| Document Title | Specification of Fixed Point Interpolation Routines |
|---|---|
| **Document Owner** | AUTOSAR |
| **Document Responsibility** | AUTOSAR |
| **Document Identification No** | 396 |
| **Document Classification** | Standard |

| | |
|---|---|
| **Document Version** | 1.2.0 |
| **Document Status** | Final |
| **Part of Release** | 4.0 |
| **Revision** | 3 |

| Document Change History | | | |
|---|---|---|---|
| **Date** | **Version** | **Changed by** | **Change Description** |
| 09.12.2011 | 1.2.0 | AUTOSAR Administration | Removal of rounding off feature from 'MAP lookup routines' |
| 15.11.2010 | 1.1.0 | AUTOSAR Administration | DPSearch function optimised using structure pointer |
| 07.12.2009 | 1.0.0 | AUTOSAR Administration | Initial Release |

## Disclaimer



## Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

# Table of Contents

Known Limitations ..... 5
1 Introduction and functional overview ..... 6
2 Acronyms and abbreviations ..... 7
3 Related documentation ..... 8
3.1 Input documents ..... 8
3.2 Related standards and norms ..... 8
4 Constraints and assumptions ..... 9
4.1 Limitations ..... 9
4.2 Applicability to car domains ..... 9
5 Dependencies to other modules ..... 10
5.1 File structure ..... 10
6 Requirements traceability ..... 11
6.1 Document: Requirements on Basic SW Modules ..... 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 ..... 18
8.5 Interpolation routines definitions ..... 19
8.5.1 Distributed data point search and interpolation ..... 20
8.5.1.1 Data Point Search ..... 20
8.5.1.2 Curve interpolation ..... 21
8.5.1.3 Curve look-up ..... 22
8.5.1.4 Map interpolation ..... 23
8.5.1.5 Map look-up ..... 24
8.5.1.6 Map look-up without rounding ..... 25
8.5.2 Integrated data point search and interpolation ..... 26
8.5.2.1 Integrated curve interpolation ..... 26
8.5.2.2 Integrated curve look-up ..... 27
8.5.2.3 Integrated fix-curve interpolation ..... 29
8.5.2.4 Integrated fix-curve look up ..... 30
8.5.2.5 Integrated fix- I curve interpolation ..... 31

8.5.2.6 Integrated fix- I curve look up ..... 32
8.5.2.7 Integrated map interpolation ..... 33
8.5.2.8 Integrated map look-up ..... 36
8.5.2.9 Integrated map look-up without rounding ..... 38
8.5.2.10 Integrated fix- map interpolation ..... 40
8.5.2.11 Integrated fix- map look up ..... 42
8.5.2.12 Integrated fix- map look up without rounding ..... 44
8.5.2.13 Integrated fix- I map interpolation ..... 45
8.5.2.14 Integrated fix- I map look up ..... 47
8.5.2.15 Integrated fix- I map look up without rounding ..... 49
8.5.3 Record layouts for interpolation routines ..... 51
8.5.3.1 Record layouts for map values ..... 51
8.5.3.2 Record layout definitions ..... 51
8.6 Examples of use of functions ..... 54
8.7 Version API ..... 54
8.7.1 Ifx_GetVersionInfo ..... 54
8.8 Call-back notifications ..... 54
8.9 Scheduled routines ..... 55
8.10 Expected Interfaces ..... 55
8.10.1 Mandatory Interfaces ..... 55
8.10.2 Optional Interfaces ..... 55
8.10.3 Configurable interfaces ..... 55
9 Sequence diagrams ..... 56
10 Configuration specification ..... 57
10.1 Published Information ..... 57
10.2 Configuration option ..... 57
11 Not applicable requirements ..... 58

# Known Limitations

# 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.

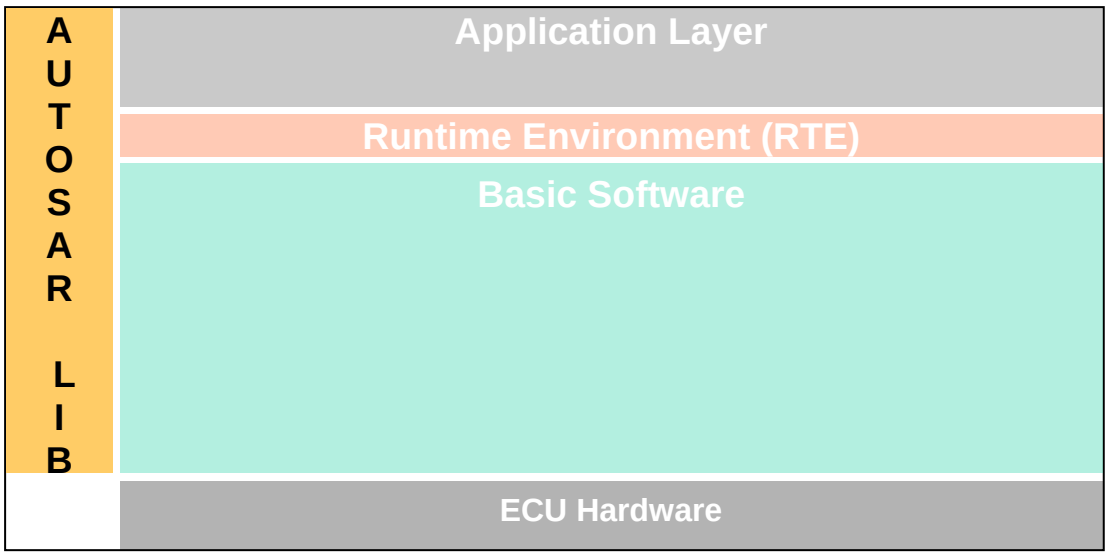


**Figure : Layered architecture**

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 |
| 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

## 3.2 Related standards and norms

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

[11] MISRA-C 2004: Guidelines for the use of the C language in critical systems, October 2004

[12] 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

**[IFX001]** ⌈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_'.
- Header file Ifx.h provides all public function prototypes and types defined by the Ifx library specification ⌋(BSW31400005)

