 0,

RatioY = 0⌋

[SWS_Ifx_00134] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1,

RatioY = 0⌋

[SWS_Ifx_00136] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x090	uint8 Ifx_IntIpoFixMap_u8u8_u8 (uint8, uint8, uint8, uint8, const uint8 *, uint8, uint8, uint8, uint8)
0x091	uint16 Ifx_IntIpoFixMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, uint16, uint16, uint16, uint16)
0x092	sint8 Ifx_IntIpoFixMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, sint8, sint8, sint8, sint8)
0x093	sint16 Ifx_IntIpoFixMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, sint16, sint16, sint16, sint16)
0x0DF	sint32 Ifx_IntIpoFixMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, sint32, sint32, sint32, sint32)
0x0E0	uint32 Ifx_IntIpoFixMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, uint32, uint32, uint32, uint32)

Table 8.17: Implemented routines for Integrated fix- map interpolation

8.3.3.11 Integrated fix- map look up

[SWS_Ifx_00139] Definition of API function Ifx_IntLkUpFixMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpFixMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpFixMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <InType> Ny, const <InType>* Val_array, <InType> OffsetX, <InType> ShiftX, <InType> OffsetY, <InType> ShiftY )`	
Service ID [hex]	0x0095 to 0x0098; 0x00E1, 0x00E2	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number to X axis samples
	Ny	Number to Y axis samples
	Val_array	Pointer to the result axis distribution array
	OffsetX	Offset of the first sampling value for X-axis

▽

△

	ShiftX	'Shift' is the power of 2, (2^ShiftX) represents X-axis distribution point interval
	OffsetY	Offset of the first sampling value for Y-axis
	ShiftY	'Shift' is the power of 2, (2^ShiftY) represents Y-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	This routine returns the value of a map at position Xin, Yin (with rounding) based on a fix axes distribution with offset and shift.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00140] ⌈X and Y axis distribution points shall be calculated based on Offset and Shift values.

X_array[index] = offsetX + index * 2^{ShiftX}

Y_array[index] = offsetY + index * 2^{ShiftY}

⌋

Example:

If Offset = 10, shift = 2 and N = 5 then,

axis = {10, 14, 18, 22, 26} (applicable to X and Y axis)

[SWS_Ifx_00141] ⌈Index calculation:

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00143] ⌈Ratio calculation:

if (indexX < (Nx - 1))

RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])

else

RatioX = 0

if (indexY < (Ny - 1))

RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])

else

RatioY = 0⌋

[SWS_Ifx_00144] ⌈if(RatioX < 0.5 && RatioY < 0.5) then

Result = Val_array [BaseIndex]

if(RatioX $\geq$ 0.5 && RatioY < 0.5) then

Result = Val_array [BaseIndex + Ny]

if(RatioX < 0.5 && RatioY $\geq$ 0.5) then

Result = Val_array [BaseIndex + 1]

if(RatioX $\geq$ 0.5 && RatioY $\geq$ 0.5) then

Result = Val_array [BaseIndex + Ny + 1]⌋

[SWS_Ifx_00145] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00146] ⌈If Xin < X_array[0], then

indexX = 0⌋

[SWS_Ifx_00147] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1⌋

[SWS_Ifx_00148] ⌈If Yin < Y_array[0], then

indexY = 0⌋

[SWS_Ifx_00149] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1⌋

[SWS_Ifx_00151] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x095	uint8 Ifx_IntLkUpFixMap_u8u8_u8 (uint8, uint8, uint8, uint8, const uint8 *, uint8, uint8, uint8, uint8)
0x096	uint16 Ifx_IntLkUpFixMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, uint16, uint16, uint16, uint16)
0x097	sint8 Ifx_IntLkUpFixMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, sint8, sint8, sint8, sint8)
0x098	sint16 Ifx_IntLkUpFixMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, sint16, sint16, sint16, sint16)
0x0E1	sint32 Ifx_IntLkUpFixMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, sint32, sint32, sint32, sint32)
0x0E2	uint32 Ifx_IntLkUpFixMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, uint32, uint32, uint32, uint32)

