ISO/TC171/SC2 N 570 E

Date: 2009-07-21

ISO/TC171/SC2
Document Management Applications
Application Issues
SECRETARIAT: ANSI

TITLE	**:**	**3 month WD ballot for Document management – 3D use of product representation compact (PRC) format – Part 1: Version 1**
SOURCE	**:**	**ISO/TC 171 SC2 Secretariat**
PROJECT	**:**	
STATUS	**:**	**Working draft**
REQUESTED ACTION	**:**	**Member countries are requested to review the draft and submit their votes via the electronic balloting system by 16 October 2009.**
DISTRIBUTION	**:**	**P, 0 and L Members**

Address Reply to:
Secretariat - ISO TC171 SC2- Association for Information and Image Management International
1100 Wayne Ave, Suite 1100, Silver Spring, MD 20910-5603
Telephone: 301-755-2682; Facsimile: 240-494-2682; e-mail: bfanning@aiim.org

ISO TC 171/SC 2 N

Date: 2008-11-8

ISO/WD

ISO TC 171/SC 2/WG 7

Secretariat: ANSI

Document management — 3D Use of Product Representation Compact (PRC) Format — Part 1: Version 1

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype:
Document stage: (20) Preparatory
Document language: E

Contents

Copyright notice 1

1 Scope 10

2 Normative references 10

3 Terms and definitions 10

4 Document Syntax Conventions 11
4.1 Conventions 11
4.2 Example Structure 11

5 PRC file concepts 12
5.1 The PRC file 12
5.2 Versioning 14
5.3 Unique Identifiers 15
5.3.1 General 15
5.3.2 File Structure 15
5.3.3 Base Entities 15
5.3.4 CAD systems 16
5.4 Current Data Values 16
5.5 UserData 16
5.6 Units 16
5.7 Tolerances 17
5.8 Compressed File Sections 17
5.9 Compressed Geometry 17
5.10 Compressed Tessellation 18

6 PRC File Contents 18
6.1 FileHeader 18
6.1.1 General 18
6.1.2 FileStructureDescription 19
6.1.3 UncompressedFiles 19
6.2 FileStructure 20
6.2.1 General 20
6.2.2 FileStructureHeader 21
6.2.3 FileStructureSchema 21
6.3 PRC Schema 22
6.3.1 General 22
6.3.2 Entity_schema_definition 22

7 Base Entities 22
7.1 General 22
7.2 Abstract Root Types 23
7.2.1 Entity Types 23
7.2.2 PRC_TYPE_ROOT 23
7.2.3 PRC_TYPE_ROOT_PRCBase 23
7.2.4 PRC_TYPE_ROOT_PRCBaseWithGraphics 24
7.2.5 PRC_TYPE_ROOT_PRCBaseNoReference 26
7.3 Structure and Assembly 26
7.3.1 Entity Types 26
7.3.2 PRC_TYPE_ASM 27
7.3.3 PRC_TYPE_ASM_ModelFile 27
7.3.4 PRC_TYPE_ASM_FileStructure 28
7.3.5 PRC_TYPE_ASM_FileStructureGlobals 29
7.3.6 PRC_TYPE_ASM_FileStructureTree 33
7.3.7 PRC_TYPE_ASM_FileStructureTessellation 34

7.3.8 PRC_TYPE_ASM_FileStructureGeometry 34
7.3.9 PRC_TYPE_ASM_FileStructureExtraGeometry 35
7.3.10 PRC_TYPE_ASM_ProductOccurrence 38
7.3.11 PRC_TYPE_ASM_PartDefinition 43
7.3.12 PRC_TYPE_ASM_Filter 44
7.3.13 CompressedUniqueId 45
7.4 Miscellaneous Data 45
7.4.1 Entity Types 45
7.4.2 PRC_TYPE_MISC 46
7.4.3 PRC_TYPE_MISC_Attribute 46
7.4.4 PRC_TYPE_MISC_EntityReference 48
7.4.5 PRC_TYPE_MISC_MarkupLinkedItem 48
7.4.6 PRC_TYPE_MISC_ReferenceOnPRCBase 49
7.4.7 PRC_TYPE_MISC_ReferenceOnTopology 50
7.4.8 PRC_TYPE_MISC_CartesianTransformation 52
7.4.9 PRC_TYPE_MISC_GeneralTransformation 52
7.4.10 ContentEntityReference 53
7.4.11 Transformation 54
7.5 Graphics 58
7.5.1 Entity Types 58
7.5.2 PRC_TYPE_GRAPH 59
7.5.3 PRC_TYPE_GRAPH_Style 59
7.5.4 PRC_TYPE_GRAPH_Material 60
7.5.5 PRC_TYPE_GRAPH_Picture 60
7.5.6 PRC_TYPE_GRAPH_TextureApplication 61
7.5.7 PRC_TYPE_GRAPH_TextureDefinition 62
7.5.8 PRC_TYPE_GRAPH_TextureTransformation 65
7.5.9 PRC_TYPE_GRAPH_LinePattern 66
7.5.10 PRC_TYPE_GRAPH_FillPattern 66
7.5.11 PRC_TYPE_GRAPH_DottingPattern 66
7.5.12 PRC_TYPE_GRAPH_HatchingPattern 67
7.5.13 PRC_TYPE_GRAPH_SolidPattern 67
7.5.14 PRC_TYPE_GRAPH_VpicturePattern 68
7.5.15 PRC_TYPE_GRAPH_AmbientLight 69
7.5.16 PRC_TYPE_GRAPH_PointLight 69
7.5.17 PRC_TYPE_GRAPH_DirectionalLight 70
7.5.18 PRC_TYPE_GRAPH_SpotLight 70
7.5.19 PRC_TYPE_GRAPH_SceneDisplayParameters 71
7.5.20 PRC_TYPE_GRAPH_Camera 71
7.6 Representation Items 72
7.6.1 Entity Types 72
7.6.2 PRC_TYPE_RI 73
7.6.3 PRC_TYPE_RI_RepresentationItem 73
7.6.4 PRC_TYPE_RI_BrepModel 74
7.6.5 PRC_TYPE_RI_Curve 74
7.6.6 PRC_TYPE_RI_Direction 75
7.6.7 PRC_TYPE_RI_Plane 75
7.6.8 PRC_TYPE_RI_PointSet 76
7.6.9 PRC_TYPE_RI_PolyBrepModel 76
7.6.10 PRC_TYPE_RI_PolyWire 77
7.6.11 PRC_TYPE_RI_Set 77
7.6.12 PRC_TYPE_RI_CoordinateSystem 77
7.7 Markup 78
7.7.1 Entity Types 78
7.7.2 PRC_TYPE_MKP 78
7.7.3 PRC_TYPE_MKP_View 79
7.7.4 PRC_TYPE_MKP_Markup 80
7.7.5 PRC_TYPE_MKP_Leader 82
7.7.6 PRC_TYPE_MKP_AnnotationItem 82
7.7.7 PRC_TYPE_MKP_AnnotationSet 83

7.7.8 PRC_TYPE_MKP_AnnotationReference ... 83
7.8 Tessellation ... 84
7.8.1 Entity Types ... 84
7.8.2 PRC_TYPE_TESS ... 84
7.8.3 PRC_TYPE_TESS_Base ... 84
7.8.4 ContentBaseTessData ... 84
7.8.5 PRC_TYPE_TESS_3D ... 85
7.8.6 PRC_TYPE_TESS_Face ... 91
7.8.7 PRC_TYPE_TESS_3D_Wire ... 93
7.8.8 PRC_TYPE_TESS_Markup ... 96
7.8.9 PRC_TYPE_TESS_3D_COMPRESSED ... 103
7.9 Topology ... 113
7.9.1 Entity Types ... 113
7.9.2 PRC_TYPE_TOPO ... 114
7.9.3 PRC_TYPE_TOPO_Context ... 114
7.9.4 PRC_TYPE_TOPO_Item ... 115
7.9.5 PRC_TYPE_TOPO_MultipleVertex ... 115
7.9.6 PRC_TYPE_TOPO_UniqueVertex ... 116
7.9.7 PRC_TYPE_TOPO_WireEdge ... 117
7.9.8 PRC_TYPE_TOPO_Edge ... 117
7.9.9 PRC_TYPE_TOPO_CoEdge ... 118
7.9.10 PRC_TYPE_TOPO_Loop ... 119
7.9.11 PRC_TYPE_TOPO_Face ... 120
7.9.12 PRC_TYPE_TOPO_Shell ... 121
7.9.13 PRC_TYPE_TOPO_Connex ... 122
7.9.14 PRC_TYPE_TOPO_Body ... 122
7.9.15 ContentBody ... 123
7.9.16 ContentWireEdge ... 123
7.9.17 PRC_TYPE_TOPO_SingleWireBody ... 124
7.9.18 PRC_TYPE_TOPO_BrepData ... 124
7.9.19 PRC_TYPE_TOPO_SingleWireBodyCompress ... 125
7.9.20 PRC_TYPE_TOPO_BrepDataCompress ... 125
7.9.21 PRC_TYPE_TOPO_WireBody ... 151
7.9.22 References ... 151
7.10 Curve ... 153
7.10.1 Entity Types ... 153
7.10.2 PRC_TYPE_CRV ... 153
7.10.3 PRC_TYPE_CRV_Base ... 153
7.10.4 PRC_TYPE_CRV_Blend02Boundary ... 154
7.10.5 PRC_TYPE_CRV_NURBS ... 157
7.10.6 PRC_TYPE_CRV_Circle ... 160
7.10.7 PRC_TYPE_CRV_Composite ... 162
7.10.8 PRC_TYPE_CRV_OnSurf ... 165
7.10.9 PRC_TYPE_CRV_Ellipse ... 166
7.10.10 PRC_TYPE_CRV_Equation ... 168
7.10.11 PRC_TYPE_CRV_Helix01 ... 169
7.10.12 PRC_TYPE_CRV_Hyperbola ... 174
7.10.13 PRC_TYPE_CRV_Intersection ... 176
7.10.14 PRC_TYPE_CRV_Line ... 183
7.10.15 PRC_TYPE_CRV_Offset ... 184
7.10.16 PRC_TYPE_CRV_Parabola ... 187
7.10.17 PRC_TYPE_CRV_PolyLine ... 189
7.10.18 PRC_TYPE_CRV_Transform ... 190
7.11 Surface ... 192
7.11.1 Entity Types ... 192
7.11.2 PRC_TYPE_SURF ... 193
7.11.3 PRC_TYPE_SURF_Base ... 193
7.11.4 PRC_TYPE_SURF_Blend01 ... 193
7.11.5 PRC_TYPE_SURF_Blend02 ... 195
7.11.6 PRC_TYPE_SURF_Blend03 ... 199

7.11.7 PRC_TYPE_SURF_NURBS 202
7.11.8 PRC_TYPE_SURF_Cone 205
7.11.9 PRC_TYPE_SURF_Cylinder 207
7.11.10 PRC_TYPE_SURF_Cylindrical 208
7.11.11 PRC_TYPE_SURF_Offset 209
7.11.12 PRC_TYPE_SURF_Pipe 210
7.11.13 PRC_TYPE_SURF_Plane 211
7.11.14 PRC_TYPE_SURF_Ruled 212
7.11.15 PRC_TYPE_SURF_Sphere 213
7.11.16 PRC_TYPE_SURF_Revolution 215
7.11.17 PRC_TYPE_SURF_Extrusion 217
7.11.18 PRC_TYPE_SURF_FromCurves 218
7.11.19 PRC_TYPE_SURF_Torus 219
7.11.20 PRC_TYPE_SURF_Transform 220
7.11.21 PRC_TYPE_SURF_Blend04 222
7.12 Mathematical Operator 222
7.12.1 Entity Types 222
7.12.2 PRC_TYPE_MATH 223
7.12.3 PRC_TYPE_MATH_FCT_1D 223
7.12.4 PRC_TYPE_MATH_FCT_1D_Polynom 223
7.12.5 PRC_TYPE_MATH_FCT_1D_Trigonometric 224
7.12.6 PRC_TYPE_MATH_FCT_1D_Fraction 224
7.12.7 PRC_TYPE_MATH_FCT_1D_ArctanCos 225
7.12.8 PRC_TYPE_MATH_FCT_1D_Combination 226
7.12.9 PRC_TYPE_MATH_FCT_3D 226
7.12.10 PRC_TYPE_MATH_FCT_3D_Linear 226
7.12.11 PRC_TYPE_MATH_FCT_3D_NonLinear 227

8 Other Data Classes 228
8.1 Other data classes 228
8.2 Parameter Range 228
8.2.1 Infinite_param 228
8.2.2 Interval 228
8.2.3 Parameterization 229
8.2.4 Domain 230
8.2.5 UVParameterization 230
8.3 BaseTopology 231
8.4 BaseGeometry 232
8.5 Basic types for geometry 232
8.5.1 Vector2d 232
8.5.2 Vector3d 233
8.5.3 BoundingBox 233
8.6 UserData 233
8.6.1 UserDataStream 234
8.6.2 UserDataSubSection 234

9 Schema Definition 234
9.1 General 234
9.2 Enumeration of Schema Tokens 236
9.3 Schema Processing 237
9.3.1 EPRCSchema_Data_Boolean 237
9.3.2 EPRCSchema_Data_Double 237
9.3.3 EPRCSchema_Data_Character 237
9.3.4 EPRCSchema_Data_Unsigned_Integer 237
9.3.5 EPRCSchema_Data_Integer 237
9.3.6 EPRCSchema_Data_String 237
9.3.7 EPRCSchema_Father_Type 237
9.3.8 EPRCSchema_ Vector_2D 237
9.3.9 EPRCSchema_ Vector_3D 237
9.3.10 EPRCSchema_Extent_1D 237
9.3.11 EPRCSchema_Extent_2D 237

9.3.12 EPRCSchema_Extent_3D 237
9.3.13 EPRCSchema_Ptr_Type 238
9.3.14 EPRCSchema_Ptr_Surface 238
9.3.15 EPRCSchema_Ptr_Curve 238
9.3.16 EPRCSchema_For 238
9.3.17 EPRCSchema_SimpleFor 239
9.3.18 EPRCSchema_If and EPRCSchema_Else 239
9.3.19 EPRCSchema_Block_Start 240
9.3.20 EPRCSchema_Block_Version 240
9.3.21 EPRCSchema_Block_End 240
9.3.22 EPRCSchema_Value_Declare 240
9.3.23 EPRCSchema_Value_Set 241
9.3.24 EPRCSchema_Value_DeclareAndSet 241
9.3.25 EPRCSchema_Value 241
9.3.26 EPRCSchema_Value_Constant 242
9.3.27 EPRCSchema_Value_For 242
9.3.28 EPRCSchema_Value_CurveIs3D 242
9.3.29 EPRCSchema_Operator_MULT 243
9.3.30 EPRCSchema_Operator_DIV 243
9.3.31 EPRCSchema_Operator_ADD 243
9.3.32 EPRCSchema_Operator_SUB 243
9.3.33 EPRCSchema_Operator_LT 243
9.3.34 EPRCSchema_Operator_LE 244
9.3.35 EPRCSchema_Operator_GT 244
9.3.36 EPRCSchema_Operator_GE 244
9.3.37 EPRCSchema_Operator_EQ 244
9.3.38 EPRCSchema_Operator_NEQ 244
9.4 Schema Examples 244
9.4.1 General 244
9.4.2 An existing entity 244
9.4.3 Existing PRC_TYPE_CRV_Polyline 245
9.4.4 Add a field to existing entity 246
9.4.5 Add a new curve 246
9.4.6 Multiple revisions to an entity type 247

10 Data Types for Physical File 247
10.1 General 247
10.2 Uncompressed Types 248
10.2.1 General 248
10.2.2 UncompressedFiles 248
10.2.3 UncompressedBlock 248
10.2.4 UncompressedUnsignedInteger 248
10.3 Compressed Types 249
10.3.1 General 249
10.3.2 Bits 249
10.3.3 Boolean 249
10.3.4 Character 249
10.3.5 CharacterArray 249
10.3.6 FloatAsBytes 249
10.3.7 String 249
10.3.8 ShortArray 250
10.3.9 Double 250
10.3.10 DoubleWithVariableBitNumber 250
10.3.11 Integer 250
10.3.12 IntegerWithVariableBitNumber 250
10.3.13 CompressedIntegerArray 250
10.3.14 UnsignedInteger 250
10.3.15 UnsignedIntegerWithVariableBitNumber 250
10.3.16 CompressedIndiceArray 250
10.3.17 NumberOfBitsThenUnsignedInteger 250

10.3.18 CompressedEntityType 250

11 I/O Algorithms 250
11.1 GetNumberOfBitsUsedToStoreUnsignedInteger 250
11.2 MakePortable32BitsUnsigned 251
11.3 WriteBits 251
11.4 WriteString 252
11.5 WriteFloatAsBytes 252
11.6 WriteCharacterArray 253
} 254
11.7 WriteShortArray 254
11.8 WriteCompressedIntegerArray 254
11.9 WriteCompressedIndiceArray 255
11.10 WriteUnsignedInteger 255
11.11 WriteInteger 256
11.12 WriteIntegerWithVariableBitNumber 256
11.13 WriteUnsignedIntegerWithVariableBitNumber 256
11.14 WriteDoubleWithVariableBitNumber 257
11.15 WriteNumberOfBitsThenUnsignedInteger 257
11.16 WriteCompressedEntityType 257
11.17 WriteDouble 259
11.17.1 General 259
11.17.2 Data definition for double storage 259
11.18 Procedure for WriteDouble 295

12 Tessellation Compression Support 299
12.1 General 299
12.2 Huffman Algorithm 299
12.3 Basis pseudocode 302

Bibliography 306

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 1000x was prepared by Technical Committee ISO/TC 171, *Document Management Applications*, Subcommittee SC 2, *Application Issues*.

Introduction

The data representations in PRC allows 3D design data, typically created in CAD and PLM systems, to be viewed and interrogated by visualization applications and to be integrated into complex documents.

This document specifies a wide range of data forms. The wide range is necessary to:

* Achieve a high fidelity, visually equivalent representation of 3D design data produced by an advanced CAD or PLM system without requiring the original application. * Allow applications to compute high accuracy product shape measurements.

PRC is intended to complement native or open standard CAD and PLM formats as a compact, concise binary form for visualization and documentation. PRC is not intended as a data format for CAD interoperability or use in factory automation systems, e.g. automated manufacturing and inspection systems, which is addressed by the ISO 10303 standards.

Document management — 3D Use of Product representation compact (PRC) format — Part 1: Version 1

1 Scope

This International Standard describes Version 1 of a product representation compact (PRC) file format for three dimensional (3D) content data. This format is designed to be included in PDF and other similar document formats for the purpose of 3D visualizaiton and exchange. It can be used for creating, viewing, and distributing 3D data in document exchange workflows. It is optimized to store, load, and display various kinds of 3D data, especially that coming from computer aided design (CAD) systems.

This International Standard does not apply to:

- Method of electronic distribution
- Converting CAD system generated datasets to the PRC format
- Specific technical design, user interface, implementation, or operational details of rendering
- Required computer hardware and/or operating systems

2 Normative references

The following referenced documents are indispensable for the application of this International Standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 12651: 1999, *Electronic imaging – Vocabulary*

ISO 24517-1: 2008, *Document management – Engineering document format using PDF – Part 1: Use of PDF 1.6 (PDF/E-1)*

ISO 32000-1: 2008, *Document management – Portable document format – Part 1: PDF 1.7*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 32000-1, ISO 24517-1 and ISO 12651 and the following apply.

4 Document Syntax Conventions

4.1 Conventions

The following conventions are used within this document to describe data within a PRC File.

Terms highlighted in bold within this document signify field names in the descriptions of entity types.

A table with three columns defines the data within a contiguous portion of the file.

The first column indicates if the data is required or is optional. Optional data is always preceeded by a flag indicating if the data is present or not. If the flag is a Boolean variable, the Option indicates what data is present if the flag is TRUE or is FALSE. If the flag is an integer, the Option indicates the class of data which is present and the flag indicates the number of items present. If the Option is an enumerated type, the Option indicates the type of data present for that value of the flag. If the flag is an UnsignedInteger, the Option indicates what array of data is present

The second column indicates the type of data that is stored. This might be

- a simple data type such as a Boolean, UnsignedInteger, or Double
- A simple class of data such as PRC_TYPE_TOPO_Body or PtrTopology where the name of the class is used to define the data stored for that class
- An ArrayOf [<data class>] which indicates an array of data of the specified class. Elements of an array are referenced beginning at 0.

The third column provides a description or other information relative to the field.

4.2 Example Structure

Table 1 — PRC file structure example

Required or Option	Data Type	Data Description
Required	Boolean	TRUE if the data is present; else FALSE
Option:TRUE	<data class 1>	Data if flag is TRUE
Option:FALSE	<data class 2>	Data if flag is FALSE
Required	PRC_TYPE_TOPO_Body	Data for a topological body
Required	<enumerated type> or integer	Flag indicating what type of data follows.
Option: enumeration 1	<data class 3>	Data for a specified value of the enumerated type
Option: enumeration 2	<data class 4>	Data for another value of the enumerated type

Required	UnsignedInteger	Number of members in array
Required	ArrayOf [<data class 5>]	An array whose members are of the specific <data class>

5 PRC file concepts

5.1 The PRC file

A PRC File is a sequential binary file, written in a way to make the file portable across machine architectures and operating systems.

PRC is optimized to store various kinds of 3D data, especially those coming from computer-aided design (CAD) systems. Most of the main constructs of CAD systems are supported within the PRC File Format:

- Assemblies and parts
- Trees of 3D entities (coordinate systems, wireframes, surfaces, and solids)
- Exact geometry representation for curves, surfaces
- Tessellated (triangulated) representation
- Markup data

PRC is meant to be multipurpose. There are two ways to store exact geometry and tessellation depending on the usage of the file and on the original information:

- Regular compression is used to directly represent CAD data without loss or transformation from the originating CAD system.
- High compression is used to store very small files, which have a specified physical tolerance from the originating shape. The tolerance is typically 0,001 mm for exact geometric data and 0,01 mm for tessellation data.

Each PRC File corresponds to a single model file (7.3.3) which defines the root product occurrences within the FileStructures of the PRC File. A PRC File is a collection of FileStructures which are independent from each other and can come from various authoring PRC File Writers. A FileStructure is the representation of an independant physical file denoting an independent 3D part, assembly, etc. within a PRC file. Hence, there is one header for each FileStructure with specific information and one global header for the model file which contains information about the PRC File Writer that assembled all these individual FileStructures into the final PRC File. Header sections gather primarily information on file version, FileStructure ids and offsets for reading / skipping sections.

Each FileStructure contains one or more product occurrences (7.3.10.2). The product occurrences denote the assembly hierarchy of the FileStructure. A product occurrence can have child nodes (called *son nodes*), which are also product occurrences. There is exactly one root product occurrence in each FileStructure. The root product occurrence is the only entry point to the FileStructure. (refer to Figure 1 below)

In parallel to the FileStructures, the model file also contains an array of root product occurrences that comprise the starting point for the entire assembly description. A product occurrence may refer to a corresponding part definition (see 7.3.11) which in turn contains:

- Geometrical data stored in a tree of representation items (see 7.6.3)
- An optional tree of part markups, grouped into annotation entities and views (see 7.7.2)

A product occurrence can contain:

- A tree of product markups
- Filters used to redefine the loading and presentation of data defined in the son product occurrences or in the part definition (see 7.3.12)

The representation items are defined through a combination of tessellation or topology and geometry data, which may be highly compressed. The markups are defined by tessellation data.