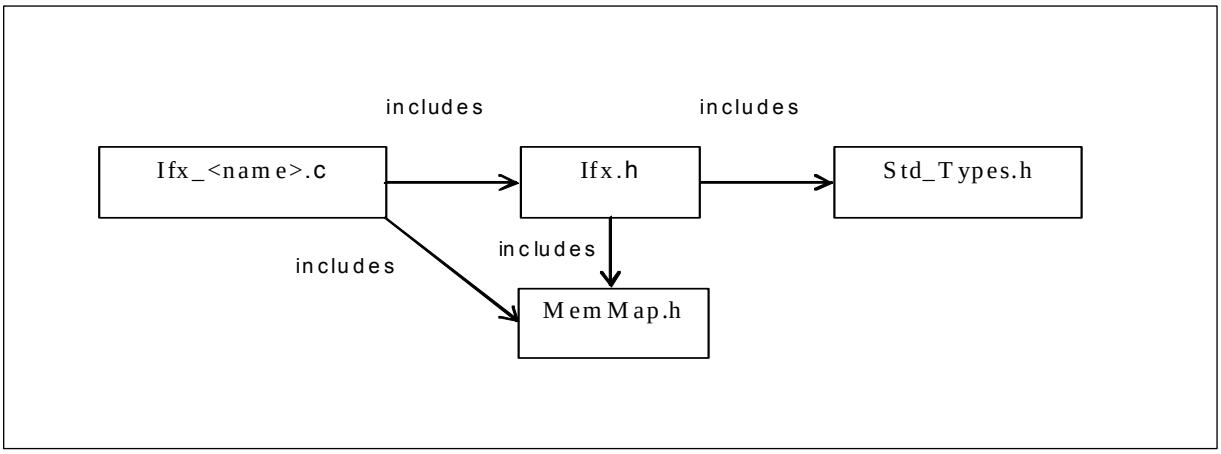


**Figure : File structure**

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 |
|---|---|---|
| BSW | These requirements are not applicable to this specification. | IFX999 |
| BSW003 | | IFX815 |
| BSW00304 | All AUTOSAR library Modules should use the AUTOSAR data types (integers, boolean) instead of nati... | IFX812 |
| BSW00306 | All AUTOSAR library Modules should avoid direct use of compiler and platform specific keyword, un... | IFX813 |
| BSW00318 | | IFX815 |
| BSW00321 | | IFX815 |
| BSW00348 | Each AUTOSAR library Module implementation *. | IFX811 |
| BSW00374 | The standardized common published parameters as required by BSW00402 in the General Requirements ... | IFX814 |
| BSW00378 | All AUTOSAR library Modules should use the AUTOSAR data types (integers, boolean) instead of nati... | IFX812 |
| BSW00379 | The standardized common published parameters as required by BSW00402 in the General Requirements ... | IFX814 |
| BSW00402 | The standardized common published parameters as required by BSW00402 in the General Requirements ... | IFX814 |
| BSW00407 | | IFX815, IFX816 |
| BSW00411 | | IFX816 |
| BSW00436 | Each AUTOSAR library Module implementation *. | IFX810 |
| BSW007 | The library, written in C programming language, should conform to the HIS subset of the MISRA C S... | IFX809 |
| BSW31400001 | The Ifx library shall not have any configuration options that may affect the functional behavior ... | IFX818 |
| BSW31400002 | Ifx library shall not require initialization phase. | IFX800 |
| BSW31400003 | Ifx library shall not require a shutdown operation phase. | IFX801 |
| BSW31400005 | The Ifx module shall provide the following files: | IFX001 |
| BSW31400013 | Error detection: Function should check at runtime (both in production and development code) the v... | IFX819, IFX817 |
| BSW31400015 | The Ifx library shall be implemented in a way that the code can be shared among callers in differ... | IFX806 |
| BSW31400017 | Usage of macros should be avoided. | IFX807 |
| BSW31400018 | A library function can call other library functions because all library functions shall be re-ent... | IFX808 |

Document: Requirements on Libraries

| *Requirement* | *Satisfied by* |
|---|---|
| [BSW31400001] The functional behavior of each library functions shall not be configurable | Section 10.2: **IFX818** |
| [BSW31400002] A library shall be opera- | Section 7.4: **IFX800** |

| *Requirement* | *Satisfied by* |
|---|---|
| tional before all BSW modules and application SWCs | |
| [BSW31400003] A library shall be operational until the shutdown | Section 7.4: **IFX801** |
| [BSW31400004] Using libraries shall not pass through a port interface | Section 7.5 |
| [BSW31400005] Library header file | Section 5.1: **IFX001** |
| [BSW31400006] The header #include is placed by the SWC developer | Section 7.5 |
| [BSW31400007] Using a library should be documented | Section 7.5 |
| [BSW31400008] Backward compatibility | The SWS owner considers this requirement during future SWS evolution. |
| [BSW31400009] Re-entrancy | All APIs in section 8 are defined to be reentrant |
| [BSW31400010] Specific types | Not relevant: section 8.3 does not define specific types |
| [BSW31400011] Naming convention for functions and types | APIs defined in section 8 are named according this requirement |
| [BSW31400012] Passing parameters with structure is allowed | Not relevant, no APIs use this possibility |
| [BSW31400013] Error detection | Section 7.2: **IFX819**,<br>Section 7.3: **IFX817** |
| [BSW31400015] Shared library code | Section 7.6: **IFX806** |
| [BSW31400016] Non AUTOSAR library | Not relevant. Ifx is an AUTOSAR standard library |
| [BSW31400017] Usage of macros should be avoided | Section 7.6: **IFX807** |
| [BSW31400018] A library function can only call library functions | Section 7.6: **IFX808** |

## 6.1 Document: Requirements on Basic SW Modules

A library is not a basic software module. Therefore, many requirements for BSW modules are simply not applicable and not listed below. However, few requirements are relevant:

| ***Requirement*** | ***Satisfied by*** |
|---|---|
| [BSW00402] Published information | Section 10.1: **IFX814** |
| [BSW00407] Function to read out published parameters | Section 8.7.1: **IFX815**, **IFX816** |
| [BSW007] HIS MISRA C | Section 7.6: **IFX809** |
| [BSW00411] Get version info keyword | Section 8.7.1: **IFX816** |
| [BSW00436] Module Header File Structure for the Basic Software Memory Mapping | Section 7.6: **IFX810** |
| [BSW00348] Standard type header | Section 7.6: **IFX811** |
| [BSW00304] AUTOSAR integer data types | Section 7.6: **IFX812** |
| [BSW00378] AUTOSAR boolean type | Section 7.6: **IFX812** |
| [BSW00306] Avoid direct use of compiler and platform specific keywords | Section 7.6: **IFX813** |
| [BSW00374] Module vendor identification | Section 10.1: **IFX814** |
| [BSW00379] Module identification | Section 10.1: **IFX814** |
| [BSW003] Version identification | Section 8.7.1: **IFX815** |
| [BSW00318] Format of module version numbers | Section 8.7.1: **IFX815** |
| [BSW00321] Enumeration of module version numbers | Section 8.7.1: **IFX815** |

# 7 Functional specification

## 7.1 Error classification

**[IFX823]** ⌈No error classification definition as DET call not supported by library

## 7.2 Error detection

**[IFX819]** ⌈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.⌋ (BSW31400013)

## 7.3 Error notification

**[IFX817]** ⌈ The functions shall not call the DET for error notification. ⌋ (BSW31400013)

## 7.4 Initialization and shutdown

**[IFX800]** ⌈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.⌋(BSW31400002)

**[IFX801]** ⌈Ifx library shall not require a shutdown operation phase.⌋(BSW31400003)

## 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-DependencyOnLibrary 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

**[IFX806]** ⌈The Ifx library shall be implemented in a way that the code can be shared among callers in different memory partitions.⌋(BSW31400015)

**[IFX807]** ⌈Usage of macros should be avoided. The function should be declared as function or inline function. Macro #define should not be used.⌋(BSW31400017)