Table 8.18: Implemented routines for Integrated fix- map look up

8.3.3.12 Integrated fix- map look up without rounding

[SWS_Ifx_00225] Definition of API function Ifx_IntLkUpFixBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpFixBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpFixBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <InType> Ny, const <InType>* Val_array, <InType> OffsetX, <InType> ShiftX, <InType> OffsetY, <InType> ShiftY )`	
Service ID [hex]	0x00B0 to 0x00B3; 0x00E3, 0x00E4	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number to X axis samples
	Ny	Number to Y axis samples
	Val_array	Pointer to the result axis distribution array
	OffsetX	Offset of the first sampling value for X-axis
	ShiftX	'Shift' is the power of 2, (2^ShiftX) represents X-axis distribution point interval
	OffsetY	Offset of the first sampling value for Y-axis
	ShiftY	'Shift' is the power of 2, (2^ShiftY) represents Y-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	This routine returns the value of a map at position Xin, Yin (without rounding) based on a fix axes distribution with offset and shift.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00226] ⌈X and Y axis distribution points shall be calculated based on Offset and Shift values.

X_array[index] = offsetX + index * 2^{ShiftX}

Y_array[index] = offsetY + index * 2^{ShiftY}

⌋

Example:

If Offset = 10, shift = 2 and N = 5 then,

axis = {10, 14, 18, 22, 26} (applicable to X and Y axis)

[SWS_Ifx_00227] ⌈Index calculation:

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00229] ⌈Return Value = Val_array [BaseIndex]⌋

[SWS_Ifx_00230] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00231] ⌈If Xin < X_array[0], then

indexX = 0⌋

[SWS_Ifx_00232] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1⌋

[SWS_Ifx_00233] ⌈If Yin < Y_array[0], then

indexY = 0⌋

[SWS_Ifx_00234] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1⌋

[SWS_Ifx_00236] ⌈Here is the list of implemented routines⌋

Routine ID[hex]	Routine prototype
0x0B0	uint8 Ifx_IntLkUpFixBaseMap_u8u8_u8 (uint8, uint8, uint8, uint8, const uint8 *, uint8, uint8, uint8, uint8)
0x0B1	uint16 Ifx_IntLkUpFixBaseMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, uint16, uint16, uint16, uint16)
0x0B2	sint8 Ifx_IntLkUpFixBaseMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, sint8, sint8, sint8, sint8)
0x0B3	sint16 Ifx_IntLkUpFixBaseMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, sint16, sint16, sint16, sint16)
0x0E3	sint32 Ifx_IntLkUpFixBaseMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, sint32, sint32, sint32, sint32)
0x0E4	uint32 Ifx_IntLkUpFixBaseMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, uint32, uint32, uint32, uint32)

Table 8.19: Implemented routines for Integrated fix- map look up without rounding

8.3.3.13 Integrated fix- I map interpolation

[SWS_Ifx_00153] Definition of API function Ifx_IntIpoFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntIpoFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntIpoFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <InType> Ny, const <InType>* Val_array, <InType> OffsetX, <InType> IntervalX, <InType> OffsetY, <InType> IntervalY )`	
Service ID [hex]	0x009A to 0x009D; 0x00E5, 0x00E6	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number to X axis samples
	Ny	Number to Y axis samples
	Val_array	Pointer to the result axis distribution array
	OffsetX	Offset of the first sampling value for X-axis
	IntervalX	represents X-axis distribution point interval
	OffsetY	Offset of the first sampling value for Y-axis
	IntervalY	represents Y-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the Interpolation
Description	This routine returns the interpolation result of a map at position Xin, Yin based on a fix axis distribution with offset and interval.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00154] ⌈X and Y axis distribution points shall be calculated based on Offset and Interval values.

