Document Title	Supplementary material of general blueprints for AUTOSAR
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	682

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History			
Date	Release	Changed by	Description
2018-10-31	4.4.0	AUTOSAR Release Management	• Multi dimensional ValueBlock • Include Physical Dimensions and Units
2017-12-08	4.3.1	AUTOSAR Release Management	• Extend description of FIX_AXIS • Include Mentioned Class Tables
2016-11-30	4.3.0	AUTOSAR Release Management	• Extended Blueprint artifacts • Composition of Blueprint artifacts • Include Test Case Blueprint artifacts
2015-07-31	4.2.2	AUTOSAR Release Management	• Initial Release

Table of Contents

References

[1] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[2] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate

[3] Specification of Floating Point Interpolation Routines
AUTOSAR_SWS_IFLLibrary

[4] Specification of Fixed Point Interpolation Routines
AUTOSAR_SWS_IFXLibrary

[5] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping

[6] Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager

[7] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[8] XML Specification of Application Interfaces
AUTOSAR_MOD_AISpecification

[9] Predefined Names in AUTOSAR
AUTOSAR_TR_PredefinedNames

[10] SW-C and System Modeling Guide
AUTOSAR_TR_SWCModelingGuide

1 Introduction

This technical report provides additional information to existing blueprints.

2 Overview General Blueprints

The General Blueprints are provided in auxiliary package AUTOSAR_MOD_GeneralBlueprints. Currently it contains

- AUTOSAR_MOD_BSWServiceInterfaces_Blueprint
- AUTOSAR_MOD_BswModuleEntrys_Blueprint
- AUTOSAR_MOD_BswServiceInterfacesMapping_Blueprint
- AUTOSAR_MOD_BswServiceDataTypes_Blueprint
- AUTOSAR_MOD_CommonDataTypes_Blueprint
- AUTOSAR_MOD_BswDataTypes_Blueprint
- AUTOSAR_MOD_IFL_RecordLayout_Blueprint
- AUTOSAR_MOD_IFX_RecordLayout_Blueprint
- AUTOSAR_MOD_Cube_SwRecordLayout_Blueprint
- AUTOSAR_MOD_ValBlk_SwRecordLayout_Blueprint
- AUTOSAR_MOD_MemoryMapping_SwAddrMethods_Blueprint
- AUTOSAR_MOD_SWCServiceRelatedInterfaces_Blueprint
- AUTOSAR_MOD_PhyiscalDimensions_Blueprint
- AUTOSAR_MOD_Units_Blueprint
- AUTOSAR_TR_PredefinedNames_Blueprint
- AUTOSAR_TP_FormulaLanguage_TestCase_Blueprint.

2.1 AUTOSAR_MOD_BSWServiceInterfaces_Blueprint

The AUTOSAR_MOD_BSWServiceInterfaces_Blueprint provides for a variety of BSW modules blueprinted specification of their Standardized AUTOSAR Interfaces which consists of `ClientServerInterface`s, `ModeDeclarationGroup`s, `ModeSwitchInterface`s, `SenderReceiverInterface`s and `ServiceSwComponentType`s. Inside these blueprints also the `BlueprintPolicy` is used. A detailed description of the `BlueprintPolicy` is given in [1]. The ARXML file is generated based on the BSW UML Model.

2.2 AUTOSAR_MOD_BswModuleEntrys_Blueprint

The AUTOSAR_MOD_BswModuleEntrys_Blueprint provides blueprints of the BswModuleDescriptions and BswModuleEntrys based on [2].

2.3 AUTOSAR_MOD_BswServiceInterfacesMapping_Blueprint

The AUTOSAR_MOD_BswServiceInterfacesMapping_Blueprint provides blueprints of the mapping per client-server-interface ClientServerInterfaceToBswModuleEntryBlueprintMappings based on [1].

2.4 AUTOSAR_MOD_BswServiceDataTypes_Blueprint

The AUTOSAR_MOD_BswServiceDataTypes_Blueprint provides blueprints of the DataConstrs, CompuMethods and ImplementationDataTypes for Services based on [1].

2.5 AUTOSAR_MOD_CommonDataTypes_Blueprint

The AUTOSAR_MOD_CommonDataTypes_Blueprint provides blueprints of the BaseTypes, CompuMethods and ImplementationDataTypes for Platform, Standard and General Definitions based on [1].

2.6 AUTOSAR_MOD_BswDataTypes_Blueprint

The AUTOSAR_MOD_BswDataTypes_Blueprint provides blueprints of the DataConstrs, CompuMethods and ImplementationDataTypes for BSW based on [1].

2.7 AUTOSAR_MOD_IFL_RecordLayout_Blueprint

The AUTOSAR_MOD_IFL_RecordLayout_Blueprint provides blueprints of the InterpolationRoutineMappingSets and SwRecordLayouts based on [3].

2.8 AUTOSAR_MOD_IFX_RecordLayout_Blueprint

The AUTOSAR_MOD_IFX_RecordLayout_Blueprint provides blueprints of the InterpolationRoutineMappingSets and SwRecordLayouts based on [4].

2.9 AUTOSAR_MOD_Cube_SwRecordLayout_Blueprint

The AUTOSAR_MOD_Cube_RecordLayout_Blueprint provides blueprints of SwRecordLayouts for cuboids.

2.10 AUTOSAR_MOD_ValBlk_SwRecordLayout_Blueprint

The AUTOSAR_MOD_ValBlk_SwRecordLayout_Blueprint provides blueprints of SwRecordLayouts for Value and Valueblocks.

2.11 AUTOSAR_MOD_MemoryMapping_SwAddrMethods_Blueprint

The AUTOSAR_MOD_MemoryMapping_SwAddrMethods_Blueprint provides blueprints of the SwAddrMethods based on [5].

2.12 AUTOSAR_MOD_SWCServiceRelatedInterfaces_Blueprint

The AUTOSAR_MOD_SWCServiceRelatedInterfaces_Blueprint provides blueprints of the ClientServerInterfaces derived from the Standardized AUTOSAR Interfaces of the NVRAM Manager [6]. Those ClientServerInterfaces are used for NvBlockSwComponentTypes as described in [7].

2.13 AUTOSAR_MOD_PhyiscalDimensions_Blueprint

The AUTOSAR_MOD_PhyiscalDimensions_Blueprint provides a collection of blueprints of Standardized AUTOSAR Physical Dimensions definitions which are used for Unit definitions as, e.g. in AUTOSAR_MOD_Units_Blueprint.

2.14 AUTOSAR_MOD_Units_Blueprint

The AUTOSAR_MOD_Units_Blueprint provides a collection of blueprints of Standardized AUTOSAR Units definitions which are used, e.g. for interface definitions as described in [8].

2.15 AUTOSAR_TR_PredefinedNames_Blueprint

The AUTOSAR_TR_PredefinedNames_Blueprint provides various predefined names used in AUTOSAR models and documents [9]. They are available as blueprints based on AUTOSAR XML model. In this model, the predefined names are represented as Keywords according to [1].

2.16 AUTOSAR_TP_FormulaLanguage_TestCase_Blueprint

The AUTOSAR_TP_FormulaLanguage_TestCase_Blueprint provides various predefined test cases to validate the formula language expressions.

2.17 Composition of Blueprints

The blueprints are composed by different elements which can be applied use case specific. Table 2.1 provides an overview of the elements decribed by blueprints.

	BswModuleDescription	BswModuleEntry	ClientServerInterface	ClientServerInterfaceToBswModuleEntryBlueprintMapping	CompuMethod	DataConstr	ImplementationDataType	ModeDeclarationGroup	ModeSwitchInterface	SenderReceiverInterface	ServiceSwComponentType
AUTOSAR_MOD_BswServiceInterfaces_Blueprint			x					x	x	x	x
AUTOSAR_MOD_BswModuleEntrys_Blueprint	x	x									
AUTOSAR_MOD_BswServiceInterfacesMapping_Blueprint				x							
AUTOSAR_MOD_SWCServiceRelatedInterfaces_Blueprint			x								
AUTOSAR_MOD_BswServiceDataTypes_Blueprint					x	x	x				
AUTOSAR_MOD_CommonDataTypes_Blueprint					x	x	x				
AUTOSAR_MOD_BswDataTypes_Blueprint					x	x	x				

Table 2.1: Overview Blueprint Elements

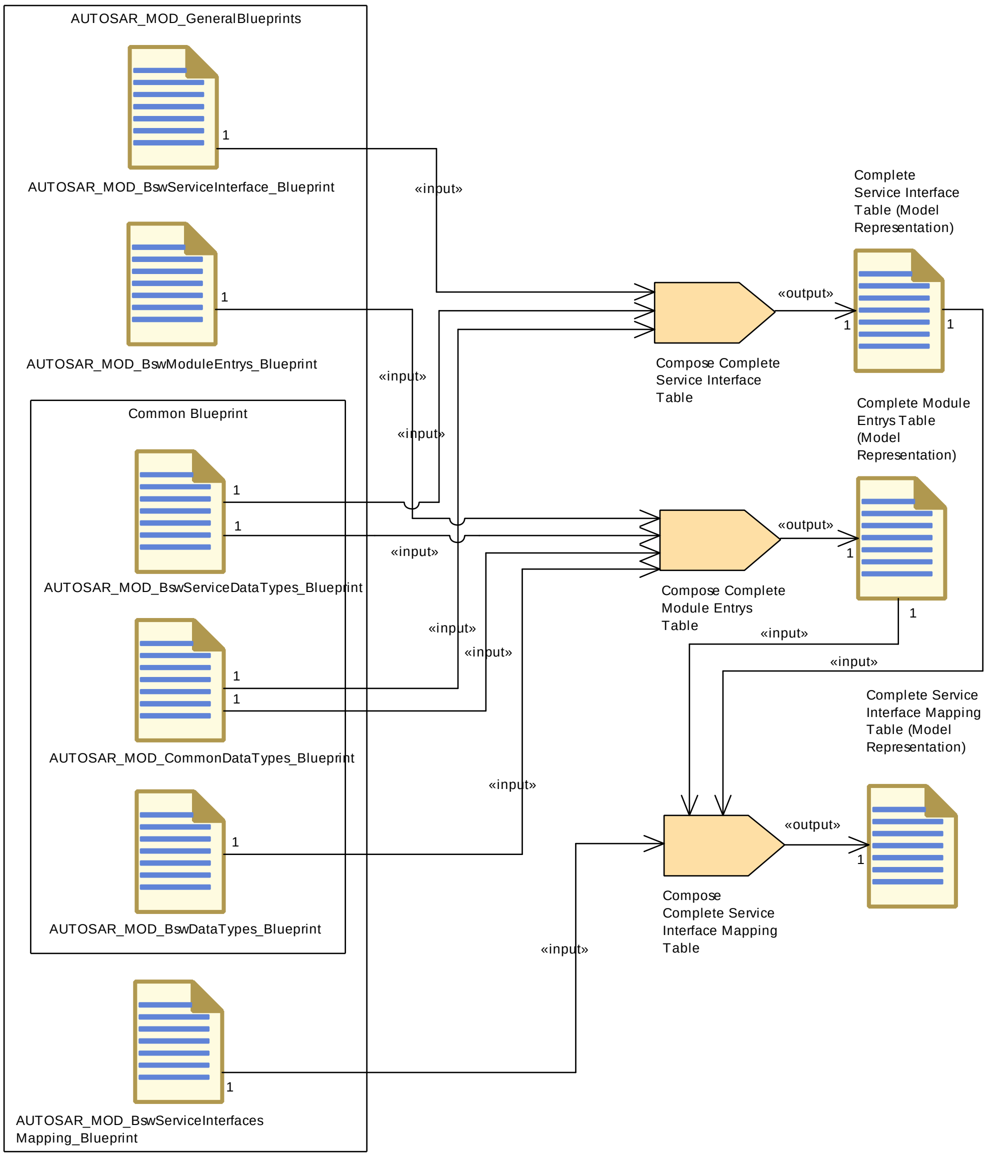

Figure 2.1: Composition of different tables based on blueprints

3 Visualization of SwRecordLayouts

The visualization of the SwRecordLayouts follows a unique representation. The used graphical elements are illustrated in figure 3.1.

