Document Title	Specification of Fixed Point Interpolation Routines
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	396
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R19-11

Document Change History

Date	Release	Changed by	Change Description
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	**Added:** • A new requirement (SWS_Ifx_00251) has been added under Section 7.6 to provide clarity on the rounding mechanism for intermediate result calculation. **Removed:** • A requirement (SWS_Ifx_00250) has been removed as it is not realizable for all the scenarios.

Document Change History

Date	Release	Changed by	Change Description
2016-11-30	4.3.0	AUTOSAR Release Management	**Added:** • 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. **Modified:** • 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:** • Added a new statement in Section 8.5 below the formula to provide more clarity to the users **Modified:** • Updated the "Requirements traceability" 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

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2014-10-31	4.2.1	AUTOSAR Release Management	**Added:** • IFX RecordLayout Blueprint reference in section 3.1 **Modified:** • 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. • Serial numbers in Section 3.2
2014-03-31	4.1.3	AUTOSAR Release Management	**Modified:** • 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 ParameterDataPrototype • Corrected for 'DependencyOnArtifact'
2011-12-22	4.0.3	AUTOSAR Administration	• Removal of rounding off feature from 'MAP lookup routines'

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
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

Table of Contents

1 Introduction and functional overview ... 8

2 Acronyms and abbreviations ... 9

3 Related documentation ... 10

3.1 Input documents ... 10
3.2 Related standards and norms ... 10

4 Constraints and assumptions ... 11

4.1 Limitations ... 11
4.2 Applicability to car domains ... 11

5 Dependencies to other modules ... 12

5.1 File structure ... 12

6 Requirements traceability ... 13

7 Functional specification ... 14

7.1 Error classification ... 14
7.2 Error detection ... 14
7.3 Error notification ... 14
7.4 Initialization and shutdown ... 14
7.5 Using Library API ... 14
7.6 library implementation ... 15

8 Routine specification ... 17

8.1 Imported types ... 17
8.2 Type definitions ... 17
8.3 Comment about rounding ... 18
8.4 Comment about routines optimization ... 18
8.4.1 Target optimization ... 18
8.4.2 Optimization for routine numbers ... 19
8.5 Interpolation routines definitions ... 20
8.5.1 Distributed data point search and interpolation ... 21
8.5.2 Integrated data point search and interpolation ... 30
8.5.3 Record layouts for interpolation routines ... 63
8.6 Examples of use of functions ... 66
8.7 Version API ... 66
8.7.1 Ifx_GetVersionInfo ... 66
8.8 Call-back notifications ... 67
8.9 Scheduled routines ... 67
8.10 Expected Interfaces ... 67
8.10.1 Mandatory Interfaces ... 67
8.10.2 Optional Interfaces ... 68
8.10.3 Configurable interfaces ... 68

9 Sequence diagrams ... 69

10 Configuration specification ... 70

10.1 Published Information 70
10.2 Configuration option 70
11 Not applicable requirements 71

1 Introduction and functional overview

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

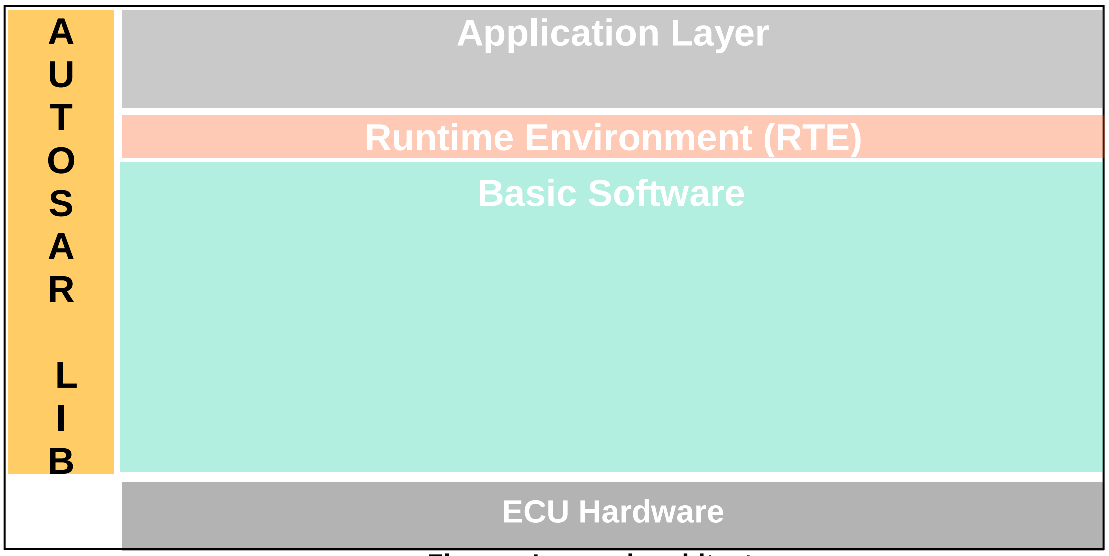

Figure : Layered architecture

Ifx routines specification specifies the functionality, API and the configuration 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 and can be used by multiple applications at the same time.