**[IFX808]** ⌈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. ⌋(BSW31400018)

**[IFX809]** ⌈The library, written in C programming language, should conform to the HIS subset of the MISRA C Standard.
Only in technically reasonable, exceptional cases MISRA violations are permissible. Such violations against MISRA rules shall be clearly identified and documented within comments in the C source code (including rationale why MISRA rule is violated). The comment shall be placed right above the line of code which causes the violation and have the following syntax:
`/* MISRA RULE XX VIOLATION: This the reason why the MISRA rule could not be followed in this special case*/`⌋(BSW007)

**[IFX810]** ⌈Each AUTOSAR library Module implementation <library>*.c and <library>*.h shall map their code to memory sections using the AUTOSAR memory mapping mechanism.⌋(BSW00436)

**[IFX811]** ⌈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.⌋(BSW00348)

**[IFX812]** ⌈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.⌋(BSW00304, BSW00378)

**[IFX813]** 「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.」(BSW00306)

**[IFX820]** 「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.」()

**[IFX821]** 「Axis distribution passed to Ifx routines shall have strong monotony sequence.」()

# 8 Routine specification

## 8.1 Imported types

In this chapter, all types included from the following files 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 PlatformTypes.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 :

**[IFX002]** ⌜

| | | | |
|---|---|---|---|
| ***Name:*** | `Ifx_DPResultU16_Type` | | |
| ***Type:*** | `Structure` | | |
| ***Element:*** | `uint16` | `Index` | Data point index |
| | `uint16` | `Ratio` | Data point ratio |
| ***Description:*** | Structure used for data point search for index and ratio<br>**IFX003:**<br>Ratio shall have resolution of $2^{-16}$<br><br>**IFX200:**<br>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

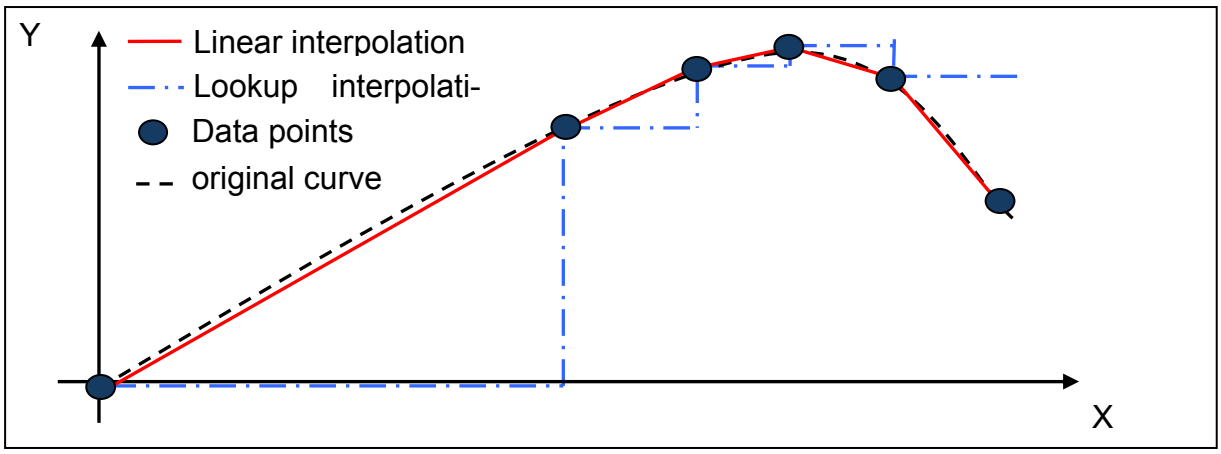


**Figure : Linear and lookup interpolation**

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

**[IFX004]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | Ifx_DPSearch_<InTypeMn> | |
| ***Syntax:*** | `void Ifx_DPSearch_<InTypeMn>(`<br>`  Ifx_DPResultU16_Type* const dpResult,`<br>`  <InType> Xin,`<br>`  <InType> N,`<br>`  const <InType>* X_array`<br>`)` | |
| ***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 (in-out):*** | None | |
| ***Parameters (out):*** | dpResult | Pointer to the result structure |
| ***Return value:*** | None | |
| ***Description:*** | **IFX005:**<br>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.<br><br>**IFX006:**<br>If (X_array[0] < Xin < X_array[N-1]), then returned Index shall be the lowest index for which (Xin < X_array[index + 1]).<br><br>dpResult ->Index = index<br>dpResult ->Ratio = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])<br><br>**IFX008:** | |

| | |
|---|---|
| | Input value matches with one of the distribution array value then return respective index and ratio = 0.<br>If (Xin == X_array[index]) then,<br>dpResult ->Index = index<br>dpResult ->Ratio = 0<br><br>**IFX009:**<br>If (Xin < X_array[0]), then return first index of an array and ratio = 0<br>dpResult ->Index = 0<br>dpResult ->Ratio = 0<br><br>**IFX010:**<br>If (Xin > X_array[N-1]), then return last index of an array and ratio = 0<br>dpResult ->Index = N - 1<br>dpResult ->Ratio = 0<br><br>**IFX011:**<br>The minimum value of N shall be 1<br><br>**IFX013:**<br>This routine returns index and ratio through the structure of type Ifx_DPResultU16_Type |

」()

**[IFX014]** 「

Here is the list of implemented routines.

| *Service ID[hex]* | *Service prototype* |
|---|---|
| 0x001 | void Ifx_DPSearch_u8 (Ifx_DPResultU16_Type* const, uint8, uint8, const uint8 *) |
| 0x002 | void Ifx_DPSearch_s8 (Ifx_DPResultU16_Type* const, sint8, sint8, const sint8 *) |
| 0x003 | void Ifx_DPSearch_u16 (Ifx_DPResultU16_Type* const, uint16, uint16, const uint16 * ) |
| 0x004 | void Ifx_DPSearch_s16 (Ifx_DPResultU16_Type* const, sint16, sint16, const sint16 *) |

」()

### 8.5.1.2 Curve interpolation

**[IFX015]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IpoCur_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_IpoCur_<OutTypeMn>(`<br>`  const Ifx_DPResultU16_Type* const dpResult,`<br>`  const <InType>* Val_array`<br>`)` | |
| ***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 (in-out):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | <OutType> | Result of the Interpolation |
| ***Description:*** | **IFX016:**<br>Based on searched index and ratio information, this routine calculates and returns interpolation for curve. | |

| | |
|---|---|
| | index = dpResult->Index<br>Result = Val_array[index] + (Val_array[index+1] - Val_array[index]) * dpResult->Ratio<br><br>**IFX201:**<br>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.

**[IFX017]** 「

| ***Routine ID[hex]*** | ***Routine prototype*** |
|---|---|
| 0x005 | sint8 Ifx_IpoCur_s8 (const Ifx_DPResultU16_Type* const, const sint8 *) |
| 0x006 | sint16 Ifx_IpoCur_s16 (const Ifx_DPResultU16_Type* const, const sint16 *) |
| 0x007 | uint16 Ifx_IpoCur_u16 (const Ifx_DPResultU16_Type* const, const uint16 *) |
| 0x008 | uint8 Ifx_IpoCur_u8 (const Ifx_DPResultU16_Type* const, const uint8 *) |

」()

### 8.5.1.3 Curve look-up

**[IFX020]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_LkUpCur_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_LkUpCur_<OutTypeMn>(`<br>`  const Ifx_DPResultU16_Type* const dpResult,`<br>`  const <InType>* Val_array`<br>`)` | |
| ***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 (in-out):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | <OutType> | Entry point of the result array |
| ***Description:*** | **IFX021:**<br>Based on searched index and ratio information, this routine calculates and returns entry point of the result array.<br>Result = Val_array[dpResult->Index]<br><br>**IFX202:**<br>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.

