Document Title	Specification of Floating Point Interpolation Library
Document Category	Software specification (SWS)
Document UID	398
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Removed API function Ifl_GetVersionInfo • Restructured chapter 8 "API specification"; interpolation routine definitions and general sections moved to chapter 7 • Removed chapter "Extended Production Errors" • Removed references to Requirements on Libraries
2025-11-27	R25-11	AUTOSAR Release Management	• Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Added import type
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes.
2022-11-24	R22-11	AUTOSAR Release Management	• Requirements added SWS_Ifl_91000 to SWS_Ifl_91003 • Requirements added SWS_Ifl_00226, SWS_Ifl_00228, SWS_Ifl_00229, SWS_Ifl_00231, SWS_Ifl_00232, SWS_Ifl_00234, SWS_Ifl_00235 • Modified SWS_Ifl_00170 , SWS_Ifl_00011 and SWS_Ifl_00221
2021-11-25	R21-11	AUTOSAR Release Management	• Editorial changes



△

2020-11-30	R20-11	AUTOSAR Release Management	• Chapter 7.1 Error sections updated
2019-11-28	R19-11	AUTOSAR Release Management	• Editorial changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Section 2 has been revisited to update Default Error Tracer instead of Development Error tracer • Updated IFL document to support MISRA 2012 standard. (Removed redundant statements in SWS_Ifl_00209 which already exist in SWS_BSW document and SWS_SRS document) • Updated the correct reference to SRS_BSW_General (SRS_BSW_00437) & (SRS_BSW_00448) for SWS_Ifl_00210 & SWS_Ifl_00224 requirements.
2015-07-31	4.2.2	AUTOSAR Release Management	• Updated Record layouts definitions for SWS_Ifx_00170 • Updated SWS_Ifl_00001 for naming convention under Section 5.1, File Structure • Updated valid range for float32 in Table 1 of Section 8.1
2014-10-31	4.2.1	AUTOSAR Release Management	• Added IFL RecordLayout Blueprint reference in section 3.1 • The usage of const is updated in function parameters for SWS_Ifl_00010, SWS_Ifl_00021 & SWS_Ifl_00025 • IFL Blueprint modified for the schema version • Serial numbers in Section 3.2

▽

△

2013-10-31	4.1.2	AUTOSAR Release Management	• Corrected array-out-of-bounds for Ifl_IpoMap function • Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• Corrected the formula for integrated map interpolation and map interpolation • Corrected array out-of-bounds for curve interpolation • Modified the reference to non-existant metamodel elementCalprmElementPrototype to Param-eterDataPrototype • Corrected for 'DependencyOnArtifact'
2011-12-22	4.0.3	AUTOSAR Administration	• Error classification support and defi-nition removed as DET call not sup-ported by library • Configuration parameter description / support removed for XXX_GetVersionInfo routine. • XXX_GetVersionInfo routine name corrected.
2010-09-30	3.1.5	AUTOSAR Administration	• DPSearch function optimised using structure pointer • Removal of normalised functions
2010-02-02	3.1.4	AUTOSAR Administration	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 7
2 Acronyms and Abbreviations 8
3 Related documentation 9
3.1 Input documents & related standards and norms . . . 9
3.2 Related specification . . . 9
4 Constraints and assumptions 10
4.1 Limitations . . . 10
4.2 Applicability to car domains . . . 10
5 Dependencies to other modules 11
5.1 File structure . . . 11
6 Requirements Tracing 12
7 Functional specification 13
7.1 Initialization and shutdown . . . 13
7.2 Using Library API . . . 13
7.3 Library implementation . . . 13
7.4 Interpolation routines definitions . . . 13
7.4.1 Record layouts for interpolation routines . . . 15
7.4.1.1 Record layout for map values . . . 15
7.4.1.2 Record layout definitions . . . 15
7.5 Error Classification . . . 16
7.5.1 Development Errors . . . 16
7.5.2 Runtime Errors . . . 16
7.5.3 Production Errors . . . 16
8 API specification 17
8.1 Imported types . . . 17
8.2 Type definitions . . . 17
8.3 Function definitions . . . 18
8.3.1 Distributed data point search and interpolation . . . 18
8.3.1.1 Data Point Search . . . 18
8.3.1.2 Curve interpolation . . . 20
8.3.1.3 Map interpolation . . . 21
8.3.1.4 Single point interpolation . . . 22
8.3.2 Integrated data point search and interpolation . . . 22
8.3.2.1 Integrated curve interpolation . . . 23
8.3.2.2 Integrated map interpolation . . . 24
8.3.2.3 Cuboid 3D interpolation . . . 26
8.3.2.4 Mixed type interpolation of integer curve . . . 27
8.3.2.5 Mixed type interpolation of integer map . . . 28