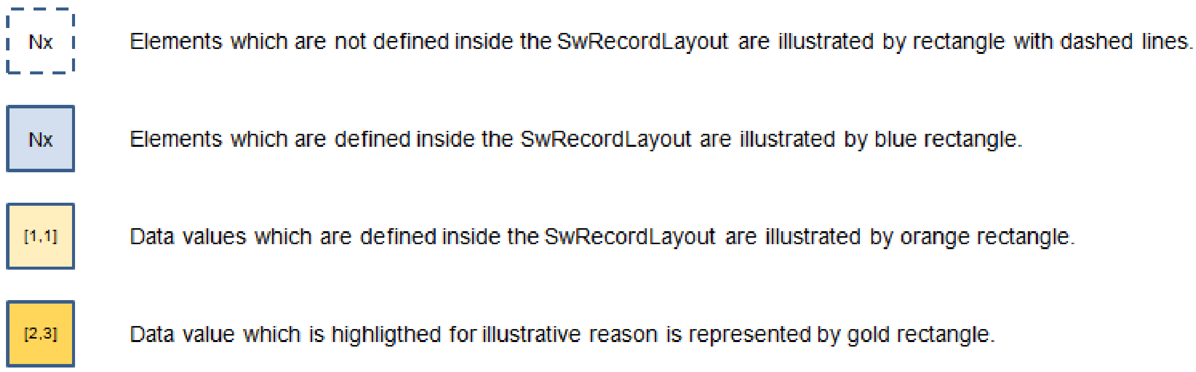

Figure 3.1: Legend of used graphical elements

The logical view represents the definitive elements as number of sampling points, axis elements and data values. The data values are arranged according to the applicable dimension. Curves are visualized one dimensional (e.g. one column, see figure 3.7). Maps are visualized in a two dimensional matrix, see figure 3.19).

The memory representation illustrates the storage of values in linear memory. In case the SwRecordLayout defines also the elements as number of sampling points and axis elements (blue rectangle) the memory representation starts with these. Subsequently the storage of data values follows (orange rectangle). In case the SwRecordLayout does not define the elements as number of sampling points and axis elements the memory representation starts with the storage of data values.

The ARXML representation lists the significant part of the ARXML file describing the SwRecordLayout.

3.1 Record Layout: Distr

This chapter describes the record layout for distributed data point search. This means that this SwRecordLayout describes only the number of sampling points and the axis values. It does not describe any values. In this case several curves can used the same axis (distributed data points), see figure 3.3.

Logical view:

The figure 3.2 illustrates the logical view of the SwRecordLayout Distr. Nx represents the standardized value of SwRecordLayoutV.swRecordLayoutVProp and is documented in [TPS_SWCT_01489]. In the scope of this example the value COUNT is used.

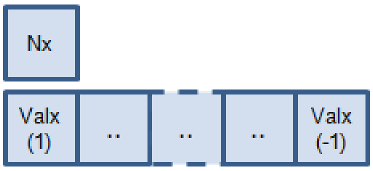

Figure 3.2: Distr Logical View

Memory representation:

Due to the fact that the number of sampling points and the axis values (content of this record layout definition) are not stored in memory without any curve definition no memory representation is defined.

ARXML representation:

Extract of the record layout Distr_s16 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.1: Record Layout: Distr_s16 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: Distr_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">Distr_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">N</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
         SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">X</SHORT-LABEL>
      <CATEGORY>INDEX_INCR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
           SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

Different curves can be assigned to one distribution.

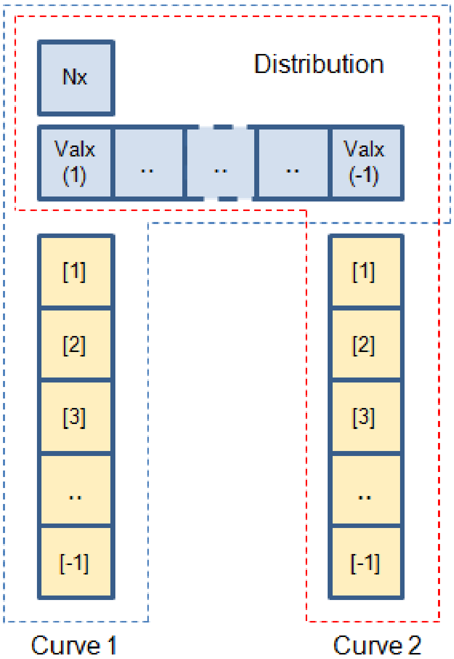

Figure 3.3: Curves assigned to Distribution Logical View

Both curves use the same distribution (AXIS 1), e.g. illustrated by the purple-dotted lines (x value 25) with different values (AXIS 0), curve values (y values 65 and 15).

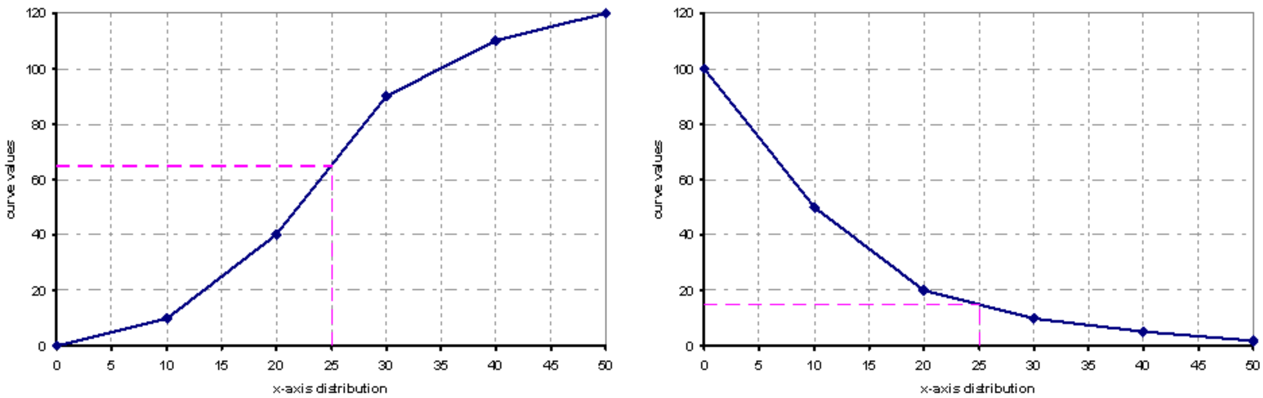

Figure 3.4: Curves assigned to same Distribution

3.2 Curves

3.2.1 Record Layout: Cur

This chapter describes the record layout for a curve.

Logical view:

The figure 3.5 illustrates the logical view of the SwRecordLayout Cur. The number of sampling points (Nx) and the elements of [AXIS 1] are not defined inside this SwRecordLayout. The SwRecordLayoutGroup with the shortLabel Val is shown in the lower part.

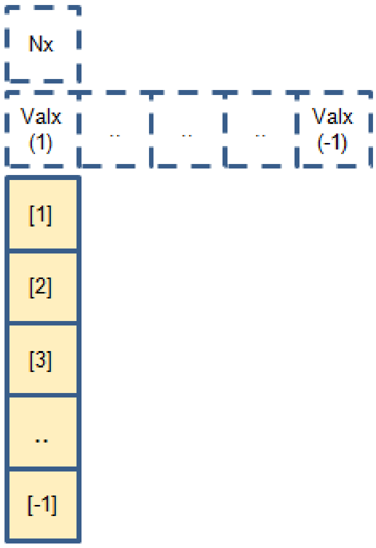

Figure 3.5: Cur Logical View

Memory representation:

The SwRecordLayout Cur illustrated in figure 3.5 is stored as follows:

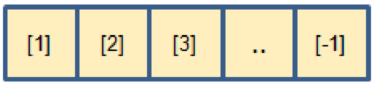

Figure 3.6: Cur Memory Representation

This means that the data is stored in direction of columns ([1],[2],[3], ...).

ARXML representation:

Extract of the record layout Cur_s16 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.2: Record Layout: Cur_s16 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: Cur_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">Cur_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
    <CATEGORY>COLUMN_DIR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
```

```
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
         SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      <SW-RECORD-LAYOUT-V-INDEX>X</SW-RECORD-LAYOUT-V-INDEX>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

3.2.2 Record Layout: IntCur

This chapter describes the record layout for a curve with integrated data point search. This means that this SwRecordLayout represents a complete curve with number of sampling points, number of axis and values. It describes all elements of the curve.

Logical view:

The figure 3.7 illustrates the logical view of the SwRecordLayout IntCur. Nx represents the number of sampling points and is given by the standardized value of SwRecordLayoutV.swRecordLayoutVProp. In the scope of this example the value COUNT is used. The SwRecordLayoutGroup with the shortLabel Val is shown in the lower part. Its elements are indexed by [AXIS 1] from value (AXIS 1: = 1) to value (AXIS 1: = -1) there -1 gives the last value.

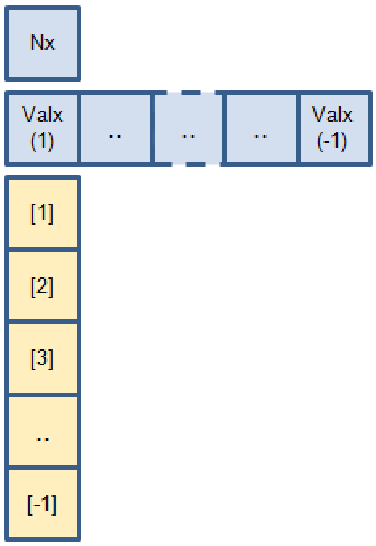

Figure 3.7: IntCur Logical View

Memory representation:

The SwRecordLayout IntCur illustrated in figure 3.7 is stored as follows:

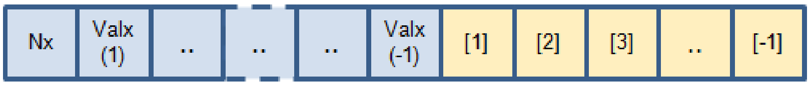

Figure 3.8: IntCur Memory Representation

This means that the data is stored in direction of columns ([1],[2],[3], ...).

ARXML representation:

Extract of the record layout IntCur_s16_s8 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.3: Record Layout: IntCur_s16_s8 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: IntCur_s16_s8 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">IntCur_s16_s8</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">N</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">X</SHORT-LABEL>
      <CATEGORY>INDEX_INCR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
            SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>0</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
            SwBaseTypes_Blueprint/sint8</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
        <SW-RECORD-LAYOUT-V-INDEX>X</SW-RECORD-LAYOUT-V-INDEX>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
```