**[IFX022]** 「

| ***Routine ID[hex]*** | ***Routine prototype*** |
|---|---|
| 0x00A | sint8 Ifx_LkUpCur_s8 (const Ifx_DPResultU16_Type* const, const sint8 *) |

| | |
|---|---|
| 0x00B | sint16 Ifx_LkUpCur_s16 (const Ifx_DPResultU16_Type* const, const sint16 *) |
| 0x00C | uint16 Ifx_LkUpCur_u16 (const Ifx_DPResultU16_Type* const, const uint16 *) |
| 0x00D | uint8 Ifx_LkUpCur_u8 (const Ifx_DPResultU16_Type* const, const uint8 *) |

」()

### 8.5.1.4 Map interpolation

**[IFX025]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IpoMap_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_IpoMap_<OutTypeMn>(`<br>`  const Ifx_DPResultU16_Type* const dpResultX,`<br>`  const Ifx_DPResultU16_Type* const dpResultY,`<br>`  uint16 num_value,`<br>`  const <InType>* Val_array`<br>`)` | |
| ***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 (in-out):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | <OutType> | Result of the Interpolation |
| ***Description:*** | **IFX026:**<br>Based on searched index and ratio information, this routine calculates and returns interpolation for map.<br><br>BaseIndex = dpResultX->Index * num_value + dpResultY->Index<br>LowerY = Val_array [BaseIndex]<br>UpperY = Val_array [BaseIndex + 1]<br>LowerX = LowerY + (UpperY - LowerY) * dpResultY->Ratio<br><br>LowerY = Val_array [BaseIndex + num_value]<br>UpperY = Val_array [BaseIndex + num_value + 1]<br>UpperX = LowerY + (UpperY - LowerY) * dpResultY->Ratio<br><br>Result = LowerX + (UpperX - LowerX) * dpResultX->Ratio<br><br>**IFX203:**<br>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.

**[IFX027]** 「

| ***Routine ID[hex]*** | ***Routine prototype*** |
|---|---|
| 0x010 | uint8 Ifx_IpoMap_u8 ( const Ifx_DPResultU16_Type* const ,<br>const Ifx_DPResultU16_Type* const ,<br>uint16,<br>const uint8 *) |

| | |
|---|---|
| 0x011 | uint16 Ifx_IpoMap_u16 ( const Ifx_DPResultU16_Type* const , const Ifx_DPResultU16_Type* const , uint16, const uint16 *) |
| 0x012 | sint8 Ifx_IpoMap_s8 ( const Ifx_DPResultU16_Type* const , const Ifx_DPResultU16_Type* const , uint16, const sint8 *) |
| 0x013 | sint16 Ifx_IpoMap_s16 ( const Ifx_DPResultU16_Type* const , const Ifx_DPResultU16_Type* const , uint16, const sint16 *) |

」()

### 8.5.1.5 Map look-up

**[IFX030]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_LkUpMap_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_LkUpMap_<OutTypeMn>( const Ifx_DPResultU16_Type* const dpResultX, const Ifx_DPResultU16_Type* const 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 (in-out):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | <OutType> | Entry point of the result array |
| ***Description:*** | **IFX031:**<br>Based on searched index and ratio information, this routine calculates and returns entry value of the result distribution array.<br><br>BaseIndex = dpResultX->Index * num_value + dpResultY->Index<br><br>**IFX033:**<br>if(dpResultX->Ratio < 0.5 && dpResultY->Ratio < 0.5) then<br>return Val_array [BaseIndex]<br><br>if(dpResultX->Ratio ≥ 0.5 && dpResultY->Ratio < 0.5) then<br>return Val_array [BaseIndex + num_value]<br><br>if(dpResultX->Ratio < 0.5 && dpResultY->Ratio ≥ 0.5) then<br>return Val_array [BaseIndex + 1]<br><br>if(dpResultX->Ratio ≥ 0.5 && dpResultY->Ratio ≥ 0.5) then<br>return Val_array [BaseIndex + num_value + 1]<br><br>**IFX204:**<br>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.

**[IFX032]** 「

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

」()

### 8.5.1.6 Map look-up without rounding

**[IFX205]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_LkUpBaseMap_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_LkUpBaseMap_<OutTypeMn>( const Ifx_DPResultU16_Type* const dpResultX, const Ifx_DPResultU16_Type* const 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 (in-out):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | <OutType> | Entry point of the result array |
| ***Description:*** | **IFX206:**<br>Based on searched index and ratio information, this routine calculates and returns entry value of the result distribution array.<br><br>BaseIndex = dpResultX->Index * num_value + dpResultY->Index<br><br>**IFX207:**<br>Return Value = Val_array [BaseIndex] | |

| | |
|---|---|
| | **IFX208:**<br>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.

**[IFX209]** ⌈

| ***Routine ID[hex]*** | ***Routine prototype*** |
|---|---|
| 0x0A5 | uint8 Ifx_LkUpBaseMap_u8 ( const Ifx_DPResultU16_Type* const ,<br>const Ifx_DPResultU16_Type* const ,<br>uint16,<br>const uint8 *) |
| 0x0A6 | uint16 Ifx_LkUpBaseMap_u16 ( const Ifx_DPResultU16_Type* const ,<br>const Ifx_DPResultU16_Type* const ,<br>uint16,<br>const uint16 *) |
| 0x0A7 | sint8 Ifx_LkUpBaseMap_s8 ( const Ifx_DPResultU16_Type* const ,<br>const Ifx_DPResultU16_Type* const ,<br>uint16,<br>const sint8 *) |
| 0x0A8 | sint16 Ifx_LkUpBaseMap_s16 ( const Ifx_DPResultU16_Type* const ,<br>const Ifx_DPResultU16_Type* const ,<br>uint16,<br>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

**[IFX035]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntIpoCur_<InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_IntIpoCur_<InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> N,`<br>`  const <InType>* X_array,`<br>`  const <InType>* Val_array`<br>`)` | |
| ***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 (in-out):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | <OutType> | Result of the Interpolation |
| ***Description:*** | **IFX036:** | |

| | |
|---|---|
| | This routine calculates interpolation of a curve at position Xin using below equation.<br>If (X_array[0] < Xin < X_array[N -1]), then<br>index = lowest index for which (Xin < X_array[index + 1]).<br>RatioX = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])<br>Result = Val_array[index] + (Val_array[index+1] - Val_array[index])*RatioX<br><br>**IFX037:**<br>Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.<br>If (Xin == X_array[index]) then,<br>Result = Val_array[index]<br><br>**IFX038:**<br>If (Xin < X_array[0]) then,<br>Result = Val_array[0]<br><br>**IFX039:**<br>If (Xin > X_array[N-1]) then,<br>Result = Val_array[N-1]<br><br>**IFX040:**<br>The minimum value of N shall be 1 |

」()

Here is the list of implemented routines.