8.3.2.6 Mixed type interpolation of integer 3D Cuboid 29
8.4 Scheduled routines 31
8.5 Expected interfaces 31
8.5.1 Mandatory interfaces 31
8.5.2 Optional interfaces 31
8.5.3 Configurable interfaces 31
9 Sequence diagrams 32
10 Configuration specification 33
10.1 How to read this chapter 33
10.2 Containers and configuration parameters 33
10.3 Published Information 33
A Not applicable requirements 34
B Change history of AUTOSAR traceable items 35
B.1 Traceable item history of this document according to AUTOSAR Release R23-11 35
B.1.1 Added Specification Items in R23-11 35
B.1.2 Changed Specification Items in R23-11 35
B.1.3 Deleted Specification Items in R23-11 35
B.2 Traceable item history of this document according to AUTOSAR Release R24-11 36
B.2.1 Added Specification Items in R24-11 36
B.2.2 Changed Specification Items in R24-11 36
B.2.3 Deleted Specification Items in R24-11 36
B.3 Traceable item history of this document according to AUTOSAR Release R25-11 36
B.3.1 Added Specification Items in R25-11 36
B.3.2 Changed Specification Items in R25-11 36
B.3.3 Deleted Specification Items in R25-11 36
B.4 Traceable item history of this document according to AUTOSAR Release R26-11 37
B.4.1 Added Specification Items in R26-11 37
B.4.2 Changed Specification Items in R26-11 37
B.4.3 Deleted Specification Items in R26-11 37
B.4.4 Added Constraints in R26-11 38
B.4.5 Changed Constraints in R26-11 38
B.4.6 Deleted Constraints in R26-11 38

1 Introduction and functional overview

This specification specifies the functionality, API of the AUTOSAR library dedicated to interpolation and lookup routines for floating point values.

The interpolation library contains the following routines:

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

All routines are re-entrant.

2 Acronyms and Abbreviations

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
ROM	Read only memory
hex	Hexadecimal
Rev	Revision
Mn	Mnemonic
Lib	Library

Table 2.1: Acronyms and Abbreviations

3 Related documentation

3.1 Input documents & related standards and norms

[1] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[2] IFL_RecordLayout_Blueprint
AUTOSAR_MOD_IFL_RecordLayout_Blueprint.arxml

[3] Specification of Platform Types for Classic Platform
AUTOSAR_CP_SWS_PlatformTypes

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [1], which is also valid for IFL Library.

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

5.1 File structure

[SWS_Ifl_00001] ⌈An implementation of the Ifl library shall follow the naming rules for files outlined in [1].⌋

6 Requirements Tracing

Requirement	Description	Satisfied by
[SRS_BSW_00448]	Module SWS shall not contain requirements from other modules	[SWS_Ifl_00224]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Initialization and shutdown

Refer to [1] chapter 7.1.20.

7.2 Using Library API

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

7.3 Library implementation

[SWS_Ifl_00220] ⌈If input value is less than first distribution entry then first value of the distribution array shall be returned or used in the interpolation routines. If input value is greater than last distribution entry then last value of the distribution array shall be returned or used in the interpolation routines.⌋

[SWS_Ifl_CONSTR_06700] Constraint on input of Ifl routines ⌈Axis distribution passed to Ifl routines shall have normal monotony sequence.⌋

7.4 Interpolation routines definitions

Interpolation between two given points is calculated as shown below.