```
</SW-RECORD-LAYOUT>
```

3.2.3 Record Layout: FixIntCur

This chapter describes the record layout for a curve with fixed axis points. Fixed axis exist in three categories: FIX_AXIS_PAR, FIX_AXIS_PAR_DIST and FIX_AXIS_PAR_LIST, see [TPS_SWCT_01748] in [7].

The number of sampling points (Nx), the Offset, the shift and the distance values are represented in the following chapters by these logical views:

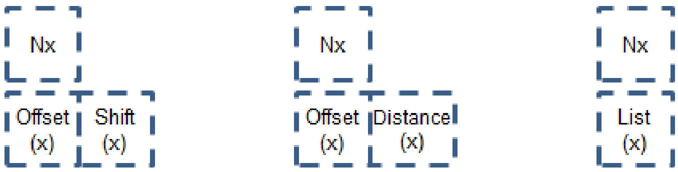

Figure 3.9: FIX_AXIS_PAR (left), FIX_AXIS_PAR_DIST (middle), FIX_AXIS_PAR_LIST (right)

These values are not defined inside `SwRecordLayout`s with fixed axis points.

Logical view:

The figure 3.10 illustrates the logical view of the `SwRecordLayout` FixIntCur. The `SwRecordLayoutGroup` with the `shortLabel` Val is shown in the lower part. Its elements are indexed by virtual [AXIS 1] which is fixed and of category FIX_AXIS_PAR and not defined inside this `SwRecordLayout`.

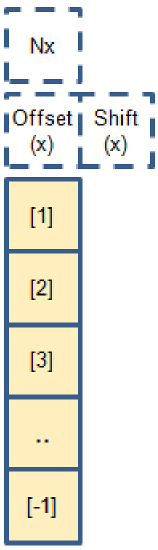

Figure 3.10: FixIntCur Logical View

Memory representation:

The SwRecordLayout FixIntCur illustrated in figure 3.10 is stored as follows:

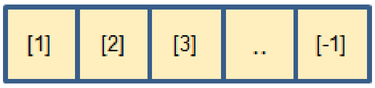

Figure 3.11: FixIntCur Memory Representation

This means that the data is stored in direction of columns ([1],[2],[3], ...).

ARXML representation:

Extract of the record layout FixIntCur_s16_s16 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.4: Record Layout: FixIntCur_s16_s16 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: FixIntCur_s16_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">FixIntCur_s16_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
    <CATEGORY>COLUMN_DIR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      <SW-RECORD-LAYOUT-V-INDEX>X</SW-RECORD-LAYOUT-V-INDEX>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

3.3 Maps

3.3.1 Definition of Indexing

To understand the visualization of SwRecordLayouts it is important to set-up a common understanding of the used indexing. There is the indexing used by matrix definition in linear algebra and by cartesian coordinate systems. In linear algebra a matrix A(m,n) is defined by the row index (m) and the column index (n).

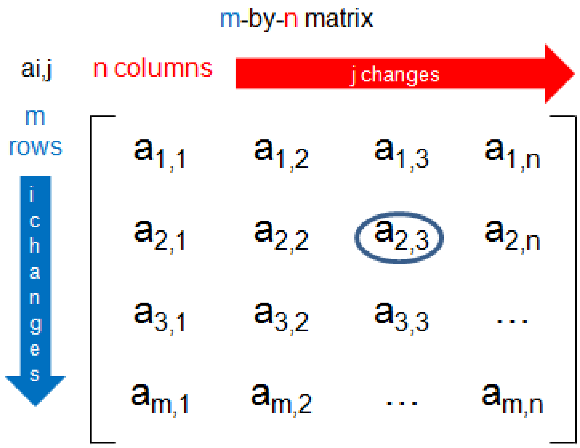

Figure 3.12: Linear Algebra Matrix

The cartesian coordinate system which is used by AUTOSAR and ASAM assigns AXIS 2 (AXIS_PTS_Y) to the row index (m) and AXIS 1 (AXIS_PTS_X) to the column index (n). This is the essential point in the transformation from indexing in matrix definition to the representation in cartesian coordinate system. The matrix element a(2,3) in figure 3.12 is represented in the cartesian coordinate system in figure 3.13 by (AXIS 1) x = 3 and (AXIS 2) y = 2.

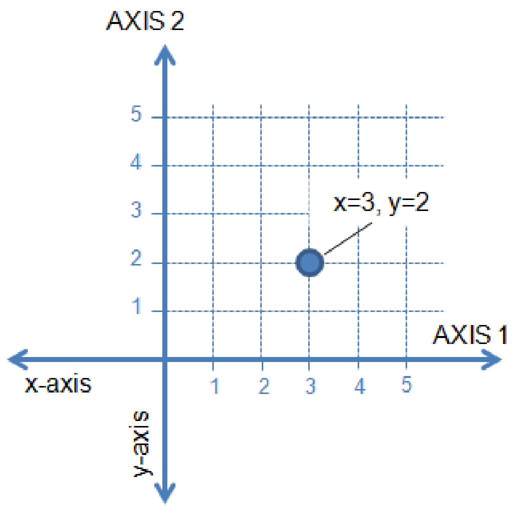

Figure 3.13: Cartesian Coordinate System

Based on this transformation definition the following visualization of `SwRecordLayouts` shall improve a better common understanding of the provided `SwRecordLayouts`.

3.3.2 Transform Logical View in Memory Representation

The logical view is represented by m-by-n matrix (two dimensional matrix) as described in 3.3.1.

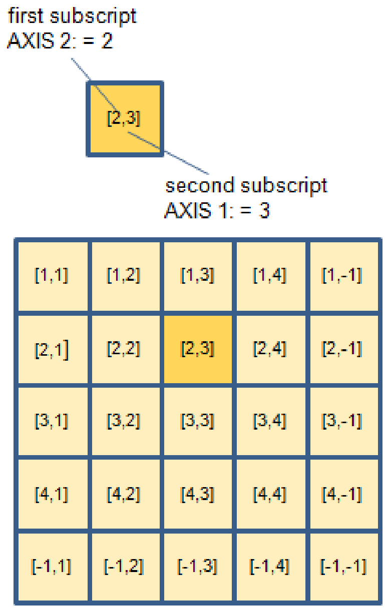

Figure 3.14: Matrix Representation

Each element of a matrix is denoted by an index with two subscripts [AXIS 2, AXIS 1]. For instance, [2,3] represents the element at the second row (AXIS 2) and third column (AXIS 1) of a matrix. The index of the matrix can be transformed to the memory representation in two different ways:

- storage of array values in column-major order in linear memory -> COLUMN_DIR
- storage of array values in row-major order in linear memory -> ROW_DIR

In column-major order[1], a multidimensional array in linear memory is organized such that columns are stored one after the other. The first element of the first column [1,1] is selected and then inside this column all elements will iterate up to the last element [-1,1] (indicated by the red arrow in figure 3.15). The last element is defined in `SwRecordLayout` by '-1'. Afterwards the first element of the second column [1,2] is selected and the iteration starts again as in the first column.

[1] The scientific programming language Fortran uses column-major ordering.

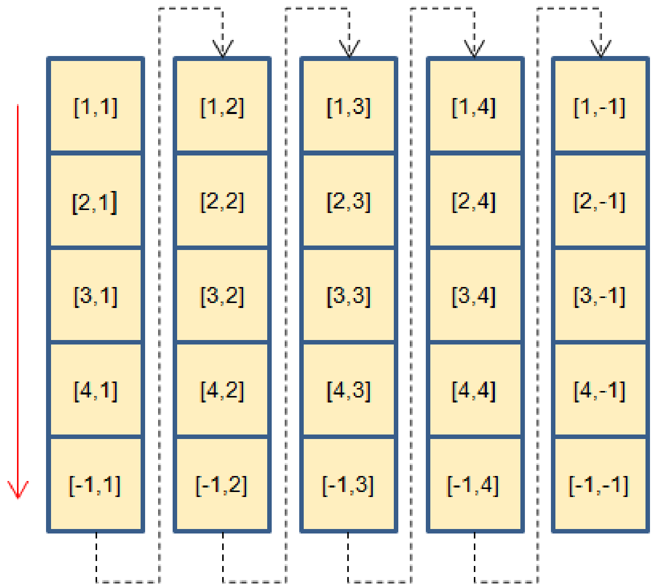

Figure 3.15: Transformation Matrix in column-major order

This listing illustrates two nested FOR-loops in case of column-major order whereas the outer loop iterates over AXIS 1 and the inner loop iterates over AXIS 2.

```
[
   (select row element; outer loop)
   iteration along row (AXIS 1 iterates, AXIS 2 is fixed !)
   start with first element (AXIS 1: = 1)
   [
      (select column element; inner loop)
      iteration along column (AXIS 2 iterates, AXIS 1 is fixed !)
      start with first element (AXIS 2: = 1)
      ...
      end with last element (AXIS 2: = -1)
   ]
   end with last element (AXIS 1: = -1)
]
```

In row-major order[2], a multidimensional array in linear memory is organized such that rows are stored one after the other. The first element of the first row [1,1] is selected and then inside this row all elements will iterate up to the last element [1,-1] (indicated by the blue arrow in figure 3.16). Afterwards the first element of the second row [2,1] is selected and the iteration starts again as in the first row.

[2]The C programming language uses row-major ordering.

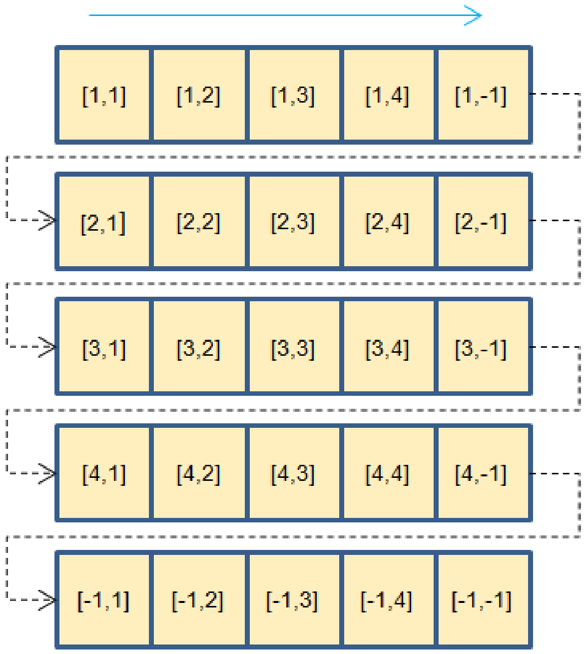

Figure 3.16: Transformation Matrix in row-major order

This listing illustrates two nested FOR-loops in case of row-major order whereas the outer loop iterates over AXIS 2 and the inner loop iterates over AXIS 1.