2 Acronyms and abbreviations

Acronyms and abbreviations, which have a local scope and therefore are not contained in the AUTOSAR glossary, must appear in a local glossary.

Abbreviation / Acronym:	***Description:***
Cur	Curve for Interpolation
DET	Default Error Tracer
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
s8	Mnemonic for the sint8, specified in AUTOSAR_SWS_PlatformTypes
s16	Mnemonic for the sint16, specified in AUTOSAR_SWS_PlatformTypes
s32	Mnemonic for the sint32, specified in AUTOSAR_SWS_PlatformTypes
u8	Mnemonic for the uint8, specified in AUTOSAR_SWS_PlatformTypes
u16	Mnemonic for the uint16, specified in AUTOSAR_SWS_PlatformTypes
u32	Mnemonic for the uint32, specified in AUTOSAR_SWS_PlatformTypes

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules,
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[5] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[6] Specification of Platform Types,
AUTOSAR_SWS_PlatformTypes.pdf

[7] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf

[8] Requirement on Libraries,
AUTOSAR_SRS_Libraries.pdf

[9] Memory mapping mechanism,
AUTOSAR_SWS_MemoryMapping.pdf

[10] Software Component Template,
AUTOSAR_TPS_SoftwareComponentTemplate.pdf

[11] Specification of C Implementation Rules,
AUTOSAR_TR_CImplementationRules.pdf

[12] IFX_RecordLayout_Blueprint,
AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml

3.2 Related standards and norms

[13] ISO/IEC 9899:1990 Programming Language – C

[14] ASAM MCD-2MC Version 1.6 : Association for Standardisation of Automation and Measuring Systems.

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

5.1 File structure

[SWS_Ifx_00001] ⌈ The Ifx module shall provide the following files:

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

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

Option 1 : <Name> can be function name providing one C file per function,
eg.: Ifx_IntIpoMap_u16u8_u8.c etc.

Option 2 : <Name> can have common name of group of functions:
2.1 Group by object family:
eg.:Ifx_IpoMap.c, Ifx_IpoCur.c, Ifx_DPSearch.c
2.2 Group by routine family:
eg.: Ifx_IpoMap.c, Ifx_IntIpoMap.c, Ifx_IpoCur.c etc.
2.3 Group by method family:
eg.: Ifx_Ipo.c, Ifx_IntIpo.c, Ifx_Lkup.c, Ifx_IntLkup.c, etc.
2.4 Group by architecture:
eg.: Ifx_IpoMap8.c, Ifx_IpoMap16.c
2.5 Group by other methods: (individual grouping allowed)

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

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00003	All software modules shall provide version and identification information	SWS_Ifx_00815
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	SWS_Ifx_00809
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_Ifx_00812
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Ifx_00813
SRS_BSW_00318	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	SWS_Ifx_00815
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_Ifx_00815
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_Ifx_00811
SRS_BSW_00374	All Basic Software Modules shall provide a readable module vendor identification	SWS_Ifx_00814
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Ifx_00812
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_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_Ifx_00815, SWS_Ifx_00816
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_Ifx_00816
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_Ifx_00810
SRS_BSW_00448	Module SWS shall not contain requirements from Other Modules	SWS_Ifx_00999
SRS_LIBS_00001	The functional behavior of each library functions shall not be configurable	SWS_Ifx_00818
SRS_LIBS_00002	A library shall be operational before all BSW modules and application SW-Cs	SWS_Ifx_00800
SRS_LIBS_00003	A library shall be operational until the shutdown	SWS_Ifx_00801
SRS_LIBS_00013	The error cases, resulting in the check at runtime of the value of input parameters, shall be listed in SWS	SWS_Ifx_00817, SWS_Ifx_00819
SRS_LIBS_00015	It shall be possible to configure the microcontroller so that the library code is shared between all callers	SWS_Ifx_00806
SRS_LIBS_00017	Usage of macros should be avoided	SWS_Ifx_00807
SRS_LIBS_00018	A library function may only call library functions	SWS_Ifx_00808

7 Functional specification

7.1 Error classification

[SWS_Ifx_00823]⌈
No error classification definition as DET call not supported by library
⌋()

7.2 Error detection

[SWS_Ifx_00819] ⌈ Error detection: Function should check at runtime (both in production and development code) the value of input parameters, especially cases where erroneous value can bring to fatal error or unpredictable result, if they have the values allowed by the function specification. All the error cases shall be listed in SWS and the function should return a specified value (in SWS) that is not configurable. This value is dependant of the function and the error case so it is determined case by case.
If values passed to the routines are not valid and out of the function specification, then such error are not detected.
E.g. If passed value > 32 for a bit-position
or a negative number of samples of an axis distribution is passed to a routine.⌋ (SRS_LIBS_00013)