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

where: x is the input value

$x_0 =$ data point before x

$x_1 =$ data point after x

$y_0 =$ value at x_0

$y_1 =$ value at x_1

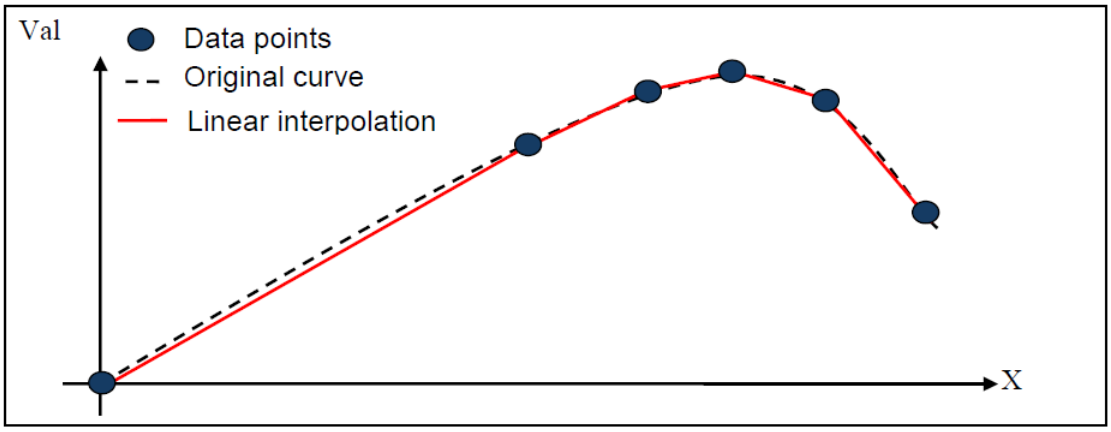


Figure 7.1: Linear interpolation

Data points can be grouped in different ways. Examples:

a) separate arrays for axis and curve

```
float32 AxisCurveA_f32[5] = {0.0,10.0,26.0,36.0,64.0};
float32 CurveA_f32[5] = {1.0,12.0,17.0,11.0,6.0};
```

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

```
float32 CurveB_f32[11] =
   {5,0.0,10.0,26.0,36.0,64.0,1.0,12.0,17.0,11.0,6.0};
```

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

```
struct{
    uint32 N = 5; // number of samples
    float32 X[5] = {0.0,10.0,26.0,36.0,64.0};
    float32 Y[5] = {1.0,12.0,17.0,11.0,6.0};
} CurveC_f32;
```

Interpolation routines have separate arguments to support above scenarios. Routine call examples are given below for array and structure grouping respectively. Example: Axis input value is 15.0

```
float32 Ifl_IntIpoCur_f32_f32(15.0, 5, &AxisCurveA_f32[0], &CurveA_f32
   [0]);
float32 Ifl_IntIpoCur_f32_f32(15.0, CurveB_f32[0], &CurveB_f32[1], &
   CurveB_f32[6]);
float32 Ifl_IntIpoCur_f32_f32(15.0, CurveC_f32.N, &CurveC_f32.X[0], &
   CurveC_f32.Y[0]);
```

Interpolation can be calculated in two ways as shown below:

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

7.4.1 Record layouts for interpolation routines

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

7.4.1.1 Record layout for map values

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

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.

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

Example for cuboids order:

2x2x2 cuboid representation in memory

Example: cub = [1 2 3 4 5 6 7 8]

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

Slice 1:

[1 2

3 4]

Slice 2:

[5 6

7 8]

7.4.1.2 Record layout definitions

Below table specifies record layouts supported for interpolation routines.

[SWS_Ifl_00170] ⌈

Record layout Name	Element1	Element2	Element3	Element4	Element5
Distr_f32	uint32 N	float32 X[]			
Curve_f32	float32 Val[]				
Map_f32	float32 Val[]				
Cub_f32	float32 Val[]				
IntCurve_f32_f32	uint32 N	float32 X[]	float32 Val[]		
IntMap_f32f32_f32	uint32 Nx	uint32 Ny	float32 X[]	float32 Y[]	float32 Val[]

⌋

Remark:

All combinations have to be defined in [2].

7.5 Error Classification

Chapter [1, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

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

7.5.1 Development Errors

There are no development errors.

7.5.2 Runtime Errors

There are no runtime errors.

7.5.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

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 [3]. The following mnemonic are used in the library routine names.

Size	Platform Type	Mnemonic
unsigned 8-Bit	boolean	NA
signed 8-Bit	sint8	s8
signed 16-Bit	sint16	s16
signed 32-Bit	sint32	s32
unsigned 8-Bit	uint8	u8
unsigned 16-Bit	uint16	u16
unsigned 32-Bit	uint32	u32
32-Bit	float32	f32

Table 8.1: Mnemonic for Base Types

As a convention in the rest of the document:

- mnemonics will be used in the name of the routines (using <InTypeMn> that means Type Mnemonic for Input)
- The real type will be used in the description of the prototypes of the routines (using <InType>).