```
[
   (select column element; outer loop)
   iteration along column (AXIS 2 iterates, AXIS 1 is fixed !)
   start with first element (AXIS 2: = 1)
   [
      (select row element; inner loop)
      iteration along row (AXIS 1 iterates, AXIS 2 is fixed !)
      start with first element (AXIS 1: = 1)
      ...
      end with last element (AXIS 1: = -1)
   ]
   end with last element (AXIS 2: = -1)
]
```

3.3.3 Record Layout: Map

This chapter describes the record layout for a map.

Logical view:

The figure 3.17 illustrates the logical view of the `SwRecordLayout` Map. The number of sampling points (Nx, Ny) and the elements of [AXIS 2, AXIS 1] are not defined inside this `SwRecordLayout`. The `SwRecordLayoutGroup` with the `shortLabel` Val is shown in the lower part.

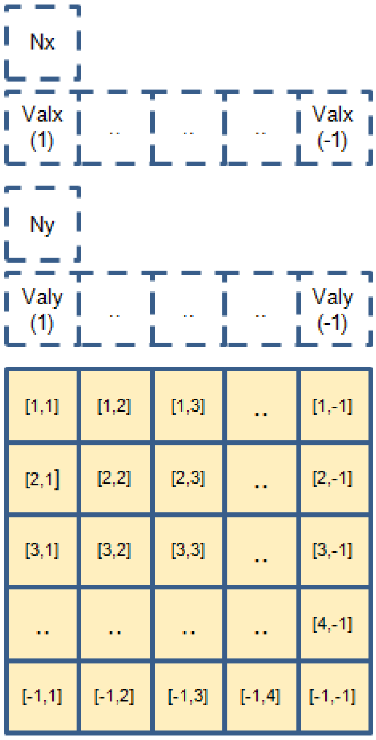

Figure 3.17: Map Logical View

The matrix element a(2,3) in figure 3.17 is represented by (AXIS 1) x = 3 and (AXIS 2) y = 2.

Memory representation (COLUMN_DIR):

The `SwRecordLayout` Map illustrated in figure 3.17 is stored in case of category COLUMN_DIR as follows:

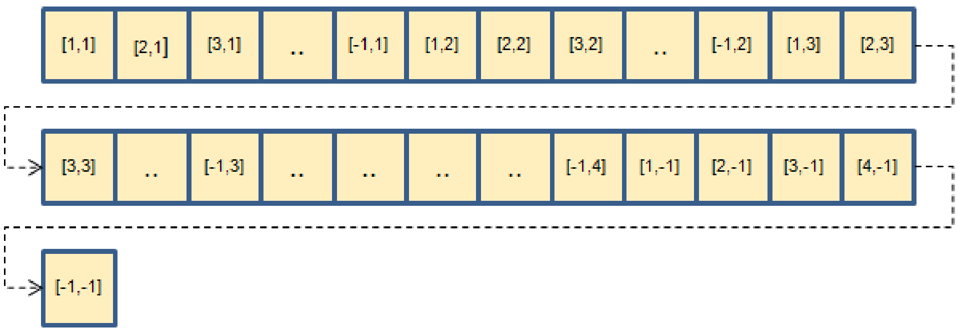

Figure 3.18: Map Memory Representation

This means that the data is stored first in direction of columns and then in direction of rows ([1,1],[2,1],[3,1], ...).

ARXML representation:

Extract of the record layout Map_s16 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.5: Record Layout: Map_s16 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: Map_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">Map_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-GROUP>
        <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
        <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
        <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
        <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
        <SW-RECORD-LAYOUT-V>
          <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
             SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
          <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
          <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
          <SW-RECORD-LAYOUT-V-INDEX>X Y</SW-RECORD-LAYOUT-V-INDEX>
        </SW-RECORD-LAYOUT-V>
      </SW-RECORD-LAYOUT-GROUP>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

3.3.4 Record Layout: IntMap

This chapter describes the record layout for a map with integrated data point search.

Logical view:

The figure 3.19 illustrates the logical view of the SwRecordLayout IntMap. Nx and Ny represent the number of sampling points given by the standardized values of SwRecordLayoutV.swRecordLayoutVProp. In the following example the dimensions of Nx and Ny are not fixed defined but given by a range indicated by index values. In the scope of this example the value COUNT is used. The SwRecordLayoutGroup with the shortLabel Val is shown in the lower part. Its elements are indexed by [AXIS 2, AXIS 1] from value (AXIS 2: = 1, AXIS

1: = 1) to value (AXIS 2: = -1, AXIS 1: = -1) there -1 gives the last value.

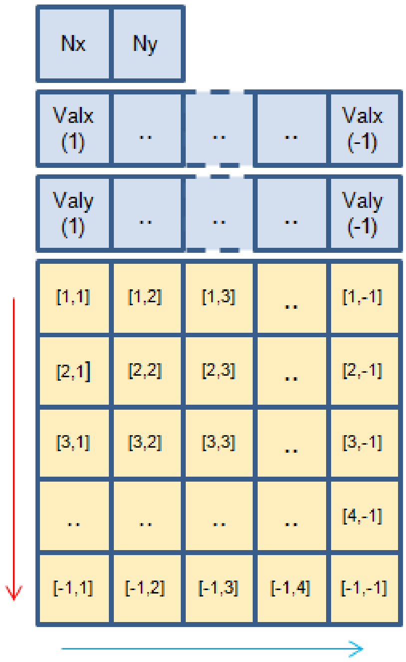

Figure 3.19: IntMap Logical View

The matrix element a(2,3) in figure 3.19 is represented by (AXIS 1) x = 3 and (AXIS 2) y = 2.

Memory representation (COLUMN_DIR):

The `SwRecordLayout` IntMap illustrated in figure 3.19 is stored in case of category COLUMN_DIR as follows:

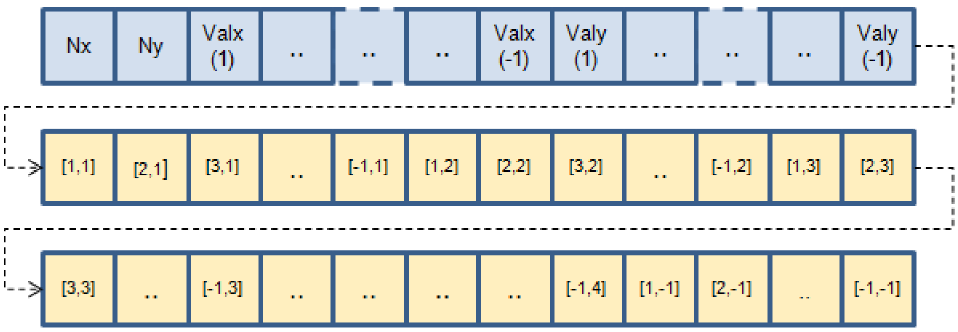

Figure 3.20: IntMap Memory Representation (COLUMN_DIR)

This means that the data is stored first in direction of columns and then in direction of rows ([1,1],[2,1],[3,1], ...).

ARXML representation:

Extract of the record layout IntMap_s16s16_s16 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.6: Record Layout: IntMap_s16s16_s16 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: IntMap_s16s16_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">IntMap_s16s16_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Nx</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
         SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Ny</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
         SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">X</SHORT-LABEL>
      <CATEGORY>INDEX_INCR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
         SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Y</SHORT-LABEL>
      <CATEGORY>INDEX_INCR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
         SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
```

```
        <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
        <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
        <SW-RECORD-LAYOUT-GROUP>
          <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
          <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
          <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
          <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
          <SW-RECORD-LAYOUT-V>
            <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
               SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
            <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
            <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
            <SW-RECORD-LAYOUT-V-INDEX>X Y</SW-RECORD-LAYOUT-V-INDEX>
          </SW-RECORD-LAYOUT-V>
        </SW-RECORD-LAYOUT-GROUP>
      </SW-RECORD-LAYOUT-GROUP>
    </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

Memory representation (ROW_DIR):

The SwRecordLayout IntMap illustrated in figure 3.19 is stored in case of category ROW_DIR as follows:

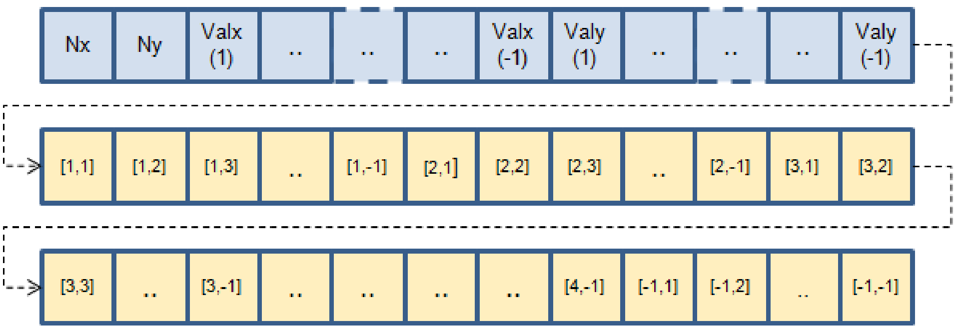

Figure 3.21: IntMap Memory Representation (ROW_DIR)

This means that the data are stored first in direction of rows and then in direction of columns ([1,1],[1,2],[1,3], ...).

ARXML representation:

Extract of the record layout IntMap_s8s16_s16 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.7: Record Layout: IntMap_s8s16_s16 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: IntMap_s8s16_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">IntMap_s8s16_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Nx</SHORT-LABEL>
```

```
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       SwBaseTypes_Blueprint/sint8</BASE-TYPE-REF>
    <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
    <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
  </SW-RECORD-LAYOUT-V>
  <SW-RECORD-LAYOUT-V>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Ny</SHORT-LABEL>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       SwBaseTypes_Blueprint/sint8</BASE-TYPE-REF>
    <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
    <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
  </SW-RECORD-LAYOUT-V>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">X</SHORT-LABEL>
    <CATEGORY>INDEX_INCR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       SwBaseTypes_Blueprint/sint8</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Y</SHORT-LABEL>
    <CATEGORY>INDEX_INCR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
    <CATEGORY>ROW_DIR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-GROUP>
      <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
           SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
```

```
          <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
          <SW-RECORD-LAYOUT-V-INDEX>X Y</SW-RECORD-LAYOUT-V-INDEX>
        </SW-RECORD-LAYOUT-V>
      </SW-RECORD-LAYOUT-GROUP>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

3.3.5 Record Layout: IntMap 3 x 4

Non-symmetrical matrices are commonly used and therefore a detailed description of their handling is given here.

The logical view is represented by 3-by-4 matrix (two dimensional matrix). Each element of a matrix is denoted by an index with two subscripts [AXIS 2, AXIS 1]. For instance, [3,2] represents the element at the third row (AXIS 2) and second column (AXIS 1) of a matrix.

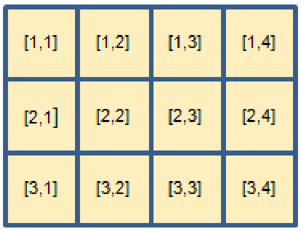

Figure 3.22: 3 x 4 Matrix Representation

In case of column-major order transformation the 3 x 4 matrix results in

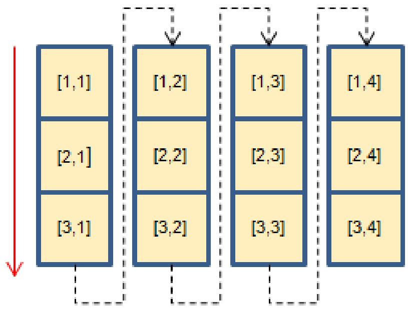

Figure 3.23: Transform 3 x 4 Matrix in column-major order

and in case of row-major order transformation the 3 x 4 matrix results in.

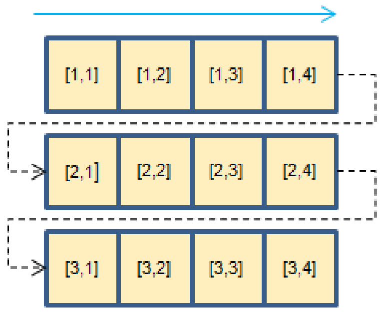

Figure 3.24: Transform 3 x 4 Matrix in row-major order

Logical view:

The figure 3.25 illustrates the logical view of the SwRecordLayout IntMap for the 3 x 4 matrix. Nx and Ny represent the number of sampling points given by the standardized values of `SwRecordLayoutV.swRecordLayoutVProp`. In the following example the dimensions are of Nx = 4 and Ny = 3. In the scope of this example the value `COUNT` is used. The `SwRecordLayoutGroup` with the `shortLabel` Val is shown in the lower part. Its elements are indexed by [AXIS 2, AXIS 1] from value (AXIS 2: = 1, AXIS 1: = 1) to value (AXIS 2: = 3, AXIS 1: = 4). AXIS 1 is assigned to Valx and shown above the values. AXIS 2 is assigned to Valy and shown on the left side of the values.

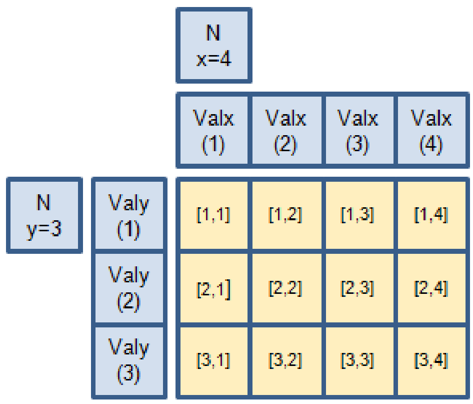

Figure 3.25: IntMap Logical View 3 x 4 Matrix

Memory representation:

The SwRecordLayout IntMap of 3 x 4 matrix illustrated in figure 3.26 is stored in case of category COLUMN_DIR as follows:

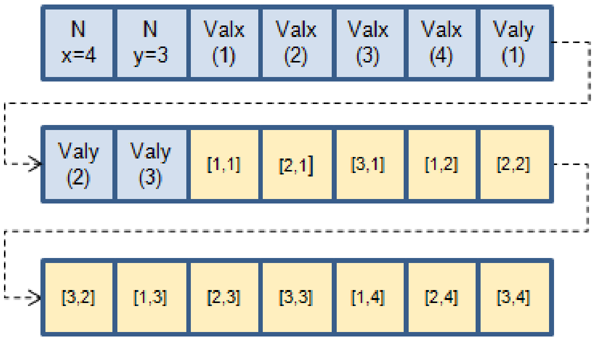

Figure 3.26: IntMap Memory Representation (COLUMN_DIR) 3 x 4 Matrix

This means that the data is stored first in direction of columns and then in direction of rows. This means for Valx(1) ([1,1],[2,1],[3,1]), for Valx(2) ([1,2],[2,2],[3,2]), for Valx(3) ([1,3],[2,3],[3,3]) and for Valx(4) ([1,4],[2,4],[3,4]).

The SwRecordLayout IntMap of 3 x 4 matrix illustrated in figure 3.27 is stored in case of category ROW_DIR as follows:

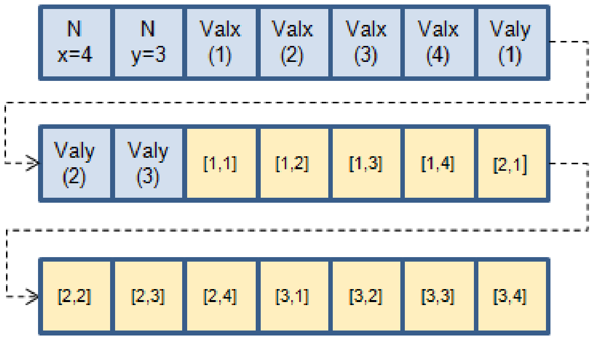

Figure 3.27: IntMap Memory Representation (ROW_DIR) 3 x 4 Matrix

This means that the data is stored first in direction of rows and then in direction of columns. This means for Valy(1) ([1,1],[1,2],[1,3],[1,4]), for Valy(2) ([2,1],[2,2],[2,3],[2,4]), for Valy(3) ([3,1],[3,2],[3,3],[3,4]).

3.3.6 Record Layout: FixIntMap

This chapter describes the record layout for a map with fixed axis points. Fixed axis exist in three categories: FIX_AXIS_PAR, FIX_AXIS_PAR_DIST and FIX_AXIS_PAR_LIST, see [TPS_SWCT_01748] in [7].

The number of sampling points (Nx, Ny), the Offset, the shift and the distance values are represented in the following chapters by these logical views:

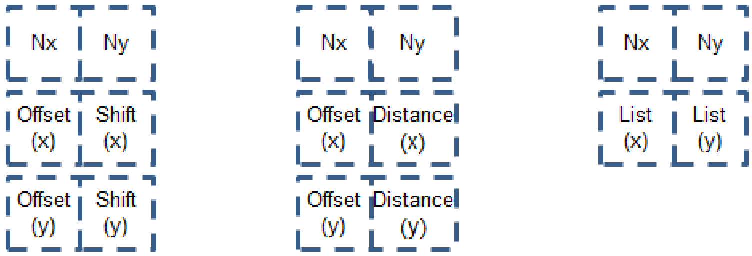

Figure 3.28: FIX_AXIS_PAR (left), FIX_AXIS_PAR_DIST (middle), FIX_AXIS_PAR_LIST (right)

These values are not defined inside `SwRecordLayout`s with fixed axis points.

Logical view:

The figure 3.29 illustrates the logical view of the `SwRecordLayout` FixIntMap. The `SwRecordLayoutGroup` with the `shortLabel` Val is shown in the lower part. Its elements are indexed by [AXIS 2, AXIS 1] from value (AXIS 2: = 1, AXIS 1: = 1) to value (AXIS 2: = -1, AXIS 1: = -1) there -1 gives the last value.

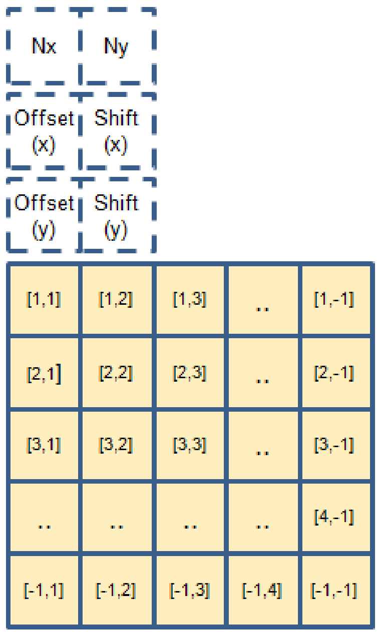

Figure 3.29: FixIntMap Logical View

The matrix element a(2,3) in figure 3.29 is represented by (AXIS 1) x = 3 and (AXIS 2) y = 2.

Memory representation (COLUMN_DIR):

The SwRecordLayout FixIntMap illustrated in figure 3.29 is stored in case of category COLUMN_DIR as follows:

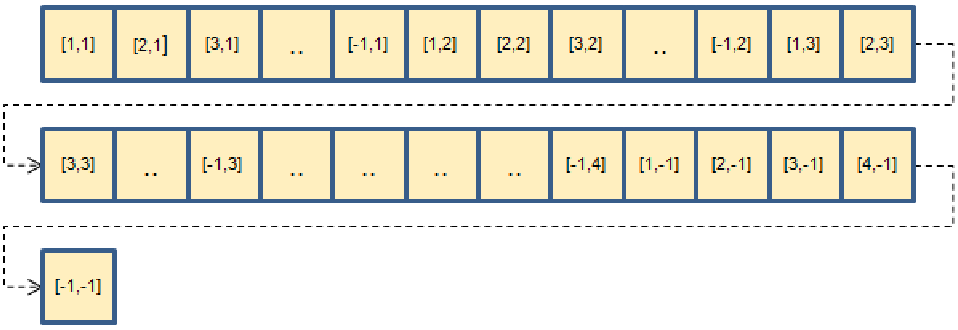

Figure 3.30: FixIntMap Memory Representation

This means that the data is stored in direction of columns and then in direction of rows ([1,1],[2,1],[3,1], ...).

ARXML representation:

Extract of the record layout FixIntMap_s16_s16 from AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

Listing 3.8: Record Layout: FixIntMap_s16_s16 in ARXML representation