7.3 Error notification

[SWS_Ifx_00817] ⌈ The functions shall not call the DET for error notification. ⌋ (SRS_LIBS_00013)

7.4 Initialization and shutdown

[SWS_Ifx_00800] ⌈ Ifx library shall not require initialization phase. A Library function may be called at the very first step of ECU initialization, e.g. even by the OS or EcuM, thus the library shall be ready.⌋ (SRS_LIBS_00002)

[SWS_Ifx_00801] ⌈ Ifx library shall not require a shutdown operation phase.⌋ (SRS_LIBS_00003)

7.5 Using Library API

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

The statement 'Ifx.h' shall be placed by the developer or an application code generator but not by the RTE generator

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

7.6 library implementation

[SWS_Ifx_00806] ⌈ The Ifx library shall be implemented in a way that the code can be shared among callers in different memory partitions.⌋ (SRS_LIBS_00015)

[SWS_Ifx_00807] ⌈ Usage of macros should be avoided. The function should be declared as function or inline function. Macro #define should not be used. ⌋ (SRS_LIBS_00017)

[SWS_Ifx_00808] ⌈ A library function can call other library functions because all library functions shall be re-entrant. A library function shall not call any BSW modules functions, e.g. the DET. ⌋ (SRS_LIBS_00018)

[SWS_Ifx_00809] ⌈ The library, written in C programming language, should conform to the MISRA C Standard.
Please refer to SWS_BSW_00115 for more details.
⌋ (SRS_BSW_00007)

[SWS_Ifx_00810] ⌈ Each AUTOSAR library Module implementation <library>*.c and
<library>*.h shall map their code to memory sections using the AUTOSAR memory mapping mechanism.⌋ (SRS_BSW_00437)

[SWS_Ifx_00811] ⌈ Each AUTOSAR library Module implementation <library>*.c, that uses AUTOSAR integer data types and/or the standard return, shall include the header file Std_Types.h.⌋ (SRS_BSW_00348)

[SWS_Ifx_00812] ⌈ All AUTOSAR library Modules should use the AUTOSAR data types (integers, boolean) instead of native C data types, unless this library is clearly identified to be compliant only with a platform. ⌋ (SRS_BSW_00304, SRS_BSW_00378)

[SWS_Ifx_00813] ⌈ All AUTOSAR library Modules should avoid direct use of compiler and platform specific keyword, unless this library is clearly identified to be compliant only with a platform. eg. #pragma, typeof etc.⌋ (SRS_BSW_00306)

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

[SWS_Ifx_00821] ⌈ Axis distribution passed to Ifx routines shall have strong monotony sequence.⌋ ()

[SWS_Ifx_00251] ⌈ The intermediate results during unscaling in interpolation calculation shall be Rounded towards zero.⌋ ()

8 Routine specification

8.1 Imported types

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

Header file	***Imported Type***
Std_Types.h	boolean, sint8, uint8, sint16, uint16, sint32, uint32

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

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

Size	**Platform Type**	**Mnemonic**	**Range**
unsigned 8-Bit	boolean	NA	[TRUE, FALSE]
signed 8-Bit	sint8	s8	[-128, 127]
signed 16-Bit	sint16	s16	[-32768, 32767]
signed 32-Bit	sint32	s32	[-2147483648, 2147483647]
unsigned 8-Bit	uint8	u8	[0, 255]
unsigned 16-Bit	uint16	u16	[0, 65535]
unsigned 32-Bit	uint32	u32	[0, 4294967295]

Table 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 <InTypeMn1> that means Type Mnemonic for 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]⌈

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_00200]⌈
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 Comment about rounding

Two types of rounding can be applied:
Results are 'rounded off', it means:

• 0 <= X < 0.5	rounded to 0
• 0.5 <= X < 1	rounded to 1
• -0.5 < X <= 0	rounded to 0
• -1 < X <= -0.5	rounded to -1

Results are rounded towards zero.

• 0 <= X < 1	rounded to 0
• -1 < X <= 0	rounded to 0

8.4 Comment about routines optimization

8.4.1 Target optimization

The routines described in this library may be realized as regular routines or inline functions. For ROM optimization purposes, it is recommended that the c routines be realized as individual source files so they may be linked in on an as-needed basis.

For example, depending on the target, two types of optimization can be done:

- Some routines can be replaced by another routine using integer promotion

- Some routines can be replaced by the combination of a limiting routine and a routine with a different signature.

8.4.2 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

8.5 Interpolation routines definitions

Interpolation between two given points is calculated as shown below.

$$result = y_0 + (y_1 - y_0) \bullet \frac{x - x_0}{x_1 - x_0}$$

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.