X_array[index] = offsetX + index * IntervalX

Y_array[index] = offsetY + index * IntervalY

⌋

Example:

If Offset = 10, Interval = 2 and N = 5 then,

axis = {10, 12, 14, 16, 18} (applicable to X and Y axis)

[SWS_Ifx_00155] ⌈Index calculation :

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00156] ⌈Ratio Calculation :

RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])

RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])⌋

[SWS_Ifx_00157] ⌈LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

LowerX = LowerY + (UpperY - LowerY) * RatioY

LowerY = Val_array [BaseIndex + Ny]

UpperY = Val_array [BaseIndex + Ny + 1]

UpperX = LowerY + (UpperY - LowerY) * RatioY

Result = LowerX + (UpperX - LowerX) * RatioX⌋

[SWS_Ifx_00158] ⌈If (Xin == X_array[indexX]) and (Y_array[indexY] < Yin < Y_array[indexY+1])

Result = Val_array [BaseIndex] + (Val_array [BaseIndex+1] - Val_array[BaseIndex]) * RatioY⌋

[SWS_Ifx_00159] ⌈If (Yin == Y_array[indexY]) and (X_array[indexX] < Xin < X_array[indexX+1])

Result = Val_array [BaseIndex] + (Val_array [BaseIndex+Ny] - Val_array[BaseIndex]) * RatioX⌋

[SWS_Ifx_00160] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00161] ⌈If Xin < X_array[0], then

indexX = 0,

RatioX = 0⌋

[SWS_Ifx_00162] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1,

RatioX = 0⌋

[SWS_Ifx_00163] ⌈If Yin < Y_array[0], then

indexY = 0,

RatioY = 0⌋

[SWS_Ifx_00164] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1,

RatioY = 0⌋

[SWS_Ifx_00166] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x09A	uint8 Ifx_IntIpoFixIMap_u8u8_u8 (uint8, uint8, uint8, uint8, const uint8 *, uint8, uint8, uint8, uint8)
0x09B	uint16 Ifx_IntIpoFixIMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, uint16, uint16, uint16, uint16)
0x09C	sint8 Ifx_IntIpoFixIMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, sint8, sint8, sint8, sint8)
0x09D	sint16 Ifx_IntIpoFixIMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, sint16, sint16, sint16, sint16)
0x0E5	sint32 Ifx_IntIpoFixIMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, sint32, sint32, sint32, sint32)
0x0E6	uint32 Ifx_IntIpoFixIMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, uint32, uint32, uint32, uint32)

Table 8.20: Implemented routines for Integrated fix- I map interpolation

8.3.3.14 Integrated fix- I map look up

[SWS_Ifx_00169] Definition of API function Ifx_IntLkUpFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <InType> Ny, const <InType>* Val_array, <InType> OffsetX, <InType> IntervalX, <InType> OffsetY, <InType> IntervalY )`	
Service ID [hex]	0x00A0 to 0x00A3; 0x00E7, 0x00E8	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number to X axis samples
	Ny	Number to Y axis samples
	Val_array	Pointer to the result axis distribution array
	OffsetX	Offset of the first sampling value for X-axis
	IntervalX	represents X-axis distribution point interval

▽

△

	OffsetY	Offset of the first sampling value for Y-axis
	IntervalY	represents Y-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	This routine returns the value of a map at position Xin, Yin (with rounding) based on a fix axis distribution with offset and interval.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00170] ⌈X and Y axis distribution points shall be calculated based on Offset and Interval values.

X_array[index] = offsetX + index * IntervalX

Y_array[index] = offsetY + index * IntervalY

⌋

Example:

If Offset = 10, Interval = 2 and N = 5 then,

axis = {10, 12, 14, 16, 18} (applicable to X and Y axis)

[SWS_Ifx_00171] ⌈Index calculation:

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00173] ⌈Ratio calculation:

if (indexX < (Nx - 1))

RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])

else

RatioX = 0

if (indexY < (Ny - 1))

RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])

else

RatioY = 0⌋

[SWS_Ifx_00174] ⌈if(RatioX < 0.5 && RatioY < 0.5) then

Result = Val_array [BaseIndex]

if(RatioX $\geq$ 0.5 && RatioY < 0.5) then

Result = Val_array [BaseIndex + Ny]

if(RatioX < 0.5 && RatioY $\geq$ 0.5) then

Result = Val_array [BaseIndex + 1]

if(RatioX $\geq$ 0.5 && RatioY $\geq$ 0.5) then

Result = Val_array [BaseIndex + Ny + 1]⌋

[SWS_Ifx_00175] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00176] ⌈If Xin < X_array[0], then

indexX = 0⌋

[SWS_Ifx_00177] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1⌋

[SWS_Ifx_00178] ⌈If Yin < Y_array[0], then

indexY = 0⌋

[SWS_Ifx_00179] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1⌋

[SWS_Ifx_00181] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x0A0	uint8 Ifx_IntLkUpFixIMap_u8u8_u8 (uint8, uint8, uint8, uint8, const uint8 *, uint8, uint8, uint8, uint8)
0x0A1	uint16 Ifx_IntLkUpFixIMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, uint16, uint16, uint16, uint16)
0x0A2	sint8 Ifx_IntLkUpFixIMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, sint8, sint8, sint8, sint8)
0x0A3	sint16 Ifx_IntLkUpFixIMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, sint16, sint16, sint16, sint16)
0x0E7	sint32 Ifx_IntLkUpFixIMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, sint32, sint32, sint32, sint32)
0x0E8	uint32 Ifx_IntLkUpFixIMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, uint32, uint32, uint32, uint32)

Table 8.21: Implemented routines for Integrated fix- I map look up

8.3.3.15 Integrated fix- I map look up without rounding

[SWS_Ifx_00249] Definition of API function Ifx_IntLkUpFixIBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> ⌈

Service Name	Ifx_IntLkUpFixIBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn>	
Syntax	`<OutType> Ifx_IntLkUpFixIBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> ( <InType> Xin, <InType> Yin, <InType> Nx, <InType> Ny, const <InType>* Val_array, <InType> OffsetX, <InType> IntervalX, <InType> OffsetY, <InType> IntervalY )`	
Service ID [hex]	0x00B4 to 0x00B7; 0x00E9, 0x00EA	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number to X axis samples
	Ny	Number to Y axis samples
	Val_array	Pointer to the result axis distribution array
	OffsetX	Offset of the first sampling value for X-axis
	IntervalX	represents X-axis distribution point interval
	OffsetY	Offset of the first sampling value for Y-axis
	IntervalY	represents Y-axis distribution point interval
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Entry point of the result array
Description	This routine returns the value of a map at position Xin, Yin (without rounding) based on a fix axis distribution with offset and interval.	
Available via	Ifx.h	

⌋

[SWS_Ifx_00237] ⌈X and Y axis distribution points shall be calculated based on Offset and Interval values.

X_array[index] = offsetX + index * IntervalX

Y_array[index] = offsetY + index * IntervalY

⌋

Example:

If Offset = 10, Interval = 2 and N = 5 then,

axis = {10, 12, 14, 16, 18} (applicable to X and Y axis)

[SWS_Ifx_00238] ⌈Index calculation:

indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])

BaseIndex = IndexX * Ny + indexY⌋

[SWS_Ifx_00240] ⌈Return Value = Val_array [BaseIndex]⌋

[SWS_Ifx_00241] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifx_00242] ⌈If Xin < X_array[0], then

indexX = 0⌋

[SWS_Ifx_00243] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1⌋

[SWS_Ifx_00244] ⌈If Yin < Y_array[0], then

indexY = 0⌋

[SWS_Ifx_00245] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1⌋

[SWS_Ifx_00247] ⌈Here is the list of implemented routines.⌋

Routine ID[hex]	Routine prototype
0x0B4	uint8 Ifx_IntLkUpFixIBaseMap_u8u8_u8 (uint8, uint8, uint8, uint8, const uint8 *, uint8, uint8, uint8, uint8)
0x0B5	uint16 Ifx_IntLkUpFixIBaseMap_u16u16_u16 (uint16, uint16, uint16, uint16, const uint16 *, uint16, uint16, uint16, uint16)
0x0B6	sint8 Ifx_IntLkUpFixIBaseMap_s8s8_s8 (sint8, sint8, sint8, sint8, const sint8 *, sint8, sint8, sint8, sint8)
0x0B7	sint16 Ifx_IntLkUpFixIBaseMap_s16s16_s16 (sint16, sint16, sint16, sint16, const sint16 *, sint16, sint16, sint16, sint16)
0x0E9	sint32 Ifx_IntLkUpFixIBaseMap_s32s32_s32 (sint32, sint32, sint32, sint32, const sint32 *, sint32, sint32, sint32, sint32)
0x0EA	uint32 Ifx_IntLkUpFixIBaseMap_u32u32_u32 (uint32, uint32, uint32, uint32, const uint32 *, uint32, uint32, uint32, uint32)