```
<!-- SW-RECORD-LAYOUT: FixIntMap_s16_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">FixIntMap_s16_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-GROUP>
        <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
        <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
        <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
        <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
        <SW-RECORD-LAYOUT-V>
          <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
              SwBaseTypes_Blueprint/sint16</BASE-TYPE-REF>
          <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
          <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
          <SW-RECORD-LAYOUT-V-INDEX>X Y</SW-RECORD-LAYOUT-V-INDEX>
        </SW-RECORD-LAYOUT-V>
      </SW-RECORD-LAYOUT-GROUP>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

The following `SwRecordLayout`s are not part of AUTOSAR_MOD_IFX_RecordLayout_Blueprint.arxml.

3.4 Multidimensional Arrays

This chapter describes record layouts for multidimensional arrays as cuboids, cube_4 and cube_5.

3.4.1 Definition of Indexing

To define record layouts for arrays with more than two dimensions the same approach is used as for maps described in 3.3.1.

In linear algebra, a 3-dimensional matrix is defined by A(l,m,n). Even though the specifics of symbolic matrix notation varies widely, the subscripts are intentionally defined as follows (see figure 3.31): slice or plane index (l), the row index (m) and the column index (n). The row index (m) and the column index (n) span maps as known from figure 3.12. The slice or plane index (l) builds an array of maps defined by the indexes (m,n).

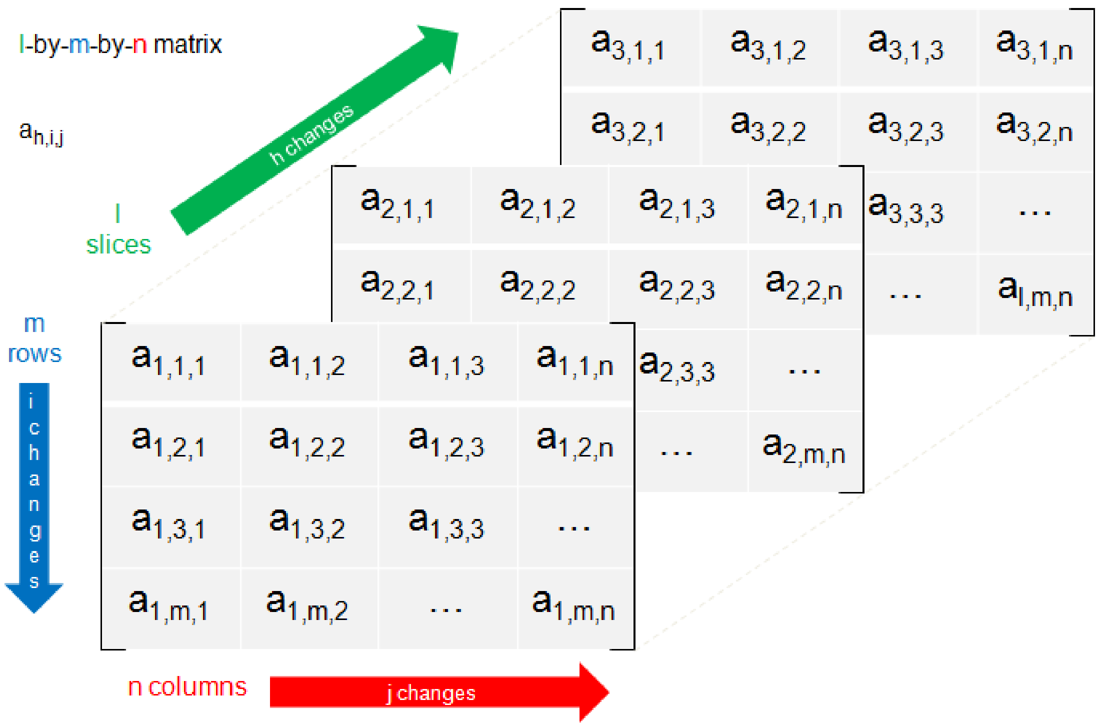

Figure 3.31: Linear Algebra Matrix with more than two dimensions

The transformation from indexing in matrix definition to the representation in Cartesian coordinate system is shown in figure 3.32.

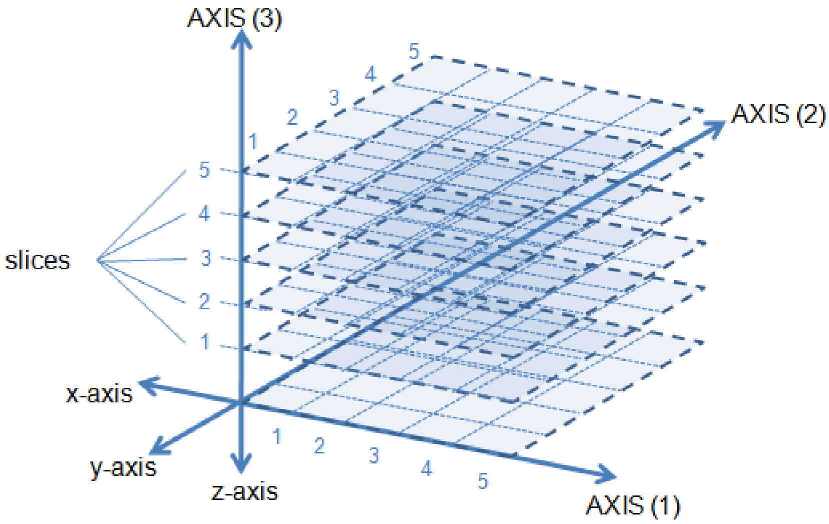

Figure 3.32: Cartesian Coordiate System with an array of maps

The (AXIS 1) and (AXIS 2) span a map. The (AXIS 3) builds an array of these maps called slices. Each of these slices will define a three-dimensional Euclidean space which determines every point by three "'coordinates"': (AXIS 1), (AXIS 2) and the value.

It is essential to understand that the (AXIS 3) is not providing the value of the data point. The (AXIS 3) gives the number of the three-dimensional Euclidean spaces in the cuboid.

3.4.2 Record Layout: Cuboid

This chapter describes the record layout for a cuboid.

Logical view:

The figure 3.33 illustrates the logical view of the `SwRecordLayout` Cuboid. The number of sampling points (Nx, Ny, Nz) are defined by separate `SwRecordLayoutV`s inside the `SwRecordLayout`. In the following example the dimensions are of Nx = 5, Ny = 4 and Nz = 2. The elements of [AXIS 1, AXIS 2, AXIS 3] are defined by separate `SwRecordLayoutGroup`s inside the enclosing `SwRecordLayoutGroup`. The `SwRecordLayoutGroup` with the `shortLabel` Val defines the values of the data points and is shown in the lower part.

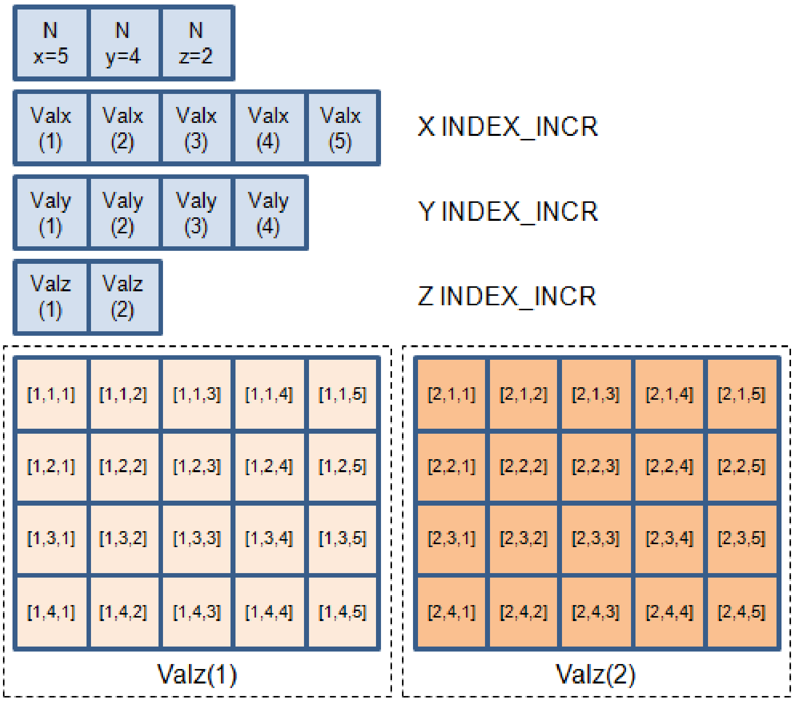

Figure 3.33: Cuboid Logical View

The first slice (AXIS 3: = 1) is illustrated by the dotted rectangular area named Valz(1), the second slice (AXIS 3: = 2) correspondently named by Valz(2).

Memory representation:

The `SwRecordLayout` Cuboid illustrated in figure 3.33 is stored in case of category COLUMN_DIR as follows:

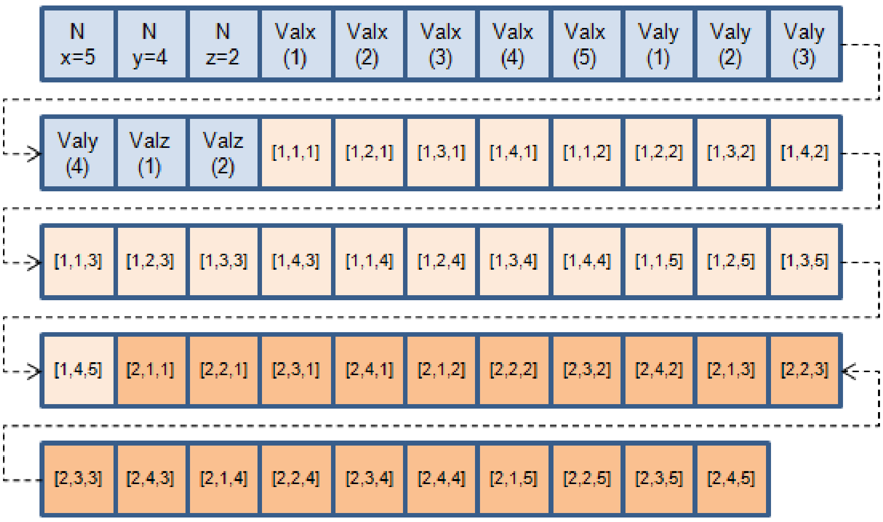

Figure 3.34: Cuboid Memory Representation (COLUMN_DIR)

This means that the data is stored in direction of columns and then in direction of rows starting with the first slice ([1,1,1],[1,2,1],[1,3,1], ... ,[1,4,5]). The second slice starts with ([2,1,1],[2,2,1],[2,3,1], ... ,[2,4,5]) and follows the same pattern.