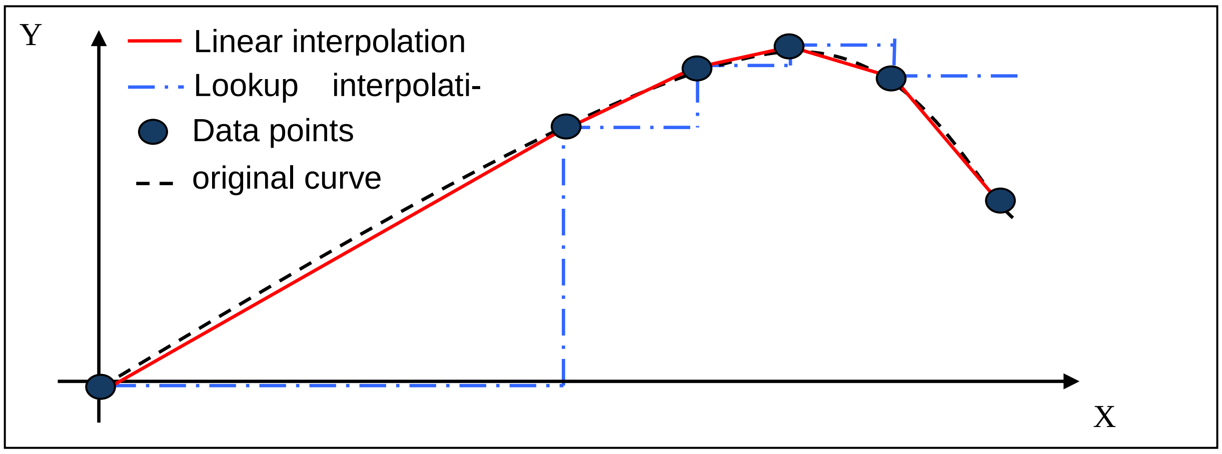

Figure : Linear and lookup interpolation

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 point arrays can be grouped as one array or one structure for all elements as shown below.

one array for all elements :

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

one structure for all elements :

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

where, number of samples = 5

X axis distribution = 0 to 64
Y axis distribution = 1 to 6

Interpolation routines accepts arguments separately to support above scenarios. Routine call example is given below for array and structure grouping respectively.

Example :
uint8 Ifx_IntIpoCur_u8_u8 (15, Curve_u8[0], &Curve_u8[1], &Curve_u8[6]);
uint8 Ifx_IntIpoCur_u8_u8 (15, Curve_u8.N, &Curve_u8.X, &Curve_u8.Y);

Interpolation can be calculated in two ways as shown below:

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

8.5.1 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.

8.5.1.1 Data Point Search

[SWS_Ifx_00004]⌈

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]	0x001 to 0x004	
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 for which (Xin < X_array[index + 1]).

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_00011]⌈
The minimum value of N shall be 1
⌋()

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

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

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 *)

⌋()

8.5.1.2 Curve interpolation

[SWS_Ifx_00015]⌈

Service Name	Ifx_IpoCur_<OutTypeMn>	

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

Service ID [hex]	0x005 to 0x008	
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 calculates and returns interpolation for 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
⌋()

[SWS_Ifx_00201]⌈
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.
⌋()

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

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 *)

⌋()

8.5.1.3 Curve look-up

[SWS_Ifx_00020]⌈

Service Name	Ifx_LkUpCur_<OutTypeMn>	

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

Service ID [hex]	0x00A to 0x00D	
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 and ratio information, this routine calculates and returns entry point of the result array.	
Available via	Ifx.h	

⌋()

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

[SWS_Ifx_00202]⌈
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.
⌋()

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

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 *)

⌋()