**[IFX041]** 「

| ***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

**[IFX045]** 「

| | |
|---|---|
| ***Service name:*** | Ifx_IntLkUpCur_<InTypeMn>_<OutTypeMn> |
| ***Syntax:*** | `<OutType> Ifx_IntLkUpCur_<InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> N,` |

| | | |
|---|---|---|
| | const <InType>* X_array,<br>const <InType>* Val_array<br>) | |
| ***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 (in-out):*** | None | |
| ***Parameters (out):*** | None | |
| ***Return value:*** | <OutType> | Entry point of the result array |
| ***Description:*** | **IFX046:**<br>This routine returns respective entry value of the result at position Xin based on below equations.<br>If (X_array[0] < Xin < X_array[N -1]), then<br>index = lowest index for which (Xin < X_array[index + 1]).<br>Result = Val_array[index]<br><br>**IFX047:**<br>Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.<br>If (Xin == X_array[index]) then,<br>Result = Val_array[index]<br><br>**IFX048:**<br>If (Xin < X_array[0]) then,<br>Result = Val_array[0]<br><br>**IFX049:**<br>If (Xin > X_array[N-1]) then,<br>Result = Val_array[N-1]<br><br>**IFX050:**<br>The minimum value of N shall be 1 | |

」()

Here is the list of implemented routines.

**[IFX051]** 「

| ***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

**[IFX055]**

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntIpoFixCur_<InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | <OutType> Ifx_IntIpoFixCur_<InTypeMn>_<OutTypeMn>(<br><InType> Xin,<br><InType> N,<br>const <InType>* Val_array,<br><InType> Offset,<br><InType> Shift<br>) | |
| ***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 (in-out):*** | 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.<br>**IFX056:**<br>X axis distribution points shall be calculated based on Offset and Shift values.<br>X_array [index] = Offset + index * $2^{Shift}$<br><br>If Offset = 10, Shift = 2 and N = 5 then,<br>X_array[5] = {10, 14, 18, 22, 26}<br><br>**IFX057:**<br>If (X_array[0] < Xin < X_array[N -1]), then<br>index = lowest index for which (Xin < X_array[index + 1]).<br>RatioX = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])<br>Result = Val_array[index] + (Val_array[index+1] - Val_array[index]) * RatioX<br><br>**IFX058:**<br>Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.<br>If (Xin == X_array[index])<br>Result = Val_array[index]<br><br>**IFX059:**<br>If (Xin < X_array[0]) then,<br>Result = Val_array[0]<br><br>**IFX060:**<br>If (Xin > X_array[N-1]) then,<br>Result = Val_array[N-1] | |

| | |
|---|---|
| | **IFX061:**<br>The minimum value of N shall be 1 |

⌋()

Here is the list of implemented routines.

**[IFX062]** ⌈

| *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

**[IFX070]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntLkUpFixCur_<InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_IntLkUpFixCur_<InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> N,`<br>`  const <InType>* Val_array,`<br>`  <InType> Offset,`<br>`  <InType> Shift`<br>`)` | |
| ***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 (in-out):*** | 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.<br>**IFX071:**<br>X axis distribution points shall be calculated based on Offset and Shift values.<br>X_array [index] = Offset + index * $2^{Shift}$<br><br>If Offset = 10, Shift = 2 and N = 5 then,<br>X_array[5] = {10, 14, 18, 22, 26}<br><br>**IFX072:**<br>If (X_array[0] < Xin < X_array[N -1]), then<br>index = lowest index for which (Xin < X_array[index + 1]).<br>Result = Val_array[index]<br><br>**IFX073:**<br>Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.<br>If (Xin == X_array[index]) then, | |

| | |
|---|---|
| | Result = Val_array[index]<br><br>**IFX074:**<br>If (Xin < X_array[0]) then,<br>Result = Val_array[0]<br><br>**IFX075:**<br>If (Xin > X_array[N-1]) then,<br>Result = Val_array[N-1]<br><br>**IFX076:**<br>The minimum value of N shall be 1 |

⌋()

Here is the list of implemented routines.

**[IFX077]** ⌈

| *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

**[IFX080]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntIpoFixICur_<InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_IntIpoFixICur_<InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> N,`<br>`  const <InType>* Val_array,`<br>`  <InType> Offset,`<br>`  <InType> Interval`<br>`)` | |
| ***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 (in-out):*** | 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.<br>**IFX081:**<br>X axis distribution points shall be calculated based on Offset and Interval values.<br>X_array [index] = offset + index * Interval<br><br>If Offset = 5, Interval = 12 and N = 5 then, | |

| | |
|---|---|
| | X_array[5] = {5, 17, 29, 41, 53}<br><br>**IFX082:**<br>If (X_array[0] < Xin < X_array[N -1]), then<br>index = lowest index for which (Xin < X_array[index + 1]).<br>RatioX = (Xin - X_array[index]) / (X_array [index+1] - X_array [index])<br>Result = Val_array[index] + (Val_array[index+1] - Val_array[index]) * RatioX<br><br>**IFX083:**<br>Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.<br>If (Xin == X_array[index])<br>Result = Val_array[index]<br><br>**IFX084:**<br>If (Xin < X_array[0]) then,<br>Result = Val_array[0]<br><br>**IFX085:**<br>If (Xin > X_array[N-1]) then,<br>Result = Val_array[N-1]<br><br>**IFX086:**<br>The minimum value of N shall be 1 |

」()

Here is the list of implemented routines.

**[IFX087]** 「

| *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

**[IFX090]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntLkUpFixICur_<InTypeMn>_<OutTypeMnt> | |
| ***Syntax:*** | `<OutType> Ifx_IntLkUpFixICur_<InTypeMn>_<OutTypeMnt>(`<br>`  <InType> Xin,`<br>`  <InType> N,`<br>`  const <InType>* Val_array,`<br>`  <InType> Offset,`<br>`  <InType> Interval`<br>`)` | |
| ***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 (in-out):*** | 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.<br>**IFX091:**<br>X axis distribution points shall be calculated based on Offset and Interval values.<br>X_array [index] = offset + index * Interval<br><br>If Offset = 5, Interval = 12 and N = 5 then,<br>X_array[5] = {5, 17, 29, 41, 53}<br><br>**IFX092:**<br>If (X_array[0] < Xin < X_array[N -1]), then<br>index = lowest index for which (Xin < X_array[index + 1]).<br>Result = Val_array[index]<br><br>**IFX093:**<br>Input value matches with one of the distribution array value then result shall be respective Y array element indicated by index.<br>If (Xin == X_array[index])<br>Result = Val_array[index]<br><br>**IFX094:**<br>If (Xin < X_array[0]) then,<br>Result = Val_array[0]<br><br>**IFX095:**<br>If (Xin > X_array[N-1]) then,<br>Result = Val_array[N-1]<br><br>**IFX096:**<br>The minimum value of N shall be 1 | |

」()

Here is the list of implemented routines.