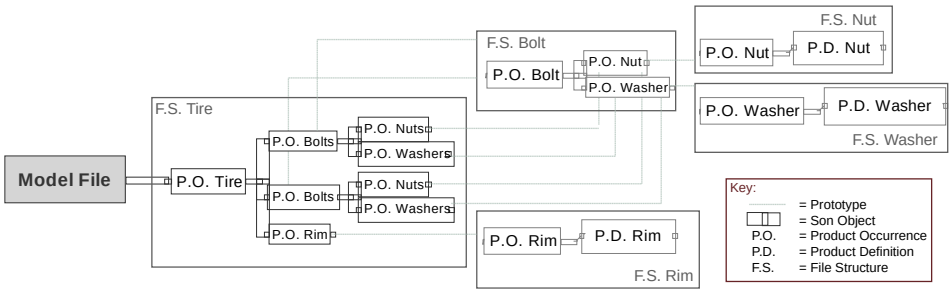


Figure 1 — Tessellation data

To optimize reading, a PRC File is arranged so that referenced entities are read before being referenced. Therefore, the FileStructures are ordered using parts, then subassemblies, and finally, the top (root) assembly.

A PRC File is composed of one header section, which starts with uncompressed data, one or more FileStructures, and one model file (PRC_TYPE_ASM_ModelFile) data section at the end, each individually compressed. Refer to 7.3.10 for more information about the PRC file structure.

Section	Sub Sections	Compression	Description
FileHeader		Uncompressed	Defines originating author of the file; specifies start, and possibly end, location of other sections in file
File Structure 1	FileStructureHeader	Uncompressed	Identifiers of other File Structures referenced from within this FileStructure;
	Schema	Compressed	Description of changes between minimal version and authoring version of the FileStructure within the PRC File Format Specification
	Globals	Compressed	Referenced FileStructures and colors, line styles, and coordinate systems for each tree entity of the FileStructure
	Tree	Compressed	a description of the tree of items (product occurrences, part definitions, representation items, and markup)
	Tesselation	Compressed	All tessellated (triangulated) data in the leaf entities of the tree (representation items and

			markups).
	Geometry	Compressed	All exact geometry and topology data of the leaf entities of the tree (representation items)
	Extra Geometry	Compressed	Geometry summary data, which allow for partial loading of the FileStructure without loading the entire geometry
.			Additional FileStructure sections in the PRC File
File Structure N		Compressed	Last FileStructure section in the PRC File
Schema	ModelFile Schema	Compressed	Description of changes between minimal version and authoring version of the ModelFile Section of the PRC File Format Specification
Model File Data	PRC_TYPE_ASM_ModelFile	Compressed	

5.2 Versioning

PRC file versions have at least 4 digits. The first digit is the year modulo 2000, and the next three digits represent the number of the day in the year. For example 10300 represents the 300th day of the year 2010.

A PRC File Writer is a software application which writes a particular PRC file. Similarly, a PRC File Reader is a software application which reads a particular PRC file.

A file version is used to define the particular version of the PRC Format Specification that a PRC File Reader or PRC File Writer comforms to. For instance, this draft version of the PRC Format Specification is file version 8137 <<Editor's note: to be updated upon submission>>. The **current_version** is the maximum file version that a PRC File Reader software or PRC File Writer software conforms to.

The **authoring_version** represents the version of the PRC Format Specification that the PRC File Writer conformed to at the time data contained in the PRC File was written. (It can also represent a FileStructure within a PRC File. Each file structure within a PRC file is independant and can be moved to another PRC file without parsing. It is thereby possible for any FileStructure within a PRC File to be written to a different PRC Format Specification).

The **minimal_version_for_read** represents the minimal file PRC Format Specification version that a PRC File Reader can successfully read. If a PRC File Reader conforms to a PRC Format Specification with a current_version number lower than this minimal_version_for_read an error should occur.

If minimal_version_for_read is lower than authoring_version, the PRC File Writer must write a schema description in the PRC File providing the differences between the two PRC Format Specifications. This description will enable a PRC File Reader to read and skip new information, since these new data cannot be interpreted as they are from a newer version.

A PRC File schema contains a description of new fields or new entity types added between the minimal_version_for_read and the authoring_version. See 6.3 for a description of a schema.

A PRC File Reader uses the information in the schema to read and skip new information:

- After reading each entity type, the schema information is queried and new data are skipped accordingly, following the tokens.
- Each time an entity type is read, if the type is unknown, the schema is searched and its data is skipped.

5.3 Unique Identifiers

5.3.1 General

A PRC File reader/writer must generate (writer) or interprete (reader) unique identifiers for information within the PRC File. Each FileStructure within a PRC File has an identifier (UUID) which uniquely identifies this particular FileStructure among all of the FileStructures within the PRC File. Within each FileStructure, unique identifiers for referenceable entities are generated using the order that they are first encountered in the FileStructure. Thus, for instance, the first referenceable entity in the FileStructure has the number 1 as its identifier. The second referenceable entitity has the number 2.

5.3.2 File Structure

A PRC FileStructure is identified by an identifier (see **FileStructureUncompressedUniqueID**) which is unique among all of the FileStructures within a single PRC File.

The method to calculate a unique identifier for a given FileStructure is not part of the PRC Format Specification. However, the last 32-bit unsigned integer of the UUID is interpreted as seconds elapsed from Jan 1st 1970. This date represents the date of creation of the PRC FileStructure. Since first PRC specification was created in June 2006, this value should be greater than 1149770868. This restriction may be removed in future versions. This methodology provides a good compromise between enforcing unique UUIDs (which can become complicated) and avoiding conflicts in practice, thereby enabling different organizations to have different strategies to handle UUIDs and still avoid conflicts in practice.

The rest of the calculation method is left to the user. This approach offers an advantage, for example, such as when there is some intent to repurpose FileStructures inside other PRC files without entirely rewriting them.

5.3.3 Base Entities

The PRC format provides support for referencing entities. See Entity Types in 7.2.1 for a list of entity types whose entities are referenceable.

The purpose of using references on entities is to enable users to handle the same entity several times without any duplication of data, either by any other program, or by another structure in the same or another PRC file.

A referenceable entity is retrieved using the following:

- The UUID of the FileStructure.
- The entity's unique identifier (a non-zero unsigned 32-bit integer) within the FileStructure.

Just as the FileStructure UUID is unique among all of the FileStructures in the PRC File, the entity identifier is unique inside a FileStructure for all referenceable entities.

A PRC File Writer must ensure that two referenceable entities inside the same FileStructure do not have the same identifier. The next available index within a FileStructure is the maximum value that has been assigned for identifiers to date, and is stored in the PRC File (see 7.3.4) so it is possible to safely add new entities to a FileStructure and assign them a unique identifier greater than this maximum value.

Data within the FileStructure tessellation and geometry sections are not accessible from outside the FileStructure using the approach for referenceable entities. These data are referenced only by data within the

tree section (see 5.1 above) of the same FileStructure. However, it is possible (see 7.4.7) to reference topological data from outside the particular FileStructure which the topological entity lies in. This is restricted to faces in the current version of PRC but may be extended to other data in future versions of the format.

5.3.4 CAD systems

In addition to the unique identifier mechanisms described above, the PRC File Format Specification provides for storage of identifiers indicating the originating CAD system. This is done using two other identifiers indicating whether the identifier is persistent. An identifier is persistent if it remains the same even if the CAD file was modified, as long as the corresponding item was not destroyed. CAD identifiers and their persistence are stored as information in PRC strictly to support external workflows. These identifiers by themselves only exist to convey information and do not play any role in PRC.

5.4 Current Data Values

A PRC File reader/writer must implement the concept of **CURRENT** for various data. This enables smaller file sizes since duplicate data need not be written to the file. It also enablesfaster readers because some of the data is not being read. Default initial values are in parentheses.

- Current name (NULL)
- Current graphics
    - Index of layer (-1)
    - Index of line style (-1)
    - Behavior bit field (-1)

Current values are updated as they are encountered in the file. Values are reset when serialization is flushed at the end of every flate-compressed section (see flate compression).

5.5 UserData

The PRC File format provides a mechanism for a PRC File Writer to write private data within various sections of a PRC File. Such data consists of writing a bit stream of data containing the size of the bit stream followed by the specified number of bits. Thus, any PRC File Reader can read the bit stream and can even resave the private data, but it may not be able to interpret the data.

Each FileStructure within a PRC File may contain UserData. UserData are defined in conjunction with an application unique identifier (UUID) which allows for their interpretation. This application identifier is stored in the FileStructurerHeading. By default, any user data within the FileStructure is assumed to be written by this one application. However, UserData may contain streams from different applications. Therefore, different application UUIDs are stored accordingly with each of these other data. UserData are meant to be interpreted only by software which is aware of its meaning according to a given authoring application UUID. A conforming PRC File Reader can either ignore those UserData, or interpret them according to their application UUID. PRC also allows applications to define special attributes which behave similarly to UserData, as discussed in 7.4.3.

Unique application identifiers are assigned through Adobe Systems, initially, and probably ISO in the long term. <<Editor's note: this is for determination by ISO>>. Each company should have it's own unique application identifier, and if they want to share data with another company, they should share application identifiers.

5.6 Units

It is mandatory that the units of a PRC File be defined. This should be done at both the model file level (see 7.3.3) AND at the product occurrence level (see 7.3.10). The unit can come from an actual CAD file and can

therefore be considered reliable to represent physical values in the data. If a unit is valid/reliable, the flag **unit_from_CAD_file** must be set to true. However, some formats (e.g. stl) do not contain any unit. Regardless, it is mandatory to define one unit at both the model file level AND at the product occurrence level .

The unit used is the first valid unit in the ModelFile / ProductOccurrence chain. If a valid unit is defined at the ModelFile level, it will apply to all product occurrences. Once a valid unit is found, the remainder of the data is interpreted with respect of that unit, even for occurrences higher in the product occurrence hierarchy. In other words, if a product occurrence having no valid unit has a son with a valid unit, it is assumed that the entire hierarchy of model file and product occurrence are to be interpreted and used according to this unit. If no valid unit is found at either the ModelFile level or any ProductOccurrence level (i.e. **unit_from_CAD_file** is always set to false), a conforming PRC File Reader must clearly indicate that the unit defined is not valid for measurement purposes.

5.7 Tolerances

PRC distinguishes between several notions of tolerances:

- A first notion of tolerance represents the maximum deviation between compressed and original data, as introduced by the lossy compression of geometry or topology. When provided, this tolerance is a user-defined value which represents a physical length given with a unit.

- A second notion of tolerance is introduced by numerical uncertainty inherent in every 3D modelling system. This form of tolerance is non-dimensional. Its purpose is to perform consistent numerical operations. This tolerance value corresponds to coincidence (e.g. of two 3D vertices) and is generally defined in conjunction with a minimal value representing zero and a maximal value representing infinity. For instance, a system might define coincidence at 1e-3, zero as any value less than 1e-12 and infinity as any value greater than 1e6. Then, additional logic outside the modelling system should define the unit so that the numerical values can be interpreted by the computer as physical values (i.e. in a particular unit).

In PRC, the 3D modelling system corresponds to entities in tessellation section (7.8) and topology section (7.9). Hence the various tolerances stored within these sections are always numerical values with no unit. PRC never stores a tolerance which can be directly interpreted as a physical value. For instance, **brep_data_compressed_tolerance** in 7.9.18 which represents the deviation introduced between original data and compressed one is stored without unit, even if it might be derived from a user interface which takes those units into account. Then, outside this 3D modelling system, unit is defined in ProductOccurrences and ModelFile as discussed in previous chapter. The physical interpretation of those tolerances should then be done from both indications. For instance, if a tolerance in a topology section is stored as 0,001 and if the unit of the ProductOccurrence it belongs to is 1000. (meter), then the physical tolerance on data is actually 1 millimeter.

5.8 Compressed File Sections

Within a PRC File, all sections, except header sections, are individually compressed with a Flate method. The Flate method is based on the public-domain zlib/deflate compression method, which is a variable-length Lempel-Ziv adaptive compression method cascaded with adaptive Huffman coding. It is fully defined in Internet RFCs 1950, *ZLIB Compressed Data Format Specification*, and 1951, *DEFLATE Compressed Data Format Specification* (see the Bibliography).

This form of compression is considered to be "lossless“. It occurs systematically whatever the actual content of the PRC file, and even if it contains compressed geometry or tessellation.

5.9 Compressed Geometry

Compression of geometry results in very small files. This compression is "lossy“ in that the geometry is an approximation to geometry to within a specified tolerance (typically 0,001mm). See 7.9.20 and 7.9.19 for entities representing compressed geometry. Note that the methodology to determine an approximation of the

original geometry (e.g. analytic recognition) is not part of PRC standard. Only the resulting entities and the method to store them are described in PRC.

5.10 Compressed Tessellation

Compression of tessellation data results in very small files. This compression is "lossy“ in that the tessellation data is an approximation to tessellation data to within a specified tolerance (typically 0,01mm). See 7.8.9 for an entity representing compressed tessellation. Tessellation data is not necessarily generated from or considered as an “approximation“ of geometry; it is an alternative way to convey data. Note that the methodology to determine an approximation of the original tessellation (e.g. polygon decimation) is not part of PRC standard. Only the resulting entities and the method to store them are described in PRC.

6 PRC File Contents

6.1 FileHeader

6.1.1 General

The Header contains the file version for the authoring version (PRC Format Specification) that the PRC File Writer is based on and the minimal version for read (PRC Format Specification) that a conforming PRC File Reader is based on that write/read the data outside of the individual FileStructures in the PRC File.

Each FileStructure has a identifier which is unique among all of the FileStructures within this PRC File.

Each application has a unique identifier which enables the interpretation of UserData. This identifier must be set to 0000 or to a valid application identifier. 0000 indicates that the authoring application is not “registered“ but (as discussed above) there can still be user data from another application in the file. Valid identifiers are distributed by Adobe Systems, Inc. upon request. <<Editor’s note: this is for determination by ISO.>>

A valid PRC File must contain at least one FileStructure.

Required or Option	Data Type	Data Description
Required	3 bytes: P, R, C	“PRC”
Required	UncompressedUnsignedInteger	Minimal version for read
Required	UncompressedUnsignedInteger	Authoring version
Required	FileStructureUncompressedUniqueId	Unique ID for file structure; The first and last sections of PRC File (before and after the FileStructures themselves) are a kind of special file structure which bear an ID as well.
Required	FileStructureUncompressedUniqueId	Unique ID for application
Required	UncompressedUnsignedInteger	Number of FileStructures in PRC File
Required	ArrayOf [FileStructureDescription]	Information describing each FileStructure in PRC File; the

		ordering of this array reflects the ordering of the FileStructure within the file.
Required	UncompressedUnsignedInteger	Start offset of ModelFileData section from beginning of PRC File (in bytes)
Required	UncompressedUnsignedInteger	End offset of ModelFileData section from beginning of PRC File (in bytes)
Required	UncompressedUnsignedInteger	Number of uncompressed files that are saved in the PRC File
Optional	ArrayOf [UncompressedFiles]	Array of uncompressed files stored in the PRC File

6.1.2 FileStructureDescription

The FileStructureDescription contains information defining a particular FileStructure within the PRC File:

- Each FileStructure has an identifier which is unique among the FileStructures within a PRC File.
- The starting offset (in bytes from the beginning of the PRC File) of each of the following sections within a FileStructure: header, globals, tree, tessellation, geometry, and extra geometry.

Required	UncompressedUniqueId	Unique id for this FileStructure
Required	UncompressedUnsignedInteger	Reserved; must be 0
Required	UncompressedUnsignedInteger	Number of sections in this FileStructure (6 for header, globals, tree, tesselation, geometry, and extra geometry)
Required	ArrayOf [UncompressedUnsignedInteger]	Start offset (in bytes) of each section from the beginning of PRC File

6.1.3 UncompressedFiles

Directly embeds private data inside a PRC File. This may be referenced by objects in the same PRC File using the index of the uncompressed file within this array, and interpreted accordingly. Up to this version of the PRC File Format Specification, only picture objects make reference to these files (see 7.5.5)

Required	UncompressedUnsignedInteger	Number of uncompressed files

Required	ArrayOf [UncompressedBlock]	Arbitrary data to embed within a PRC File

6.2 FileStructure

6.2.1 General

A file structure is comprised of one **FileStructureHeader** section, a **Schema** section and the following data sections:

- **Globals:** referenced file structures and colors, line styles, and coordinate systems for each tree entity of the file structure.
- **Tree:** a description of the tree of items (product occurrences, part definitions, representation items, and markup).
- **Tessellation:** all tessellated (triangulated) data in the leaf entities of the tree (representation items and markups).
- **Geometry:** all exact geometry and topology data of the leaf entities of the tree (representation items).
- **Extra geometry:** geometry summary data, which allow for partial loading of the file structure without loading the entire geometry.

Required or Option	Data Type	Data Description
Required	FileStructureHeader	
Required	FileStructureSchema	Define the schema for the entities in this FileStructure which have changed between the minimal_version_to_read and the authoring_version
Required	PRC_TYPE_ASM_FileStructureGlobals	Referenced FileStructures and colors, line styles, and coordinate systems for each tree entity of the file structure
Required	PRC_TYPE_ASM_FileStructureTree	a description of the tree of items (product occurrences, part definitions, representation items, and markup)
Required	PRC_TYPE_ASM_FileStructureTessellation	All tessellated (triangulated) data in the leaf entities of the tree (representation items and markups).

Required	PRC_TYPE_ASM_FileStructureGeometry	All exact geometry and topology data of the leaf entities of the tree (representation items)
Required	PRC_TYPE_ASM_FileStructureExtraGeometry	Geometry summary data, which allow for partial loading of the file structure without loading the entire geometry

6.2.2 FileStructureHeader

A FileStructureHeader defines various properties of a FileStructure:

- The minimal file version of the a conforming PRC File Reader;
- The authoring version of a conforming PRC File Writer that wrote this FileStucture;
- The unique ID of this FileStructure; this ID must be the same as the identifier in the PRC File Header section;
- The unique ID of the conforming PRC File Writer;
- A variable number of private uncompressed data files.

Required or Option	Data Type	Data Description
Required	3 bytes (P, R, C)	"PRC"
Required	UncompressedUnsignedInteger	Minimal version to read
Required	UncompressedUnsignedIngeger	Authoring version
Required	FileStructureUncompressedUniqueId	Unique ID of this FileStructure
Required	FileStructureUncompressedUniqueId	Unique ID for application
Required	UncompressedUnsignedInteger	Number of uncompressed files
Required	ArrayOf [UncompressedFile]	Array of uncompressed data

6.2.3 FileStructureSchema

Each FileStructure in a PRC File may represent a different version of the PRC Format Specification written by a different application (PRC File Writer). Each FileStructure must define the schema for changes (new entities and new data fields to existing entities) for the entities that are stored within it. See 6.3 for a description of the facilities for describing a schema.

6.3 PRC Schema

6.3.1 General

A schema describes changes between versions of PRC File Format Specification. See 6 for a basic description of versioning in PRC. This mechanism allows information to be added for a given entity type (**schema_type**) and still be readable by previous versions of the software.

Only those types which have changed between the **minimal_version_for_read and authoring_version** require a schema description, and only if they are present in the file.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number of entity types that have changed between minimal version for read and authoring version
Required	ArrayOf[Entity_Schema_definition]	Schema definition of each type that has changed

6.3.2 Entity_schema_definition

This provides the schema definition for an entity type in a PRC File. The entity is described by an array of schema tokens which in turn should be viewed as a list of versioned blocks which describe versions of the entity. See 9.3.20 for a description.

Entity_type represents the entity type, such as 7.10.14, that is being described by the array of schema tokens.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Entity_type
Required	UnsignedInteger	Number of tokens describing this entity type
Required	ArrayOf[UnsignedInteger]	Array of schema tokens describing this entity type

7 Base Entities

7.1 General

BASE ENTITIES represent high level concepts such as curves, surfaces, topology, parts, assemblies, markups, or tessellation data which are stored in a PRC File. Entities are defined by a type name used for

descriptive purposes, a type value which is stored in the PRC File to indicate that the data defining the entity follows, and an indication if the entity is referenceable. An entity is referenceable if it may be referenced using a unique identifier.

7.2 Abstract Root Types

7.2.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_ROOT	0	N/A
PRC_TYPE_ROOT_PRCBase	PRT_TYPE_ROOT + 1	N/A
PRC_TYPE_ROOT_PRCBaseWithGraphics	PRT_TYPE_ROOT + 2	N/A
PRC_TYPE_ROOT_PRCBaseNoReference	PRT_TYPE_ROOT + 3	N/A

7.2.2 PRC_TYPE_ROOT

An entity of this type in a PRC File is to be interpreted as a NULL pointer or entity depending on the specific circumstances.

7.2.3 PRC_TYPE_ROOT_PRCBase

7.2.3.1 General

This is the abstract root type for any PRC entity that can be referenced.

7.2.3.2 ContentPRCBase

This represents common data for all PRC base entities. All PRC base entities have attribute and name information. However, only entities which are eligible to be referenced will have a PRC FileStructure unique identifier and a persistent and non-persistent identifier from the originating CAD file.

Required or Option	Data Type	Data Description
Required	AttributeData	Attribute data associated with the entity
Required	Name	Entity Name
OPTION: if type of entity is elegible for reference	Unsignedinteger	non-persistent CAD identifier from originating CAD file
OPTION: if type of entity is elegible for reference	Unsignedinteger	persistent CAD identifier from originating CAD file
OPTION: if type of entity is elegible for reference	Unsignedinteger	PRC FileStructure unique identifier

7.2.3.3 AttributeData

A base entity may have zero or more associated attributes.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number of attributes
Required	ArrayOf [PRC_TYPE_MISC_Attribute]	An array of attribues

7.2.3.4 Name

PRC employs the concept of a current_name which retains the name of the last entity being read or written. If the name of the subsequent entity being read or written is the same as the current name, no name will be in the file. Otherwise, a name for the entity is read from the file and the current name is updated to this new name. This is done to optimize on space within the file.

Required or Option	Data Type	Data Description
Required	Boolean	TRUE implies the name of this entity is the same as the current name; FALSE implies that a new name is in the file
OPTION: FALSE	String	Name of the entity if it is not the same as the current name; this name will become the current name

7.2.4 PRC_TYPE_ROOT_PRCBaseWithGraphics

7.2.4.1 General

Information for any base PRC entity which can be referenced and which contains graphics.

PRC employs the concept of current graphics content which retains the graphics content of the last entity being read or written. If the graphics content of the subsequent entity being read or written is the same as that the current graphics content, no graphics content will be in the file. Otherwise, a graphics content for the entity is read from the file and the current graphics content is updated to this new graphics content. This is done to optimize on space within the file.

Required or Option	Data Type	Data Description
Required	ContentPRCBase	Base information associated with the entity
Required	Boolean	SameGraphicsAsCurrent
OPTION:FALSE	GraphicsContent	Graphical data associated with the entity

7.2.4.2 GraphicsContent

Layer_index represents the layer the entity lies on. It should have a value less than 65535.

index_of_line_style represents the index into the array of styles, which is stored in **FileStructureInternalGlobalData (**See section 7.3.5.2 below). The array of styles is a set of **PRC_TYPE_GRAPH_Style** entities (See 7.5.3) and should have a value less than 65535. (See section 7.5.3)

behaviour_bit_field is an unsigned short integer (2 bytes). This bit field defines the behavior of a given entity, given its position in the tree of entities. The inheritance works as follows:

If the **SonHerit** flag is set, the value corresponds to the youngest son.
If the **SonHerit** flag is not set but the **FatherHerit** flag is set, the value corresponds to the oldest father.
If no flag is set, the value is set to the son, if any exists.