8.5.1.4 Map interpolation

[SWS_Ifx_00025]⌈

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]	0x010 to 0x013	
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 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 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
```
⌋()

[SWS_Ifx_00203]⌈
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.
⌋()

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

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

⌋()

8.5.1.5 Map look-up

[SWS_Ifx_00030]⌈

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]	0x015 to 0x018	
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 entry value of the result distribution array.	
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]
⌋()

[SWS_Ifx_00204]⌈
Do not call this routine until you have searched the axis to ensure the search result contains valid data and is not used uninitialized.
⌋()

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

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 *)

⌋()

8.5.1.6 Map look-up without rounding

[SWS_Ifx_00205]⌈

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]	0x0A5 to 0x0A8

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 entry value of the result distribution array.	
Available via	Ifx.h	

⌋()

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

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

[SWS_Ifx_00208]⌈
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.
⌋()

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

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 *)

⌋()

8.5.2 Integrated data point search and interpolation

In this method of interpolation, single routine does data point search (e.g. Index and ratio) and interpolation for curve, map or look-up table.

8.5.2.1 Integrated curve interpolation

[SWS_Ifx_00035]⌈

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]	0x01A to 0x029	
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 calculates interpolation of a curve at position Xin using below equation.	
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_00040]⌈
The minimum value of N shall be 1
⌋()

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

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 *)

⌋()

8.5.2.2 Integrated curve look-up

[SWS_Ifx_00045]⌈

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]	0x030 to 0x03F	
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 respective entry value of the result at position Xin based on below equations.	
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_00050]⌈
The minimum value of N shall be 1
⌋()

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

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 *)

⌋()

8.5.2.3 Integrated fix-curve interpolation

[SWS_Ifx_00055]⌈

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]	0x040 to 0x043	
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	<Out Type>	Result of the Interpolation
Description	This routine calculates interpolation of a curve at position Xin using below equations.	
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}

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_00061]⌈
The minimum value of N shall be 1
⌋()

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

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)

⌋()

8.5.2.4 Integrated fix-curve look up

[SWS_Ifx_00070]⌈

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]	0x045 to 0x048	
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 respective entry value of the result distribution array at position Xin based on below equations.	
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}

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_00076]⌈
The minimum value of N shall be 1
⌋()

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

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)

⌋()

8.5.2.5 Integrated fix- I curve interpolation

[SWS_Ifx_00080]⌈

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]	0x04A to 0x04D	
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 calculates interpolation of a curve at position Xin using below equations.	
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

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 re-spective 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_00086]⌈
The minimum value of N shall be 1
⌋()

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

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)

⌋()

8.5.2.6 Integrated fix- I curve look up

[SWS_Ifx_00090]⌈

Service Name	Ifx_IntLkUpFixICur_<InTypeMn>_<OutTypeMnt>	

Syntax	`<OutType> Ifx_IntLkUpFixICur_<InTypeMn>_<OutTypeMnt> (`
	`  <InType> Xin,`
	`  <InType> N,`
	`  const <InType>* Val_array,`
	`  <InType> Offset,`
	`  <InType> Interval`
	`)`	

Service ID [hex]	0x050 to 0x053	
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 respective entry value of the result distribution array at position Xin based on below equations.	