**[IFX097]** 「

| ***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

**[IFX098]** 「

| | |
|---|---|
| ***Service name:*** | Ifx_IntIpoMap_<InTypeMn><InTypeMn>_<OutTypeMn> |
| ***Syntax:*** | `<OutType> Ifx_IntIpoMap_<InTypeMn><InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> Yin,`<br>`  <InType> Nx,` |

| | | |
|---|---|---|
| | <InType> Ny,<br>const <InType>* X_array,<br>const <InType>* Y_array,<br>const <InType>* Val_array<br>) | |
| ***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 (in-out):*** | 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.<br>**IFX099:**<br>Index calculation :<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX100:**<br>Ratio calculation :<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX101:**<br>LowerY = Val_array [BaseIndex]<br>UpperY = Val_array [BaseIndex + 1]<br>LowerX = LowerY + (UpperY - LowerY) * RatioY<br><br>LowerY = Val_array [BaseIndex + Ny]<br>UpperY = Val_array [BaseIndex + Ny + 1]<br>UpperX = LowerY + (UpperY - LowerY) * RatioY<br><br>Result = LowerX + (UpperX - LowerX) * RatioX<br><br>**IFX102:**<br>If (Xin == X_array[indexX]) and (Y_array[indexY] < Yin < Y_array[indexY+1])<br>Result = Val_array [BaseIndex] + (Val_array [BaseIndex+1] - Val_array[BaseIndex]) * RatioY<br><br>**IFX103:**<br>If (Yin == Y_array[indexY]) and (X_array[indexX] < Xin < X_array[indexX+1])<br>Result = Val_array [BaseIndex] + (Val_array [BaseIndex+Ny] - Val_array[BaseIndex]) * RatioY<br><br>**IFX104:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX105:**<br>If Xin < X_array[0], then<br>indexX = 0, | |

| | |
|---|---|
| | RatioX = 0<br><br>**IFX106:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1,<br>RatioX = 0<br><br>**IFX107:**<br>If Yin < Y_array[0], then<br>indexY = 0,<br>RatioY = 0<br><br>**IFX108:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1,<br>RatioY = 0<br><br>**IFX109:**<br>The minimum value of Nx and Ny shall be 1 |

」()

Here is the list of implemented routines.

**[IFX110]** 「

| ***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 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

**[IFX111]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntLkUpMap_<InTypeMn><InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | <OutType> Ifx_IntLkUpMap_<InTypeMn><InTypeMn>_<OutTypeMn>(<br><InType> Xin,<br><InType> Yin,<br><InType> Nx,<br><InType> Ny,<br>const <InType>* X_array,<br>const <InType>* Y_array,<br>const <InType>* Val_array<br>) | |
| ***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 (in-out):*** | 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.<br><br>**IFX112:**<br>Index calculation :<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX113:**<br>Ratio calculation:<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX114:**<br>if(RatioX < 0.5 && RatioY < 0.5) then<br>Result = Val_array [BaseIndex]<br><br>if(RatioX ≥ 0.5 && RatioY < 0.5) then<br>Result = Val_array [BaseIndex + Ny]<br><br>if(RatioX < 0.5 && RatioY ≥ 0.5) then<br>Result = Val_array [BaseIndex + 1]<br><br>if(RatioX ≥ 0.5 && RatioY ≥ 0.5) then<br>Result = Val_array [BaseIndex + Ny + 1]<br><br>**IFX116:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX117:**<br>If Xin < X_array[0], then<br>indexX = 0<br><br>**IFX118:** | |

| | |
|---|---|
| | If Xin > X_array[Nx-1], then<br>indexX = Nx - 1<br><br>**IFX119:**<br>If Yin < Y_array[0], then<br>indexY = 0<br><br>**IFX120:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1<br><br>**IFX121:**<br>The minimum value of Nx and Ny shall be 1 |

」()

Here is the list of implemented routines.

**[IFX122]** 「

| *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

**[IFX211]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntLkUpBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | `<OutType>`<br>`Ifx_IntLkUpBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> Yin,`<br>`  <InType> Nx,`<br>`  <InType> Ny,`<br>`  const <InType>* X_array,`<br>`  const <InType>* Y_array,`<br>`  const <InType>* Val_array`<br>) | |
| ***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 (in-out):*** | 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.<br><br>**IFX212:**<br>Index calculation:<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX213:**<br>Ratio calculation:<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX214:**<br>Return Value = Val_array [BaseIndex]<br><br>**IFX216:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX217:**<br>If Xin < X_array[0], then<br>indexX = 0<br><br>**IFX218:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1<br><br>**IFX219:**<br>If Yin < Y_array[0], then<br>indexY = 0<br><br>**IFX220:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1<br><br>**IFX221:**<br>The minimum value of Nx and Ny shall be 1 | |

Here is the list of implemented routines.

**[IFX222]** ⌈

| ***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

**[IFX123]** ⌜

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntIpoFixMap_<InTypeMn><InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | `<OutType> Ifx_IntIpoFixMap_<InTypeMn><InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> Yin,`<br>`  <InType> Nx,`<br>`  <inType> Ny,`<br>`  const <InType>* Val_array,`<br>`  <InType> OffsetX,`<br>`  <InType> ShiftX,`<br>`  <InType> OffsetY,`<br>`  <InType> ShiftY`<br>) | |
| ***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 (in-out):*** | 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.<br>**IFX124:**<br>X and Y axis distribution points shall be calculated based on Offset and Shift values.<br><br>X_array[index] = OffsetX + index * $2^{ShiftX}$<br>Y_array[index] = OffsetY + index * $2^{ShiftY}$<br><br>If Offset = 10, Shift = 2 and N = 5 then,<br>axis = {10, 14, 18, 22, 26} (applicable to X and Y axis)<br><br>**IFX125:**<br>Index calculation :<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX126:**<br>Ratio calculation :<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX127:**<br>LowerY = Val_array [BaseIndex] | |

| | |
|---|---|
| | UpperY = Val_array [BaseIndex + 1]<br>LowerX = LowerY + (UpperY - LowerY) * RatioY<br><br>LowerY = Val_array [BaseIndex + Ny]<br>UpperY = Val_array [BaseIndex + Ny + 1]<br>UpperX = LowerY + (UpperY - LowerY) * RatioY<br><br>Result = LowerX + (UpperX - LowerX) * RatioX<br><br>**IFX128:**<br>If (Xin == X_array[indexX]) and (Y_array[indexY] < Yin < Y_array[indexY+1])<br>Result = Val_array [BaseIndex] + (Val_array [BaseIndex+1] -<br>Val_array[BaseIndex]) * RatioY<br><br>**IFX129:**<br>If (Yin == Y_array[indexY]) and (X_array[indexX] < Xin < X_array[indexX+1])<br>Result = Val_array [BaseIndex] + (Val_array [BaseIndex+Ny] -<br>Val_array[BaseIndex]) * RatioY<br><br>**IFX130:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX131:**<br>If Xin < X_array[0], then<br>indexX = 0,<br>RatioX = 0<br><br>**IFX132:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1,<br>RatioX = 0<br><br>**IFX133:**<br>If Yin < Y_array[0], then<br>indexY = 0,<br>RatioY = 0<br><br>**IFX134:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1,<br>RatioY = 0<br><br>**IFX135:**<br>The minimum value of Nx and Ny shall be 1 |

」()

Here is the list of implemented routines.