Potential values and meanings of this bit field are:

- **PRC_GRAPHICS_Show** 0x0001
  The entity is shown.
- **PRC_GRAPHICS_SonHeritShow** 0x0002
  Shown entity son inheritance.
- **PRC_GRAPHICS_FatherHeritShow** 0x0004
  Shown entity father inheritance.
- **PRC_GRAPHICS_SonHeritColor** 0x0008
  Color/material son inheritance.
- **PRC_GRAPHICS_FatherHeritColor** 0x0010
  Color/material father inheritance.
- **PRC_GRAPHICS_SonHeritLayer** 0x0020
  Layer son inheritance.
- **PRC_GRAPHICS_FatherHeritLayer** 0x0040
  Layer father inheritance.
- **PRC_GRAPHICS_SonHeritTransparency** 0x0080
  Transparency son inheritance.
- **PRC_GRAPHICS_FatherHeritTransparency** 0x0100
  Transparency father inheritance.
- **PRC_GRAPHICS_SonHeritLinePattern** 0x0200
  Line pattern son inheritance.
- **PRC_GRAPHICS_FatherHeritLinePattern** 0x0400
  Line pattern father inheritance.
- **PRC_GRAPHICS_SonHeritLineWidth** 0x0800
  Line width son inheritance.
- **PRC_GRAPHICS_FatherHeritLineWidth** 0x1000

Line width father inheritance.

- **PRC_GRAPHICS_Removed** 0x2000
  The entity has been removed and no longer appears in the tree.

Required or Optional	Data Type	Data Description
Required	UnsignedInteger	Layer_index+1
Required	UnsignedInteger	Index_of_line_style+1
Required	Unsigned Character	Behavior_bit_field
Required	Unsigned Character	Behavior_bit_field >> 8

7.2.5 PRC_TYPE_ROOT_PRCBaseNoReference

This is the abstract root type for any PRC entity that can not be referenced.

This type is useful for schema descriptions. For example, **EPRCSchema_Father_Type** can be used to define a new type which has one of three different ancestor types. It can be a son of either **PRC_TYPE_RootPRCBase**, **PRC_TYPE_RootPRCBaseWithGraphics** , or **PRC_TYPE_RootPRCBaseNoReference**.

- **PRC_TYPE_RootPRCBase:** ancestor type would indicate that the new type is a referencable type
- **PRC_TYPE_RootPRCBaseNoReference:** ancestor type would indicate that the new type is not referencable.
- **PRC_TYPE_RootPRCBaseWithGraphics:** ancestor type would indicate that the new type is a referencable type which bear graphics

Examples :

PRC_TYPE_GRAPH_LinePattern has **PRC_TYPE_RootPRCBase** in its ancestor chain

PRC_TYPE_ASM_FileStructure has **PRC_TYPE_ROOT_PRCBaseNoReference** in its ancestor chain

PRC_TYPE_RI_Curve has PRC_TYPE_RootPRCBaseWithGraphics in its ancestors chain

7.3 Structure and Assembly

7.3.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_ASM	PRC_TYPE_ROOT + 300	
PRC_TYPE_ASM_ModelFile	PRC_TYPE_ASM + 1	
PRC_TYPE_ASM_FileStructure	PRC_TYPE_ASM + 2	
PRC_TYPE_ASM_FileStructureGlobals	PRC_TYPE_ASM + 3	
PRC_TYPE_ASM_FileStructureTree	PRC_TYPE_ASM + 4	
PRC_TYPE_ASM_FileStructureTessellation	PRC_TYPE_ASM + 5	

PRC_TYPE_ASM_FileStructureGeometry	PRC_TYPE_ASM + 6	
PRC_TYPE_ASM_FileStructureExtraGeometry	PRC_TYPE_ASM + 7	
PRC_TYPE_ASM_ProductOccurrence	PRC_TYPE_ASM + 8	yes
PRC_TYPE_ASM_PartDefinition	PRC_TYPE_ASM + 9	yes
PRC_TYPE_ASM_ModelFile	PRC_TYPE_ASM + 10	

7.3.2 PRC_TYPE_ASM

This is the abstract type for the top level PRC structure.

7.3.3 PRC_TYPE_ASM_ModelFile

7.3.3.1 General

A model file (**PRC_TYPE_ASM_ModelFile**) is typically created by importing data from a CAD file. There is only a single **PRC_TYPE_ASM_ModelFile** entity in a PRC File. The model file contains product occurrences, which are split into different FileStructures.

units_from_CAD_file is to be interpreted as discussed in section **Units**

number_of_product_occurrences is the number of root product occurrences in the model file.

product_occurrences represents the root product occurrences in the model file.

file_structure_index_in_model_file indicates the index at which the FileStructure should be stored in memory within a ModelFile (**PRC_TYPE_ASM_ModelFile**). A conforming PRC File Reader should reconstitute / maintain FileStructures in memory in the order that they appear in the ModelFile, regardless of the order in the physical PRC file. In the physical PRC file, the order of file structures must be in accordance with their dependancies with other file structures as follows: if a given file structure A depends on another file structure B (in the sense that A references elements of B), B must appear first in the PRC physical file. This restriction does not exist for the memory storage of the file structures.

number_of_file_structures is obtained directly from the header of the PRC File.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_ModelFile
Required	ContentPRCBase	Base information associcated with entities
Required	Boolean	Units_from_CAD_file is TRUE if the units come from a CAD system and are thus suitable for measurement; else FALSE
Option:	Double	Units in multiple of mm

TRUE		
Required	UnsignedInteger	number_of_root_product_occurrences
Required	ArrayOf[ProductOccurrenceReference]	References to the root product occurrences in the model file
Required	ArrayOf [UnsignedInteger]	`file_structure_index_in_model_file`: Indicies to to FileStructure within the model file; the size of the array is obtained directly from the header of the PRC Fie
Required	UserData	User defined data

7.3.3.2 ProductOccurrenceReference

This defines the unique identifier of the **PRC_TYPE_ASM_FileStructure** and the index of the root product occurence in the array of product occurrences within the **PRC_TYPE_ASM_FileStructureTree**, as contained within the **PRC_TYPE_ASM_FileStructure**.

Product_occurrence_is_active is reserved for future use, and is currently unused. Its default value should be TRUE. Its use will be to control storing different versions/configurations of a product in the same PRC File. Currently, only one can be active at a time but you can switch from one to the other.

Required or Option	Data Type	Data Description
Required	CompressedUniqueId	Unique identifier of the FileStructure that contains this root product occurrence.
Required	UnsignedInteger	Index of the root product occurrence within the FileStructure
Required	Boolean	TRUE if the product occurrence is active; else FALSE

7.3.4 PRC_TYPE_ASM_FileStructure

This type gathers internal data of a file structure as described in **PRC_TYPE_ASM_FileStructureTree**

The parameter **Next_available_index** is used when re-opening the FileStructure, to be able to safely add entities with a unique id without overlap to pre-existing entities.

The parameter **Index_product_occurrence** is used to denote which product occurrence inside the FileStructure is the unique root.

Required or Option	Data Type	Data Description

Required	UnsignedInteger	PRC_TYPE_ASM_FileStructure
Required	ContentPRCBase	Base information associated with entities
Required	UnsignedInteger	Next available index
Required	UnsignedInteger	Index of root product occurrence

7.3.5 PRC_TYPE_ASM_FileStructureGlobals

7.3.5.1 General

This type gathers global data of a file structure as described in Section 6.2.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_FileStructureGlobals
Required	ContentPRCBase	Base information associated with entities
Required	UnsignedInteger	Number of referenced FileStructures
Required	ArrayOf [CompressedUniqueId]	Unique ids for FileStructures within the PRC File which are referenced by entities in this FileStructure
Required	FileStructureInternalGlobalData	
Required	UserData	User defined data

7.3.5.2 FileStructureInternalGlobalData

7.3.5.2.1 General

This internal structure is used in **PRC_TYPE_Asm_FileStructureGlobals.**

Required or Option	Data Type	Data Description
Required	Double	Tessellation chord height
Required	Double	Tessellation angle (degrees)

Required	MarkupSerializationHelper	See below
Required	UnsignedInteger	Number of colors
Required	ArrayOf [RgbColor]	Array of color definitions
Required	UnsignedInteger	Number of pictures
Required	ArrayOf [PRC_TYPE_GRAPH_Picture]	Array of pictures
Required	UnsignedInteger	Number of textures definitions
Required	ArrayOf [PRC_TYPE_GRAPH_TextureDefinition]	Array of texture definitions
Required	UnsignedInteger	Number of materials
Required	ArrayOf [PRC_TYPE_GRAPH_Material]	Array of materials
Required	UnsignedInteger	Number of line patterns
Required	ArrayOf [PRC_TYPE_GRAPH_LinePattern]	Array of line patterns
Required	UnsignedInteger	Number of styles
Required	ArrayOf [PRC_TYPE_GRAPH_Style]	Array of category 1 line styles
Required	UnsignedInteger	Number of fill patterns
Required	ArrayOf [PRC_TYPE_GRAPH_FillPattern]	Array of fill patterns
Required	UnsignedInteger	Number of reference coordinate systems
Required	ArrayOf [PRC_TYPE_RI_CoordinateSystem]	Array of reference coordinate systems

7.3.5.2.2 MarkupSerializationHelper

7.3.5.2.2.1 General

This Global data is composed of font information for markup. The following example shows how the global data is serialized for markup font information.

- **default_font_family_name** defines the case-sensitive default font family name used for

text if a given font family is not available on the computer. If this default font family itself is not available on the computer, the font will be MyriadPro from Adobe Systems, Inc.

- **font_keys_of_font** represents several font keys sharing the same base font name.
  Indices for font keys used in **ContentMarkupTess** are calculated from this array.
  For example, if there are two font names representing 4 and 5 font keys respectively, index #7 would be represented by font_keys[1][3].

- **attributes** represents the font attributes, and is a combination of the values in the table below.

Required or Option	Data Type	Data Description
Required	String	default_font_family_name
Required	UnsignedInteger	Number of fonts
Required	ArrayOf[FontKeysSameFont]	Font_key_of_font[i]

7.3.5.2.2.2 FontKeySameFont

This type describes a list of usages of the same font (referred to by its name) with different metrics and attributes.

Required or Option	Data Type	Data Description
Required	String	Font Name
Required	UnsignedInteger	Character Set
Required	UnsignedInteger	Number-of-font-keys
Required	ListOf[UnsignedInteger, Character]	Font_key[i].Font-size + 1, Font-key[i].Attributes

The following table contains the possible values for the **Character Set**.

Value	Description
0	Roman
1	Japanese
2	Traditional Chinese

3	Korean
4	Arabic
5	Hebrew
6	Greek
7	Cyrillic
8	RightLeft
9	Devanagari
10	Gurmukhi
11	Gujarati
12	Oriya
13	Bengali
14	Tamil
15	Telugu
16	Kannada
17	Malayalam
18	Sinhalese
19	Burmese
20	Khmer
21	Thai
22	Laotian
23	Georgian
24	Armenian
25	Simplified Chinese
26	Tibetan
27	Mongolian
28	Geez
29	EastEuropeanRoman

30	Vietnamese
31	ExtendedArabic

attributes represents the font attributes, and is a combination of the following values.

Font attribute value	Description
2	Bold
4	Italic
8	Underlined
16	Strike-Out
32	Overlined
64	Stretch. In case the font to be used is not the original font, this attribute value indicates that the text must be stretched to fit within its bounding box<./TD>
128	Wire. This attribute value indicates that the original font is a wireframe font.

7.3.5.2.3 RgbColor

Color definition with 3 components.

Required or Option	Data Type	Data Description
Required	Double	Red
Required	Double	Green
Required	Double	Blue

7.3.6 PRC_TYPE_ASM_FileStructureTree

Each FileStructure within a PRC file has a FileStructureTree which defines

- the number of parts and part data
- the number of product occurrences and product occurence data

within this FileStructure.

By convention, the data within PRC Files are ordered so that data is defined before it is referenced. In the case of a FileStructureTree, part data is defined before product occurrence data which may refer to it, but within the array of part data, the order of the parts is immaterial. This is not the case for product occurrence data.

Product occurrence data represent an assembly, with a single product occurrence being the root. Each product occurrence (except the root) may be referenced by only one product occurrence. However, each

product occurrence may refer to multiple product occurrences. The array of product occurence data should be ordered so that any product occurrence is defined before the product occurrence referring to it.

The FileStructureInternalData defines the index of the root product occurrence and the next available index available to use when assigning unique index (identifiers) to referenceable entities within the FileStructure.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_FileStructureTree
Required	ContentPRCBase	Base information associated with entities
Required	UnsignedInteger	Number of part definitions
Required	ArrayOf [PRC_TYPE_ASM_PartDefinition]	An array of part definitions
Required	UnsignedInteger	Number of product occurrences
Required	ArrayOf [PRC_TYPE_ASM_ProductOccurrence]	An array of product occurrences
Required	PRC_TYPE_ASM_FileStructure	FileStructureInternalData
Required	UserData	User defined data

7.3.7 PRC_TYPE_ASM_FileStructureTessellation

This type gathers tessellation data of a file structure as described in section 6.2**.**

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_FileStructureTesssellation
Required	ContentPRCBase	Base information associated with entities
Required	UnsignedInteger	Number_of_tessellations
Required	ArrayOf[PRC_TYPE_TESS]	Content of all tessellations
Required	UserData	User defined data

7.3.8 PRC_TYPE_ASM_FileStructureGeometry

7.3.8.1 General

This type gathers geometry data of a file structure as described in section 6.2.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_FileStructureGeometry
Required	ContentPRCBase	Base information associcated with entities
Required	FileStructureExactGeometry	Topological context entities and their associated brep bodies
Required	UserData	User defined data

7.3.8.2 FileStructureExactGeometry

The FileStructureExactGeometry section consists of an array of "topological contexts". Each "topological context" contains an array of brep bodies contained within that topological context. Every geometrical and topological entity within a FileStructure may only belong to a single topological context.

A pair of indices (topological context, brep body) uniquely identifies a brep body within the topololological entities of a FileStructure.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number of topological contexts
Required	ArrayOf [TopologicalContext]	Array of topological contexts together with its associated brep bodies

7.3.9 PRC_TYPE_ASM_FileStructureExtraGeometry

7.3.9.1 General

The extra geometry data is summary data pertaining to the geometry which can be used to enable partial loading of the file structure without loading the entire geometry. This type gathers summary information of the exact geometry section, by topological context.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_FileStructureExtraGeometry
Required	ContentPRCBase	See Section **ContentPRCBase**
Required	UnsignedInteger	Number of extra geometry contexts
Required	ArrayOf [ExtraGeometry]	

Required	UserData	See Section 5.5 for details

7.3.9.1.1 ExtraGeometry

This is the summary of a topological context.

Required	GeometrySummary	Summary Data
Required	ContextGraphics	Graphics Context Data

7.3.9.1.2 GeometrySummary

7.3.9.1.2.1 General

This describes the summary list of the bodies of the topological context.

The **number_of_bodies** must be the same as the number of bodies in the context.

Required	UnsignedInteger	Number_of_bodies
Required	ArrayOf [BodyInformation]	Graphics information specific to each body

7.3.9.1.2.2 BodyInformation

This describes the summary information of a body.

The **body_serial_type** must be the type of the body that is in the topological context or it must be set to **PRC_ROOT_TYPE** if the body has no geometry. See the section 7.9.14 for details.

Required	UnsignedInteger	body_serial_type
Optional (if body_serial_type is one of the following: PRC_TYPE_TOPO_BrepDataCompress PRC_TYPE_TOPO_SingleWireBodyCompress PRC_TYPE_TESS_3D_Compress)	Double	tolerance which corresponds to the compression tolerance for the corresponding entity

7.3.9.1.3 ContextGraphics

7.3.9.1.3.1 General

This describes the summary list of the graphical attributes of entities within the topological context.

The following loop shows how to traverse a topological context to gather **GraphicInformation**. A **GraphicInformation** is gathered as soon as there is a graphic content provided for a particular entity :

```
For (i=0; i<number_of_body; i++) {
    If (body[i] is PRC_TYPE_TOPO_BrepData) {
        For (j=0; j<body[i].number_of_connex; j++) {
            For (k=0; k<body[i].connex[j].number_of_shell; k++) {
                For (l=0; l<body[i].connex[j].shell[k].number_of_face; k++) {
                    Add_to_output(body[i].connex[j].shell[k].face[l])
                }
            }
        }
    }
}
```

Number_of_treat_type corresponds to the number of entity types for which **GraphicsInformation** is stored (currently, only **PRC_TYPE_TOPO_Face** is supported, so if there are graphics on some faces, **Number_of_treat_type** is 1, else it is 0).

The current graphics as explained in 7.5 is reset prior to writing/reading the context graphics.

Required	UnsignedInteger	Number_of_treat_type
Required	ArrayOf[GraphicsInformation]	Graphic information for each type

7.3.9.1.3.2 GraphicsInformation

This describes the particular graphics for one particular type of the topological context.

The **number_of_element** represents the number of elements collected during a recursive search on the Context data (including duplicated elements) as shown in 7.5.

Required	UnsignedInteger	Element type
Required	UnsignedInteger	Number of elements
Required	ArrayOf[ElementInformation]	Graphics Information for each Element

7.3.9.1.3.3 ElementInformation

This describes the particular graphics for one particular element of the topological context.

Required	Boolean	**TRUE** if element[i] has graphics
Optional	ArrayOf[ElementGraphicsBehavior]	If element[i] has Graphics

7.3.9.1.3.4 ElementGraphicsBehavior

This describes graphics for one particular element of the topological context.

layer_index represents the index in the array of layers stored in **FileStructureInternalGlobalData.**

Index_of_line_style represents the index in the array of styles stored in **FileStructureInternalGlobalData**.

behaviour_bit_field is an unsigned short integer (2 bytes) . See Section **ContextGraphics** for details.

Required	Boolean	**TRUE** => use current graphic context
Optional (if Boolean = **FALSE**)	UnsignedInteger UnsignedInteger UnsignedCharacter UnsignedCharacter	layer_index + 1 Index_of_line_style + 1 Behavior_bit_field Behavior_bit_field >> 8

7.3.10 PRC_TYPE_ASM_ProductOccurrence

7.3.10.1 General

A product occurrence defines an assembly tree. In the case of a single part, the product occurrence points directly to a part definition (**PRC_TYPE_ASM_PartDefinition**). In the case of a more complex assembly, a product occurrence is comprised of a list of product occurences.

A product occurrence is comprised of the following data:

- Part definition: A pointer to the corresponding part definition. It can be null.
- Product prototype: A pointer to the corresponding product occurrence prototype. It can be null.
- External data: A pointer to the corresponding external product occurrence. It can be null.
- Sons: An array of pointers to the son product occurrences.

The product prototype is the product occurrence of a subassembly or part to be used in the parent assembly. This prototype acts as a template for a given product occurrence, and lets you link to information inside the subpart or assembly, such as geometry. When building assemblies from subassemblies, the tree of sons of the product occurrence is duplicated from the prototype description. For external data, the tree is only described inside the external data product occurrence. (refer to Figure 1 in Section 5.1)

When the assembly is heterogeneous (originating from different CAD systems), the link is specified through the external data rather than the prototype.

A product occurrence has at most one father.

Product behavior represents the various flags for the product. In this version of PRC, only **PRC_PRODUCT_BEHAVIOUR_SUPPRESSED** == 0x0001 is used. The other flags should be set to 0.

Location represents the transformation between the product occurrence and its father.

Entity_reference is the referenced entities with possible modifiers towards their nominal definition, which may include location, color, and visibility.

Markups represents the markups of this product occurrence (as opposed to part definition markups); these are grouped into annotations.

Views represents the views which can contain annotations or specific display parameters.

Entity_filter is a specific filter applied when loading data from product prototypes denoting sub-assemblies.

Display_filters represents the filters to use for display. Several filters can be specified, but only one can be active (see 7.3.12).

Scene_display_parameters is reserved for future use.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_ProductOccurrence
Required	PRCBaseWithGraphics	
Required	ReferencesOfProductOcurrence	
Required	Character	Behavior
Required	ProductInformation	
Required	Bit	TRUE if there is a transformation; else FALSE
OPTION:TRUE	Transformation	Location
Required	UnsignedInteger	Number of references
Required	ArrayOf [PRC_TYPE_MISC_EntityReference]	Array of entity references
Required	MarkupData	Markups for this product occurrence
Required	UnsignedInteger	Number of views
Required	ArrayOf [PRC_TYPE_MKP_View]	Array of view data
Required	Bit	TRUE if product occurrence has entity filter; else FALSE
OPTION: TRUE	PRC_TYPE_ASM_Filter	
Required	UnsignedInteger	Number of display filters
Required	ArrayOf [PRC_TYPE_ASM_Filter]	Array of display filters
Required	UnsignedInteger	Number of scene display parameters

Required	ArrayOf [PRC_TYPE_GRAPH_SceneDisplayParameter]	Array of scene display parameters
Required	UserData	User defined data

7.3.10.2 ReferencesOfProductOcurrence

7.3.10.2.1 General

For a given file structure, the product occurrences should be ordered based on the following criteria:

- A product prototype in the same file structure should be stored before any occurrences that use it.
- External data in the same file structure should be stored before any occurrences that use it.
- A son occurrence should be stored before its father.
- A part definition can be referenced several times in the same file structure.

(Refer to Figure 1 in Section 5.1.)

index_part represents the index of the part definition in the array of part definitions of the same FileStructure.

index_prototype represents the index of the product prototype in the FileStructure product occurrences array.

prototype_in_same_file_structure indicates whether the prototype is in the same FileStructure.

index_external_data represents the index of the external data.

external_data_in_same_file_structure indicates whether the external data is in the same FileStructure.

index_son_occurrence, which is mandatory in the same file structure, represents the index of the son product occurrence.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Index_part + 1
Required	UnsignedInteger	Index_prototype + 1
Option: (index_prototype != -1)	SaveFileIdentifier	Save prototype_in_same_file_structure
Required	UnsignedInteger	Index_external_data + 1
Option: (index_external_data != -	SaveFileIdentifier	Save external_data_in_same_file_structure

1)		
Required	UnsignedInteger	Number_of_son_product_occurrences
Required	ArrayOf[UnsignedInteger]	Array of index_son_occurrence

7.3.10.2.2 SaveFileIdentifier

This saves the identifier of the FileStructure that the prototype or external data if it is different from the FileStructure that the product occurrence lies in.

Required or Option	Data Type	Data Description
Required	Boolean	TRUE if the entity exists in the same FileStructure;
Option: FALSE	CompressedUniqueId	Save the identifier of the FileStructure that the entity lies in

7.3.10.3 ProductInformation

7.3.10.3.1 General

This is used to save general information associated with a product occurrence.

product_information_flags is described in **PRCProductFlag**.

product_load_status is described in **EPRCProductLoadStatus**.

unit_from_CAD_file indicates whether the unit is read from the native CAD file.

unit represents the units in mm.

Required or Option	Data Type	Data Description
Required	Boolean	Unit_from_CAD_file
Required	Double	Units
Required	Character	`product_information_flags`
Required	Integer	`product_load_status`

7.3.10.3.2 PRCProductFlag

These flags represent characteristics of product occurrences.

A product occurrence can be:

- Default: The product occurrence is the default container, configuration, or view. This means that it is loaded by default in the originating CAD system.
- Internal: when used as a prototype of another product occurrence, this product occurrence does not come from a different physical file. Hence it should belong to the same file structure.
- Container: The product occurrence acts as a repository of son occurrences that do not necessarily have relationships between them. This is useful for situations where a single CAD file can correspond to a whole database of parts and assemblies.
- Configuration: This is a specific arrangement of a product with respect to its whole hierarchy. Some parts may differ or are in a different position. For example, consider the case of an automobile where the steering wheel may be either on the left or right side.
- View: A product occurrence which is a view refers to another product occurrence (its prototype) to denote a particular setting of visibilities and position within the same hierarchy.

If none of these flags is specified, a product occurrence is referred to as regular. If the product occurrence has no father, it is similar to a configuration. A product occurrence with no father leads to a different FileStructure, unless it is internal, meaning that it represents a part, subassembly, or assembly hierarchy inside the same FileStructure.