Table 8.22: Implemented routines for Integrated fix- l map look up without rounding

8.3.3.16 Cuboid 3D interpolation

[SWS_Ifx_91002] Definition of API function Ifx_IpoCub_<OutTypeMn> ⌈

Service Name	Ifx_IpoCub_<OutTypeMn>	
Syntax	`<OutType> Ifx_IpoCub_<OutTypeMn> ( const Ifx_DPResultU16_Type* dpResultX, const Ifx_DPResultU16_Type* dpResultY, const Ifx_DPResultU16_Type* dpResultZ, uint16 num_x, uint16 num_y, const <InType>* Val_array )`	
Service ID [hex]	0x00F1 to 0x00F4	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResultX	Data point search result for X axis
	dpResultY	Data point search result for Y axis
	dpResultZ	Data point search result for Z axis
	num_x	Number of X axis points
	num_y	Number of Y axis points
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	<OutType>	Result of the interpolation
Description	Based on searched indices and ratios information using the relevant Ifx_DPSearch routine, this routine calculates and returns the interpolation result for a 3D cuboid.	
Available via	Ifx.h	

⌋

[SWS_Ifx_91003] ⌈

Based on searched indices and ratios information using the relevant Ifx_DPSearch routine, this routine calculates and returns the interpolation result for 3D cuboids.

The axis order memory representation is [z][x][y]. This is the column-major orientation COLUMN_DIR from the ASAM standard. The first axis z specifies the selected slice.

Implementation:

Linear interpolation along z-axis between the result of two 2D interpolations between neighbouring X/Y Maps.

num_slice = num_x * num_y

if(dpResultZ->Ratio==0)

Result=Ifx_IpoMap_<OutTypeMn> (dpResultX, dpResultY, num_y, Val_array[num_slice * dpResultZ->Index])

else

LowerXY=Ifx_IpoMap_<OutTypeMn> (dpResultX, dpResultY, num_y, Val_array[num_slice * dpResultZ ->Index])

UpperXY=Ifx_IpoMap_<OutTypeMn> (dpResultX, dpResultY, num_y, Val_array[num_slice * dpResultZ ->Index + 1])

Result=LowerXY + dpResultZ->Ratio * (UpperXY - LowerXY)

⌋

Note: Do not call this routine without using the Ifx_DPSearch routine. It is ensures a valid search result (Ifx_DPResultU16_Type) and initialization.

[SWS_Ifx_91005] ⌈

Routine ID[hex]	Routine prototype
0x0F1	sint8 Ifx_IpoCub_s8 (const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, uint16, uint16, const sint8 *)
0x0F2	uint8 Ifx_IpoCub_u8 (const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, uint16, uint16, const uint8 *)
0x0F3	sint16 Ifx_IpoCub_s16 (const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, uint16, uint16, const sint16 *)
0x0F4	uint16 Ifx_IpoCub_u16 (const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, const Ifx_DPResultU16_Type *, uint16, uint16, const uint16 *)

⌋

8.3.4 Callback notifications

None

8.4 Scheduled functions

The Ifx library does not have scheduled functions.

8.5 Expected Interfaces

None

8.5.1 Mandatory Interfaces

None

8.5.2 Optional Interfaces

None

8.5.3 Callout interfaces

None

8.5.4 Configurable interfaces

None

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 Ifx.

Chapter 10.3 specifies published information of the module Ifx.

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 Ifx Library has no standardized configuration options. 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 [2] Chapter 10.3 *"Published Information"*.