[SWS_Ifl_91004] Definition of imported datatypes of module Ifl ⌈

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

⌋

8.2 Type definitions

Structure definition:

[SWS_Ifl_00005] Definition of datatype Ifl_DPResultF32_Type ⌈

Name	Ifl_DPResultF32_Type
Kind	Structure
Elements	Index

▽

△

	Type	uint32
	Comment	Data point index
	Ratio	
	Type	float32
	Comment	Data point ratio
Description	Structure used for data point search for index and ratio	
Available via	Ifl.h	

⌋

Note: Ifl_DPResultF32_Type structure shall not be read/write/modified by the user directly. Only Ifl routines shall have access to this structure.

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Distributed data point search and interpolation

In this interpolation method data point search (e.g. index and ratio) is calculated using routine Ifl_DPSearch_f32 which returns result structure Ifl_DPResultF32_Type. It contains index and ratio information. This result can be used by curve interpolation and map interpolation.

8.3.1.1 Data Point Search

[SWS_Ifl_00010] Definition of API function Ifl_DPSearch_f32 ⌈

Service Name	Ifl_DPSearch_f32	

Syntax	`void Ifl_DPSearch_f32 (`
	`  Ifl_DPResultF32_Type* dpResult,`
	`  float32 Xin,`
	`  uint32 N,`
	`  const float32* X_array`
	`)`	

Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples; Valid range N>0
	X_array	Pointer to distribution array
Parameters (inout)	None	
Parameters (out)	dpResult	Pointer to the result structure

▽

△

Return value	None
Description	This routine searches the position of input Xin within the given distribution array X_array, and returns index and ratio necessary for interpolation.
Available via	Ifl.h

⌋

[SWS_Ifl_00011] ⌈Returned Index shall be the lowest index for which (X_array[index] < Xin < X_array[index + 1]).

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

dpResult ->Index = index

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

Example:

For a given array float32 X[] ={0.0,10.0,26.0,36.0,64.0}:

If Xin = 20.0 then

dpResult ->Index = 1

dpResult ->Ratio = (20.0 - 10.0) / (26.0 - 10.0) = 0.625

[SWS_Ifl_00012] ⌈If the input value matches with one of the distribution array values, then return respective index and ratio as 0.0.

If Input Xin == X_array[index], then

dpResult ->Index = index (Index of the set point)

dpResult ->Ratio = 0.0⌋

[SWS_Ifl_00013] ⌈If (Xin < X_array[0]), then return first index of an array and ratio = 0.0

dpResult ->Index = 0

dpResult ->Ratio = 0.0⌋

[SWS_Ifl_00014] ⌈If (Xin > X_array[N-1]), then return last index of an array and ratio = 0.0

dpResult ->Index = N - 1

dpResult ->Ratio = 0.0⌋

[SWS_Ifl_00016] ⌈If X_array[Index+1] == X_array[Index], then the Ratio shall be zero.

dpResult->Ratio = 0.0⌋

[SWS_Ifl_00017] ⌈This routine returns index and ratio through the structure of type Ifl_DPResultF32_Type⌋

8.3.1.2 Curve interpolation

[SWS_Ifl_00021] Definition of API function Ifl_IpoCur_f32 ⌈

Service Name	Ifl_IpoCur_f32	
Syntax	`float32 Ifl_IpoCur_f32 ( const Ifl_DPResultF32_Type* dpResult, const float32* Val_array )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResult	Data point search result
	Val_array	Pointer to the result distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the Interpolation
Description	Based on searched index and ratio information, this routine returns the interpolation result for a curve.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00022] ⌈index = dPResult->Index

if dPResult->Ratio == 0.0

Result = Val_array[index]

else

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

Note: Do not call this routine until you have searched the axis using the Ifl_DPSearch routine. Only then it is ensured that the search result (Ifl_DPResultF32_Type) contains valid data and is not used uninitialized.

8.3.1.3 Map interpolation

[SWS_Ifl_00025] Definition of API function Ifl_IpoMap_f32 ⌈

Service Name	Ifl_IpoMap_f32	
Syntax	`float32 Ifl_IpoMap_f32 ( const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, uint32 num_value, const float32* Val_array )`	
Service ID [hex]	0x05	
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 result distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the Interpolation
Description	Based on searched indices and ratios information using the Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for a map.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00026] ⌈Based on searched indices and ratios information using the Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for map.