ARXML representation:

The ARXML representation of the record layout Cuboid_s16s16s16_s16 is given in two parts for illustrative reason. The first part defines the number of sampling points and the elements of axis.

Listing 3.9: Record Layout of Cuboid - part one

```
<!-- SW-RECORD-LAYOUT: Cuboid_s16s16s16_s16 COLUMN_DIR -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">Cuboid_s16s16s16_s16</
      SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Nx</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Ny</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Nz</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>3</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">X</SHORT-LABEL>
      <CATEGORY>INDEX_INCR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Y</SHORT-LABEL>
      <CATEGORY>INDEX_INCR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
```

```
    <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
        BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Z</SHORT-LABEL>
    <CATEGORY>INDEX_INCR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>3</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>Z</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
        BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>3</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
```

The second part defines the values of the data points. Inside the SwRecordLayoutGroup with the shortLabel Val the definition of memory representation (COLUMN_DIR or ROW_DIR) has to be unique. This means that memory representation of the map (AXIS 1 and AXIS 2) and those of the slice (AXIS 3) have to be equal. In case of listing 3.10 the memory representation COLUMN_DIR is defined.

Listing 3.10: Record Layout of Cuboid - part two

```
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
    <CATEGORY>COLUMN_DIR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>3</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>Z</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-GROUP>
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-GROUP>
        <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
        <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-
            INDEX>
        <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
        <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
        <SW-RECORD-LAYOUT-V>
          <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
              BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
          <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
```

```
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
        <SW-RECORD-LAYOUT-V-INDEX>Z X Y</SW-RECORD-LAYOUT-V-INDEX>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT-GROUP>
```

The combination of different base types e.g. a Cuboid_u8s16u16_u32 are technically possible but not further described in this document.

3.4.3 Record Layout: Cube_4 and Cube_5

This chapter describes the record layouts for Cube_4 and Cube_5. The Cube_4 stores an array of Cuboids with incremented or decremented (AXIS 4). The Cube_5 correspondingly stores an array of Cube_4s with incremented or decremented (AXIS 5). In this version of the document only Cube_4 is described.

Logical view:

The figure 3.35 illustrates the logical view of the SwRecordLayout Cube_4. The number of sampling points (Nx, Ny, Nz1, Nz2) are defined by separate SwRecordLayoutVs inside the SwRecordLayout. In the following example the dimensions are of Nx = 5, Ny = 4, Nz1 = 2 and Nz2 = 3. The elements of [AXIS 1, AXIS 2, AXIS 3, AXIS 4] are defined by separate SwRecordLayoutGroups inside the SwRecordLayout. The SwRecordLayoutGroup with the shortLabel Val defines the values of the data points and is shown at the right side.

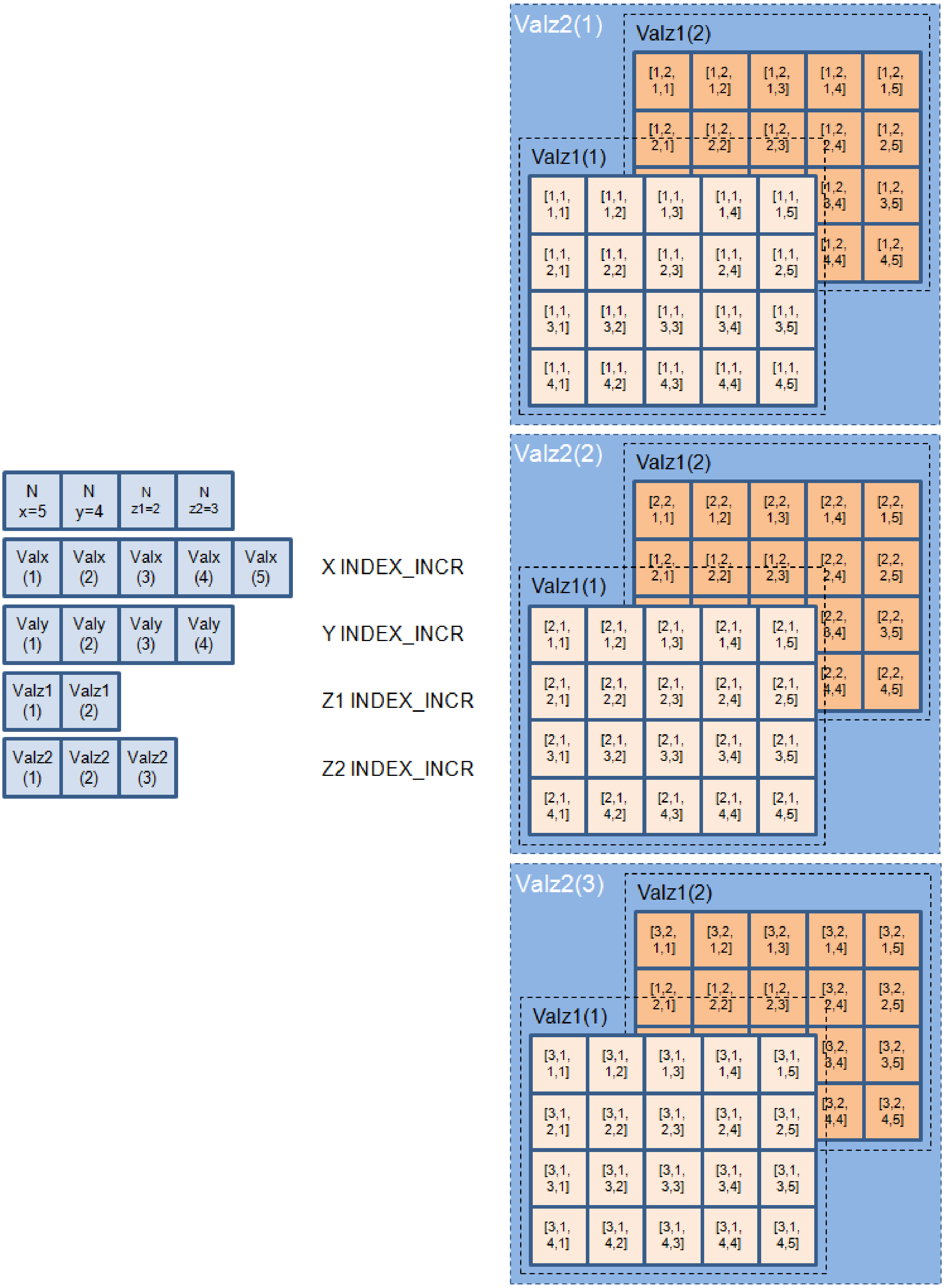

Figure 3.35: Cube_4 Logical View

The first array of cuboids (AXIS 4: = 1) is illustrated by the blue rectangular area named Valz2(1) at the top in figure 3.35. It contains the cuboid with the slices Valz1(1) and Valz1(2). The second array of cuboids (AXIS 4: = 2) and the third one (AXIS 4: = 3) are illustrated in the middle and at the bottom in figure 3.35. Both contain cuboids with the slices Valz1(1) and Valz1(2). Each element of a matrix is denoted by an index with four subscripts [AXIS 4, AXIS 3, AXIS 2, AXIS 1].

Memory representation:

The `SwRecordLayout` Cube_4 illustrated in figure 3.35 is stored in case of category COLUMN_DIR as follows:

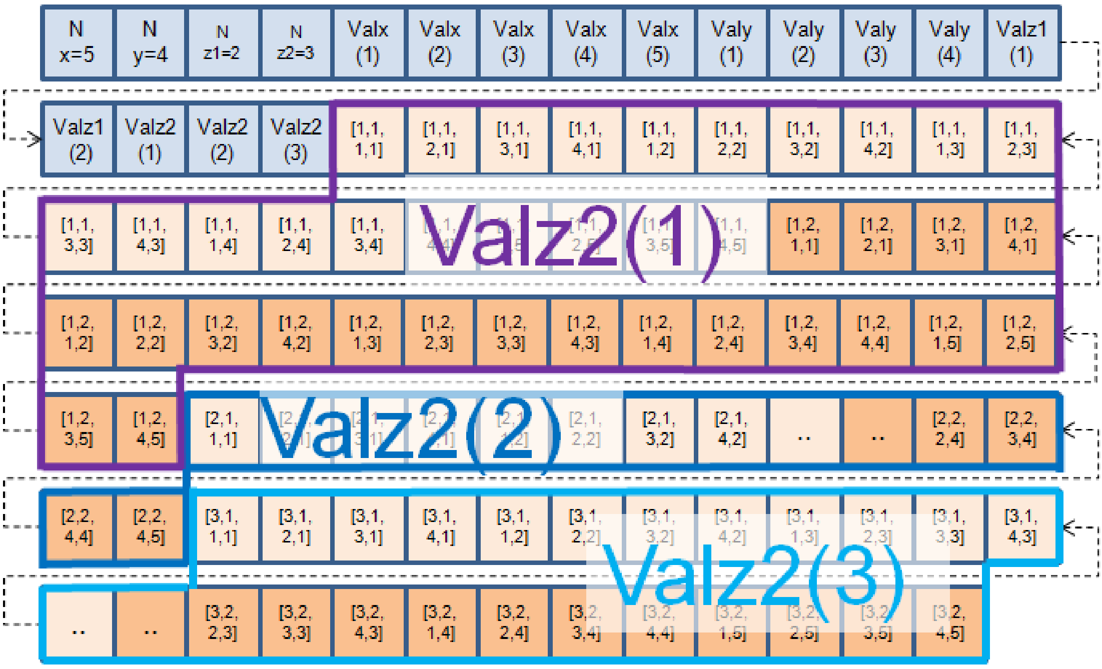

Figure 3.36: Cube_4 Memory Representation (COLUMN_DIR)

The data values are stored in the following order: starting with the iteration along cuboids Valz2(1), Valz2(2) and Valz2(3). Inside each of these iterations the iteration along slices Valz1(1) and Valz1(2) run. Inside each of these iterations the iteration along the maps is executed as known from 3.3.2. The data values of cuboids Valz2(2) and Valz2(3) are intentionally not completely illustrated in figure 3.36.

```
[
   (select cuboid; loop level 4)
   iteration along cubuids
   (AXIS 4 iterates, AXIS 3, AXIS 2 and AXIS 1 are fixed !)
   start with first cuboid (AXIS 4: = 1)
   [
      (select slice; loop level 3)
      iteration along slices
      (AXIS 3 iterates, AXIS 4, AXIS 2 and AXIS 1 are fixed !)
```

```
      start with first slice (AXIS 3: = 1)
      [
         (select row element; loop level 2)
         iteration along row
         (AXIS 1 iterates, AXIS 4, AXIS 3 and AXIS 2 are fixed !)
         start with first row (AXIS 1: = 1)
         [
            (select column element; loop level 1)
            iteration along column
            (AXIS 2 iterates, AXIS 4, AXIS 3 and AXIS 1 are fixed !)
            start with column element (AXIS 2: = 1)
            ...
            end with last column (AXIS 2: = 5)
         ]
         end with last row (AXIS 1: = 4)
      ]
      end with last slice (AXIS 3: = 2)
   ]
   end with last cuboid (AXIS 4: = 3)
]
```

ARXML representation:

The ARXML representation of the record layout Cube_4_s16s16s16s16_s16 is given in three parts for illustrative reason. The first part defines the number of sampling points (Nx, Ny, Nz1, Nz2).

Listing 3.11: Record Layout of Cube_4 - part one

```
<!-- SW-RECORD-LAYOUT: Cube_4_s16s16s16s16_s16 COLUMN_DIR -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">Cube_4_s16s16s16s16_s16</
      SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Nx</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Ny</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Nz1</SHORT-LABEL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>3</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-V>
```

```
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Nz2</SHORT-LABEL>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
    <SW-RECORD-LAYOUT-V-AXIS>4</SW-RECORD-LAYOUT-V-AXIS>
    <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
  </SW-RECORD-LAYOUT-V>
```

The second part defines the elements of axis [AXIS 1, AXIS 2, AXIS 3, AXIS 4].