Value	Type Name
0x0000	PRC_PRODUCT_FLAG_REGULAR
0x0001	PRC_PRODUCT_FLAG_DEFAULT
0x0002	PRC_PRODUCT_FLAG_INTERNAL
0x0004	PRC_PRODUCT_FLAG_CONTAINER
0x0008	PRC_PRODUCT_FLAG_CONFIG
0x0010	PRC_PRODUCT_FLAG_VIEW

7.3.10.3.3 EPRCProductLoadStatus

This represents the status of a loading (i.e. file reading) operation.

Value	Type Name	Description
0	KEPRCProductLoadStatus_Error	Unknown status
1	KEPRCProductLoadStatus_NotLoaded	Loading error. For example, there is a missing file
2	KEPRCProductLoadStatus_NotLoadable	Not loadable. For example, something prevents the file from being loaded.
3	KEPRCProductLoadStatus_Loaded	The product was successfully loaded

7.3.10.4 MarkupData

This is all the data that are related to Markup for a part or a product.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number_of_linked_items
Required	ArrayOf[PRC_TYPE_MISC_MarkupLinkedItem]	Array of linked items
Required	UnsignedInteger	Number_of_leaders
Required	ArrayOf[PRC_TYPE_MKP_Leader]	Array of leaders
Required	UnsignedInteger	Number_of_markups
Required	ArrayOf[PRC_TYPE_MKP_Markup]	Array of markups
Required	UnsignedInteger	Number_of_annotation_entities
Required	ArrayOf[AnnotationEntities]	Array of annotation entities

7.3.10.5 AnnotationEntities

An annotation entity can be a

- **PRC_TYPE_MKP_AnnotationItem**
- **PRC_TYPE_MKP_AnnotationSet**
- **PRC_TYPE_MKP_AnnotationReference**

7.3.11 PRC_TYPE_ASM_PartDefinition

This represents a part definition.

A part consists of:

- A Bounding_box (reserved for future use);
- A collection of visible representation items containing geometrical data;
- Markups representing the markups of this part definition (as opposed to product occurrence markups); these are grouped into annotations.
- Views represents the views which can contain annotations or specific display parameters.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_PartDefinition

Required	PRC_TYPE_ROOT_PRCBaseWithGraphics	
Required	BoundingBox	Currently not used
Required	UnsignedInteger	Number of representational items
Required	ArrayOf [PRC_TYPE_RI]	Array of representational items
Required	MarkupData	Markups for this part definition (see PRC_TYPE_ASM_ProductOccurrence)
Required	UnsignedInteger	Number of views
Required	ArrayOf [PRC_TYPE_MKP_View]	Array of view data
Required	UserData	User defined data

7.3.12 PRC_TYPE_ASM_Filter

7.3.12.1 General

This entity specifies the filtering between parts and assemblies. A filter denotes the particular usage of a subpart or product occurrence within a more complex one. It has the following purposes:

- To represent only those items that are of interest in the complex assembly.
- To configure the display accordingly.

Is_active: indicates whether this filter should correspond to the active layout when loading the file.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_ASM_Filter
Required	ContentPRCBase	
Required	ContentLayerFilterItems	Layer filter
Required	ContentEntityFilterItems	Entity filter
Optional	UserData	See Section 8.6 for details

7.3.12.2 ContentLayerFilterItems

This saves information for filtering of entities by layer: only entities having certain layer specifications will be retained by the filter.

b_is_inclusive indicates whether the elements inside the filter must be retained.

Required	Boolean	b_is_inclusive
Required	UnsignedInteger	Number of layers

Required	ArrayOf[UnsignedInteger]	Layer index

7.3.12.3 ContentEntityFilterItems

This saves information for a filtering directly by entities: only entities referred to in the array will be retained by the filter.

b_is_inclusive indicates whether the elements inside the filter must be retained.

Required	Boolean	b_is_inclusive
Required	UnsignedInteger	Number of entities
Required	ArrayOf[PRC_TYPE_MISC_EntityReference]	Basic entity information

7.3.13 CompressedUniqueId

This saves information on unique id in compressed mode. Please refer to section 5.3.

Required	UnsignedInteger	Unique id[0]
Required	UnsignedInteger	Unique id[1]
Required	UnsignedInteger	Unique id[2]
Required	UnsignedInteger	Unique id[3]

7.4 Miscellaneous Data

7.4.1 Entity Types

This section gather types allowing for entities' referencing and positioning.

Type Name	Type Value	Referenceable
PRC_TYPE_MISC	PRC_TYPE_ROOT + 200	
PRC_TYPE_MISC_Attribute	PRC_TYPE_ MISC + 1	
PRC_TYPE_MISC_EntityReference	PRC_TYPE_ MISC + 2	
PRC_TYPE_MISC_MarkupLinkedItem	PRC_TYPE_ MISC + 3	
PRC_TYPE_MISC_ReferenceOnPRCBase	PRC_TYPE_ MISC + 4	
PRC_TYPE_MISC_ReferenceOnTopology	PRC_TYPE_ MISC + 5	
PRC_TYPE_MISC_CartesianTransformation	PRC_TYPE_ MISC + 6	

PRC_TYPE_MISC_GeneralTransformation	PRC_TYPE_ MISC + 7	

7.4.2 PRC_TYPE_MISC

This is the abstract base type for **PRC_TYPE_Misc** entity types.

7.4.3 PRC_TYPE_MISC_Attribute

7.4.3.1 General

This represents the storage of an attribute which has a single title and a variable number of key/value pairs.

For example an attribute might contain the coordinates of the center of gravity, which would be represented by an attribute with title "Center of Gravity" and three key/value pairs (X =, 5), (Y =, 10) and (Z =, 20).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_Attribute
Required	AttributeEntry	Attribute title
Required	UnsignedInteger	Number of attribute Key/Value pairs
Required	ArrayOf [Attribute Key/Value]	Array of Key/Value pairs

7.4.3.2 AttributeEntry

This represents the storage of an attribute title represented by either a string or an integer containing a predefined string.

The following are valid integer titles and their predefined character strings:

Integer Value	Title
2	Title
3	Subject
4	Author
5	Keywords
6	Comments
7	Template
8	Last Saved By
9	Revision Number

10	Total Editing Time
11	Last Printed
12	Create Time/Date
13	Last saved Time/Date
14	Number of Pages
15	Number of Words
16	Number of Characters
17	Thumbnail
18	Name of Creating Application
19	Security

Attributes with a title beginning with either "__**PRC_RESERVED_ATTRIBUTE**" or "__**PRC_EXTERNAL_ATTRIBUTE"** can be used in the same way as UserData as discussed in section 5.5. They are considered as conveying proprietary information which should not be interpreted by a conforming PRC File Reader. This proprietary information is to be interpreted together with the application UUID of the FileStructure the attribute belongs to, unless the first 4 key/value pairs of the attribute are:

- "__PRC_APPLICATION_UUID_1" integer,
- "__PRC_APPLICATION_UUID_2" integer,
- "__PRC_APPLICATION_UUID_3" integer,
- "__PRC_APPLICATION_UUID_4" integer

which indicates an alternate application UUID for the data to be interpreted.

Required or Option	**Data Type**	**Data Description**
Required	Boolean	TRUE if tiltle of the attribute is an integer else false
Option: TRUE	UnsignedInteger	Title is an integer
Option: FALSE	String	Title is a string

7.4.3.3 AttributeKey/Value

PRC allows for five different kinds of attribute data, represented by the following key:

- 0 represents an invalid type

- 1 represents a 32 bit integer
- 2 represents a floating point
- 3 represents a 32 bit integer interpreted like time_t
- 4 represents a UTF-8 character string

Required or Option	Data Type	Data Description
Required	AttributeEntry	Title of Key/Value Pair
Required	UnsignedInteger	Key determines what of 4 legal types of attributes is to follow
Option: 1	Integer	
Option: 2	Double	
Option: 3	Integer	Interpreted as time_t
Option: 4	String	

7.4.4 PRC_TYPE_MISC_EntityReference

This general type can be used to reference any referenceable entity. The data stored in the reference may include a line style, visibility, position or other property, and can be used to overwrite properties of the referenced entity.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_EntityReference
Required	ContentEntityReference	
Required	UserData	User defined data

7.4.5 PRC_TYPE_MISC_MarkupLinkedItem

7.4.5.1 General

This is used to establish a cross reference between markup and geometry. It contains a reference to the geometry in a PRC File as well as a reference to the product occurrence to which the given instance of the geometry belongs.

For example, consider the case of a distance dimension between a part contained in two product occurrences (assemblies). The dimension will have two **MarkupLinkedItems**, the first pointing to the first product occurrence and referencing the part, the second pointing to the second product occurrence and referencing the part as well.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_MarkupLinkedItem
Required	ContentExtendedEntityReference	Reference to a remote product occurrence
Required	Boolean	If true, show/hide markup when showing/hiding the referenced entity
Required	Boolean	If true, delete markup when deleting the referenced entity
Required	Boolean	If true, show the leader when showing/hiding the referenced entity
Required	Boolean	If true, delete the leader when deleting the referenced entity
Required	UserData	User defined data

7.4.5.2 ContentExtendedEntityReference

Stores data to reference entities in remote product occurrences.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_MarkupLinkedItem
Required	ContentEntityReference	Reference to a remote product occurrence
Required	ReferenceData	Reference data for the entity

7.4.6 PRC_TYPE_MISC_ReferenceOnPRCBase

This describes a reference to a referenceable entity. Referenceable entity types are a subset of the PRC base entities (see section 7.2.1 for the specific subset). Referenceable topological entities are handled separately (see 7.4.7).

A reference to an entity consists of the UUID of the FileStructure the referenced entity lies in (if different from the FileStructure this entity is in) and the index of the referenced entity within this FileStructure.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_ReferenceOnPRCBase
Required	UnsignedInteger	This is the type of the target entity
Required	Boolean	TRUE if this reference is to an entity in same File Structure as this entity exists in else FALSE

Option: FALSE	CompressedUniqueID	Unique identifier of target FileStructure if different from this FileStructure; See Section 6.2 for details
Required	UnsignedInteger	Unique identifer within target File Structure

7.4.7 PRC_TYPE_MISC_ReferenceOnTopology

7.4.7.1 General

This describes a reference to a topological entity.

The following describe the data needed to locate the target entity:

- The type of topological entity being referenced must be one of those in **ReferenceOnTopology Entities**
- If the target entity has a body in the Exact Geometry Section of the target FileStructure, additional information is required to locate the target entity.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_ReferenceOnTopology
Required	UnsignedInteger	Type of the topolocial entity being referenced
Required	Boolean	TRUE if the target entity has a body in the Exact Geometry Section of the target FileStructure
OPTION: TRUE	AdditionalTargetData	Data defining the reference to the target entity

7.4.7.2 AdditionalTargetData

This is used only within **PRC_TYPE_MISC_ReferenceOnTopology**.

The unique identifier of the target FileStructure is stored here if it is different from the FileStructure of the entity currently being read or written.

To locate the target entity within the target FileStructure requires the following:

- The index of the topological context of the target entity.
- The index of the body within the topological context.

- In addition, most topological entities need other indices within the body to identify themselves. The number of additional indices and an array of index values indicate other index values that are needed to uniquely identify the target entity. At present, the only topological entity which may be referenced is a **PRC_TYPE_TOPO_Face**. For this, the requisite data would be:

    - Number of additional indices is 1

    - The array would consist of one Unsigned Integer which would be the index of the face within the body.

Required or Option	Data Type	Data Description
Required	Boolean	TRUE if the target entity is in the same FileStructure as this entity
OPTION: FALSE	CompressedUniqueId	Unique identifier of the FileStructure the target entity lies in
Required	UnsignedInteger	Index of the topological context of the target entity within the FileStructure
Required	UnsignedInteger	Index of the body within the topological context of the target entity
Required	UnsignedInteger	Number of additional indices needed to locate the target entity
Required	ArrayOf [UnsignedInteger]	Array of additional indices

7.4.7.3 ReferenceOnTopology Entities

The only topological entity which may be referenced is a **PRC_TYPE_TOPO_Face**.

The follow topological entities may be referenced in future versions:

- PRC_TYPE_TOPO_MultipleVertex
- PRC_TYPE_TOPO_UniqueVertex
- PRC_TYPE_TOPO_WireEdge
- PRC_TYPE_TOPO_Edge
- PRC_TYPE_TOPO_Loop
- PRC_TYPE_TOPO_Shell
- PRC_TYPE_TOPO_Connex

7.4.8 PRC_TYPE_MISC_CartesianTransformation

This represents a 3D transformation. Only the following flags are acceptable in defining a **PRC_TYPE_MISC_CartesianTransformation**:

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x04	PRC_TRANSFORMATION_Mirror	Mirror
0x08	PRC_TRANSFORMATION_Scale	Uniform scale
0x10	PRC_TRANSFORMATION_NonUniformScale	Non uniform scale

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_CartesianTransformation
Required	Transformation	Data defining the cartesian transformation which is limited to the above table

7.4.9 PRC_TYPE_MISC_GeneralTransformation

This is a general 3D transformation consisting of the sixteen coordinates of a 4x4 matrix.

To use a 4x4 matrix to convert a 3D position of vector, one pre multiplies by the matrix, that is,

New_3D_PointOrVector = matrix * Old_3D_PointOrVector

This type exists from documented version 8137 and above. Hence every PRC writer with minimal version lower than 8137 must write the schema of this type before writing such entity.

The coefficients are stored in the following order:

Matrix (First number is row, second number is column). For example, translation is represented by Tx=M[0][3], Ty = M[1][3], Tz=M[2][3]

M[0][0] M[0][1] M[0][2] M[0][3]

M[1][0] M[1][1] M[1][2] M[1][3]

M[2][0] M[2][1] M[2][2] M[2][3]

M[3][0] M[3][1] M[3][2] M[3][3]

Storage order:

M[0][0]

M[1][0]

M[2][0]

M[3][0]

M[0][1]

M[1][1]

....

M[1][3]

M[2][3]

M[3][3]

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_GeneralTransformation
Required	Double[16]	16 Coefficients of transformation

7.4.10 ContentEntityReference

7.4.10.1 General

This represents the data defining a reference to any referenceable entity.

The **index_of_local_coordinate_system** may be -1 indicating no local coordinate system is present. Otherwise, the value given is the index into the array of reference coordinate systems defined in 7.3.5.2.

If the referenced entity does not exist, no futher information should be stored. If the reference does exist, data describing the unique identifier of the referenced entity will be present in the PRC File.

Required or Option	Data Type	Data Description
Required	PRCBaseWithGraphics	
Required	UnsignedInteger	Index_of_local_coordinate_system or -1 if none present
Required	Boolean	TRUE if the referenced entity exists (i.e. is not NULL)
OPTION:TRUE	ReferenceData	Define the unique identifier of the

		reference entity.

7.4.10.2 ReferenceData

PRC_TYPE_MISC_ReferenceOnTopology should be used to reference an entity, whenever referencing a referencable topological entity. Any other value is an error.

Required or Option	Data Type	Data Description
OPTION: topology reference	PRC_TYPE_MISC_ReferenceOnTopology	Reference to a referenceabe topological entity
OPTION: non topology reference	PRC_TYPE_MISC_ReferenceOnPRCBase	Reference to a non-topological entity

7.4.11 Transformation

7.4.11.1 General

The Transformation associated with an entity is defined as either 2D or 3D depending upon the dimension of the entity containing the transformation.

Notes:

- A cartesian transformation that is used to represent a 3D cartesian transformation must be orthogonal and must not have homogeneous values. Non-orthogonal or homogeneous transformations are used only for the transformations used in textures.
- **PRC_TYPE_MISC_GeneralTransformation** is a transformation but only matrix coefficients are stored, as described in section 7.4.9.

The Transformation is defined by its behavior which can be any combination (except as noted above) of

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x04	PRC_TRANSFORMATION_Mirror	Mirror
0x08	PRC_TRANSFORMATION_Scale	Uniform scale
0x10	PRC_TRANSFORMATION_NonUniformScale	Non uniform scale
0x20	PRC_TRANSFORMATION_NonOrtho	Non orthogonal

0x40	PRC_TRANSFORMATION_Homogeneous	Homogeneous

This parameter is stored at the beginning of the transformation and conditions the way in which the data are read and written. The procedure is explained with the following pseudocode:

```
if (behavior & PRC_TRANSFORMATION_Translate)

   read vector



if (behavior & PRC_TRANSFORMATION_NonOrtho)

   Read vectors

else if (behavior & PRC_TRANSFORMATION_Rotate)

   Read vectors



if (behavior & PRC_TRANSFORMATION_NonUniformScale)

   Read vector

else if (behavior & PRC_TRANSFORMATION_Scale)

   Read double



if (behaviour & PRC_TRANSFORMATION_Homogenous)

   Read vectors
```

For 3D geometry, a 4x4 matrix containing a translation, rotation, mirror, or scaling component can be inferred from the data as follows:

- If translation, translate -> mat[0][3] mat[1][3] mat[2][3]
- If rotation,
    - rotate.xaxis -> mat[0][0] mat[1][0] mat[2][0]
    - rotate.yaxis -> mat[0][1] mat[1][1] mat[2][1]
    - If no mirror, rotate.xaxis X rotate.yaxis -> mat[0][2] mat[1][2] mat[2][2] where X is the cross product
    - If mirror, rotate.yaxis X rotate.xaxis -> mat[0][2] mat[1][2] mat[2][2] where X is the cross product
- If non-orthogonal,
    - nonortho.xaxis -> mat[0][0] mat[1][0] mat[2][0]

    - nonortho.yaxis -> mat[0][1] mat[1][1] mat[2][1]
    - nonortho.zaxis -> mat[0][2] mat[1][2] mat[2][2]
- Mirror is not applicable and must not be set in the case of non-orthognal rotation.If scale, apply scale to the 3x3 submatrix mat[0][0] ... mat[3][3]
- If non-uniform scaling
    - Apply non_uniform_scale.x to column mat[0][0] mat[1][0] mat[2][0]
    - Apply non_uniform_scale.y to column mat[0][1] mat[1][1] mat[2][1]
    - Apply non_uniform_scale.z to column mat[0][2] mat[1][2] mat[2][2]
- If non-homogeneous,
    - Homogeneous.x -> mat[3][0]
    - Homogeneous.y -> mat[3][1]
    - Homogeneous.z -> mat[3][2]
    - Homogeneous.w -> mat[3][3]

Similarly, for 2D geometry, a 3x3 matrix containing a translation, rotation or scaling component can be inferred from the data as follows:

- If translation, origin -> mat[0][2] mat[1][2]
- If rotation,
    - xaxis -> mat[0][0] mat[1][0]
    - yaxis -> mat[0][1] mat[1][1]
- If non-orthogonal,
    - nonortho.xaxis -> mat[0][0] mat[1][0]
    - nonortho.yaxis -> mat[0][1] mat[1][1]

      Mirror is not applicable and must not be set in the case of non-orthogonal rotation.
- If scale, apply scale to the 2x2 submatrix mat[0][0] ... mat[1][1].
- If non-uniform scaling
    - Apply non_uniform_scale.x to column mat[0][0] mat[1][0]
    - Apply non_uniform_scale.y to column mat[0][1] mat[1][1]
- If non-homogeneous,
    - Homogeneous.x -> mat[2][0]
    - Homogeneous.y -> mat[2][1]

- Homogeneous.w -> mat[2][2]

Required or Option	Data Type	Data Description
Required	Character	Behavior determines the type of data used to define the transformation; each bit of the behavior determines if the transformation has that data

See below for the contents of transformation entities. Vector3d is available for 3D transformation, Vector2D is available for 2D transformation.

7.4.11.2 Translation

Required	Vector3d or Vector2d	origin

7.4.11.3 Rotation

Required	Vector3d or Vector2d	X axis; must be a unit vector
Required	Vector3d or Vector2d	Y axis; must be a unit vector

7.4.11.4 NonOrtho

Required	Vector3d or Vector2d	X axis; must be a unit vector
Required	Vector3d or Vector2d	Y axis; must be a unit vector
Required (if 3D transformation)	Vector3d	Z axis; must be a unit vector

7.4.11.5 Scale

Required	Double	Uniform scale

7.4.11.6 NonUniformScalePart

Required	Vector3d or Vector2d	Scale factor for x, y, and z for 3D transformation

7.4.11.7 HomogeneousPart

Required	Double	X_homogeneous coordinate
Required	Double	Y_homogeneous_coordinate
Required (if 3D transformation)	Double	Z _homogeneous_coordinate
Required	Double	W _ homogeneous_coordinate

7.5 Graphics

7.5.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_GRAPH	PRC_TYPE_ROOT + 700	
PRC_TYPE_GRAPH_Style	PRC_TYPE_GRAPH + 1	yes
PRC_TYPE_GRAPH_Material	PRC_TYPE_GRAPH + 2	yes
PRC_TYPE_GRAPH_Picture	PRC_TYPE_GRAPH + 3	
PRC_TYPE_GRAPH_TextureApplication	PRC_TYPE_GRAPH + 11	yes
PRC_TYPE_GRAPH_TextureDefinition	PRC_TYPE_GRAPH + 12	yes
PRC_TYPE_GRAPH_TextureTransformation	PRC_TYPE_GRAPH + 13	
PRC_TYPE_GRAPH_LinePattern	PRC_TYPE_GRAPH + 21	yes
PRC_TYPE_GRAPH_FillPattern	PRC_TYPE_GRAPH + 22	
PRC_TYPE_GRAPH_DottingPattern	PRC_TYPE_GRAPH + 23	yes
PRC_TYPE_GRAPH_HatchingPattern	PRC_TYPE_GRAPH + 24	yes
PRC_TYPE_GRAPH_SolidPattern	PRC_TYPE_GRAPH + 25	yes
PRC_TYPE_GRAPH_VpicturePattern	PRC_TYPE_GRAPH +26	yes
PRC_TYPE_GRAPH_AmbientLight	PRC_TYPE_GRAPH + 31	yes
PRC_TYPE_GRAPH_PointLight	PRC_TYPE_GRAPH + 32	yes
PRC_TYPE_GRAPH_DirectionalLight	PRC_TYPE_GRAPH + 33	yes
PRC_TYPE_GRAPH_SpotLight	PRC_TYPE_GRAPH + 34	yes
PRC_TYPE_GRAPH_SceneDisplayParameters	PRC_TYPE_GRAPH + 35	yes

PRC_TYPE_GRAPH_Camera	PRC_TYPE_GRAPH + 36	yes

7.5.2 PRC_TYPE_GRAPH

The abstract type for miscellaneous graphic elements not included in part geometry, topology, tesselation, or markups. Includes line and fill styles and patterns, colors, textures, pictures, lighting scenes, and camera angles. Graphic elements may be applied to other elements, such as part surfaces or markups.

7.5.3 PRC_TYPE_GRAPH_Style

This type contains all information used to describe the style of a line.

- **line_width** represents the line width in millimeters.
- **is_vpicture** indicates that the drawing style is a VPicture pattern instead of a line pattern. This style is to be found in the pattern array instead of the line pattern array (see **FileStructureInternalGlobalData** Section 7.5.3.2).
- **is_material** indicates that the color style is a material instead of a plain color. This style is to be found in the material array instead of the color array (see **FileStructureInternalGlobalData** Section 7.3.5.2).
- **material_index** is the index into the material array. (see **FileStructureInternalGlobalData Section** 7.3.5.2).
- **color_index** is the index into the color array. (see **FileStructureInternalGlobalData** Section 7.3.5.2).
- **transparency** values can range from 0 (transparent) to 255 (opaque).
- **Rendering parameters** holds values from **PRC** documented version 8137 and above.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_Style
Required	Double	Line width
Required	Boolean	Is_vpicture
Required	UnsignedInteger	Line_pattern_index + 1 or Vpicture_index + 1
Required	Boolean	Is_material
Required	UnsignedInteger	Color_index + 1 or Material_index + 1
Required	Boolean	If true, transparency is defined
Optional	Character	transparency
Required	Boolean	If true, rendering parameters are defined
Optional	Character	Rendering parameter
Required	Boolean	Not currently used (set false)
Required	Boolean	Not currently used (set false)

.