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

if (dpResultX->Ratio == 0)

if (dpResultY->Ratio == 0)

Result = Val_array [BaseIndex]

else

LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

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

else

if (dpResultY->Ratio == 0)

LowerX = Val_array [BaseIndex]

UpperX = Val_array [BaseIndex + num_value]

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

else

LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

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

LowerY = Val_array [BaseIndex + num_value]

UpperY = Val_array [BaseIndex + num_value + 1]

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

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

Note: Do not call this routine until you have searched the axis using the Ifl_DPSearch routine. Only then it is ensured that the search result (Ifl_DPResultF32_Type) contains valid data and is not used uninitialized.

8.3.1.4 Single point interpolation

[SWS_Ifl_00030] Definition of API function Ifl_Interpolate_f32 ⌈

Service Name	Ifl_Interpolate_f32	

Syntax	`float32 Ifl_Interpolate_f32 (`
	`  float32 Value1,`
	`  float32 Value2,`
	`  float32 Coef`
	`)`	

Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Value1	First value to be used in the interpolation.
	Value2	Second value to be used in the interpolation.
	Coef	Interpolation coefficient.
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Interpolated value
Description	Returns the result of a linear interpolation between two values based on a coefficient.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00031] ⌈Result = Value1 + (Coef * (Value2 - Value1))⌋

8.3.2 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.

8.3.2.1 Integrated curve interpolation

[SWS_Ifl_00035] Definition of API function Ifl_IntIpoCur_f32_f32 ⌈

Service Name	Ifl_IntIpoCur_f32_f32	
Syntax	`float32 Ifl_IntIpoCur_f32_f32 ( float32 Xin, uint32 N, const float32* X_array, const float32* Val_array )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value
	N	Number of samples; Valid range N>0
	X_array	Pointer to X distribution
	Val_array	Pointer to Y values
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the Interpolation
Description	This routine returns the interpolation result of a curve at position Xin.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00036] ⌈index = minimum value of integer index if (X_array[index] < Xin < X_array[index+1])

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

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

[SWS_Ifl_00037] ⌈If the input value matches with one of the distribution array values, then result will be the respective Y array element indicated by the index.

If (Xin == X_array[index]),

Result = Val_array[index]⌋

[SWS_Ifl_00038] ⌈If Xin < X_array[0], then

Result = Val_array[0]⌋

[SWS_Ifl_00039] ⌈If Xin > X_array[N-1], then

Result = Val_array[N-1]⌋

8.3.2.2 Integrated map interpolation

[SWS_Ifl_00041] Definition of API function Ifl_IntIpoMap_f32f32_f32 ⌈

Service Name	Ifl_IntIpoMap_f32f32_f32	
Syntax	`float32 Ifl_IntIpoMap_f32f32_f32 ( float32 Xin, float32 Yin, uint32 Nx, uint32 Ny, const float32* X_array, const float32* Y_array, const float32* Val_array )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Xin	Input value for X axis
	Yin	Input value for Y axis
	Nx	Number of X axis intervals; Valid range Nx>0
	Ny	Number of Y axis intervals; Valid range Ny>0
	X_array	Pointer to the X axis distribution array
	Y_array	Pointer to the Y axis distribution array
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the Map Interpolation
Description	This routine returns the interpolation result of a map at position Xin, Yin.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00042] ⌈indexX = minimum value of index if (X_array[indexX] < Xin < X_array[indexX+1])

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

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

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

BaseIndex = IndexX * Ny + indexY

LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

LowerX = LowerY + (UpperY - LowerY) * RatioY

LowerY = Val_array [BaseIndex + Ny]

UpperY = Val_array [BaseIndex + Ny + 1]

UpperX = LowerY + (UpperY - LowerY) * RatioY

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

[SWS_Ifl_00043] ⌈If (Xin == X_array[indexX]) and (Y_array[indexY] < Yin < Y_array[indexY+1])

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

[SWS_Ifl_00044] ⌈If (Yin == Y_array[indexY]) and (X_array[indexX] < Xin < X_array[indexX+1])

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

[SWS_Ifl_00045] ⌈If (Xin == X_array[indexX]) and (Yin == Y_array[indexY])