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

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 re-spective 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_00096]⌈
The minimum value of N shall be 1
⌋()

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

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)

⌋()

8.5.2.7 Integrated map interpolation

[SWS_Ifx_00098]⌈

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]	0x060 to 0x087	
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 calculates Interpolation of a map at position X and Y using below equations.	
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
⌋()

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

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 *)
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 *)

⌋()

8.5.2.8 Integrated map look-up

[SWS_Ifx_00111]⌈

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]	0x08A to 0x08D	
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 respective entry value of the result distribution array at position Xin and Yin based on below equations.	
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_00121]⌈
The minimum value of Nx and Ny shall be 1
⌋()

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

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 *)

⌋()

8.5.2.9 Integrated map look-up without rounding

[SWS_Ifx_00211]⌈

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]	0x0AA to 0x0AD	
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 respective entry value of the result distribution array at position Xin and Yin based on below equations.	
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_00221]⌈
The minimum value of Nx and Ny shall be 1
⌋()

Here is the list of implemented routines.

[SWS_Ifx_00222]⌈

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 *)

⌋()

8.5.2.10 Integrated fix- map interpolation

[SWS_Ifx_00123]⌈

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]	0x090 to 0x093	
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	<Out Type>	Result of the Interpolation
Description	This routine calculates Interpolation of a map at position X and Y using below equations.	
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}

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_00135]⌈
The minimum value of Nx and Ny shall be 1
⌋()

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

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)

⌋()

8.5.2.11 Integrated fix- map look up

[SWS_Ifx_00139]⌈

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]	0x095 to 0x098	
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 respective entry value of the result distribution array at position Xin and Yin based on below equations.	
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}

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 ≥ 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_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_00150]⌈
The minimum value of Nx and Ny shall be 1
⌋()

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

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)

⌋()

8.5.2.12 Integrated fix- map look up without rounding

[SWS_Ifx_00225]⌈

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]	0x0B0 to 0x0B3	
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 respective entry value of the result distribution array at position Xin and Yin based on below equations.	
Available via	Ifx.h	

⌋()

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

$X\_array[index] = offsetX + index * 2^{ShiftX}$
$Y\_array[index] = offsetY + index * 2^{ShiftY}$

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_00235]⌈
The minimum value of Nx and Ny shall be 1
⌋()

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

Routine	***Routine prototype***

ID[hex]	
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)

⌋()

8.5.2.13 Integrated fix- I map interpolation

[SWS_Ifx_00153]⌈

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]	0x09A to 0x09D	
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 calculates Interpolation of a map at position X and Y using below equations.
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

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_00165]⌈
The minimum value of Nx and Ny shall be 1
⌋()

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

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)

⌋()

8.5.2.14 Integrated fix- I map look up

[SWS_Ifx_00169]⌈

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]	0x0A0 to 0x0A3	
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 respective entry value of the result distribution array at position Xin and Yin based on below equations.	
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