Rendering parameter	Value
special-culling strategy applies	0x0001
front culling applies (ignored if no special-culling strategy)	0x0002

back culling applies (ignored if no special-culling strategy)	0x0004
no light applied to the corresponding object	0x0008

7.5.4 PRC_TYPE_GRAPH_Material

This type defines basic material appearance with colors and alphas.

- **ambient_index:** index into the RGB array (see Section 7.3.5.2)
- **diffuse_index:** index into the RGB array (see Section 7.3.5.2)
- **emissive_index:** index into the RGB array (see Section 7.3.5.2)
- **specular_index:** index into the RGB array (see Section 7.3.5.2)

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_Material
Required	ContentPRCBase	Basic information associated with the entity
Optional	UnsignedInteger	ambient_index + 1
Required	UnsignedInteger	diffuse_index + 1
Required	UnsignedInteger	emissive_index + 1
Required	UnsignedInteger	specular_index + 1
Required	Double	shininess
Required	Double	ambient_alpha (0.0 –> 1.0)
Required	Double	diffuse_alpha (0.0 –> 1.0)
Required	Double	emissive_alpha (0.0 –> 1.0)
Required	Double	specular_alpha (0.0 –> 1.0)

The definitions for shininess, anbient_alpha, diffuse_alpha, emissive_alpha , and specular_alpha are identical to the definitions in **OPENGL**.

7.5.5 PRC_TYPE_GRAPH_Picture

7.5.5.1 General

This type is used to define pictures embedded in the file.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_Picture
Required	ContentPRCBase	Basic information associated with the entity
Required	Integer	Format

Required	UnsignedInteger	uncompressed_file_index + 1
Required	UnsignedInteger	pixel width
Required	UnsignedInteger	pixel height

Pixel_width and pixel_height are the size of the picture expressed in pixels. When **Format** is 0 or 1, pixel width and pixel height fields are ignored. When **Format i**s one of {2,3,4,5} the size of the picture buffer when uncompressed must be at least pixel width * pixel height * number of components per pixel .

7.5.5.2 EPRCPictureDataFormat

This object is used for the format of the Picture.

Value	Type Name	Type Description
0	*KEPRCPicture_PNG*	PNG format buffer
1	*KEPRCPicture_JPG*	JPEG format buffer
2	*KEPRCPicture_BITMAP_RGB_BYTE*	flate-formatted pixel data. Each element is an RGB triplet (3 components).
3	*KEPRCPicture_BITMAP_RGBA_BYTE*	flate-formatted pixel data. Each element is an RGBA triplet (4 components).
4	*KEPRCPicture_BITMAP_GREY_BYTE*	flate-formatted pixel data. Each element is a single luminance value (1 component).
5	*KEPRCPicture_BITMAP_GREYA_BYTE*	flate-formatted pixel data. Each element is a luminance/alpha pair (2 components).

7.5.6 PRC_TYPE_GRAPH_TextureApplication

This type contains a definition of the complete texture pipe (multiple texturing) to be applied.

A definition of the unique variable follows:.

- **material_generic_index** represents an index in the material array (see **FileStructureInternalGlobalData** Section 7.3.5.2). This index should correspond to a **PRC_TYPE_GRAPH_Material**, which defines the basic material parameters of the texture.
- **texture_definition_index** represents an index in the texture definition array (see Section

7.3.5.2).

- **next_texture_index** represents an index in the material array (see Section 7.3.5.2). This index should correspond to a **PRC_TYPE_GRAPH_TextureApplication**, which is used as the next level of texture in multiple texturing. This index is set to -1 if it is the last level of texture.
- **UV_coordinates_index** represents the texture mapping coordinates index (see PRC_TYPE_TESS_FACE section 7.8.6 and below).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_TextureApplication
Required	ContentPRCBase	Basic information associated with the entity
Required	UnsignedInteger	material_generic_index + 1
Required	UnsignedInteger	texture_definition_index + 1
Required	UnsignedInteger	next_texture_index + 1
Optional	UnsignedInteger	UV_coordinates_index + 1

UV_coordinates_index denotes the set of UV coordinates to consider in the **PRC_TYPE_TESS_Face** for textured entities, as there might be several UV coordinates for each point.

See: **number_of_texture_coordinate_indexes** in **PRC_TYPE_TESS_Face**.

For example, a simple triangle with TWO texture coordinates index is described by (normal,{texture1,texture2},point,

normal, {texture1,texture2},point,

normal, {texture1,texture2},point).

UV_coordinate_index indicates which of texture1 or texture2 should be used.

7.5.7 PRC_TYPE_GRAPH_TextureDefinition

This type contains a single set of texture parameters to be used in a TextureApplication.

A definition for the unique variables follows:

- **picture_index** represents the index in the picture array (see **FileStructureInternalGlobalData** Section 7.3.5.2).
- **texture_dimension** represents the dimension of the image. It's possible values are 1, 2, and 3 (1 and 3 are reserved for future use).
- **texture_mapping_attributes** is a bit field that represents the procedure used to apply the texture (see texture mapping attributes table below). This information can be combined with additional information, such as **intensity**, and involves color or alpha components.
- **size_texture_mapping_attributes_intensities** can be set either to 0 or to the number of procedures deduced from texture_mapping_attributes. If it is set to 0, the intensity is set to 1. Otherwise, its values should be in the range [0.0,1.0] and should correspond to each nonzero bit of **texture_mapping_attributes**, respectively. The same is true for **size_texture_mapping_attributes_components**, for which the default value is

PRC_TEXTURE_MAPPING_COMPONENTS_RGBA (see texture mapping attributes table below)
Multiple procedures for texture application are reserved for future use.
Therefore **size_texture_mapping_attributes_intensities** and
size_texture_mapping_attributes_components contain at most one element.
If **texture_mapping_attributes = PRC_TEXTURE_MAPPING_DIFFUSE**,
then **size_texture_mapping_attributes_intensities** = 0. For each bit of
texture_mapping_attributes with a value of 1, **intensity** will be 1.0 by default.
If **size_texture_mapping_attributes_components** = 0, then for each bit of
texture_mapping_attributes with a value of 1, components will be
PRC_TEXTURE_MAPPING_COMPONENTS_RGBA by default.

Or:
texture_mapping_attributes = PRC_TEXTURE-MAPPING_DIFFUSE
size_texture_mapping_attributes_intensities = 1
texture_mapping_attributes_intensities[0] = 1.0
size_texture_mapping_attributes_components = 1
texture_mapping_attributes_components[0] =
PRC_TEXTURE_MAPPING_COMPONENTS_RGBA

- **texture_function** : see texture function table below.
- **blend_src_rgb, blend_dst_rgb, blend_src_alpha, blend_dst_alpha**;
  blending modes are **reserved for future use**.
- **texture_applying_mode** : see texture application mode table below.
- **alpha_test : reserved for future use.**
- **alpha_test_reference** : threshold value for alpha test; used in conjunction with alpha_test.
- **texture_wrapping_mode_S** : Repeating mode; U direction; see wrapping mode table below.
- **texture_wrapping_mode_T** : Repeating mode; V direction; see wrapping mode table below.
- **texture_wrapping_mode_R** : Repeating mode; W direction (for multi dimension textures) ; see wrapping mode table below.
- **texture_transformation** : optional transformation on texture coordinates.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_TextureDefinition
Required	ContentPRCBase	Basic information associated with the entity
Required	Unsigned integer	picture_index + 1
Required	Character	texture dimension = 2 (1 and 3 are reserved for future use)
Required	Integer	texture mapping type
Optional	Integer	If[texture mapping type == TEXTURE_MAPPING_OPERATOR] texture mapping operator
Optional	Bit	has_transformation
Optional	Cartesian Transformation	If(has_transformation != 0)
Required	UnsignedInteger	texture mapping attributes
Required	UnsignedInteger	number of texture mapping attributes intensities

		(must be 0 or 1)
Optional	ArrayOf[Doubles]	texture_mapping_attributes_intensities[0] = 1.0
Required	UnsignedInteger	number of texture mapping attributes components (must be 0 or 1)
Optional	ArrayOf[Characters]	texture_mapping_attributes_components[0] = 0x000F
Required	Integer	texture function (reserved for future use)
Optional	ArrayOf[Double]	If(texture_function ==KEPRCTextureFunctionBlend) [red, green,blue,alpha] blend color components In the range (0.0, 1.0)
Required	Integer	blend_src_rgb (reserved for future use)
Required	Integer	blend_src_alpha (reserved for future use)
Required	Character	texture application mode
Optional	Integer	If(texture application mode & PRC_TEXTURE_APPLYING_MODE_ALPHATEST) alpha_test
Optional	Double	alpha_test_reference
Required	Character	texture wrapping mode
Required	integer	texture_wrapping_mode_S
Optional	integer	If(texture_dimension > 1) texture_wrapping_mode_T
Optional	Integer	If(texture_dimension>2) texture_wrapping_mode_R
Required	Bit	texture_transfomation
Optional	PRC_TYPE_GRAPH_ TextureTransformation	If(texture_transfomation) (see section PRC_TYPE_GRAPH_TextureTransformation 7.5.8)

Texture mapping type	**Integer value**
Let the application choose.	1
Use the mapping coordinates that are stored on a 3D tessellation object.	2
Retrieve the UV coordinates on the surface as mapping coordinates (reserved for future use).	3
Use the defined Texture mapping operator to calculate mapping coordinates (reserved for future use)	4

Texture mapping operator	**Integer value**
Unknown (default value)	1
Planar	2
Cylindrical	3
Spherical	4

Cubic	5

Texture mapping attributes	Integer value
Red component	0x0001
Green component	0x0002
Blue component	0x0004
RGB component	0x0007
Alpha component	0x0008
RGBA component	0x000F

Texture function	Integer value
Unknown - Let the application choose.	1
Modulate - Combine lighting with texturing (default value).	2
Replace the object color with texture color data.	3
Blend	4
Decal	5

Texture application mode	Character value
Let the application choose. (All states disabled.)	0x0000
Use lighting mode.	0x0001
Use alpha test.	0x0002
Combine a texture with one-color-per-vertex mode.	0x0004

Texture wrapping mode	Integer value
Unknown - Let the application choose.	1
Repeat - Display the repeated texture on the surface.	2
ClampToBorder - Clamp the texture to the border. Display the surface color along the texture limits.	3
Clamp	4
Clamp to edge	5
Mirrored repeat	6

7.5.8 PRC_TYPE_GRAPH_TextureTransformation

This type contains the transformation data used in a texture definition. In the current release, texture transformations are limited to two dimensions.

Required or	Data Type	Data Description

Option		
Required	UnsignedInteger	PRC_TYPE_GRAPH_TextureTransformation
Required	Boolean	If(**true**) the S coordinate parameter is inverted.
Required	Boolean	If(**true**) the T coordinate parameter is inverted.
Required	Boolean	If(**true**) the matrix transformation contains only 2-dimensional terms. (Always true in this version.)
Required	Transformation	2d transformation (see section 7.4.11 Transformations)

7.5.9 PRC_TYPE_GRAPH_LinePattern

This type contains the information used to display the dashes and gaps that comprise a line pattern.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_LinePattern
Required	ContentPRCBase	Basic information associated with the entity
Required	UnsignedInteger	number of unique dash-array elements
Optional	ArrayOf[Doubles]	lengths of each type of alternating dashes and gaps, length
Required	Double	the offset within the dash pattern at which to start the dash, phase
Required	Boolean	If(**true**) the pattern aspect that scales with the view.

If a pattern scales with the view, the unit of length is the same as the product occurrence it is associated with: otherwise, lengths are to be interpreted as a ratio.

7.5.10 PRC_TYPE_GRAPH_FillPattern

Abstract class for a two-dimensional display style. This type contains information related to a fill pattern, which can be one of the following types of patterns:

- Dotting pattern (PRC_TYPE_GRAPH_DottingPattern)
- Hatching pattern (PRC_TYPE_GRAPH_HatchingPattern)
- Solid pattern (PRC_TYPE_GRAPH_SolidPattern)
- Vectorized picture pattern (PRC_TYPE_GRAPH_VPicturePattern)

7.5.11 PRC_TYPE_GRAPH_DottingPattern

This type describes a two-dimensional filling pattern with points. By default, this pattern describes a regular grid of points spaced with pitch (zizag==false). If zizag is true, the points are offset in X by pitch/2.0 for the odd row.

- **next_pattern_index** represents the index of the next pattern (superimposed) in the pattern array (see **FileStructureInternalGlobalData** Section 7.3.5.2).

- **Color_index** represents the index into the color array (see **FileStructureInternalGlobalData** Section 7.3.5.2).

Required or Option	Data Type	Data Description
Required	Unsigned integer	PRC_TYPE_GRAPH_DottingPattern
Required	ContentPRCBase	Basic information associated with the entity
Required	Unsigned integer	next_pattern_index + 1
Required	Double	pitch of point spacing
Required	Boolean	If (**true**), the points are offset in X by (pitch/2.0) for the odd row.
Required	Integer	color_index + 1

7.5.12 PRC_TYPE_GRAPH_HatchingPattern

This type describes a two-dimensional filling pattern with hatches. This pattern is defined by a group of infinite lines, each having its own dash pattern and color.

- **next_pattern_index** represents the index of the next pattern (superimposed) in the pattern array (see **FileStructureInternalGlobalData** Section 7.3.5.2).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_HatchingPattern
Required	ContentPRCBase	Basic information associated with the entity
Required	Unsigned integer	next_pattern_index + 1
Required	UnsignedInteger	number of pattern hatching lines
Required	ArrayOf[groups of 5 Doubles 1 Integer]	(2 D vector **start point** 2 D vector **end point** Double **angle** Index_of_line_style + 1)

7.5.13 PRC_TYPE_GRAPH_SolidPattern

This type defines a two-dimensional filling pattern with a particular style (color, material, texture).

- **next_pattern_index** represents the index of the next pattern (superimposed) in the pattern array (see **FileStructureInternalGlobalData** Section 7.3.5.2)
- **material_index** is the index into the material array. (see **FileStructureInternalGlobalData** Section 7.3.5.2)
- **color_index** is the index into the color array. (see **FileStructureInternalGlobalData** Section 7.3.5.2)

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_SolidPattern
Required	ContentPRCBase	Basic information associated with the entity
Required	UnsignedInteger	next_pattern_index + 1

Required	Boolean	If (**true**)
		the fill is a material
		else
		a plain color.

Required	UnsignedInteger	material_index+1 OR color_index+1

7.5.14 PRC_TYPE_GRAPH_VpicturePattern

This type defines a two-dimensional filling pattern consisting of a vectorized picture. In this version a restricted version of **PRC_TYPE_TESS_Markup** is used. The allowed types are:

- Polyline
- Triangles
- Color
- Line Stipple
- Points
- Polygon
- Line Width

next_pattern_index represents the index of the next pattern (superimposed) in the pattern array (see **FileStructureInternalGlobalData** Section 7.3.5.2).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_VpicturePattern
Required	ContentPRCBase	Basic information associated with the entitiy
Required	UnsignedInteger	next_pattern_index + 1
Required	ArrayOf[Doubles]	X and Y dimensions of the pattern
Required	PRC_TYPE_TESS_Markup	PRC_TYPE_TESS_Markup object (See MARKUP Section for types)

7.5.15 PRC_TYPE_GRAPH_AmbientLight

This type defines the ambient illumination of a scene.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_AmbientLight
Required	ContentPRCBase	Basic information associated with the entity
Optional	UnsignedInteger	ambient_index + 1
Required	UnsignedInteger	diffuse_index+ 1
Required	UnsignedInteger	emissive_index + 1
Required	UnsignedInteger	specular_index + 1

7.5.16 PRC_TYPE_GRAPH_PointLight

This type defines scene light from a point with attenuation factors.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_PointLight
Required	ContentPRCBase	Basic information associated with the entity
Optional	UnsignedInteger	Ambient_index + 1
Required	UnsignedInteger	diffuse_index + 1
Required	UnsignedInteger	emissive_index + 1
Required	UnsignedInteger	specular_index + 1
Required	ArrayOf[Doubles (3D point)]	location of light
Required	Double	constant light attenuation factor in the range [0.0,1.0]
Required	Double	linear light attenuation factor in the range [0.0,1.0]
Required	Double	quadratic light attenuation factor in the range [0.0,1.0]

The attenuation factor is defined (like OpenGl) as:

F = 1/(Cc + Cl*d +Cq*d*d)

Where:

d = postive distance between the light's position and the vertex

Cc = constant light attenuation

Cl = linear light attenuation.

Cq = quadratic light attenuation

7.5.17 PRC_TYPE_GRAPH_DirectionalLight

This type defines scene directional illumination.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_DirectionalLight
Required	ContentPRCBase	Basic information associated with the entity
Optional	UnsignedInteger	ambient_index + 1
Required	UnsignedInteger	diffuse_index + 1
Required	UnsignedInteger	emissive_index + 1
Required	UnsignedInteger	specular_index + 1
Required	ArrayOf[Doubles (3D vector)]	direction of light
Required	Double	light intensity, a coefficient for the light in the range [0.0,1.0]

7.5.18 PRC_TYPE_GRAPH_SpotLight

This type defines scene light from a spot illumination, a point, with angle, intensity and attenuation parameters.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_SpotLight
Required	ContentPRCBase	Basic information associated with the entity
Optional	UnsignedInteger	ambient_index + 1
Required	UnsignedInteger	diffuse_index + 1
Required	UnsignedInteger	emissive_index + 1
Required	UnsignedInteger	specular _index+ 1
Required	ArrayOf[Doubles (3D point)]	location of light
Required	Double	constant light attenuation factor in the range [0.0,1.0]
Required	Double	linear light attenuation factor in the range [0.0,1.0]
Required	Double	quadratic light attenuation factor in the range [0.0,1.0]
Required	ArrayOf[Doubles (3D vector)]	direction of light
Required	Double	**fall-off angle**: the maximum spread angle of the light source in degrees in the range [0.0,90.0] or 180,0 degrees.
Required	Double	**fall-off exponent**: intensity distribution of the light in the range [0.0,128.0]

The **fall_off angle** is the angle between the axis of the cone and a ray along the edge of the cone. A value of 180 degrees specifies that the light is emitted in all directions.

7.5.19 PRC_TYPE_GRAPH_SceneDisplayParameters

Type defines parameters used for scene visualization, including ambient light and camera.

index_of_line_style: index into the line style array stored in the **FileStructureInternalGlobalData** Section 7.3.5.2. This array contains a list of PRC_TYPE_GRAPH_Style objects.

is_active: since there can be more than one object of this type, this boolen is used to specify if this object is the currently active scene.

rotation center: This define the center of rotation of the scenegraph. In other words, all objects in the scenegraph must turn around this point if this SceneDisplayParameters is activated.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_GRAPH_SceneDisplayParameters
Required	ContentPRCBase	Basic information associated with the entity
Required	Boolean	Is_active
Required	UnsignedInteger	number of lights
Required	ArrayOf[PRC_TYPE_GRAPH_light objects]	(see the sections on the light objects for details)
Required	Boolean	If (**true)**, a camera is defined.
Optional	PRC_TYPE_GRAPH_Camera	(see Section 7.5.20 for details)
Required	Boolean	If (**true**), a rotation center is defined
Optional	ArrayOf[Doubles (3D vector)]	rotational center
Required	UnsignedInteger	number of clipping planes
Optional	ArrayOf[PRC_TYPE_SURF_ Plane]	(see Section 7.11.13 for details)
Required	UnsignedInteger	Index_of_line_style+1 (background)
Required	UnsignedInteger	Index_of_line_style+1 (default)
Required	UnsignedInteger	number of default styles per type
Optional	ArrayOf[UnsignedInteger]	List of (t**ype**, **line_style_index**+1) pairs (see section 7 for a list of base entities)
Optional	Boolean	If (**true**), the position of lights, camera and clipping planes are absolute even when those parameters belong to a sub assembly.

7.5.20 PRC_TYPE_GRAPH_Camera

This type defines the camera used in scene visualization. It contains attributes such as its position, view angle, and zoom.

Required or	Data Type	Data Description

Option		
Required	UnsignedInteger	PRC_TYPE_GRAPH_Camera
Required	ContentPRCBase	Basic information associated with the entity
Required	Boolean	If (**true**), projection is orthographic, else perspective.
Required	ArrayOf[Doubles]	position of the camera (3D Position)
Required	ArrayOf[Doubles]	"look at" point (3D Position)
Required	ArrayOf[Doubles]	up vector (3D Vector)
Required	Double	field of view angle in radian (X direction) if perspective, Scale X if orthographic
Required	Double	field of view angle in radian (Y direction) if perspective, Scale Y if orthographic
Required	Double	ratio of X to Y
Required	Double	near clipping plane distance from the viewer (positive value)
Required	Double	far clipping plane distance from the viewer (positive value)
Required	Double	zoom factor (default 1.0)

7.6 Representation Items

7.6.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_RI	PRC_TYPE_ROOT + 230	
PRC_TYPE_RI_RepresentationalItem	PRC_TYPE_RI + 1	
PRC_TYPE_RI_BrepModel	PRC_TYPE_RI + 2	yes
PRC_TYPE_RI_Curve	PRC_TYPE_RI + 3	yes
PRC_TYPE_RI_Directioni	PRC_TYPE_RI + 4	yes
PRC_TYPE_RI_Plane	PRC_TYPE_RI + 5	yes
PRC_TYPE_RI_PointSet	PRC_TYPE_RI + 6	yes
PRC_TYPE_RI_PolyBrepModel	PRC_TYPE_RI + 7	yes
PRC_TYPE_RI_PolyWire	PRC_TYPE_RI + 8	yrs
PRC_TYPE_RI_Set	PRC_TYPE_RI + 9	yes
PRC_TYPE_RI_CoordinateSystem	PRC_TYPE_RI + 10	yes

7.6.2 PRC_TYPE_RI

This is an abstract base class. When PRC_TYPE_RI class is referenced in this documentation of the PRC File Format Specification, one of its constituent classes will be physically present in the file.

This is an abstract class to group the following classes:

- PRC_TYPE_RI_RepresentationItem
- PRC_TYPE_RI_BrepModel
- PRC_TYPE_RI_Curve
- PRC_TYPE_RI_Direction
- PRC_TYPE_RI_Plane
- PRC_TYPE_RI_PointSet
- PRC_TYPE_RI_PolyBrepModel
- PRC_TYPE_RI_PolyWire
- PRC_TYPE-RI_Set
- PRC_TYPE_RI_CoordinateSystem

7.6.3 PRC_TYPE_RI_RepresentationItem

7.6.3.1 General

This is an abstract class for all representation items. PRC_TYPE_RI_RepresentationItem denotes the abstract type from which any RI type derives and gathers all data common to any RI type.

7.6.3.2 RepresentationItemContent

This represents common data for all PRC_TYPE_RI entities.

Index_local_coordinate_system represents, if defined with a value other than -1, the index of the coordinate system as stored in **FileStructureInternalGlobalData.** The transformation is used to position geometry or tessellation. The general principal is that this transformation (LocalMatrix) must be multiplied by the global matrix to obtain the transformation using

GlobalMatrix x LocalMatrix

Index_tessellation represents, if defined with a value other that -1, the index in the **FileTessellation** section within a **FileStructure**

Required or Option	Data Type	Data Description
Required	PRC_TYPE_ROOT_PRCBaseWithGraphics	

Required	UnsignedInteger	Index_local_coordinate_system
Required	UnsignedInteger	Index_tessellation

7.6.4 PRC_TYPE_RI_BrepModel

This type represents a brep model.