Result = Val_array [BaseIndex]⌋

[SWS_Ifl_00046] ⌈If Xin < X_array[0], then

indexX = 0,

RatioX = 0.0⌋

[SWS_Ifl_00047] ⌈If Xin > X_array[Nx-1], then

indexX = Nx - 1,

RatioX = 0.0⌋

[SWS_Ifl_00048] ⌈If Yin < Y_array[0], then

indexY = 0,

RatioY = 0.0⌋

[SWS_Ifl_00049] ⌈If Yin > Y_array[Ny-1], then

indexY = Ny - 1,

RatioY = 0.0⌋

8.3.2.3 Cuboid 3D interpolation

[SWS_Ifl_91000] Definition of API function Ifl_IpoCub_f32 ⌈

Service Name	Ifl_IpoCub_f32	

Syntax	`float32 Ifl_IpoCub_f32 (`
	`  const Ifl_DPResultF32_Type* dpResultX,`
	`  const Ifl_DPResultF32_Type* dpResultY,`
	`  const Ifl_DPResultF32_Type* dpResultZ,`
	`  uint16 num_x,`
	`  uint16 num_y,`
	`  const float32* Val_array`
	`)`	

Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResultX	Data point search result for X axis
	dpResultY	Data point search result for Y axis
	dpResultZ	Data point search result for Z axis
	num_x	Number of X axis points
	num_y	Number of Y axis points
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the interpolation
Description	Based on searched indices and ratios information using the relevant Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for a 3D cuboid.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00226] ⌈Based on searched indices and ratios information using the Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for 3D cuboids.

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

Implementation:

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

num_slice = num_x * num_y

if(dpResultZ->Ratio==0.0)

Result=Ifl_IpoMap_f32 (dpResultX, dpResultY, num_y, Val_array[num_slice * dpResultZ->Index])

else

LowerXY=Ifl_IpoMap_f32 (dpResultX, dpResultY, num_y, Val_array[num_slice * dpResultZ ->Index])

UpperXY=Ifl_IpoMap_f32 (dpResultX, dpResultY, num_y, Val_array[num_slice * dpResultZ ->Index + 1])

Result=Ifl_Interpolate_f32 (LowerXY, UpperXY, dpResultZ->Ratio)⌋

8.3.2.4 Mixed type interpolation of integer curve

[SWS_Ifl_91001] Definition of API function Ifl_IpoCur_<InTypeMn>_f32 ⌈

Service Name	Ifl_IpoCur_<InTypeMn>_f32	
Syntax	`float32 Ifl_IpoCur_<InTypeMn>_f32 ( const Ifl_DPResultF32_Type* dpResult, const <InType>* Val_array )`	
Service ID [hex]	0x13 to 0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResult	Data point search result
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the interpolation
Description	Based on searched indices and ratios information using the relevant Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for a curve.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00228] ⌈index=dpResult->Index

if dpResult->Ratio==0.0

Result=Val_array[index]

else

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

Here is the list of implemented routines:

[SWS_Ifl_00229] ⌈

Routine ID[hex]	Routine prototype
0x013	float32 Ifl_IpoCur_u8_f32 (const Ifl_DPResultF32_Type* dpResult, const uint8* Val_array)
0x014	float32 Ifl_IpoCur_u16_f32 (const Ifl_DPResultF32_Type* dpResult, const uint16* Val_array)
0x015	float32 Ifl_IpoCur_s8_f32 (const Ifl_DPResultF32_Type* dpResult, const sint8* Val_array)
0x016	float32 Ifl_IpoCur_s16_f32 (const Ifl_DPResultF32_Type* dpResult, const sint16* Val_array)

⌋

8.3.2.5 Mixed type interpolation of integer map

[SWS_Ifl_91002] Definition of API function Ifl_IpoMap_<InTypeMn>_f32 ⌈

Service Name	Ifl_IpoMap_<InTypeMn>_f32	

Syntax	`float32 Ifl_IpoMap_<InTypeMn>_f32 (`
	`  const Ifl_DPResultF32_Type* dpResultX,`
	`  const Ifl_DPResultF32_Type* dpResultY,`
	`  uint32 num_value,`
	`  const <InType>* Val_array`
	`)`	

Service ID [hex]	0x18 to 0x1b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResultX	Data point search result for X axis
	dpResultY	Data point search result for Y axis
	num_value	Number of Y axis points
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the interpolation
Description	Based on searched indices and ratios information using the relevant Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for a map.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00231] ⌈Based on searched indices and ratios information using the Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for map.