[SWS_Ifx_00814]

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 [REF] 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 [REF].⌋

A Not applicable requirements

[SWS_Ifx_00999]

Upstream requirements: SRS_BSW_00448

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable 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 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

Number	Heading
[SWS_Ifx_00004]	Definition of API function Ifx_DPSearch_<InTypeMn>
[SWS_Ifx_91001]	Definition of imported datatypes of module Ifx
[SWS_Ifx_91002]	Definition of API function Ifx_IpoCub_<OutTypeMn>
[SWS_Ifx_91003]	

Table B.1: Changed Specification Items in R26-11

B.1.3 Deleted Specification Items in R26-11

Number	Heading
[SWS_Ifx_00001]	
[SWS_Ifx_00011]	
[SWS_Ifx_00040]	
[SWS_Ifx_00050]	
[SWS_Ifx_00061]	
[SWS_Ifx_00076]	
[SWS_Ifx_00086]	
[SWS_Ifx_00096]	
[SWS_Ifx_00121]	
[SWS_Ifx_00135]	
[SWS_Ifx_00150]	
[SWS_Ifx_00165]	

▽

△

Number	Heading
[SWS_Ifx_00180]	
[SWS_Ifx_00186]	
[SWS_Ifx_00200]	
[SWS_Ifx_00201]	
[SWS_Ifx_00202]	
[SWS_Ifx_00203]	
[SWS_Ifx_00204]	
[SWS_Ifx_00208]	
[SWS_Ifx_00221]	
[SWS_Ifx_00235]	
[SWS_Ifx_00246]	
[SWS_Ifx_00800]	
[SWS_Ifx_00801]	
[SWS_Ifx_00806]	
[SWS_Ifx_00807]	
[SWS_Ifx_00808]	
[SWS_Ifx_00809]	
[SWS_Ifx_00810]	
[SWS_Ifx_00811]	
[SWS_Ifx_00812]	
[SWS_Ifx_00813]	
[SWS_Ifx_00815]	Definition of API function Ifx_GetVersionInfo
[SWS_Ifx_00816]	
[SWS_Ifx_00818]	
[SWS_Ifx_00821]	
[SWS_Ifx_00823]	
[SWS_Ifx_91004]	

Table B.2: Deleted Specification Items in R26-11

B.1.4 Added Constraints in R26-11

Number	Heading
[SWS_Ifx_CONSTR_04099]	Monotony of axis distribution

▽

△

Number	Heading
[SWS_Ifx_CONSTR_04100]	Access to data point search result structure

Table B.3: Added Constraints in R26-11

B.1.5 Changed Constraints in R26-11

none

B.1.6 Deleted Constraints in R26-11

none

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

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

none

B.2.3 Deleted Specification Items in R25-11

none

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

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

none

B.3.3 Deleted Specification Items in R24-11

none

B.4 Traceable item history of this document according to AUTOSAR Release R23-11

B.4.1 Added Specification Items in R23-11

none

B.4.2 Changed Specification Items in R23-11

Number	Heading
[SWS_Ifx_00002]	Definition of datatype Ifx_DPResultU16_Type
[SWS_Ifx_00014]	
[SWS_Ifx_00017]	
[SWS_Ifx_00022]	
[SWS_Ifx_00027]	
[SWS_Ifx_00032]	
[SWS_Ifx_00041]	
[SWS_Ifx_00051]	
[SWS_Ifx_00062]	
[SWS_Ifx_00077]	
[SWS_Ifx_00087]	
[SWS_Ifx_00097]	
[SWS_Ifx_00110]	
[SWS_Ifx_00122]	
[SWS_Ifx_00136]	
[SWS_Ifx_00151]	
[SWS_Ifx_00166]	
[SWS_Ifx_00181]	
[SWS_Ifx_00209]	
[SWS_Ifx_00222]	

▽

△

Number	Heading
[SWS_Ifx_00236]	
[SWS_Ifx_00247]	
[SWS_Ifx_91001]	Definition of imported datatypes of module Ifx

Table B.4: Changed Specification Items in R23-11

B.4.3 Deleted Specification Items in R23-11

none