If the brep model has a body in the exact geometry section of the FileStructure, the index of the topological context and the index of the body within the topological context identify the body.

A boolean flag indicates if the body is open or closed. Even if there is no body in the exact geometry section, tessellation data may represent a closed body.

Required or Option	**Data Type**	**Data Description**
Required	UnsignedInteger	PRC_TYPE_RI_BrepModel
Required	**RepresentationItemContent**	
Required	Bit	TRUE if brep model has a body in the exact geometry section of the File Structure ; else FALSE
Option: TRUE	UnsignedInteger	Index of the topological context in the exact geometry section of the File Structure
Option: TRUE	UnsignedInteger	Index of the body within the topological context
Required	Boolean	TRUE if the body is closed; else FALSE
Required	UserData	User defined data

7.6.5 PRC_TYPE_RI_Curve

This type represents a curve.

If there is a wire body in the exact geometry section of the FileStructure, the index of the topological context and the index of the body within the topological context identify the wire body.

Required or Option	**Data Type**	**Data Description**

Required	UnsignedInteger	PRC_TYPE_RI_Curve
Required	**RepresentationItemContent**	
Required	Bit	TRUE if curve has a wire body in the exact geometry section of the File Structure ; else FALSE
Option: TRUE	UnsignedInteger	Index of the topological context in the exact geometry section of the File Structure
Option: TRUE	UnsignedInteger	Index of the wire body within the topological context
Required	UserData	User defined data

7.6.6 PRC_TYPE_RI_Direction

This type represents a direction vector with an optional origin. This is used to define an axis.

This entity can be used to define infinite construction lines.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_RI_Direction
Required	**RepresentationItemContent**	
Required	Bit	TRUE if the direction has an origin; else FALSE
OPTION: TRUE	Vector3d	Direction origin
Required	Vector3d	Direction vector
Required	UserData	User defined data

7.6.7 PRC_TYPE_RI_Plane

This type represents a construction plane as opposed to a planar surface.

If the plane has an associated body in the exact geometry section of the FileStructure, the index of a topological context and an index of the body within the topological context identify the body.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_RI_Plane
Required	**RepresentationItemContent**	Common data
Required	Bit	TRUE if plane has associated body in the B-rep model; else FALSE
OPTION: TRUE	UnsignedInteger	Index of a topological context in the exact geometry section containing the body
OPTION: TRUE	UnsignedInteger	Index of a body within the topological context
Required	UserData	Users defined data

7.6.8 PRC_TYPE_RI_PointSet

This type represents a set of 3D points.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_RI_PointSet
Required	**RepresentationItemContent**	
Required	UnsignedInteger	Number of points
Required	ArrayOf [Vector3d]	Array of points in the set
Required	UserData	User defined data

7.6.9 PRC_TYPE_RI_PolyBrepModel

This type represents a PolyBrepModel defined by the tessellation data stored in the **RepresentationItemContent**. A boolean flag indicates if the tessellation is closed or open.

Required or Option	Data Type	Data Description

Required	UnsignedInteger	PRC_TYPE_RI_PolyBrepModel
Required	**RepresentationItemContent**	
Required	Boolean	TRUE if the tessellation is closed; else FALSE
Required	UserData	User defined data

7.6.10 PRC_TYPE_RI_PolyWire

This type represents a PolyWire defined by the tessellation data stored in the **RepresentationItemContent**.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_RI_PolyWire
Required	**RepresentationItemContent**	
Required	UserData	User defined data

7.6.11 PRC_TYPE_RI_Set

This represents the logical grouping of an arbitrary number of representational items.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_RI_Set
Required	**RepresentationItemContent**	
Required	UnsignedInteger	Number of representation items in the set
Required	ArrayOf [PRC_TYPE_RI_RepresentationItem]	An array of any of the PRC_TYPE_RI_xx items
Required	UserData	User defined data

7.6.12 PRC_TYPE_RI_CoordinateSystem

A coordinate system can have one of two distinct roles

- As a representation item belonging to the tree of a part definition.

- An entity to position other representation items. In this role, the coordinate system exists in the global section of the FileStructure (see **FileStructureInternalGlobalData** and **PRC_TYPE_RI** description).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_RI_CoordinateSystem
Required	**RepresentationItemContent**	
Required	Transformation	PRC_TYPE_MISC_GeneralTransformation or PRC_TYPE_MISC_CartesianTransformation
Required	UserData	User defined data

7.7 Markup

7.7.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_MKP	PRC_TYPE_ROOT + 500	
PRC_TYPE_MKP_View	PRC_TYPE_MKP + 1	yes
PRC_TYPE_MKP_Markup	PRC_TYPE_MKP + 2	yes
PRC_TYPE_MKP_Leader	PRC_TYPE_MKP + 3	yes
PRC_TYPE_MKP_AnnotationItem	PRC_TYPE_MKP + 4	yes
PRC_TYPE_MKP_AnnotationSet	PRC_TYPE_MKP + 5	yes
PRC_TYPE_MKP_AnnotationReference	PRC_TYPE_MKP + 6	yes

7.7.2 PRC_TYPE_MKP

This is the basic type for all 3D markups (annotations). Markups are non-geometric entities that aid viewers in understanding PRC model geometry. Markup types and subtypes include notes, dimensional annotations, geometric tolerance blocks, and weld symbols. Markups are linked to items, such as part geometry or assemblies. Markups may be attached to linked items by leaders (leader lines) for clarity.

Markups may contain tesselation data as patterns to define a vectorized picture incorporated in the markup. In this version, only the following entities may incorporate tesselated data: polyline, triangles, color, line style, points, polygon, line width.

7.7.3 PRC_TYPE_MKP_View

3D markups can be grouped into views that are associated with planes in which markup annotations lie. A view contains an array of annotation entities. A view can also define visibilities and positions of entities.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MKP_View
Required	PRC_TYPE_ROOT_ PRCBaseWithGraphics	See Section 7.2.4 for details.
Required	UnsignedInteger	Number of annotations
Required	Arrayof[ReferenceUnique Identifiers]	Unique identifiers for annotation entities
Required	Annotation Plane	See Section 7.3.10.5 for data definition.
Required	Bit	**scene_display_parameters**
Optional	If[**scene_display_parameters**] SceneDisplayParameters	See Section 7.5.19 for data definition
Required	Boolean	If true then view is an annotation view
Required	Boolean	If true the view is the default view
Required	Boolean	If true the plane is only indicating a direction
Required	UnsignedInteger	Number of linked items in markup view
Required	Arrayof [ReferenceUniqueIdentifiers]	Unique identifiers of linked items
Required	UnsignedInteger	Number of display filters
Required	Arrayof[PRC_TYPE_ASM_Filter]	Display Filters
Optional	User Data	See Section 8.6 for details

Definition of **ReferenceUniqueIdentifier**:

reference_in_same_file_structure indicates whether the object is in the same file structure.

Required or Optional	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MISC_ReferenceOnPRCBase
Required	UnsignedInteger	Reference type
Required	Boolean	**reference_in_same_file_structure**
Optional	If[!**reference_in_same_file_structure**] CompressedUniqueID	target_file_structure See Section 8.2.2 for details
Required	UnsignedInteger	Unique_identifier for entity

7.7.4 PRC_TYPE_MKP_Markup

This is the Basic type for simple markups. Each markup is defined by a type and a subtype. For instance, a markup may by of the type "dimension" and the subtype "dimension radius edge" indicating that this annotation points to the radius arc of the edge of an object.

Markup types are as follows:

Enum Label	Description (value)
KEPRCMarkupType_Unknown	Unknown value (0)
KEPRCMarkupType_Text	Plain text (1)
KEPRCMarkupType_Dimension	Dimension (2)
KEPRCMarkupType_Arrow	Arrow (3)
KEPRCMarkupType_Balloon	Balloon (4)
KEPRCMarkupType_CircleCenter	Center of Circle (5)
KEPRCMarkupType_Coordinate	Coordinate (6)
KEPRCMarkupType_Datum	Datum (7)
KEPRCMarkupType_Fastener	Fastener (8)
KEPRCMarkupType_Gdt	Geometric Dimensioning and Tolerance (GD&T) Block (9)
KEPRCMarkupType_Locator	Locator (10)
KEPRCMarkupType_MeasurementPoint	Point (11)
KEPRCMarkupType_Roughness	Roughness (12)
KEPRCMarkupType_Welding	Welding (13)
KEPRCMarkupType_Table	Table (15)
KEPRCMarkupType_Other	Other (16)

Markup subtypes are as follows:

Enum Label	Description (value)
KEPRCMarkupSubType_Datum_Ident	Datum Identifier subtype (1)
KEPRCMarkupSubType_Datum_Target	Datum Target subtype (2)
KEPRCMarkupSubType_Dimension_Distance	Distance Dimension (1)

KEPRCMarkupSubType_Dimension_Distance_Offset	Dimension offset distance (2)
KEPRCMarkupSubType_Dimension_Distance_Cumulate	Dimension cumulative distance (3)
KEPRCMarkupSubType_Dimension_Chamfer	Dimension chamfer callout (4)
KEPRCMarkupSubType_Dimension_Slope	Dimension slope (5)
KEPRCMarkupSubType_Dimension_Ordinate	Dimension ordinate (6)
KEPRCMarkupSubType_Dimension_Radius	Dimension radius (7)
KEPRCMarkupSubType_Dimension_Radius_Tangent	Tangent radius dimension (8)
KEPRCMarkupSubType_Dimension_Radius_Cylinder	Cylinder radius dimension (9)
KEPRCMarkupSubType_Dimension_Radius_Edge	Radius edge dimension (10)
KEPRCMarkupSubType_Dimension_Diameter	Diameter dimension (11)
KEPRCMarkupSubType_Dimension_Diameter_Tangent	Tangent diameter dimension (12)
KEPRCMarkupSubType_Dimension_Diameter_Cylinder	Cylinder diameter dimension (13)
KEPRCMarkupSubType_Dimension_Diameter_Edge	Diameter edge dimension (14)
KEPRCMarkupSubType_Dimension_Diameter_Cone	Cone diameter dimension (15)
KEPRCMarkupSubType_Dimension_Length	Length dimension (16)
KEPRCMarkupSubType_Dimension_Length_Curvilinear	Curvilinear length dimension (17)
KEPRCMarkupSubType_Dimension_Length_Circular	Circular length dimension (18)
KEPRCMarkupSubType_Dimension_Angle	Angle Dimension (19)
KEPRCMarkupSubType_Gdt_Fcf	Geometric Dimensioning and Tolerancing (1)
KEPRCMarkupSubType_Welding_Line	Welding line (1)
KEPRCMarkupSubType_Welding_Spot	Welding Spot (2)
KEPRCMarkupSubType_Other_Symbol_User	Symbol User (1)
KEPRCMarkupSubType_Other_Symbol_Utility	(2)
KEPRCMarkupSubType_Other_Symbol_Custom	(3)

KEPRCMarkupSubType_Other_GeometricReference	Geometric Reference (4)

index_tessellation represents, if defined (by specifying a value other than -1), the index of the tessellation in the tessellation section of the file structure. This index should point to a **PRC_TYPE_TESS_Markup** type object associated with this markup.

Required or Option	**Data Type**	**Data Description**
Required	UnsignedInteger	PRC_TYPE_MKP_Markup
Required	PRC_TYPE_ROOT_ PRCBaseWithGraphics	See Section 7.2.4 for details
Required	UnsignedInteger	type
Required	UnsignedInteger	sub_type
Required	UnsignedInteger	Number_of_linked_items
Required (at least one)	Arrayof[ReferenceUniqueIdentifiers]	Unique identifiers for each linked item
Required	UnsignedInteger	Number_of_leaders
Optional	Arrayof[ReferenceUniqueIdentifiers]	Unique identifiers for each leader
Required	UnsignedInteger	index_tessellation + 1
Optional	User Data	See Section 8.6 for details

7.7.5 PRC_TYPE_MKP_Leader

This is the basic type for a 3D markups leader. Leaders attach the markup annotation item to the annotation reference.

Required or Option	**Data Type**	**Data Description**
Required	UnsignedInteger	PRC_TYPE_MKP_Leader
Required	PRC_TYPE_ROOT_ PRCBaseWithGraphics	See Section 7.2.4 for details
Required	Arrayof[ReferenceUniqueIdentifiers]	Unique identifiers for each linked item
Optional	Arrayof[ReferenceUniqueIdentifiers]	Unique identifiers for each leader
Required	UnsignedInteger	index_tessellation + 1
Optional	User Data	See Section 8.6 for details

7.7.6 PRC_TYPE_MKP_AnnotationItem

This section contains the data for a single annotation item.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MKP_AnnotationItem
Required	PRC_TYPE_ROOT_ PRCBaseWithGraphics	See Section 7.2.4 for details
Required	ReferenceUniqueIdentifier	Unique identifier for the annotation item
Optional	User Data	See Section 8.6 for details

7.7.7 PRC_TYPE_MKP_AnnotationSet

An annotation set is a group of annotation items or subsets. For example, a tolerance defined by a datum and a feature control frame are described by an annotation set with two annotation items, where the items point respectively to a markup of type "datum" and a markup of type "feature control frame."

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MKP_AnnotationSet
Required	PRC_TYPE_ROOT_ PRCBaseWithGraphics	See Section 7.2.4 for details
Required	unsigned integer	Number of entities in the annotation set
Optional	ArrayOf[AnnotationEntity]	For each entity in the annotation set.
Optional	User Data	See Section 8.6 for details

The **AnnotationEntity** entry above will be one of the type: **PRC_TYPE_MKP_AnnotationItem** or **PRC_TYPE_MKP_AnnotationSet** or **PRC_TYPE_MKP_AnnotationReference**.

7.7.8 PRC_TYPE_MKP_AnnotationReference

An annotation reference stores explicit combinations of markup data with modifiers that can then be used to define other annotations. An example would be a feature control frame.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MKP_AnnotationReference
Required	PRC_TYPE_ROOT_ PRCBaseWithGraphics	See Section 7.2.4 for details
Required	unsigned integer	Number of linked items in the annotation reference
Optional	Arrayof[ReferenceUniqueIdentifiers]	List of the identifiers of the linked items in the reference

7.8 Tessellation

7.8.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_TESS	PRC_TYPE_ROOT + 230	
PRC_TYPE_TESS_Base	PRC_TYPE_TESS + 1	
PRC_TYPE_TESS_3D	PRC_TYPE_TESS + 2	
PRC_TYPE_TESS_3D_Compressed	PRC_TYPE_TESS + 3	
PRC_TYPE_TESS_Face	PRC_TYPE_TESS + 4	
PRC_TYPE_TESS_3D_Wire	PRC_TYPE_TESS + 5	
PRC_TYPE_TESS_Markup	PRC_TYPE_TESS + 6	

7.8.2 PRC_TYPE_TESS

7.8.3 PRC_TYPE_TESS_Base

Abstract root type for any tessellated entity.

7.8.4 ContentBaseTessData

This base class stores the coordinates of the tessellated data.

is_calculated is a flag denoting whether the tessellation was calculated during import or read directly from the native CAD file.

number_of_coordinates represents the number of doubles in the coordinate array.

coordinates is an array of doubles.

The interpretation of the **coordinates** data depends upon the entity type containing this array. See PRC_TYPE_TESS_3D, PRC_TYPE_TESS_3D_Compressed, PRC_TYPE_TESS_3D_Wire, or PRC_TYPE_TESS_Markup for a description of the interpretation of the coordinates array within these contexts.

Required or Option	Data Type	Data Description
Required	Boolean	is_calculated
Required	Unsignedinteger	number_of_coordinates
Required	ArrayOf[Double]	coordinates

7.8.5 PRC_TYPE_TESS_3D

7.8.5.1 General

A PRC_TYPE_TESS_3D entity contains tessellation data for an ordered collection of faces (PRC_TYPE_TESS_Face) as well as tessellation data for the wire boundaries of the faces. The notion of face does not necessarily reflect that the data comes from geometrical faces; it is also possible to store tessellation data within this entity which are an unordered set of triangles.

The following is a description of the data in the file:

- The **ContentBaseTessData** class defines the **number_of_coordinates** and **coordinates** of the tessellation data. It also defines a flag, **is_calculated**, indicating whether the data was calculated during import or comes directly from a CAD system. Data in the **coordinates** array are interpreted as the x, y, and z coordinates of the 3D points for the entire tessellation.

- **number_of_normal_coordinates** is size of the normal_coordinates array

- **normal_coordinates is an** array of doubles. Data in the **normal_coordinates** array are interpreted as the (nx, ny, nz) values of a normal vector at a 3D point. A 3D point may have multiple normal values each associated with a different triangularization within the tessellated data.

- **number_of_triangulated_indices** is the size of the **triangulated_index_array.**

- **triangulated_index_array** is an array of integers which are an index into the **coordinates** or **normal_coordinates** arrays. Because these arrays represents triples of numbers of the (x, y, z) of a point or the (nx, ny, nz) values of a normal vector, the index is always a multiple of 3. The interpretation of the data in this array is described below.

- **number_of_wire_indices** is the size of the wire index array

- **wire_indices** are indices into the coordinates array. The indicies in this array are grouped into the indices for a wire contour of the face. The array **wire_index** within the PRC_TYPE_TESS_Face indicates the start of the wire for each of the wire contours within a specific face.

- **has_faces** is true if this entity is built using geometrical faces.

- **has_loops** is true if this entity is built using geometrical faces and loops (wires of faces denote the loops).

- **number_of_face_tessellation_data** is the faces in the array of face_tessellation_data

- **face_tessellation_data** an array of **PRC_TYPE_TESS_Face** objects

- **Number_of_texture_coordinates** is the size of the texture coordinate array

- **Texture_coordinates** texture coordinate **(see PRC_TYPE_GRAPH_TextureApplication)**

- **crease_angle** is the threshold angle between two faces.

When recalculating the normals at points, the angle between two adjacent triangles is calculated and compared to the **crease_angle**. If it is below crease_angle, the normal would be shared at this point for the two triangles; otherwise, two distinct normals will exist.

- If **must_recalculate_normals** is set to **true**, the normals must be recalculated at loading according to the **crease_angle**. In this case, no normal indices are stored in the **triangulated_index_array** and the **normal_coordinate** array size is set to 0.

  However, all the indices stored in **PRC_TYPE_TESS_Face** are not affected by the value of **must_recalculate_normals**. Specifically, **used_entities_flag** and **start_triangulated** are set as if normal indices were stored.

  For instance, when storing a tessellation data with two faces, with one triangle each that have a common edge, both with a flag **used_entities_flag** = **PRC_FACETESSDATA_Triangle**, and with **must_recalculate_normals = true,** this is what will be stored :

  - **number_normal_coordinates** = 0; (It Would be 12 with **must_recalculate_normals = false)**
  - **number_of_coordinates** = 12;
  - **number_of_triangulated_indicies** = 6. (It Would be 12 with **must_recalculate_normals = false)**
  - all of the data for PRC_TYPE_TESS_Face is identical regardless of the must_recalculate_normals flag setting.

The basic tessellation data consists of

- an array **coordinates** representing the (x, y, z) coordinates of the 3D points ot the tessellation;
- an array **normal_coordinates** representing the (nx, ny, nz) components of normal vectors at the points; a given point may have multiple normal vectors, one for each vertex of the point in the triangularization data of the tessellation;
- an array **triangulated_index_array** of indicies into either the **coordinates** or **normal_coordinates** array. The entries in this array are grouped into one of the types of triangularization data (**PRC Tessellation Type**).
  - The type of triangularization defines the sequence and type of data (point or normal) of the triangularization data. For instance, a **PRC_FACETESSDATA_Triangle** is described with 6 indices (normal, point, normal, point, normal, point). Note that it is mandatory to specify at least one normal per triangularization data.
  - The order of tessellated faces in **face_tessellation_data** defines the order of triangularization data in the **triangulated_index_array**. The triangularization data for the first face is first in the **triangulated_index_array**, followed by the data for the second face, etc.
  - The triangularization data within a face consists of multiple triangulations. Each triangulation is of one of the types described in **PRC Tessellation Types** and identical types are grouped together. The bit fields of the **used_entities_flag** indicates if that type of triangularization data is present in the triangularization data for the face and the order of the bit fields from low to high (0 to 31) indicate the order of data in the **TriangulatedData** array. See **PRC_TYPE_TESS_Face** for a description of the face data.

A face tessellation corresponds to a geometrical face if faces are used (as denoted by **has_faces**). Otherwise, it is a large container that can be used for any tessellated data.

Wire_indices are the indicies describing the face's wire contours. See **PRC_TYPE_TESS_3D_Wire** and **PRC_TYPE_TESS_Face** for respective descriptions of how to interpret the data in the **wire_indices array**.

Texture coordinates are also to be interpreted according to the final graphics of each **face_tessellation**. Those graphics are specified either in **face_tessellation** or by the representation item owning the **PRC_TYPE_TESS_3D**. Then, the graphics will correspond to a texture with an appropriate number of coordinates as explained in **PRC_TYPE_GRAPH_TextureApplication** type description.

Required or Option	**Data Type**	**Data Description**
Required	Unsignedinteger	PRC_TYPE_TESS_3D
Required	ContentBaseTessData	Tessellation coordinates
Required	Boolean	has_faces
Required	Boolean	has_loops
Required	Boolean	Must_calculate_cormals
Option: Must_calculate_normals	Character	Normal_recalculation_flags (not used should be zero)
Option: Must_calculate_normals	Double	Crease_angle
Required	Unsignedinteger	Number_of_normal_coordinates
Required	ArrayOf[Double]	Normal_coordinates
Required	Unsignedinteger	Number_of_wire_indices
Required	ArrayOf[Unsigninteger]	Wire_indices
Required	Unsignedinteger	Number_of_triangulated_indicies
Required	ArrayOf[Unsignedinteger]	Triangulated_index_array
Required	Unsignedinteger	Number_of_face_tessellation
Required	ArrayOf[PRC_TYPE_TESS_Face]	Face _tessellation_data
Required	Unsignedinteger	Number_of_texture_coordinates
Required	ArrayOf[Double]	Texture coordinates

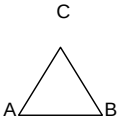
7.8.5.2 Example: triangle

triangle_indice = [0 3 6 0 3 9] (It Would be = [0 0 3 3 6 6 0 0 3 3 9 9] *with* ***must_recalculate_normals = false)***

Triangulated indices are the point indices and normal indices describing the face's triangulated representation (triangles, triangle fans, triangle stripes) in the array of points. Therefore, all these indices are multiples of 3.Triangles indices have to be ordered particulary such that they are consistent with triangle's normal orientation. For instance, in the following figure, vertices have to be stored counterclockwise if the Triangle normal is such that:

normal.(AB^AC) > 0 with . is the scalar product

and ^ is the vector product



7.8.5.3 Example : triangle fan

A **triangle fan** describes a set of connected triangles that share one central vertex. If N is the number of triangles in the fan, the number of vertices describing it is N+2. Triangle indices have to respect a precise order.The index that references the central vertex is stored first; then [F E D C B] or [B C D E F] are stored depending on the triangle's normal orientation.

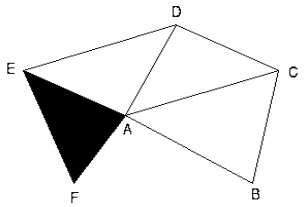


Diagram of four triangles with a common vertex A.

7.8.5.4 Example: triangle strip

A **triangle strip** is a series of connected triangles, sharing vertices.