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

if (dpResultX->Ratio == 0.0)

if (dpResultY->Ratio == 0.0)

Result = Val_array [BaseIndex]

else

LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

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

else

if (dpResultY->Ratio == 0.0)

LowerX = Val_array [BaseIndex]

UpperX = Val_array [BaseIndex + num_value]

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

else

LowerY = Val_array [BaseIndex]

UpperY = Val_array [BaseIndex + 1]

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

LowerY = Val_array [BaseIndex + num_value]

UpperY = Val_array [BaseIndex + num_value + 1]

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

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

Here is the list of implemented routines.

[SWS_Ifl_00232] ⌈

Routine ID[hex]	Routine prototype
0x018	float32 Ifl_IpoMap_u8_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, uint32 num_value, const uint8* Val_array)
0x019	float32 Ifl_IpoMap_u16_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, uint32 num_value, const uint16* Val_array)
0x01A	float32 Ifl_IpoMap_s8_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, uint32 num_value, const sint8* Val_array)
0x01B	float32 Ifl_IpoMap_s16_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, uint32 num_value, const sint16* Val_array)

⌋

8.3.2.6 Mixed type interpolation of integer 3D Cuboid

[SWS_Ifl_91003] Definition of API function Ifl_IpoCub_<InTypeMn>_f32 ⌈

Service Name	Ifl_IpoCub_<InTypeMn>_f32	
Syntax	`float32 Ifl_IpoCub_<InTypeMn>_f32 ( const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, const Ifl_DPResultF32_Type* dpResultZ, uint16 num_x, uint16 num_y, const <InType>* Val_array )`	
Service ID [hex]	0x1C to 0x1F	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	dpResultX	Data point search result for X axis
	dpResultY	Data point search result for Y axis

▽

△

	dpResultZ	Data point search result for Z axis
	num_x	Number of X axis points
	num_y	Number of Y axis points
	Val_array	Pointer to the result axis distribution array
Parameters (inout)	None	
Parameters (out)	None	
Return value	float32	Result of the interpolation
Description	Based on searched indices and ratios information using the relevant Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for a 3D cuboid.	
Available via	Ifl.h	

⌋

[SWS_Ifl_00234] ⌈Based on searched indices and ratios information using the Ifl_DPSearch_f32 routine, this routine calculates and returns the interpolation result for 3D cuboids.

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

Implementation:

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

num_slice = num_x*num_y

if (dpResultZ->Ratio == 0.0)

Result = Ifl_IpoMap_<InTypeMn>_f32 (dpResultX, dpResultY, num_y, Val_array[num_slice*dpResultZ->Index])

else

LowerXY = Ifl_IpoMap_<InTypeMn>_f32 (dpResultX, dpResultY, num_y, Val_array[num_slice*dpResultZ->Index])

UpperXY = Ifl_IpoMap_<InTypeMn>_f32 (dpResultX, dpResultY, num_y, Val_array[num_slice*dpResultZ->Index + 1])

Result = Ifl_Interpolate_f32 (LowerXY, UpperXY, dpResultZ->Ratio)⌋

Here is the list of implemented routines.

[SWS_Ifl_00235] ⌈

Routine ID[hex]	Routine prototype
0x01C	float32 Ifl_IpoCub_u8_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, const Ifl_DPResultF32_Type* dpResultZ, uint16 num_x, uint16 num_y, const uint8* Val_array)

▽

△

Routine ID[hex]	Routine prototype
0x01D	float32 Ifl_lpoCub_u16_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, const Ifl_DPResultF32_Type* dpResultZ, uint16 num_x, uint16 num_y, const uint16* Val_array)
0x01E	float32 Ifl_lpoCub_s8_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, const Ifl_DPResultF32_Type* dpResultZ, uint16 num_x, uint16 num_y, const sint8* Val_array)
0x01F	float32 Ifl_lpoCub_s16_f32 (const Ifl_DPResultF32_Type* dpResultX, const Ifl_DPResultF32_Type* dpResultY, const Ifl_DPResultF32_Type* dpResultZ, uint16 num_x, uint16 num_y, const sint16* Val_array)

⌋

8.4 Scheduled routines

The Ifl library does not have scheduled routines.

8.5 Expected interfaces

None.

8.5.1 Mandatory interfaces

None.

8.5.2 Optional interfaces

None.