**[IFX136]** 「

| ***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

**[IFX139]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntLkUpFixMap_<InTypeMn><InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | <OutType> Ifx_IntLkUpFixMap_<InTypeMn><InTypeMn>_<OutTypeMn>(<br><InType> Xin,<br><InType> Yin,<br><InType> Nx,<br><InType> Ny,<br>const <InType>* Val_array,<br><InType> OffsetX,<br><InType> ShiftX,<br><InType> OffsetY,<br><InType> ShiftY<br>) | |
| ***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 (in-out):*** | 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.<br>**IFX140:**<br>X and Y axis distribution points shall be calculated based on Offset and Shift values.<br><br>$X\_array[index] = offsetX + index * 2^{ShiftX}$<br>$Y\_array[index] = offsetY + index * 2^{ShiftY}$<br><br>If Offset = 10, shift = 2 and N = 5 then,<br>axis = {10, 14, 18, 22, 26} (applicable to X and Y axis)<br><br>**IFX141:**<br>Index calculation :<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX143:** | |

| | |
|---|---|
| | Ratio calculation :<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX144:**<br>if(RatioX < 0.5 && RatioY < 0.5) then<br>Result = Val_array [BaseIndex]<br><br>if(RatioX ≥ 0.5 && RatioY < 0.5) then<br>Result = Val_array [BaseIndex + Ny]<br><br>if(RatioX < 0.5 && RatioY ≥ 0.5) then<br>Result = Val_array [BaseIndex + 1]<br><br>if(RatioX ≥ 0.5 && RatioY ≥ 0.5) then<br>Result = Val_array [BaseIndex + Ny + 1]<br><br>**IFX145:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX146:**<br>If Xin < X_array[0], then<br>indexX = 0<br><br>**IFX147:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1<br><br>**IFX148:**<br>If Yin < Y_array[0], then<br>indexY = 0<br><br>**IFX149:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1<br><br>**IFX150:**<br>The minimum value of Nx and Ny shall be 1 |

」()

Here is the list of implemented routines.

**[IFX151]** 「

| ***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

**[IFX225]** ⌈

| | | |
|---|---|---|
| ***Service name:*** | Ifx_IntLkUpFixBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> | |
| ***Syntax:*** | <OutType><br>Ifx_IntLkUpFixBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn>(<br><InType> Xin,<br><InType> Yin,<br><InType> Nx,<br><InType> Ny,<br>const <InType>* Val_array,<br><InType> OffsetX,<br><InType> ShiftX,<br><InType> OffsetY,<br><InType> ShiftY<br>) | |
| ***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 (in-out):*** | 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.<br><br>**IFX226:**<br>X and Y axis distribution points shall be calculated based on Offset and Shift values.<br><br>X_array[index] = offsetX + index * $2^{ShiftX}$<br>Y_array[index] = offsetY + index * $2^{ShiftY}$<br><br>If Offset = 10, shift = 2 and N = 5 then,<br>axis = {10, 14, 18, 22, 26} (applicable to X and Y axis)<br><br>**IFX227:**<br>Index calculation:<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX228:**<br>Ratio calculation:<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX]) | |

| | |
|---|---|
| | RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX229:**<br>Return Value = Val_array [BaseIndex]<br><br>**IFX230:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX231:**<br>If Xin < X_array[0], then<br>indexX = 0<br><br>**IFX232:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1<br><br>**IFX233:**<br>If Yin < Y_array[0], then<br>indexY = 0<br><br>**IFX234:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1<br><br>**IFX235:**<br>The minimum value of Nx and Ny shall be 1 |

Here is the list of implemented routines.

**[IFX236]** ⌈

| *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) |

### 8.5.2.13 Integrated fix- I map interpolation

**[IFX153]** ⌈

| | |
|---|---|
| ***Service name:*** | Ifx_IntIpoFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn> |
| ***Syntax:*** | `<OutType> Ifx_IntIpoFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> Yin,`<br>`  <InType> Nx,`<br>`  <InType> Ny,`<br>`  const <InType>* Val_array,`<br>`  <InType> OffsetX,`<br>`  <InType> IntervalX,`<br>`  <InType> OffsetY,`<br>`  <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 (in-out):*** | 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.<br>**IFX154:**<br>X and Y axis distribution points shall be calculated based on Offset and Interval values.<br><br>X_array[index] = offsetX + index * IntervalX<br>Y_array[index] = offsetY + index * IntervalY<br><br>If Offset = 10, Interval = 2 and N = 5 then,<br>axis = {10, 12, 14, 16, 18} (applicable to X and Y axis)<br><br>**IFX155:**<br>Index calculation :<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX156:**<br>Ratio Calculation :<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX157:**<br>LowerY = Val_array [BaseIndex]<br>UpperY = Val_array [BaseIndex + 1]<br>LowerX = LowerY + (UpperY - LowerY) * RatioY<br><br>LowerY = Val_array [BaseIndex + Ny]<br>UpperY = Val_array [BaseIndex + Ny + 1]<br>UpperX = LowerY + (UpperY - LowerY) * RatioY<br><br>Result = LowerX + (UpperX - LowerX) * RatioX<br><br>**IFX158:**<br>If (Xin == X_array[indexX]) and (Y_array[indexY] < Yin < Y_array[indexY+1])<br>Result = Val_array [BaseIndex] + (Val_array [BaseIndex+1] -<br>Val_array[BaseIndex]) * RatioY<br><br>**IFX159:**<br>If (Yin == Y_array[indexY]) and (X_array[indexX] < Xin < X_array[indexX+1])<br>Result = Val_array [BaseIndex] + (Val_array [BaseIndex+Ny] -<br>Val_array[BaseIndex]) * RatioY | |

| | |
|---|---|
| | **IFX160:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX161:**<br>If Xin < X_array[0], then<br>indexX = 0,<br>RatioX = 0<br><br>**IFX162:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1,<br>RatioX = 0<br><br>**IFX163:**<br>If Yin < Y_array[0], then<br>indexY = 0,<br>RatioY = 0<br><br>**IFX164:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1,<br>RatioY = 0<br><br>**IFX165:**<br>The minimum value of Nx and Ny shall be 1 |

」()

Here is the list of implemented routines.