Listing 3.12: Record Layout of Cube_4 - part two

```
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">X</SHORT-LABEL>
    <CATEGORY>INDEX_INCR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Y</SHORT-LABEL>
    <CATEGORY>INDEX_INCR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>2</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Z1</SHORT-LABEL>
    <CATEGORY>INDEX_INCR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>3</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>Z1</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>3</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Z2</SHORT-LABEL>
    <CATEGORY>INDEX_INCR</CATEGORY>
```

```
    <SW-RECORD-LAYOUT-GROUP-AXIS>4</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>Z2</SW-RECORD-LAYOUT-GROUP-INDEX>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
       BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>4</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
```

The third part defines the values of the data points. Inside the SwRecordLayout-Group with the shortLabel Val the nesting of the axis definies the memory representation. In case of listing 3.13 the memory representation COLUMN_DIR is defined. The (AXIS 2) iterates along the column.

Listing 3.13: Record Layout of Cube_4 - part three

```
<SW-RECORD-LAYOUT-GROUP>
  <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
  <CATEGORY>COLUMN_DIR</CATEGORY>
  <SW-RECORD-LAYOUT-GROUP-AXIS>4</SW-RECORD-LAYOUT-GROUP-AXIS>
  <SW-RECORD-LAYOUT-GROUP-INDEX>Z2</SW-RECORD-LAYOUT-GROUP-INDEX>
  <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
  <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
  <SW-RECORD-LAYOUT-GROUP>
    <CATEGORY>COLUMN_DIR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-AXIS>3</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-INDEX>Z1</SW-RECORD-LAYOUT-GROUP-INDEX
       >
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-GROUP>
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-INDEX>X</SW-RECORD-LAYOUT-GROUP-
         INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-GROUP>
        <SW-RECORD-LAYOUT-GROUP-AXIS>2</SW-RECORD-LAYOUT-GROUP-
           AXIS>
        <SW-RECORD-LAYOUT-GROUP-INDEX>Y</SW-RECORD-LAYOUT-GROUP-
           INDEX>
        <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-
           FROM>
        <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
        <SW-RECORD-LAYOUT-V>
          <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
             BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
          <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
          <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
          <SW-RECORD-LAYOUT-V-INDEX>Z2 Z1 X Y</SW-RECORD-LAYOUT-V-
             INDEX>
        </SW-RECORD-LAYOUT-V>
```

```
              </SW-RECORD-LAYOUT-GROUP>
            </SW-RECORD-LAYOUT-GROUP>
          </SW-RECORD-LAYOUT-GROUP>
        </SW-RECORD-LAYOUT-GROUP>
```

3.5 Value and ValueBlock

3.5.1 Record Layout: Value

Logical view:

The figure 3.37 illustrates the logical view of the SwRecordLayout Value. This SwRecordLayout contains only one value.

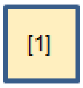

Figure 3.37: Value Logical View

Memory representation:

The SwRecordLayout Val illustrated in figure 3.37 is stored as follows:

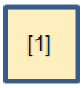

Figure 3.38: Value Memory Representation

ARXML representation:

Listing 3.14: Record Layout of Value

```
<!-- SW-RECORD-LAYOUT: Val_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">Val_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
    <SW-RECORD-LAYOUT-V>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
```

```
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

3.5.2 Record Layout: One dimensional ValueBlock

Logical view:

The figure 3.39 illustrates the logical view of the `SwRecordLayout` ValueBlock. This `SwRecordLayout` is an array of values (similar to an axis but without the number of axis points).

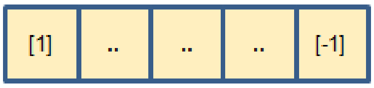

Figure 3.39: ValueBlock Logical View

Memory representation:

The `SwRecordLayout` ValueBlock illustrated in figure 3.40 is stored as follows:

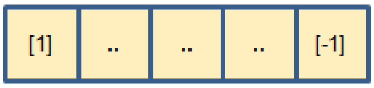

Figure 3.40: Value Memory Representation

ARXML representation:

Listing 3.15: Record Layout of ValueBlock

```
<!-- SW-RECORD-LAYOUT: ValBlk_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">ValBlk_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
    <CATEGORY>COLUMN_DIR</CATEGORY>
    <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
          BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
      <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

3.5.3 Record Layout: Multi dimensional ValueBlock

Logical view:

The figure 3.41 illustrates the logical view of the SwRecordLayout Multi dimensional ValueBlock. This SwRecordLayout is an array of values (similar to a map or cuboid but without axis points). N_i defines the dimension of the ValueBlock. N_j defines the number of values in a dedicated dimension.

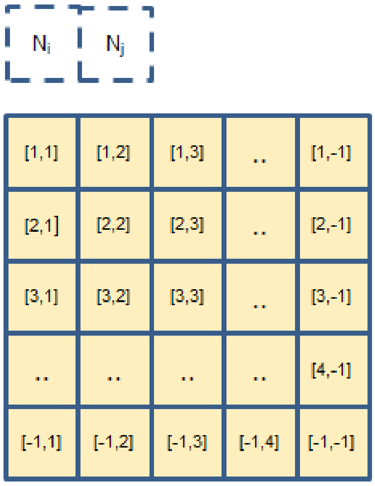

Figure 3.41: Multi dimensional ValueBlock Logical View

Memory representation:

The SwRecordLayout Multi dimensional ValueBlock illustrated in figure 3.41 is stored as follows:

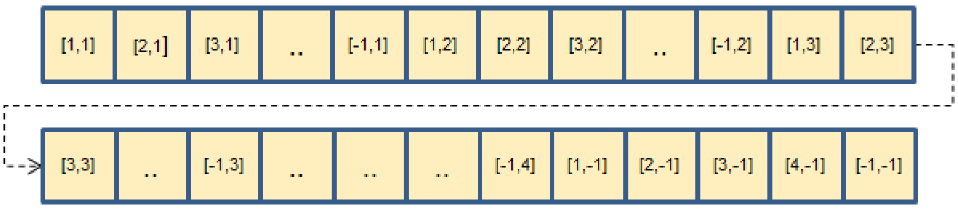

Figure 3.42: Multi dimensional ValueBlock Memory Representation

ARXML representation:

Listing 3.16: Record Layout of Multi dimensional ValueBlock

```
<!-- SW-RECORD-LAYOUT: ValBlkMDim_s16 -->
<SW-RECORD-LAYOUT>
  <SHORT-NAME NAME-PATTERN="{blueprintName}">ValBlkMDim_s16</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL NAME-PATTERN="{blueprintName}">Val</SHORT-LABEL>
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-INDEX>i</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
```

```
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-GROUP>
      <SW-RECORD-LAYOUT-GROUP-INDEX>j</SW-RECORD-LAYOUT-GROUP-INDEX>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR/Platform/
            BaseTypes_Blueprint/sint16</BASE-TYPE-REF>
        <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS>
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
        <SW-RECORD-LAYOUT-V-INDEX>i j</SW-RECORD-LAYOUT-V-INDEX>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
```

Hint: Comparable with FixIntMap. Due to ASAM the dimension is not limited, the examples in ASAM will represent up to 3 dimension.

4 Additional SwRecordLayouts

In this chapter further `SwRecordLayout` will be described which are not covered by dedicated SWS documents.

Contents will be updated.

5 Units and Physical Dimensions

The Units and Physical Dimension are defined accordingly to the description in [10] and [7] by [TPS_SWCT_01285], [TPS_SWCT_01056], [TPS_SWCT_01057], [TPS_SWCT_01058], [TPS_SWCT_01736], [TPS_SWCT_01059], [TPS_SWCT_01737], [TPS_SWCT_01060], [TPS_SWCT_01060], [TPS_SWCT_01061], [TPS_SWCT_01068], [constr_1026] and [constr_1255].

A Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	***BaseType*** (abstract)			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This abstract meta-class represents the ability to specify a platform dependant base type.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	SwBaseType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
baseType Definition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table A.1: BaseType

Class	***BlueprintPolicy*** (abstract)			
Package	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to indicate whether blueprintable elements will be modifiable or not modifiable.			
Base	*ARObject*			
Subclasses	*BlueprintPolicyModifiable*, BlueprintPolicyNotModifiable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
attributeName	String	1	attr	This identifies the related attribute of a BlueprintPolicy. For navigation over the model a subset of xpath expressions is used.

Table A.2: BlueprintPolicy

Class	**BswModuleDescription**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:** atp.recommendedPackage=BswModuleDescriptions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
bswModule Dependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

▽

△

Class	**BswModuleDescription**			
bswModule Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDocumentation, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6
expectedEntry	BswModuleEntry	*	ref	Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=expectedEntry, variationPoint.short Label vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	*	ref	Specifies an entry provided by this module which can be called by other modules. This includes "main" functions, interrupt routines, and callbacks. Replacement of providedEntry / expectedCallback. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implementedEntry, variation Point.shortLabel vh.latestBindingTime=preCompileTime
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a Bsw ModuleDescription can be distributed over several physical files. Therefore the aggregation is «atpSplitable». **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName xml.sequenceOffset=65
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:** xml.sequenceOffset=5
providedClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module provides a client server entry which can be called from another parition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55

▽

△

Class	**BswModuleDescription**			
providedMode Group	ModeDeclarationGroup Prototype	*	aggr	A set of modes which is owned and provided by this module or cluster. It can be connected to the required ModeGroups of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with modes provided via ports by an associated ServiceSwComponentType, EcuAbstraction SwComponentType or ComplexDeviceDriverSw ComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, Ecu AbstractionSwComponentType or ComplexDeviceDriver SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another parition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The required Data is declared locally to this context and will be connected to the providedData of another or the same module via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredMode Group	ModeDeclarationGroup Prototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

▽

△

Class	**BswModuleDescription**			
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table A.3: BswModuleDescription

Class	**BswModuleEntry**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	This class represents a single API entry (C-function prototype) into the BSW module or cluster. The name of the C-function is equal to the short name of this element with one exception: In case of multiple instances of a module on the same CPU, special rules for "infixes" apply, see description of class BswImplementation. **Tags:** atp.recommendedPackage=BswModuleEntrys			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
argument (ordered)	SwServiceArg	*	aggr	An argument belonging to this BswModuleEntry. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=45
bswEntryKind	BswEntryKindEnum	0..1	attr	This describes whether the entry is concrete or abstract. If the attribute is missing the entry is considered as concrete. **Tags:** xml.sequenceOffset=40
callType	BswCallType	1	attr	The type of call associated with this service. **Tags:** xml.sequenceOffset=25
execution Context	BswExecutionContext	1	attr	Specifies the execution context which is required (in case of entries into this module) or guaranteed (in case of entries called from this module) for this service. **Tags:** xml.sequenceOffset=30
function Prototype Emitter	NameToken	0..1	attr	This attribute is used to control the generation of function prototypes. If set to "RTE", the RTE generates the function prototypes in the Module Interlink Header File.
isReentrant	Boolean	1	attr	Reentrancy from the viewpoint of function callers: • True: Enables the service to be invoked again, before the service has finished. • False: It is prohibited to invoke the service again before is has finished. **Tags:** xml.sequenceOffset=15

▽

△

Class	**BswModuleEntry**			
isSynchronous	Boolean	1	attr	Synchronicity from the viewpoint of function callers: • True: This calls a synchronous service, i.e. the service is completed when the call returns. • False: The service (on semantical level) may not be complete when the call returns. **Tags:** xml.sequenceOffset=20
returnType	SwServiceArg	0..1	aggr	The return type belonging to this bswModuleEntry. **Tags:** xml.sequenceOffset=40
role	Identifier	0..1	attr	Specifies the role of the entry in the given context. It shall be equal to the standardized name of the service call, especially in cases where no ServiceIdentifier is specified, e.g. for callbacks. Note that the ShortName is not always sufficient because it maybe vendor specific (e.g. for callbacks which can have more than one instance). **Tags:** xml.sequenceOffset=10
serviceId	PositiveInteger	0..1	attr	Refers to the service identifier of the Standardized Interfaces of AUTOSAR basic software. For non-standardized interfaces, it can optionally be used for proprietary identification. **Tags:** xml.sequenceOffset=5
swServiceImplPolicy	SwServiceImplPolicyEnum	1	attr	Denotes the implementation policy as a standard function call, inline function or macro. This has to be specified on interface level because it determines the signature of the call. **Tags:** xml.sequenceOffset=35

Table A.4: BswModuleEntry

Class	**ClientServerInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
operation	ClientServerOperation	1..*	aggr	ClientServerOperation(s) of this ClientServerInterface. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface.