8.5.3 Configurable interfaces

None.

9 Sequence diagrams

Not applicable.

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module Ifl.

Chapter 10.3 specifies published information of the module Ifl.

10.1 How to read this chapter

For details refer to [1] Chapter 10.1 "Introduction to configuration specification".

10.2 Containers and configuration parameters

The Ifl Library has no standardized configuration options.

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

10.3 Published Information

For details refer to [1] Chapter 10.3 "Published Information".

A Not applicable requirements

[SWS_Ifl_00224]

Upstream requirements: SRS_BSW_00448

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R23-11

B.1.1 Added Specification Items in R23-11

none

B.1.2 Changed Specification Items in R23-11

Number	Heading
[SWS_Ifl_00005]	Definition of datatype Ifl_DPResultF32_Type
[SWS_Ifl_00021]	Definition of API function Ifl_IpoCur_f32
[SWS_Ifl_00025]	Definition of API function Ifl_IpoMap_f32
[SWS_Ifl_00030]	Definition of API function Ifl_Interpolate_f32
[SWS_Ifl_00035]	Definition of API function Ifl_IntIpoCur_f32_f32
[SWS_Ifl_00041]	Definition of API function Ifl_IntIpoMap_f32f32_f32
[SWS_Ifl_00170]	
[SWS_Ifl_91000]	Definition of API function Ifl_IpoCub_f32
[SWS_Ifl_91001]	Definition of API function Ifl_IpoCur_<InTypeMn>_f32
[SWS_Ifl_91002]	Definition of API function Ifl_IpoMap_<InTypeMn>_f32
[SWS_Ifl_91003]	Definition of API function Ifl_IpoCub_<InTypeMn>_f32

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

B.1.3 Deleted Specification Items in R23-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R24-11

B.2.1 Added Specification Items in R24-11

Number	Heading
[SWS_Ifl_91004]	Definition of imported datatypes of module Ifl

Table B.2: Added Specification Items in R24-11

B.2.2 Changed Specification Items in R24-11

Number	Heading
[SWS_Ifl_00214]	

Table B.3: Changed Specification Items in R24-11

B.2.3 Deleted Specification Items in R24-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R25-11

B.3.1 Added Specification Items in R25-11

none

B.3.2 Changed Specification Items in R25-11

none

B.3.3 Deleted Specification Items in R25-11

none

B.4 Traceable item history of this document according to AUTOSAR Release R26-11

B.4.1 Added Specification Items in R26-11

none

B.4.2 Changed Specification Items in R26-11

Number	Heading
[SWS_Ifl_00001]	
[SWS_Ifl_00010]	Definition of API function Ifl_DPSearch_f32
[SWS_Ifl_00035]	Definition of API function Ifl_IntIpoCur_f32_f32
[SWS_Ifl_00041]	Definition of API function Ifl_IntIpoMap_f32f32_f32
[SWS_Ifl_91004]	Definition of imported datatypes of module Ifl

Table B.4: Changed Specification Items in R26-11

B.4.3 Deleted Specification Items in R26-11

Number	Heading
[SWS_Ifl_00006]	
[SWS_Ifl_00015]	
[SWS_Ifl_00040]	
[SWS_Ifl_00050]	
[SWS_Ifl_00180]	
[SWS_Ifl_00181]	
[SWS_Ifl_00200]	
[SWS_Ifl_00201]	
[SWS_Ifl_00206]	
[SWS_Ifl_00207]	
[SWS_Ifl_00208]	
[SWS_Ifl_00209]	
[SWS_Ifl_00210]	
[SWS_Ifl_00211]	
[SWS_Ifl_00212]	
[SWS_Ifl_00213]	
[SWS_Ifl_00214]	
[SWS_Ifl_00215]	Definition of API function Ifl_GetVersionInfo

▽

△

Number	Heading
[SWS_Ifl_00216]	
[SWS_Ifl_00217]	
[SWS_Ifl_00218]	
[SWS_Ifl_00219]	
[SWS_Ifl_00221]	
[SWS_Ifl_00223]	

Table B.5: Deleted Specification Items in R26-11

B.4.4 Added Constraints in R26-11

Number	Heading
[SWS_Ifl_CONSTR_06700]	Constraint on input of Ifl routines

Table B.6: Added Constraints in R26-11

B.4.5 Changed Constraints in R26-11

none

B.4.6 Deleted Constraints in R26-11

none