**[IFX166]** 「

| *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

**[IFX169]** 「

| | |
|---|---|
| ***Service name:*** | Ifx_IntLkUpFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn> |
| ***Syntax:*** | `<OutType>`<br>`Ifx_IntLkUpFixIMap_<InTypeMn><InTypeMn>_<OutTypeMn>(`<br>`  <InType> Xin,`<br>`  <InType> Yin,`<br>`  <InType> Nx,`<br>`  <InType> Ny,`<br>`  const <InType>* Val_array,` |

| | | |
|---|---|---|
| | <InType> OffsetX,<br><InType> IntervalX,<br><InType> OffsetY,<br><InType> IntervalY<br>) | |
| ***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 (in-out):*** | 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.<br>**IFX170:**<br>X and Y axis distribution points shall be calculated based on Offset and Interval values.<br><br>X_array[index] = offsetX + index * IntervalX<br>Y_array[index] = offsetY + index * IntervalY<br><br>If Offset = 10, Interval = 2 and N = 5 then,<br>axis = {10, 12, 14, 16, 18} (applicable to X and Y axis)<br><br>**IFX171:**<br>Index calculation :<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX173:**<br>Ratio calculation:<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX174:**<br>if(RatioX < 0.5 && RatioY < 0.5) then<br>Result = Val_array [BaseIndex]<br><br>if(RatioX ≥ 0.5 && RatioY < 0.5) then<br>Result = Val_array [BaseIndex + Ny]<br><br>if(RatioX < 0.5 && RatioY ≥ 0.5) then<br>Result = Val_array [BaseIndex + 1]<br><br>if(RatioX ≥ 0.5 && RatioY ≥ 0.5) then<br>Result = Val_array [BaseIndex + Ny + 1]<br><br>**IFX175:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex] | |

| | |
|---|---|
| | **IFX176:**<br>If Xin < X_array[0], then<br>indexX = 0<br><br>**IFX177:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1<br><br>**IFX178:**<br>If Yin < Y_array[0], then<br>indexY = 0<br><br>**IFX179:**<br>If Yin > Y_array[Ny-1], then<br>indexY = Ny - 1<br><br>**IFX180:**<br>The minimum value of Nx and Ny shall be 1 |

⌋()

Here is the list of implemented routines.

**[IFX181]** ⌈

| *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

**[IFX236]** ⌈

| | |
|---|---|
| ***Service name:*** | Ifx_IntLkUpFixIBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn> |
| ***Syntax:*** | <OutType><br>Ifx_IntLkUpFixIBaseMap_<InTypeMn><InTypeMn>_<OutTypeMn>(<br><InType> Xin,<br><InType> Yin,<br><InType> Nx,<br><InType> Ny,<br>const <InType>* Val_array,<br><InType> OffsetX,<br><InType> IntervalX,<br><InType> OffsetY,<br><InType> IntervalY<br>) |
| ***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 (in-out):*** | 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.<br><br>**IFX237:**<br>X and Y axis distribution points shall be calculated based on Offset and Interval values.<br><br>X_array[index] = offsetX + index * IntervalX<br>Y_array[index] = offsetY + index * IntervalY<br><br>If Offset = 10, Interval = 2 and N = 5 then,<br>axis = {10, 12, 14, 16, 18} (applicable to X and Y axis)<br><br>**IFX238:**<br>Index calculation:<br>indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])<br>indexY = minimum value of index if (Y_array[indexY] < Yin < Y_array[indexY+1])<br>BaseIndex = IndexX * Ny + indexY<br><br>**IFX239:**<br>Ratio calculation:<br>RatioX = (Xin - X_array[indexX]) / (X_array [indexX+1] - X_array [indexX])<br>RatioY = (Yin - Y_array[indexY]) / (Y_array [indexY+1] - Y_array [indexY])<br><br>**IFX240:**<br>Return Value = Val_array [BaseIndex]<br><br>**IFX241:**<br>If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])<br>Result = Val_array [BaseIndex]<br><br>**IFX242:**<br>If Xin < X_array[0], then<br>indexX = 0<br><br>**IFX243:**<br>If Xin > X_array[Nx-1], then<br>indexX = Nx - 1<br><br>**IFX244:**<br>If Yin < Y_array[0], then<br>indexY = 0<br><br>**IFX245:**<br>If Yin > Y_array[Ny-1], then | |

| | |
|---|---|
| | indexY = Ny - 1<br><br>**IFX246:**<br>The minimum value of Nx and Ny shall be 1 |

Here is the list of implemented routines.

**[IFX247]** ⌈

| *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 CalprmElementPrototype. 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.

**[IFX185]** ⌈

| 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 | uint16 N | uint8 Val[] |
| Map_u16 | uint16 N | uint16 Val[] |
| Map_s8 | uint16 N | sint8 Val[] |
| Map_s16 | uint16 N | sint16 Val[] |

**Table: Record layouts for distributed interpolation routines ]()**

Below table specifies record layouts supported for integrated interpolation routines.

**[IFX186]** 「

| S.No | Record Layout Name | Element1 | Element2 | Element3 | Element4 | Element5 | Element6 | Element7 |
|---|---|---|---|---|---|---|---|---|
| 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[] | uint8 Offset | uint8 Shift/Intv | | | |
| 18 | FixIntCur_u16_u16 | uint16 N | uint16 Val[] | uint16 Offset | uint16 Shift/Intv | | | |
| 19 | FixIntCur_s8_s8 | sint8 N | sint8 Val[] | sint8 Offset | sint8 Shift/Intv | | | |
| 20 | FixIntCur_s16_s16 | sint16 N | sint16 Val[] | sint16 Offset | sint16 Shift/Intv | | | |
| 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[] | uint8 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[] | uint8 OffsetX | uint8 Shift/IntvX | uint8 OffsetY | uint8 Shift/IntvY |
| 86 | FixIntMap_u16_u16 | uint16 Nx | uint16 Ny | uint16 Val[] | uint16 OffsetX | uint16 Shift/IntvX | uint16 OffsetY | uint16 Shift/IntvY |
| 87 | FixIntMap_s8_s8 | sint8 Nx | sint8 Ny | sint8 Val[] | sint8 OffsetX | sint8 Shift/IntvX | sint8 OffsetY | sint8 Shift/IntvY |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 88 | FixIntMap_s16_s16 | sint16 Nx | sint16 Ny | sint16 Val[] | sint16 OffsetX | sint16 Shift/IntvX | sint16 OffsetY | sint16 Shift/IntvY |

**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

**[IFX815]** 「

| | | |
|---|---|---|
| ***Service name:*** | Ifx_GetVersionInfo | |
| ***Syntax:*** | `void Ifx_GetVersionInfo( Std_VersionInfoType* versioninfo )` | |
| ***Service ID[hex]:*** | 0xff | |
| ***Sync/Async:*** | Synchronous | |
| ***Reentrancy:*** | Reentrant | |
| ***Parameters (in):*** | None | |
| ***Parameters (in-out):*** | 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. | |

」(BSW00407, BSW003, BSW00318, BSW00321)

The version information of a BSW module generally contains:
Module Id
Vendor Id
Vendor specific version numbers (BSW00407).

**[IFX816]** 「
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.」(BSW00407, BSW00411)

## 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

**[IFX814]** ⌈ The standardized common published parameters as required by BSW00402 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].⌋(BSW00402, BSW00374, BSW00379)

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

## 10.2 Configuration option

**[IFX818]** ⌈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.⌋(BSW31400001)

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

# 11 Not applicable requirements

**[IFX999]**「These requirements are not applicable to this specification.」(BSW)