Table A.5: ClientServerInterface

Class	**ClientServerInterfaceToBswModuleEntryBlueprintMapping**
Package	M2::AUTOSARTemplates::StandardizationTemplate::ClientServerInterfaceToBswModuleEntryMapping

▽

△

Class	**ClientServerInterfaceToBswModuleEntryBlueprintMapping**			
Note	This represents a mapping between one ClientServerInterface blueprint and BswModuleEntry blueprint in order to express the intended implementation of ClientServerOperations by specific BswModuleEntries under consideration of PortDefinedArguments. Such a mapping enables the formal check whether the number of arguments and the data types of arguments of the operation + additional PortDefinedArguments matches the signature of the BswModule Entry. **Tags:** atp.recommendedPackage=BlueprintMappingSets			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
clientServer Interface	ClientServerInterface	1	ref	The referenced ClientServerInterface represents the client server interface the mapping is dedicated to.
operation Mapping	ClientServerOperation BlueprintMapping	1..*	aggr	This specifies the operations used in the mapping between the ClientServerInterface and the BswModule Entry. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
portDefined Argument Blueprint (ordered)	PortDefinedArgument Blueprint	*	aggr	This specifies the PortDefinedArguments used in the mapping between the ClientServerInterface and the Bsw ModuleEntry. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table A.6: ClientServerInterfaceToBswModuleEntryBlueprintMapping

Class	**CompuMethod**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
compuInternal ToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:** xml.sequenceOffset=80
compuPhysTo Internal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:** xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:** xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:** xml.sequenceOffset=30

Table A.7: CompuMethod

Class	DataConstr			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints on data. **Tags:** atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table A.8: DataConstr

Class	ImplementationDataType			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to True, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional. **Tags:** atp.Status=draft
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table A.9: ImplementationDataType

Class	**InterpolationRoutineMappingSet**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::InterpolationRoutineMappingSet			
Note	This meta-class specifies a set of interpolation routine mappings. **Tags:** atp.recommendedPackage=InterpolationRoutineMappingSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
interpolationRoutineMapping	InterpolationRoutineMapping	*	aggr	This specifies one particular mapping of recordlayout and its matching interpolationRoutines.

Table A.10: InterpolationRoutineMappingSet

Class	**Keyword**			
Package	M2::AUTOSARTemplates::StandardizationTemplate::Keyword			
Note	This meta-class represents the ability to predefine keywords which may subsequently be used to construct names following a given naming convention, e.g. the AUTOSAR naming conventions. Note that such names is not only shortName. It could be symbol, or even longName. Application of keywords is not limited to particular names.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
abbrName	NameToken	1	attr	This attribute specifies an abbreviated name of a keyword. This abbreviation may e.g. be used for constructing valid shortNames according to the AUTOSAR naming conventions. Unlike shortName, it may contain any name token. E.g. it may consist of digits only.
classification	NameToken	*	attr	This attribute allows to attach classification to the Keyword such as MEAN, ACTION, CONDITION, INDEX, PREPOSITION

Table A.11: Keyword

Class	**ModeDeclarationGroup**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:** atp.ManifestKind=ExecutionManifest,MachineManifest atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
initialMode	ModeDeclaration	1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
modeDeclaration	ModeDeclaration	1..*	aggr	The ModeDeclarations collected in this ModeDeclarationGroup. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=blueprintDerivationTime
modeManagerErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user).

▽

△

Class	**ModeDeclarationGroup**			
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup
modeUserErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager).
onTransitionValue	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses.

Table A.12: ModeDeclarationGroup

Class	**ModeSwitchInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A mode switch interface declares a ModeDeclarationGroupPrototype to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
modeGroup	ModeDeclarationGroupPrototype	1	aggr	The ModeDeclarationGroupPrototype of this mode interface.

Table A.13: ModeSwitchInterface

Class	**NvBlockSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The NvBlockSwComponentType defines non volatile data which data can be shared between SwComponentPrototypes. The non volatile data of the NvBlockSwComponentType are accessible via provided and required ports. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
nvBlockDescriptor	NvBlockDescriptor	*	aggr	Specification of the properties of exactly one NVRAM Block. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table A.14: NvBlockSwComponentType

Class	**SenderReceiverInterface**
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*

▽

△

Class	**SenderReceiverInterface**			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	VariableDataPrototype	1..*	aggr	The data elements of this SenderReceiverInterface.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement

Table A.15: SenderReceiverInterface

Class	**ServiceSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *Atp Type*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *Sw ComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table A.16: ServiceSwComponentType

Class	**SwAddrMethod**			
Package	M2::MSR::DataDictionary::AuxillaryObjects			
Note	Used to assign a common addressing method, e.g. common memory section, to data or code objects. These objects could actually live in different modules or components. **Tags:** atp.recommendedPackage=SwAddrMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
memory Allocation KeywordPolicy	MemoryAllocation KeywordPolicyType	0..1	attr	Enumeration to specify the name pattern of the Memory Allocation Keyword.
option	Identifier	*	attr	This attribute introduces the ability to specify further intended properties of the MemorySection in with the related objects shall be placed. These properties are handled as to be selected. The intended options are mentioned in the list. In the Memory Mapping configuration, this option list is used to determine an appropriate MemMapAddressing ModeSet.
section Initialization Policy	SectionInitialization PolicyType	0..1	attr	Specifies the expected initialization of the variables (inclusive those which are implementing VariableData Prototypes). Therefore this is an implementation constraint for initialization code of BSW modules (especially RTE) as well as the start-up code which initializes the memory segment to which the AutosarData Prototypes referring to the SwAddrMethod's are later on mapped. If the attribute is not defined it has the identical semantic as the attribute value "INIT"
sectionType	MemorySectionType	0..1	attr	Defines the type of memory sections which can be associated with this addresssing method.

Table A.17: SwAddrMethod

Class	**SwRecordLayout**			
Package	M2::MSR::DataDictionary::RecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:** atp.recommendedPackage=SwRecordLayouts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
swRecord LayoutGroup	SwRecordLayoutGroup	1	aggr	This is the top level record layout group. **Tags:** xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table A.18: SwRecordLayout

▽

Class	SwRecordLayoutGroup			
Package	M2::MSR::DataDictionary::RecordLayout			
Note	Specifies how a record layout is set up. Using SwRecordLayoutGroup it recursively models iterations through axis values. The subelement swRecordLayoutGroupContentType may reference other Sw RecordLayouts, SwRecordLayoutVs and SwRecordLayoutGroups for the modeled record layout.			
Base	*ARObject*			
Attribute	Type	Mul.	Kind	Note
desc	MultiLanguageOverview Paragraph	0..1	aggr	This aggregation allows a brief description about the particular record layout group which can help to identify the entry. In-depth documentation should be added to the introduction of the surrounding record layout. **Tags:** xml.sequenceOffset=20
category	AsamRecordLayout Semantics	0..1	attr	This attribute denotes the semantics in particular in terms of the corresponding A2L-Keyword. This is to support the mapping of the more general record layouts in AUTOSAR/MSR to the specific A2l keywords. It is possible to express the specific semantics of A2l recordlayout keywords in swRecordlayoutGroup but not always vice versa. Therefore the mapping is provided in this optional attribute. **Tags:** xml.sequenceOffset=5
shortLabel	Identifier	1	attr	This attribute specifies a name which can be used e.g. when ECU code is generated from the record layout group. **Tags:** xml.sequenceOffset=3
swGenericAxis ParamType	SwGenericAxisParam Type	0..1	ref	This association allows to specify record layout groups to iterate over generic axis parameters. For example, if the generic axis parameter is an array, the record layout group will iterate over this array. Obviously, the axis referred to by swRecordLayoutGroup Axis shall be a generic axis in which the referenced Sw GenericAxisType is aggregated. **Tags:** xml.sequenceOffset=50
swRecord Layout Component	Identifier	0..1	attr	This attribute is used to denote the component to which the group in question applies. Thus, the record layout supports structured objects. This secures independence from the sequence of components, because they can be referred to via name. **Tags:** xml.sequenceOffset=90

▽

△

Class	**SwRecordLayoutGroup**			
swRecordLayoutGroupAxis	AxisIndexType	0..1	attr	This attribute specifies the iteration axis number for a SwRecordLayoutGroup. The current record layout group then refers exactly to the axis with this number. This means that the values are taken by iterating along the thus referenced axis. **Tags:** xml.sequenceOffset=30
swRecordLayoutGroupContentType	SwRecordLayoutGroupContent	0..1	aggr	This is the contents of the recordLayout which is produced for every step of iteration. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=100 xml.typeElement=false xml.typeWrapperElement=false
swRecordLayoutGroupFrom	RecordLayoutIteratorPoint	0..1	attr	This attribute specifies the iterator index for the point in the axis from which a record layout group is commenced. Negative values are also possible, i.e. the value -4 counts from the fourth value from the end. If this property is missing, the iteration starts with '1'. **Tags:** xml.sequenceOffset=60
swRecordLayoutGroupIndex	NameToken	0..1	attr	This attribute attributes a symbolic name to the iterator of the superimposed record layout group. This can be referenced as a loop index in contained SwRecordLayoutV elements. **Tags:** xml.sequenceOffset=40
swRecordLayoutGroupStep	Integer	0..1	attr	This attribute specifies the step width for the iterator index that is used for the current record layout group. Note that negative values are also possible, in case of the starting point is higher than the endpoint. If the property is missing, the step width is "1". **Tags:** xml.sequenceOffset=80
swRecordLayoutGroupTo	RecordLayoutIteratorPoint	0..1	attr	This attribute specifies the end point for the iteration. Negative values are also possible, i.e. the value -4 counts up to the fourth value from the end. If this property is not there, the iteration ends at "-1" which is the last element. Note that depending on the arraySizeSemantics of SwTextProps the iteration ends at the value specified in swMaxTextSize. **Tags:** xml.sequenceOffset=70

Table A.19: SwRecordLayoutGroup

Class	**SwRecordLayoutV**			
Package	M2::MSR::DataDictionary::RecordLayout			
Note	This element specifies which values are stored for the current SwRecordLayoutGroup. If no baseType is present, the SwBaseType referenced initially in the parent SwRecordLayoutGroup is valid. The specification of swRecordLayoutVAxis gives the axis of the values which shall be stored in accordance with the current record layout SwRecordLayoutGroup. In swRecordLayoutVProp one can specify the information which shall be stored.			
Base	*ARObject*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

▽

△

Class	SwRecordLayoutV			
desc	MultiLanguageOverview Paragraph	0..1	aggr	This aggregation allows for a brief description about the particular record layout value which can help to identify the entry. In-depth documentation should be added to the introduction of the surrounding record layout. **Tags:** xml.sequenceOffset=20
category	AsamRecordLayout Semantics	0..1	attr	This attribute denotes the semantics in particular in terms of the corresponding A2L-Keyword. This is to support the mapping of the more general record layouts in AUTOSAR/MSR to the specific A2l keywords. It is possible to express the specific semantics of A2l Record Layout keywords in swRecordlayoutGroup but not always vice versa. Therefore the mapping is provided in this optional attribute. **Tags:** xml.sequenceOffset=5
baseType	SwBaseType	0..1	ref	This association allows to refer to a base type in case a specific encoding is intended. If no base type is referred, the base type referenced initially in the corresponding DataPrototype is to be used. **Tags:** xml.sequenceOffset=30
shortLabel	Identifier	1	attr	This attribute specifies a name which can be used e.g. when ECU code is generated from the record layout value. **Tags:** xml.sequenceOffset=3
swGenericAxis ParamType	SwGenericAxisParam Type	0..1	ref	This association supports the case that a value from a generic axis definition shall be stored. This value is denoted by a particular generic axis parameter type. **Tags:** xml.sequenceOffset=70
swRecord LayoutVAxis	AxisIndexType	0..1	attr	This attribute gives the index of the axis of which values that are stored in the record. swRecordVIndex refers to the symbolic names of the iterators for which the axis value shall be stored in the record. In case of nested iterators (mainly for multidimensional objects) the iterator names are specified as whitespace-separated names. These symbolic names relate to swRecordLayoutGroup Index. The iterators are processed from left to right in such a manner that they symbolize the loop index from the outside to the inside. It is considered an error if more components are specified than axes exist in the related ApplicationDataType. **Tags:** xml.sequenceOffset=40
swRecord LayoutVFix Value	Integer	0..1	attr	This attribute specifies the filler character for the current record layout, in the form of hex digits. It is also used to specify the fix value for e.g. FIXRIGHTDIFF. **Tags:** xml.sequenceOffset=80
swRecord LayoutVIndex	NameTokens	0..1	attr	The symbolic value for iteration, or the symbolic values separated by whitespaces, refer to the symbolic values given in swRecordLayoutGroupIndex . The iterators are processed from left to right, in such a manner that they symbolize the loop index from the outside to the inside. ▽

▽

△

Class	**SwRecordLayoutV**			
				△ It is considered an error if the record layout is referenced by an entity which has less number of axes than index names referenced here. **Tags:** xml.sequenceOffset=60
swRecord LayoutVProp	NameToken	0..1	attr	This attribute describes the kind of values to be stored. More details see below. The standardized values foreseen for this attribute are defined in [TPS_SWCT_01489]. **Tags:** xml.sequenceOffset=50

Table A.20: SwRecordLayoutV