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 ≥ 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_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_00180]⌈
The minimum value of Nx and Ny shall be 1
⌋()

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

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)

⌋()

8.5.2.15 Integrated fix- I map look up without rounding

[SWS_Ifx_00249]⌈

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]	0x0B4 to 0x0B7	
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 respective entry value of the result distribution array at position Xin and Yin based on below equations.	
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

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_00246]⌈
The minimum value of Nx and Ny shall be 1
⌋()

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

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)

⌋()

8.5.3 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.

8.5.3.1 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 8.4.2.

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.

According to ASAM standard [ASAM MCD-2MC Version 1.5.1 and 1.6], COLUMN_DIR and ROW_DIR are formats of storing map values (Val[]) and more information can be found in ASAM standard.

8.5.3.2 Record layout definitions

Below table specifies record layouts supported for distributed interpolation routines.

[SWS_Ifx_00185] ⌈

Record layout Name	Element1	Element2
Distr_s8	sint8 N	sint8 X[]
Distr_u8	uint8 N	uint8 X[]
Distr_s16	sint16 N	sint16 X[]
Distr_u16	uint16 N	uint16 X[]
Cur_u8	uint8 Val[]	
Cur_u16	uint16 Val[]	
Cur_s8	sint8 Val[]	
Cur_s16	sint16 Val[]	
Map_u8	uint8 Val[]	
Map_u16	uint16 Val[]	
Map_s8	sint8 Val[]	
Map_s16	sint16 Val[]	

Table: Record layouts for distributed interpolation routines⌋ ()

Below table specifies record layouts supported for integrated interpolation routines.

[SWS_Ifx_00186] ⌈

S.No	Record Layout Name	Element1	Element2	Element3	Element4	Element5
1	IntCur_u8_u8	uint8 N	uint8 X[]	uint8 Val[]		
2	IntCur_u8_u16	uint8 N	uint8 X[]	uint16 Val[]		
3	IntCur_u8_s8	uint8 N	uint8 X[]	sint8 Val[]		
4	IntCur_u8_s16	uint8 N	uint8 X[]	sint16 Val[]		
5	IntCur_u16_u8	uint16 N	uint16 X[]	uint8 Val[]		
6	IntCur_u16_u16	uint16 N	uint16 X[]	uint16 Val[]		
7	IntCur_u16_s8	uint16 N	uint16 X[]	sint8 Val[]		

8	IntCur_u16_s16	uint16 N	uint16 X[]	sint16 Val[]		
9	IntCur_s8_u8	sint8 N	sint8 X[]	uint8 Val[]		
10	IntCur_s8_u16	sint8 N	sint8 X[]	uint16 Val[]		
11	IntCur_s8_s8	sint8 N	sint8 X[]	sint8 Val[]		
12	IntCur_s8_s16	sint8 N	sint8 X[]	sint16 Val[]		
13	IntCur_s16_u8	sint16 N	sint16 X[]	uint8 Val[]		
14	IntCur_s16_u16	sint16 N	sint16 X[]	uint16 Val[]		
15	IntCur_s16_s8	sint16 N	sint16 X[]	sint8 Val[]		
16	IntCur_s16_s16	sint16 N	sint16 X[]	sint16 Val[]		
17	FixIntCur_u8_u8	uint8 N	uint8 Val[]			
18	FixIntCur_u16_u16	uint16 N	uint16 Val[]			
19	FixIntCur_s8_s8	sint8 N	sint8 Val[]			
20	FixIntCur_s16_s16	sint16 N	sint16 Val[]			
21	IntMap_u8u8_u8	uint8 Nx	uint8 Ny	uint8 X[]	uint8 Y[]	uint8 Val[]
22	IntMap_u8u8_u16	uint8 Nx	uint8 Ny	uint8 X[]	uint8 Y[]	uint16 Val[]
23	IntMap_u8u8_s8	uint8 Nx	uint8 Ny	uint8 X[]	uint8 Y[]	sint8 Val[]
24	IntMap_u8u8_s16	uint8 Nx	uint8 Ny	uint8 X[]	uint8 Y[]	sint16 Val[]
25	IntMap_u8s8_u8	uint8 Nx	uint8 Ny	uint8 X[]	sint8 Y[]	uint8 Val[]
26	IntMap_u8s8_u16	uint8 Nx	uint8 Ny	uint8 X[]	sint8 Y[]	uint16 Val[]
27	IntMap_u8s8_s8	uint8 Nx	uint8 Ny	uint8 X[]	sint8 Y[]	sint8 Val[]
28	IntMap_u8s8_s16	uint8 Nx	uint8 Ny	uint8 X[]	sint8 Y[]	sint16 Val[]
29	IntMap_u16u8_u8	uint16 Nx	uint16 Ny	uint16 X[]	uint8 Y[]	uint8 Val[]
30	IntMap_u16u8_u16	uint16 Nx	uint16 Ny	uint16 X[]	uint8 Y[]	uint16 Val[]
31	IntMap_u16u8_s8	uint16 Nx	uint16 Ny	uint16 X[]	uint8 Y[]	sint8 Val[]
32	IntMap_u16u8_s16	uint16 Nx	uint16 Ny	uint16 X[]	uint8 Y[]	sint16 Val[]
33	IntMap_u16u16_u8	uint16 Nx	uint16 Ny	uint16 X[]	uint16 Y[]	uint8 Val[]
34	IntMap_u16u16_u16	uint16 Nx	uint16 Ny	uint16 X[]	uint16 Y[]	uint16 Val[]
35	IntMap_u16u16_s8	uint16 Nx	uint16 Ny	uint16 X[]	uint16 Y[]	sint8 Val[]
36	IntMap_u16u16_s16	uint16 Nx	uint16 Ny	uint16 X[]	uint16 Y[]	sint16 Val[]
37	IntMap_u16s8_u8	uint16 Nx	uint16 Ny	uint16 X[]	sint8 Y[]	uint8 Val[]
38	IntMap_u16s8_u16	uint16 Nx	uint16 Ny	uint16 X[]	sint8 Y[]	uint16 Val[]
39	IntMap_u16s8_s8	uint16 Nx	uint16 Ny	uint16 X[]	sint8 Y[]	sint8 Val[]
40	IntMap_u16s8_s16	uint16 Nx	uint16 Ny	uint16 X[]	sint8 Y[]	sint16 Val[]
41	IntMap_u16s16_u8	uint16 Nx	uint16 Ny	uint16 X[]	sint16 Y[]	uint8 Val[]
42	IntMap_u16s16_u16	uint16 Nx	uint16 Ny	uint16 X[]	sint16 Y[]	uint16 Val[]
43	IntMap_u16s16_s8	uint16 Nx	uint16 Ny	uint16 X[]	sint16 Y[]	sint8 Val[]
44	IntMap_u16s16_s16	uint16 Nx	uint16 Ny	uint16 X[]	sint16 Y[]	sint16 Val[]
45	IntMap_s8s8_u8	sint8 Nx	sint8 Ny	sint8 X[]	sint8 Y[]	uint8 Val[]
46	IntMap_s8s8_u16	sint8 Nx	sint8 Ny	sint8 X[]	sint8 Y[]	uint16 Val[]
47	IntMap_s8s8_s8	sint8 Nx	sint8 Ny	sint8 X[]	sint8 Y[]	sint8 Val[]
48	IntMap_s8s8_s16	sint8 Nx	sint8 Ny	sint8 X[]	sint8 Y[]	sint16 Val[]