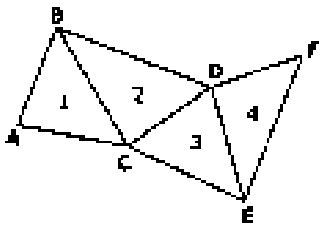


Diagram of four triangles, 1, 2, 3, and 4, with vertices A, B, C, D, E, F

7.8.5.5 PRC Tessellation Types

0x40000000	PRC_FACETESSDATA_NORMAL_Single	If this flag is set, the corresponding `OneNormal` entity (see PRC Tessellation Types) is planar and only one normal is defined for the entity. Otherwise, one normal per point is

		defined. This flag is only used for PRC_FACETESSDATA_*OneNormal entities

0x0001	PRC_FACETESSDATA_Polyface	Not used
0x0002	PRC_FACETESSDATA_Triangle	described with 6 indices (normal,point,normal,point,normal,point).
0x0004	PRC_FACETESSDATA_TriangleFan	described with 2*n indices (normal,point,... ,normal,point).
0x0008	PRC_FACETESSDATA_TriangleStripe	described with 2*n indices (normal,point,... ,normal,point).
0x0010	PRC_FACETESSDATA_PolyfaceOneNormal	Not used
0x0020	PRC_FACETESSDATA_TriangleOneNormal	described with 4 indices (normal,point,point,point).
0x0040	PRC_FACETESSDATA_TriangleFanOneNormal	described with n+1 indices (normal,point,point,..., point) if PRC_FACETESSDATA_NORMAL_Single is set described with 2*n indices (normal,point,... ,normal,point) if PRC_FACETESSDATA_NORMAL_Single is not set, in which case normal is to be interpreted as triangle normal (last normal is repeated)
0x0080	PRC_FACETESSDATA_TriangleStripeOneNormal	Described with n+1 indices (normal,point,point,..., point) if PRC_FACETESSDATA_NORMAL_Single is set Described with 2*n indices (normal,point,... ,normal,point)if PRC_FACETESSDATA_NORMAL_Single is not set, in which case normal is to be interpreted as triangle normal (last normal is repeated)
0x0100	PRC_FACETESSDATA_PolyfaceTextured	Not used
0x0200	PRC_FACETESSDATA_TriangleTextured	This is the same as PRC_FACETESSDATA_Triangle except that there are texture coordinate indices between normal and point indices. The variable **number_of_texture_coordinate_indexes** in PRC_TYPE_TESS_Face specifies the number of indices. For example, a simple triangle with one texture coordinate index is described by (normal,texture,point,normal,texture,point,normal,texture,point).
0x0400	PRC_FACETESSDATA_TriangleFanTextured	This is the same as PRC_FACETESSDATA_TriangleFan except that there are texture coordinate indices between normal and point indices. The variable **number_of_texture_coordinate_indexes** in

		PRC_TYPE_TESS_Face specifies the number of indices. For example, a triangle fan with one texture coordinate index is described by (normal,texture,point,normal,texture,point,normal,texture,point).
0x0800	PRC_FACETESSDATA_TriangleStripeTextured	This is the same as PRC_FACETESSDATA_TriangleStripe except that there are texture coordinate indices between normal and point indices. The variable **number_of_texture_coordinate_indexes** in PRC_TYPE_TESS_Face specifies the number of indices. For example, a triangle stripe with one texture coordinate index is described by (normal,texture,point,normal,texture,point,...,normal,texture,point).
0x1000	PRC_FACETESSDATA_PolyfaceOneNormalTextured	Not used
0x2000	PRC_FACETESSDATA_TriangleOneNormalTextured	This is the same as PRC_FACETESSDATA_TriangleOneNormal except that there are texture coordinate indices between normal and point indexes. The variable **number_of_texture_coordinate_indexes** in PRC_TYPE_TESS_Face specifies the number of indices. For example, a simple triangle with one texture coordinate index is described by (normal,texture,point,texture,point,texture,point).
0x4000	PRC_FACETESSDATA_TriangleFanOneNormalTextured	This is the same as PRC_FACETESSDATA_TriangleFanOneNormal except that there are texture coordinate indices between normal and point indexes. The variable **number_of_texture_coordinate_indexes** in PRC_TYPE_TESS_Face specifies the number of indices. For example, a triangle fan with one texture coordinate index is described as follows: (normal,texture,point,... ,normal,texture,point) if PRC_FACETESSDATA_NORMAL_Single is not set. (normal,texture,point,... ,texture,point) if PRC_FACETESSDATA_NORMAL_Single is set.
0x8000	PRC_FACETESSDATA_TriangleStripeOneNormalTextured	This is the same as PRC_FACETESSDATA_TriangleStripeOneNormal except that there are texture coordinate indices between normal and point

		indexes. The variable **number_of_texture_coordinate_indexes** in PRC_TYPE_TESS_Face specifies the number of indices. For example, a triangle stripe with one texture coordinate index is described as follows: (normal,texture,point,... ,normal,texture,point) if PRC_FACETESSDATA_NORMAL_Single is not set. (normal,texture,point,... ,texture,point) if PRC_FACETESSDATA_NORMAL_Single is set

7.8.6 PRC_TYPE_TESS_Face

7.8.6.1 General

This represents tessellation data for a face. An entity of this type only exists in a PRC File because it is referenced by a PRC_TYPE_TESS_3D. The coordinates, normals, and indices of the triangulated data are found in the PRC_TYPE_TESS_3D which references this entity.

The following is a description of the variables in the file:

- **size_of_line_attributes** is the number of entries in **line_attributes**
- **Line_attributes** is an array of line styles
- **Start_of_wire_data** represents the starting index for the wire data in the array of **wire_indices** of the **PRC_TYPE_TESS_3D** entity. Using **sizes_wire**, and **start_of_wire_data** determines where to retrieve wire point coordinates**.**
- **Size_of_sizes_wire** is the number of entries in **sizes_wire**
- **sizes_wire** is an integer array of the number of indices for each wire edge of this face. The indicies are stored in the array **wire_indicies** within the **PRC_TYPE_TESS_3D** entity containing this face.
- **used_entities_flag** is a flag that indicates the types of triangulated entities in the array **TriangulatedData;** the various bits of this flag are defined in **PRC Tessellation Types**; the order of the bits in this flag correspond with the ordering of triangulation data within the **TriangulateData** arrays.
- **start_triangulated** represents the starting index for the triangulated data of this face within the array **triangulated_index_array** of the **PRC_TYPE_TESS_3D** entity containing this face.
- **Size_of_TriangulatedData** is the number of entries in the array **TriangulatedData**.
- **TriangulatedData** is an integer array describing the tessellation data for a face. See below for a description of the data within this array.
- **number_of_texture_coordinate_indexes** represents the number of texture coordinate indices (see **PRC Tessellation Types**).
- **has_vertex_colors** is a flag indicating if colors are stored directly in the vertices. Either there is no color for the vertices, or every vertex must have a color.

- **behavior** denotes the graphics behaviour, such as inheritance, for the entity in the tree owning the face tessellation, as described in **behavior_bit_field** of **GraphicsContent** section. Note that this is not relevant if **size_of_line_attributes** is 0 (meaning that there are no graphic attributes for the face).

The tessellation data for a face consists of a number of triangulations. Each triangulation is of one of the types described in **PRC Tessellation Types**. The bit fields of the **used_entities_flag** indicate if that type of triangularization data is present and the order of the bit fields from low to high (0 to 31) indicate the order of data in the **TriangulatedData** array if such data is present.

The first entry of the **TriangulatedData** array indicates the number of triangles (**PRC_FACETESSDATA_Triangle**); other entries will indicate either the number of triangularizations of a specific type or the number of indices for a triangularization type.

For example, consider a face whose tessellation data contains 5 triangles, two fans of 5 and 7 indices, and 1 stripe of 11 indices. In this case,

- **used_entities_flag** = PRC_FACETESSDATA_Triangle & PRC_FACETESSDATA_TriangleFan & PRC_FACETESSDATA_TriangleStripe
- **start_triangulated** = index into **triangulated_index_array** of the start of data for this face; this would be 0 for a single face in a **PRC_TYPE_TESS_3D** entity.
- **TriangulatedData** = (5, 2, 5, 7, 1, 11)

size_of_line_attributes can have one of following values.

- **0** if there are no graphics. In this case, all graphics are inherited from the owner of the **PRC_TYPE_TESS_3D** data.
- **1** if there is one graphic associated with the whole face tessellation data**.**
- **2** or higher : in this case, the number of graphics entities must be equal to the number of entities stored in the current face. For instance, if the face contains 3 triangles, 2 fans and 7 stripes, this number must be set to 12.

The size of a wire edge of a FaceTessData is limited to 16383 (0x3FFF) points. For wire edges, two flags denote the drawing **behaviour** (see **Special flags for 3DwireTessData wire tessellation.**).

For example, if there are two loops having 2 and 1 wire edges, respectively: For the first loop, the first edge would have 10 points and the second edge would have 20 points. For the second loop there would be 12 points. The array would be [10, 20 | **PRC_FACETESSDATA_WIRE_IsClosing**, 12 | **PRC_FACETESSDATA_WIRE_IsClosing**] Note that the indices for the edge extremes are always stored. Therefore, the 10th point of the first edge should be at the same location as the first point of the second edge.

In the cases where the tessellation type contains one normal, the number of points is combined with the flag **PRC_FACETESSDATA_NORMAL_Single**. Hence the number of points is always limited to 0x3FFFFFFF whatever the **PRC tessellation type for FaceTessData**.

Required or Option	Data Type	Data Description
Required	Unsignedinteger	PRC_TYPE_TESS_Face

Required	Unsignedinteger	Size_of_line_attributes
Required	ArrayOf[Unsignedinteger]	array of Line_attributes where each entry is (Index_of_line_style+1) into the array of line styles (see **GraphicsContent**).
Required	Unsignedinteger	Start_of_wire_data
Required	Unsignedinteger	Size_of_sizes_wire
Required	ArrayOf[Unsignedinteger]	Sizes_wire
Required	Unsignedinteger	Used_entities_flag
Required	Unsignedinteger	Start_triangulated
Required	Unsignedinteger	Size_of_TriangulatedData
Required	ArrayOf[Unsignedinteger]	TriangulatedData
Required	Unsignedinteger	Number_of_textured_coordinate_indexes
Required	Boolean	has_vertex_colors
Required	VertexColors	Vertex color data
Option: size_of_line_attributes > 0	UnsignedInteger	behavior

7.8.6.2 Face Wire Tessellation Flags

0x4000	PRC_FACETESSDATA_WIRE_IsNotDrawn	Indicates that the edge should not be drawn (its neighbor will be drawn).
0x8000	PRC_FACETESSDATA_WIRE_IsClosing	Indicates that this is the last edge of a loop.

7.8.7 PRC_TYPE_TESS_3D_Wire

7.8.7.1 General

Tessellation for a 3D wire edge

The following is a dscription of the variables in the file:

- The **ContentBaseTessData** class defines the **number_of_coordinates** and **coordinates** of the tessellation data. It also defines a flag, **is_calculated**, indicating whether the data was calculated during import or comes directly from a CAD system. Data in the **coordinates** array is interpreted as the x, y, and z coordinates of the 3D points in the tessellation.

- **number_of_wire_indexes** is the number of integers in the wire_indexes array.

- **wire_indexes** is an array of integers which is defined below.
.
- **has_vertex_colors** is a flag indicating if colors are stored directly in the vertices. Either there is no color for the vertices, or every vertex must have a color.

If **number_of_wire_indexes** is zero, the tessellation **coordinates** represents a single wire edge. If **number_of_wire_indexes** is not zero, the array **wire_indexes** defines a sequence of wire edges by specifying the number_of_indices_per_wire_edge followed by the indicies for that wire edge. The indicies define the index into the **coordinates** array for the (x, y, z) of a point along the wire edge. The indices must be a multiple of 3.

The number_of_indices_per_wire_edge is an encoded 32 bit integer containing the following:

Flag	Number_of indicies_per_wire_edge

The flag is the leftmost 4 bits and is interpreted using **3D Wire Tess Flags** to indicate

- if the first point of this wire should be linked to the last point of the preceeding wire (PRC_3DWIRETESSDATA_IsContinuous)
- if the last point of this wire should be linked to the first point of this wire (PRC_3DWIRETESSDATA_IsClosing)

Required Or Option	**Data Type**	**Data Description**
Required	Unsignedinteger	PRC_type_tess_3D_wire
Required	ContentBaseTessData	Tessellation coordinates
Required	Unsignedinteger	Number_of_wire_indexes
Required	ArrayOf[Integer]	Wire_indexes
Required	Boolean	has_vertex_colors
Option: TRUE	VertexColors	Vertex color data

7.8.7.2 VertexColors

- **is_rgba:** TRUE implies the color is 4 characters (RGBA); FALSE implies the color is 3 characters (RGB).
- **is_segment_color:** TRUE implies there is a color for every two points of the wire; FALSE implies there is a color for every point of the wire.
- **b_optimised:** reserved for future use; it should always be false.
- **color_array** is a sequence of characters indicating the RGB or RGBA values for each of the vertexes or segments in the tessellation.

The number_of_colors stored in the **color_array** must be calculated from the from the number of point indices

- found in the **wire_indexes** array in the case of a **PRC_TYPE_TESS_3D_Wire**
- found in the **`sizes_triangulated`** `in the case of a` **`PRC_TYPE_TESS_Face`**

If **is_segment_color** is FALSE, there is a color for every point in the appropriagte array; otherwise, there is a color for every segment in the array. It is important to remember that implicit points must also have a color. An implicit point is a point that is implied in the sequence of wire points but is not stored in the file, such as when a wire is of type **PRC_3DWIRETESSDATA_IsClosing** (i.e. last point connects to first point, but the first point is not repeated in the file).

Required Or Option	Data Type	Data Description
Required	Boolean	Is_rgba
Required	Boolean	Is_segment_color
Required	Boolean	B_optimized
Option: !B_optimized	ArrayOf[ColorDdata]	

7.8.7.3 ColorData:

Required or Option	Data Type	Data Description
Required	Color	Color of first vertex; **is_RGBA** indicates either 3 characters (FALSE) or 4 characters (TRUE)
Required	ArrayOf[ColorDataRemainder]	Color of remaining vertexes

7.8.7.4 ColorDataRemainder

Required or Option	Data Type	Data Description
Required	Boolean	TRUE implies this entry has the same color as the previous one
Optional:FALSE	Color	Color of vertex; **is_RGBA** indicates either 3 characters (FALSE) or 4 characters (TRUE)

7.8.7.5 3D Wire Tess Flags

0x10000000	**PRC_3DWIRETESSDATA_IsClosing**	Indicates that the first point is implicitly repeated after the last one to close the wire edge.
0x20000000	**PRC_3DWIRETESSDATA_IsContinuous**	Indicates that the last point of the preceding wire should be linked with the first point of the current one.

7.8.8 PRC_TYPE_TESS_Markup

7.8.8.1 General

Contains information describing the graphical behavior for the tessellation associated to a markup (**PRC_TYPE_MKP_Markup**).

The tessellation of a markup uses two arrays containing the codes and the coordinates.

The codes array contains a description of the entities used in the tessellation.

The coordinates array (**ContentBaseTessData**) contains point coordinates as well as other floating point values used by entities.

Each entity has at least two codes. The first code contains the entity type and the number of specific inner codes. The second code is the number of doubles (coordinates) for this entity. These doubles are located in the coordinates array.

- The **ContentBaseTessData** class defines the **number_of_coordinates** and **coordinates** of the tessellation data. In the case of markup, the flag **is_calculated**, is meaningless. Data in the **coordinates** array is normally interpreted as x,y,z data, but can also contain data such as the 16 elements of a matrix.
- **Number _of_codes** specifies the size of the code array
- **Code Numbers** is an integer array of code numbers for the markup entity
- **Number_of_text_strings** specfies the size of the string array
- **Text_strings** is an array that contains the text strings for any text entities contained current markup object.
- **Tessellation label** is the name of the corresponding **PRC_TYPE_MKP_Markup**. **Behavior** is the bit field describes the graphical behavior of the tessellation.

Required	Data Type	Data Description
Required	Unsignedinteger	PRC_TYPE_TESS_Markup
Required	ContentBaseTessData	Tessellation coordinates
Required	Unsignedinteger	Number_of_codes
Required	ArrayOf[UnsignedInteger]	Code Numbers associated with the current markup object
Required	Unsignedinteger	Number_of_text_strings
Required	ArrayOf[String]	Text_strings
Required	String	Tessellation label
Required	Character	behavior

7.8.8.2 Markup Flags

Special flags for various markup conditions. These flags are used to extract the corresponding information from the integer code array as explained in **Markup tessellation codes**.

0x08000000	**PRC_MARKUP_IsMatrix**	Bit to denote that the current markup entity is a matrix
0x04000000	**PRC_MARKUP_IsExtraData**	Bit to denote that the current markup entity is extra data (it is neither a matrix nor a polyline).
0xFFFFF	**PRC_MARKUP_IntegerMask**	Integer mask to retrieve the number of inner codes for a given entity
0x3E00000	**PRC_MARKUP_ExtraDataType**	Mask to retrieve the integer type of the markup entity

7.8.8.3 Markup Tessellation Behavior

Special flags for handling the graphical behavior of the tessellation associated with the markup object. these flags are represented by bits in the variable **Behavior**.

0x01	**PRC_MARKUP_IsHidden**	The tessellation is hidden
0x02	**PRC_MARKUP_HasFrame**	The tessellation has a frame
0x04	**PRC_MARKUP_IsNotModifiable**	tessellation is given and should not be modified
0x08	**PRC_MARKUP_IsZoomable**	tessellation has zoom capability
0x10	**PRC_MARKUP_IsOnTop**	The tessellation is on top of the geometry
0x20	**PRC_MARKUP_IsFlipable**	The text tessellation can be flipped to always be readable on screen. This value is currently unused.

7.8.8.4 Description of the first Markup code.

There are three masks needed to identify the entity type.
PRC_MARKUP_IsMatrix
PRC_MARKUP_IsExtraData
PRC_MARKUP_ExtraDataType

If none of these masks is set, the entity is a polyline.
PRC_MARKUP_IsMatrix should not be set if **PRC_MARKUP_IsExtraData.**

If **PRC_MARKUP_IsExtraData** is set then **PRC_MARKUP_ExtraDataType mask should be used to retrieve** the type of markup entity.

7.8.8.5 Description of the second Markup code.

The second code is the number of doubles needed by the entity.
The following table shows, for each defined entity, the extra data type, the number of inner codes , and the number doubles in the coordinate array.

The extra data type is set using the **PRC_MARKUP_ExtraDataType** mask.

7.8.8.6 Table of entities

In the table below, [1] indicates entity types which are used to define blocks. The notion of block is discussed in next section. [2] indicates entity modes as discussed in further section as well.

.

Entity	Extra Data Type	Number of inner codes	Number of Doubles
Polyline	None	0	Points*3
Matrix mode [1]	None	0 or number of entities in the block	0 or number of doubles used in the block (at least 16)
Pattern	O	3+number of loops	Points in loop*3
Picture	1	1	0
Triangles	2	0	Number triangle*9
Quads	3	0	Number of quads*12
Face view model[1]	6	0 or number of entities in the block	0 or number of doubles used in the block
Frame draw model[1]	7	0 or number of entities in the block	0 or number of doubles used in the block
Fixed Size Model[1]	8	0 or number of entities in the block	0 or number of doubles used in the block
Symbol	9	1	3
Cylinder	10	0	3
Color	11	1	0
Line stipple[2]	12	0	10
Font	13	1	0
Text	14	1	2
Points	15	0	Number Points*3
Polygon	16	0	Number points*3
Linewidth[2]	17	0	0 or 1

7.8.8.7 Block and entity modes

7.8.8.7.1 Description of a block.

Blocks are defined by face view, frame draw, fixed size and matrix modes (described below).
Each block is surrounded by the corresponding entity. At the start of a block, the entity modifies the state, which may include the line style, or current transformation matrix. The state is restored at the end of the block.

For example, a matrix mode starts by defining a matrix that will multiply the current transformation matrix, draws some entities, and ends with another matrix mode entity indicating the end of the mode.

7.8.8.7.2 Description of modes used in block definitions.

Because the face view, frame draw, fixed size, and matrix modes start with the corresponding entity and end when the same entity is encountered, they define blocks.

The starting entity has a non-zero number of inner codes. It represents the number of codes until the end of the block, not counting the two mandatory codes for each entity. The same rule applies to the doubles. The ending entity has no inner codes and no doubles.

The number of inner codes makes it possible to skip a block when reading a tessellation. To treat the content of a block, use the numbers as shown in the following table.

Mode	Number of inner codes	Number of doubles
Face view (starting)	number of entities in the block	number of doubles in the block (at least 3)
Face view (ending)	0	0
Frame draw(starting)	number of entities in the block	number of doubles in the block (at least 3)
Frame draw(ending)	0	0
Fixed size(starting)	number of entities in the block	number of doubles in the block (at least 3)
Fixed size(ending)	0	0
Matrix(starting)	number of entities in the block	number of doubles in the block (at least 16)
Matrix(ending)	0	0

The following example shows the codes for defining a matrix mode and then 3 points in the block.

(0x08000000 + 3) (begin matrix block; 3 entities to follow), 16 + 3*3 (number of doubles in block)

(0x04000000 + 15) (first point) , 3 (it uses 3 doubles)

(0x04000000 + 15) (second point) , 3 (it uses 3 doubles)

(0x04000000 + 15) (third point) , 3 (it uses 3 doubles)

(0x08000000) (end matrix block), 0 (matrix ending; no double)

7.8.8.7.3 Description of entity modes.

The line stipple and line width modes operate identically to the modes used in block definitions, but the numbers correspond only to the entity and not to the block.
For the line stipple mode, the number of inner codes denotes the start (1) or the end (0) of the block.
For the line width mode, the number of doubles denotes the start (1) or the end (0) of the block.

7.8.8.8 Entity description

7.8.8.8.1 General

For each entity, the following tables show the mandatory codes and the inner codes, as well as the doubles needed by the entity.

7.8.8.8.2 Polyline

There is an (x,y,z) triplet for each point of the polyline.

Codes	**Doubles**
0	X
Number of points*3	Y
	Z

7.8.8.8.3 Triangles

A list of triangles. There is an (x,y,z) triplet for each point of the triangle list.

Codes	Doubles
0	X
Number of points*9	Y
	Z

7.8.8.8.4 Quads

A list of quads. There is an (x,y,z) triplet for each point of the quad list.

Codes	Doubles
0	X
Number of points*12	Y
	Z

7.8.8.8.5 Polygon

There is an (x,y,z) triplet for each point of the polygon.

Codes	Doubles
0	X
Number of points*3	Y
	Z

7.8.8.8.6 Points

A list of points. There is a (x,y,z) triplet for each point.

Codes	Doubles
0	X
Number of points*3	Y
	Z

7.8.8.8.7 Face view mode

In this mode, all the drawing entities are parallel to the screen (billboard). The point given in the doubles corresponds to the origin of the new coordinate system in which entries are drawn parallel to the screen.

Codes	Doubles
0 or number of entities in block	X
0 or number of doubles in block	Y
	Z

7.8.8.8.8 Frame draw mode

In this mode, all the drawing entities are given in 2-dimensional space. The point given in the doubles corresponds to a 3D point projected onto the screen, providing the origin of the 2-dimensional coordinate system in which to draw (viewport).

Codes	Doubles
0 or number of entities in block	X
0 or number of doubles in block	Y
	Z

7.8.8.8.9 Fixed size mode

In this mode, all the drawing entities are drawn at a fixed size, independent of zoom. The point given in the doubles corresponds to the origin of the new coordinate system in which to draw at fixed size.

Codes	Doubles
0 or number of entities in block	X
0 or number of doubles in block	Y
	Z

7.8.8.8.10 Matrix mode

In this mode, all the drawing entities are transformed by the current transformation matrix multiplied by the matrix given in the doubles. At the end of the mode, the transformation matrix that was previously active is restored.