49	IntMap_s16u8_u8	sint16 Nx	sint16 Ny	sint16 X[]	uint8 Y[]	uint8 Val[]
50	IntMap_s16u8_s8	sint16 Nx	sint16 Ny	sint16 X[]	uint8 Y[]	sint8 Val[]
51	IntMap_s16u8_u16	sint16 Nx	sint16 Ny	sint16 X[]	uint8 Y[]	uint16 Val[]
52	IntMap_s16u8_s16	sint16 Nx	sint16 Ny	sint16 X[]	uint8 Y[]	sint16 Val[]
53	IntMap_s16s8_u8	sint16 Nx	sint16 Ny	sint16 X[]	sint8 Y[]	uint8 Val[]
54	IntMap_s16s8_u16	sint16 Nx	sint16 Ny	sint16 X[]	sint8 Y[]	uint16 Val[]
55	IntMap_s16s8_s8	sint16 Nx	sint16 Ny	sint16 X[]	sint8 Y[]	sint8 Val[]
56	IntMap_s16s8_s16	sint16 Nx	sint16 Ny	sint16 X[]	sint8 Y[]	sint16 Val[]
57	IntMap_s16s16_u8	sint16 Nx	sint16 Ny	sint16 X[]	sint16 Y[]	uint8 Val[]
58	IntMap_s16s16_u16	sint16 Nx	sint16 Ny	sint16 X[]	sint16 Y[]	uint16 Val[]
59	IntMap_s16s16_s8	sint16 Nx	sint16 Ny	sint16 X[]	sint16 Y[]	sint8 Val[]

60	IntMap_s16s16_s16	sint16 Nx	sint16 Ny	sint16 X[]	sint16 Y[]	sint16 Val[]
61	IntMap_u8u16_u8	uint8 Nx	uint8 Ny	uint8 X[]	uint16 Y[]	uint8 Val[]
62	IntMap_u8u16_u16	uint8 Nx	uint8 Ny	uint8 X[]	uint16 Y[]	uint16 Val[]
63	IntMap_u8u16_s8	uint8 Nx	uint8 Ny	uint8 X[]	uint16 Y[]	sint8 Val[]
64	IntMap_u8u16_s16	uint8 Nx	uint8 Ny	uint8 X[]	uint16 Y[]	sint16 Val[]
65	IntMap_u8s16_u8	uint8 Nx	uint8 Ny	uint8 X[]	sint16 Y[]	uint8 Val[]
66	IntMap_u8s16_u16	uint8 Nx	uint8 Ny	uint8 X[]	sint16 Y[]	uint16 Val[]
67	IntMap_u8s16_s8	uint8 Nx	uint8 Ny	uint8 X[]	sint16 Y[]	sint8 Val[]
68	IntMap_u8s16_s16	uint8 Nx	uint8 Ny	uint8 X[]	sint16 Y[]	sint16 Val[]
69	IntMap_s8u8_u8	sint8 Nx	sint8 Ny	sint8 X[]	uint8 Y[]	uint8 Val[]
70	IntMap_s8u8_u16	sint8 Nx	sint8 Ny	sint8 X[]	uint8 Y[]	uint16 Val[]
71	IntMap_s8u8_s8	sint8 Nx	sint8 Ny	sint8 X[]	uint8 Y[]	sint8 Val[]
72	IntMap_s8u8_s16	sint8 Nx	sint8 Ny	sint8 X[]	uint8 Y[]	sint16 Val[]
73	IntMap_s8s16_u8	sint8 Nx	sint8 Ny	sint8 X[]	sint16 Y[]	uint8 Val[]
74	IntMap_s8s16_u16	sint8 Nx	sint8 Ny	sint8 X[]	sint16 Y[]	uint16 Val[]
75	IntMap_s8s16_s8	sint8 Nx	sint8 Ny	sint8 X[]	sint16 Y[]	sint8 Val[]
76	IntMap_s8s16_s16	sint8 Nx	sint8 Ny	sint8 X[]	sint16 Y[]	sint16 Val[]
77	IntMap_s8u16_u8	sint8 Nx	sint8 Ny	sint8 X[]	uint16 Y[]	uint8 Val[]
78	IntMap_s8u16_u16	sint8 Nx	sint8 Ny	sint8 X[]	uint16 Y[]	uint16 Val[]
79	IntMap_s8u16_s8	sint8 Nx	sint8 Ny	sint8 X[]	uint16 Y[]	sint8 Val[]
80	IntMap_s8u16_s16	sint8 Nx	sint8 Ny	sint8 X[]	uint16 Y[]	sint16 Val[]
81	IntMap_s16u16_u8	sint16 Nx	sint16 Ny	sint16 X[]	uint16 Y[]	uint8 Val[]
82	IntMap_s16u16_u16	sint16 Nx	sint16 Ny	sint16 X[]	uint16 Y[]	uint16 Val[]
83	IntMap_s16u16_s8	sint16 Nx	sint16 Ny	sint16 X[]	uint16 Y[]	sint8 Val[]
84	IntMap_s16u16_s16	sint16 Nx	sint16 Ny	sint16 X[]	uint16 Y[]	sint16 Val[]
85	FixIntMap_u8_u8	uint8 Nx	uint8 Ny	uint8 Val[]		
86	FixIntMap_u16_u16	uint16 Nx	uint16 Ny	uint16 Val[]		
87	FixIntMap_s8_s8	sint8 Nx	sint8 Ny	sint8 Val[]		
88	FixIntMap_s16_s16	sint16 Nx	sint16 Ny	sint16 Val[]		

Table: Record layouts for integrated interpolation routines⌋ ()

Note: As mentioned in in chapter 8.4, interpolation routines optimization is achieved by swaping X and Y axis during function call for Call-back notifications for below mentioned record layouts.
From Map_u8u16_u8 (S. No 61) to Map_s16u16_s16 (S. No 84)

8.6 Examples of use of functions

None

8.7 Version API

8.7.1 Ifx_GetVersionInfo

[SWS_Ifx_00815]⌈

Service Name	Ifx_GetVersionInfo
Syntax	`void Ifx_GetVersionInfo (`

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

⌋(SRS_BSW_00407, SRS_BSW_00003, SRS_BSW_00318, SRS_BSW_00321)
The version information of a BSW module generally contains:
Module Id
Vendor Id
Vendor specific version numbers (SRS_BSW_00407).

[SWS_Ifx_00816] ⌈
If source code for caller and callee of Ifx_GetVersionInfo is available, the Ifx library should realize Ifx_GetVersionInfo as a macro defined in the module's header file.⌋
(SRS_BSW_00407, SRS_BSW_00411)

8.8 Call-back notifications

None.

8.9 Scheduled routines

The Ifx library does not have scheduled routines.

8.10 Expected Interfaces

None

8.10.1 Mandatory Interfaces

None

8.10.2 Optional Interfaces

None

8.10.3 Configurable interfaces

None

9 Sequence diagrams

Not applicable.

10 Configuration specification

10.1 Published Information

[SWS_Ifx_00814] ⌈ The standardized common published parameters as required by SRS_BSW_00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1]. ⌋ (SRS_BSW_00402, SRS_BSW_00374, SRS_BSW_00379)

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

10.2 Configuration option

[SWS_Ifx_00818] ⌈ The Ifx library shall not have any configuration options that may affect the functional behavior of the routines. I.e. for a given set of input parameters, the outputs shall be always the same. For example, the returned value in case of error shall not be configurable.⌋ (SRS_LIBS_00001)

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

11 Not applicable requirements

[SWS_Ifx_00999] ⌈ These requirements are not applicable to this specification. ⌋ (SRS_BSW_00448)