Codes	Doubles
0 or number of entities in block	A(1,1)
0 or number of doubles in block	A(2,1)
	A(3,1)
	A(4,1)
	A(1,1)
	A(1,2)
	A(1,3)
	…
	A(4,3)
	A(4,4)

7.8.8.8.11 Symbol

The point given in the doubles corresponds to the position of the symbol in 3D.
The pattern identifier is an index into the picture array stored in **FileStructureInternalGlobalData**
The symbol is a **VPicturePattern** type.

Codes	Doubles
1	X
1	Y
Pattern identifier	Z

7.8.8.8.12 Color

This entity defines a color that will be effective until a new one is defined.
The color identifier is an index into the color array stored in **FileStructureInternalGlobalData.**

Codes	Doubles
1	
0	
Color identifier	

7.8.8.8.13 Line style mode

This entity defines the line style that will be effective inside the block.
The first code is 1 for beginning the block and 0 for ending.
The line style identifier is an index into the line stype array store in **FileStructureInternalGlobalData**.

Codes	Doubles
0 or 1	
0	
Line Style identifier	

7.8.8.8.14 Font

This entity defines the font used for the next Text entity.
The font identifier is and index into the font array stored in **FileStructureInternalGlobalData**.

Codes	Doubles
1	
0	
Font identifier	

7.8.8.8.15 Text

This entity defines text to be rendered using the current font (defined by the Font entity).
The text index refers to the text number in the string array.
W and H correspond to the width and height, respectively, of the text in real display coordinates.

Codes	Doubles
1	W
2	H
Text Index	

7.8.8.8.16 Line width mode

This entity defines the line width that will be effective inside the block.
The number of doubles is 1 for the beginning of the block and 0 for the ending of the block.
W is the line width to use in the block. It is not used when ending the block.

Codes	Doubles
0	W
0 or 1	

7.8.8.8.17 Cylinder

The cylinder is positioned by a matrix mode, oriented with the z-axis, with the base at Z = 0 and the top at Z = Height.

Codes	Doubles
0	Base radius
3	Top radius
	Height

7.8.8.8.18 Image

This entity defines an image positioned at the current position.
The picture identifier is an index into the picture array stored in the **FileStructureInternalGlobalData** section of the file..

Codes	Doubles
1	
0	
Picture identifier	

7.8.8.8.19 Pattern

The pattern identifier is an index into the fill pattern array stored in the **FileStructureInternalGlobalData**.section of the file.

The filled mode is one of the following values: 0 = OR, 1 = AND, 2 = XOR.

The behavior is a bit field, with the 0x1 bit indicating whether to ignore the view transformation. If it is true, the pattern is not transformed by the current view transformation. The other bits should be set to zero.There is an (x,y,z) triplet for each point in the loops, and they are listed in sequential order.

Codes	Doubles
3 + number of loops	X
Number of points in loop * 3	Y
Pattern identifier	Z
Fill mode	
Behavior	
Number of points for loop 1	
…	
Number of points for loop n	

7.8.9 PRC_TYPE_TESS_3D_COMPRESSED

A highly compressed tessellation which is a compact approximation of a **PRC_TYPE_TESS_3D** object.The starting point is a mesh described with points, normals and triangles, with implicit topology. Each triangle has 3 normals (one for each point). The triangle normal is determined by cross-product on its vertices, oriented in conjunction with one of its 3 normals (it is assumed that the calculation gives the same sign whatever the normal). A tolerance for approximation is also given as input. All triangles are supposed to be not-degenerated relative to this tolerance : they must have edge length and height greater than the tolerance.
The input non-compressed mesh is duplicated into a working structure which will be traversed as described below. At each step, approximation on points, normals and textures occur and the results of these approximations are reinjected into this working structure and used in further calculations until traversal is completed, producing an output compressed mesh. The approximation algorithms on points and normals are described in the following sections. The code corresponding to the basic functions used in those algorithms is given as pseudo code in the Section 9.

7.8.9.1 Mesh Traversal

The input mesh is traversed as follows (see figure 1):

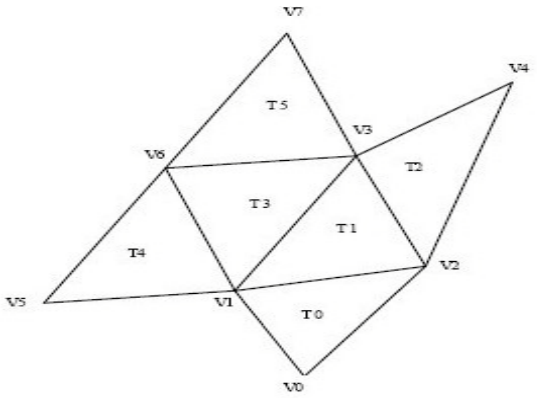


Figure 1

Let T0 [V0 V1 V2] be the first triangle and [V0 V1] be the first edge of T0 (this edge is arbitrarily chosen). Then, T1 is the left neighbor of T0 and the edge between V1 and V2 is the first edge of T1. T3 is the left neighbor of T1 and the edge between V1 and V3 is the first edge of T3. T2 is the right neighbor of T1 and the edge between V2 and V3 is the first edge of T2, and so on... Left / right characteristic for the neighbor is determined using the triangle normal.

To traverse the mesh structure, 3 cases are considered.

- If the current triangle has only one neighbor which is not treated, this neighbor will be treated just after the current triangle.
- If the current triangle has both a left and a right neighbors which are not treated, the left triangle will be treated just after the current triangle. The right triangle is put in a "last in first out" stack.
- If the current triangle has no neighbor which are not treated, the last triangle in the stack is treated.

At each stage, the stack is updated accordingly.

7.8.9.2 Mesh points and triangles:

The following table contains a description of the variables used while computing the compressed point mesh.

Name	Type	Description
Tolerance	Double	3D point tolerance
Point_array	ArrayOf[Integer]	Array of points
Edge_status_array	ArrayOf[Character]	Flags to describe a Triangle

		neighbor
Point_reference_array	ArrayOf[Integer]	Point relative reference
Point_is_a_reference	ArrayOf[Boolean]	Indicates whether a point is a reference

point_array describes the vertex coordinates of each point. Coordinates are stored only if necessary, if the point has not been encountered before. As denoted in previous section, the first triangle [V0 V1 V2] of a mesh, its first edge [V0 V1] and first point V0 are chosen arbitrarily. This first triangle is stored following way : For V0, its coordinates X,Y,Z are divided by the **tolerance** the nearest 3 integers are stored as V0app and V0 is updated in the working structure (the coordinates of V0 are replaced by their truncated values). This assumes that those coordinates divided by tolerance do not overflow a 32 bit integer. This is a condition at every step of the compression process. For V1, DV1 = V1-V0 is computed and the result is compressed and stored like the first point as DV1app; then V1 is updated in the working structure. For V2 : DV2 = V2 - (V0+V1) / 2 is computed, compressed and stored the same way as DV2app; then V2 is updated in the working structure. For subsequent triangles, they are always entered through an edge as explained in previous section. Let [V0 V1 V2] be the current triangle to treat, [V0 V1] be the entering edge and Tn [V0 V1 V3] the already-treated triangle which is the neighbor of the current towards [V0 V1]. If V2 is not a reference as denoted in **point_is_a_reference** array, V2 is stored following way : A coordinate system is defined using Tn. Origin O = (V1 + V0) * 0.5. Others axis are defined below.

$$\vec{X} = (\overrightarrow{V1} - \overrightarrow{V0}) / \left\| \overrightarrow{V1} - \overrightarrow{V0} \right\| \qquad (1)$$

$$\overrightarrow{Z_{temp}} = \overrightarrow{V3} - \vec{O} \qquad and \qquad \vec{Z} = \overrightarrow{Z_{temp}} \wedge \vec{X} / \overrightarrow{\|X\|} \qquad \vec{Z} = \vec{Z} / \overrightarrow{\|Z\|} \qquad (2)$$

$$\vec{Y} = \vec{Z} / \overrightarrow{\|Z\|} \wedge \vec{X} / \overrightarrow{\|X\|} \qquad (3)$$

In the equation (1), V0 and V1 are taken so that V0 has a treatement index less than V1 (which means that V0 has been treated before V1). A particular case occurs if the Z axis
or Y are null (length less than FLT_EPSILON). In these cases, they are computed by the function MakeOrthoRep() described in annex, using the unit axis X as input.
Then V2 is expressed in this coordinate system, compressed the same way as before and updated in the working structure. Then the next triangle is traversed as explained above.

edge_status_array describes triangles' neighbors. Each triangle has a flag which is initialized to 0 and then set to:

- |= *0x1 if Triangle has a Right Neighbor*
- |= *0x2 if Triangle has a Left Neighbor*

point_reference_array is used to store treatment indexes of points which have been stored by processing a previous triangle.

point_is_a_reference indicates if a point has been already treated.

7.8.9.3 Mesh Normal Description

The following table contains a description of the variables used while computing the compressed normal mesh.

Name	Type	Description
Normal_binary_data	ArrayOf[Boolean]	Information used to compute normal

Normal_angle_array	ArrayOf[ShortInteger]	Spherical coordinates of the normal
Is_face_planar	ArrayOf[Boolean]	Is associated face planar

normal_binary_data is a bit field used to store information types on normals.

- Bit *has_multiple_normal* is true if the current vertex has many normals. This bit is added only if the current vertex is encountered for the first time.
- Bit *triangle_normal_reversed* is true if the computed triangle normal used to define a local coordinate system must be reversed. See next paragraph for determination of the local coordinate system.
- Bit *is_a_reference* is true if the current normal is stored as a reference on another normal of the current vertex. In this case, reference_index denotes the value of the reference. It is stored in **normal_binary_data** on a variable number of bit : *number_of_bits*. *Number_of_bits* is computed using number_of_stored_normals : number of already actually stored normals (without references) on the current vertex.
- Bit *x_is_reversed* is true if the x-coordinate of the normal in the local coordinate system is reversed. (true if x is reversed). Same for *y_is_reversed*.

Normal_angle_array describe spherical coordinates of normals (normals are unit vectors). Values stored are comprised between 0 and PI / 2. For each triangle, a local coordinate system is computed and used to calculate these two angles. Finally, these two angles are compressed and temporarily stored in a short. The compressed value is computed using *normal_angle_number_of_bits*. This number must be less than 16, (default value is 10) to be stored in an array of shorts.

Is_face_planar is true if corresponding face is planar. In this case, only one normal is stored for all triangles of this face. It is stored when treating the first vertex of the first triangle of this face.

7.8.9.4 Mesh Normal Construction

For each triangle, 3 normals are computed and stored. The first one corresponds to the vertex that has the min treatement index in the first edge. The second one corresponds to the max treatement index in the first edge. Then, a local coordinate system X,Yand Z is defined from the triangles vertices in the working structure.

$$\vec{V1} = \frac{\overrightarrow{SecondVertex - firstVertex}}{\left\|\overrightarrow{SecondVertex - firstVertex}\right\|} \qquad \vec{V2} = \frac{\overrightarrow{ThirdVertex - firstVertex}}{\left\|\overrightarrow{ThirdVertex - firstVertex}\right\|}$$

$$\vec{V3} = \frac{\overrightarrow{ThirdVertex - SecondVertex}}{\left\|\overrightarrow{ThirdVertex - SecondVertex}\right\|}$$

$$\theta 1 = \left|\left(\vec{V1}, \vec{V2}\right) - \frac{\pi}{2}\right| \qquad \theta 2 = \left|\left(\vec{V3}, -\vec{V1}\right) - \frac{\pi}{2}\right| \qquad \theta 3 = \left|\left(\overrightarrow{-V2}, -\vec{V3}\right) - \frac{\pi}{2}\right|$$

$$If(\theta 1 < \theta 2) and (\theta 1 < \theta 3) \Rightarrow \vec{X} = \vec{V3} \qquad \vec{Z} = (\vec{V1} \wedge \vec{V2})$$

$$ElseIf(\theta 2 < \theta 3) \Rightarrow \vec{X} = \vec{V3} \qquad \vec{Z} = -\vec{V3} \wedge \vec{V1}$$

$$Else \Rightarrow \vec{X} = -\vec{V2} \qquad \vec{Z} = \left(\vec{V2} \wedge \vec{V3}\right)$$

Z is reversed to have a scalar product positive or null with the current vertex normal. Note that this coordinate system is the same for the 3 vertices of the triangle as a consequence of the primary condition on triangle normal. The result is stored in ***triangle_normal_reversed***.

$$\vec{Y} = \frac{\vec{Z}}{\|\vec{Z}\|} \wedge \vec{X}$$

A particular case occurs if the Z axis or Y are null (length less than FLT_EPSILON). In these cases, they are computed by the function MakeOrthoRep() described in annex, using the unit axis X as input. For each vertex normal, the angles in normal_angle_array are computed as described below (n denotes the triangle normal) :

$$\phi = \text{asin}(\vec{n} \cdot \vec{Z}) \qquad with \qquad \vec{n} \cdot \vec{Z} \in [0, 1], \phi \in \left[0, \frac{\pi}{2}\right]$$

$$\theta = \text{asin}\left(\frac{(\vec{n} - (\vec{n} \cdot \vec{Z}) \times \vec{Z})}{\|\vec{n} - (\vec{n} \cdot \vec{Z}) \times \vec{Z}\|} \cdot \vec{Y}\right) \qquad with \qquad \frac{\vec{n} - (\vec{n} \cdot \vec{Z}) \times \vec{Z}}{\|\vec{n} - (\vec{n} \cdot \vec{Z}) \times \vec{Z}\|} \cdot \vec{Y} \in [0, 1]$$

$$and \qquad \theta \in \left[0, \frac{\pi}{2}\right]$$

Spherical angles Theta and Phi are then compressed and stored the same way as follows (same formula for Phi) :

$$\theta_{short} = \frac{|\theta| \times (2^{normalAngleNumberOfBits} - 1)}{\frac{\pi}{2}}$$

These values are then written in unsigned short integers on 16 bits with a cast. Then, the nearest unsigned short values to these angles are stored in ***normal_angle_array***.

$$\theta_{short} \times \tfrac{\pi}{2} / (2^{normalAngleNumberOfBits} - 1) - |\theta| > \frac{1}{2} \Rightarrow \theta_{short} = \theta_{short} + 1$$

For each vertex in each triangle, Theta and then Phi are added in **normal_angle_array** if the normal is not a reference. The pseudo code below describes how ***normal_binary_data*** and ***normal_angle_array*** are filled.

```
If (number_of_stored_normal == 0 || !has_multiple_normal )
{
    Add has_multiple_normal in normal_binary_data

    Add triangle_normal_reversed in normal_binary_data

    Add x_is_reversed in normal_binary_data

    Add y_is_reversed in normal_binary_data

    Add Angles in normal_angle_array
}
```

```
else
{
    Add is_a_reference in normal_binary_data
    if (is_a_reference)
    {
    for( i = 0; i < number_of_stored_normal; i++)
            Add reference_index&(1<<i) in normal_binary_data
        }
        else
        {
        Add triangle_normal_reversed in normal_binary_data
                Add x_is_reversed in normal_binary_data
                Add y_is_reversed in normal_binary_data
                Add Angles in normal_angle_array
                }

            }
        }
```

After the compressed normal calculation, a compressed normal is computed and re-injected in the working structure as follow.

$$\theta_{comp} = \frac{\theta_{short} \times \frac{\pi}{2}}{2^{normalAngleNumberOfBits} - 1}$$

$$\phi_{comp} = \frac{\phi_{short} \times \frac{\pi}{2}}{2^{normalAngleNumberOfBits} - 1}$$

$$\overrightarrow{n_{comp}} = \cos(\theta_{comp})\cos(\phi_{comp}) \times \vec{X} + \sin(\theta_{comp})\cos(\phi_{comp}) \times \vec{Y} + \sin(\phi_{comp}) \times \vec{Z}$$

The cos and sinus functions are computed using a taylor expansion with 4 terms and the following expression.

if($\alpha > \pi/4$) => $\sin(\alpha) = \cos(\pi/2 - \alpha)$ $\cos(\alpha) = \sin(\pi/2 - \alpha)$

7.8.9.5 Mesh texture structure

The following table contains a description of the variables used for textures storage. This structure is used to describe the textures' UV parameters.

Name	Type	Description
All_face_has_texture	Boolean	False if there is at least one face without texture
Face_has_texture	ArrayOf[Boolean]	Does corresponding face have texture
Texture_data	CompressedTextureParameter	Information to retrieve UV texture parameters.

The combination of All_face_has_texture and Face_has_texture determines whether a face has textures. Texture_data contains information to retrieve UV textures' parameters. See **CompressedTextureParameter** for more details.

7.8.9.6 Mesh Attribute structure

The following table contains a description of the variables used to contain mesh attribute data.

Name	Type	Description
Is_point_color	Boolean	TRUE if there is at least one face with point color
Is_point_color_on_face	ArrayOf[Boolean]	TRUE if corresponding face has point color
Point_color_array	ArrayOf[Integer]	RGB or RGBA
Is_multiple_attribute	Boolean	TRUE if there is at least one face with multiple attributes
Line_attribute_array	ArrayOf[Short]	Indexes in the graphics array

point_color_array describes colors on vertices for each triangle. For each triangle vertex with point color, 5 characters are stored. The first character describe if the vertex has got RGB or RGBA components. Then 4 components are used to stored R, G, B, and alpha.
line_attribute_array describe indexes in a graphic array. If a face contains multiple attributes, one index per triangle is added in line_attribute_array. Otherwise, one index per face is added, when encountering the first triangle of this face.

7.8.9.7 Description of the data written to the file

The following is a description of the data in the file:

- **is_calculated** indicates whether the tessellation has been calculated during the import or has been read directly from a native file.
- **has_faces** is **true** if the entity is built using geometrical faces.
- **tolerance** represents the tolerance of the approximation of the original tessellation.
- **origin_array** contains three floating point coordinates that describe the bounding boxcenter of the compressed 3D tessellation data.
- **points_array** contains the array of vertex points.
- **edge_status_array** for each triangle**,** used to describe the triangles neighbors.
- **point_is_referenced_array_size** size of the reference array.
- **point_is_referenced_array** indicates whether a point is a reference.
- **number_of_referenced_points** size of the point reference array.
- **point_reference_array** relative point references.
- **triangle_face_array** represents, for each triangle, the index of the face to which it belongs.
- **character_array** is calculated as shown below.
- **character_array_compressed** is an integer array obtained by the Huffman algorithm, with 6 bits in **character_array**.
- **must_recalculate_normals** and **crease_angle** are described in **3D_TESS_FACE**.
- **number_implicit_normal** is reserved for future use.
- **normal_is_reversed** is reserved for future use.
- **normal_binary_data** information used to compute normal.
- **normal_angle_array** spherical coordinates.
- **normal_angle_number_of_bits** is the number of bits used to approximate the triangles normals. It must be lower than 16 and should be set to 10 bits to ensure good

performance.

- **normal_angle_array** is an unsigned short array containing values less than (1 << **normal_angle_number_of_bit**) - 1). This array is optionally compressed with a Huffman algorithm using **normal_angle_number_of_bit** bits.
- **is_normal_angle_array_compressed** indicates whether **normal_angle_array** is compressed.
- **normal_angle_array_compressed** is integer array obtained by the Huffman algorithm on normal_angle_array.
- **face_number** is derived from the maximum value in **triangle_face_array**.
- **is_face_planar** is true if the face is planar. In this case, only one normal per face is stored.
- **is_point_color** is true if at least one face has vertices with colors (RGB or RGBA).
- **is_point_color_on_face** is true if the corresponding face has vertices with colors (RGB or RGBA).
- **point_color_array** contains an RGB or RGBA component compressed using a Huffman algorithm with 8 bits.
- **is_multiple_line_attribute** indicates if there is at least one face with multiple line attributes.
- **is_multiple_line_attribute_on_face** is true if the face has multiple line attributes. If there is one line attribute on the face, one graphic referenced in **line_attributes_array** is associated with the face. Otherwise, the number of graphics referenced in **line_attributes_array** is equal to the number of triangles in the face .
- **no_texture** is true if there is no texture.
- **texture_data**. See corresponding chapter.
- **all_faces_have_texture** is true if all faces have a texture.
- **face_has_texture** is a boolean array. It indicates which faces have texure when no_texture is true and all_faces_have_texture is false.
- **has_behaviours** is true if special graphics behaviors or inheritances exist on faces or triangles. See **3D_TESS_FACE** for more information**.**

- **behaviours_array** represents the behavior for each face.

Required or Option	**Data Type**	**Data Description**
Required	Unsignedinteger	PRC_TYPE_TESS_3D_COMPRESSED
Required	Boolean	Is_calculated
Required	Boolean	Has_faces
Required	Boolean	Tolerance
Required	ArrayOf[FloatAsBytes]	Origin_Array The data is compressed as follows: FloatAsBytes(Origin_Array[i]) for i = 0, 1, 2 See Chapter **FloatAsBytes**
Required	CompressedIntegerArray	Point_array
Required	CharacterArray	Edge_status_array (2 bits per character only)
Required	CompressedIndiceArray	Triangle_face_array

Required	Unsignedinteger	Reference array size
Required	ArrayOf[Boolean]	Points_is_reference_array
Required	CompressedIndiceArray	Point_reference_array CompressedIndiceArray (see **CompressedIndiceArray**) invokes WriteCharacterArray (see **WriteCharacterArray**); in this case, the boolean value which indicates whether the character array is compressed is not stored. Its value is implicit and set to number_of_reference_points=>3
Required	Boolean	Must_recalculate_normals
Option:TRUE	ArrayOf[Boolean]	normal_is_reversed The number of normals is implicit, depending of the number of triangles and faces. Vertices have always as many normals as number of faces to which they belong.
Option:TRUE	Double	Crease_angle
Option:TRUE	Character	Normal recalculation flags (not used; should be zero)
Option:FALSE	Character	Normal_angle_number_of_bits
Option:FALSE	Unsignedinteger	Normal_binary_data_size
Option:FALSE	ArrayOf[Boolean]	Normal_binary_data
Option:FALSE	ShortArray	Normal_angle_array (size 16 bits)
Option:FALSE	ArrayOf[Boolean]	Is_face_planar
Required	Boolean	Is_point_color
Option:TRUE	ArrayOf[Boolean]	Is_point_color_on_face
Option:TRUE	CharacterArray	Point_color_array (size 8 Bits)
Required	Boolean	Is_multiple_line_attribute
Option:TRUE	ArrayOf[Boolean]	Is_multiple_line_attribute_on_face
Required	ShortArray	Line_attribute_array (size 16 bits)

Required	Boolean	No_texture
Option:FALSE	CompressedTexturePara meter	Texture_data
Option:FALSE	Boolean	All_faces_have_texture
Option: (!No_texture)&& (!All_faces_have_text ure)	ArrayOf[Boolean]	Face_has_texture
Required	Boolean	Has_behaviors
Option:TRUE	CharacterArray	Behaviours_array (size 8 Bits)

7.8.9.8 CompressedTextureParameter

The following table contains a description of the variables used to store UV textures' parameters.

- **binary_texture_data** represents a bit field. During mesh traversal, if the current vertex has a texture, a bit *texture_is_reference* is set in this array.This bit is true if the same UV parameter has already been stored during mesh traversal for the same vertex. In this case the reference index is stored in reference_array. Otherwise, this bit is set to false and UV parameters are stored in Texture_parameters.
- **`reference_array`** is used to reference UV parameters. The references on UV parameters are done per vertex. A UV parameter for the current vertex is referenced only if the same UV parameter has already been stored for the same vertex during the treatment of another triangle. This treatment is performed during mesh traversal, the same way as normals' treatment.
- **texture_parameters_tolerance** is reserved for future use and should be set to zero.
- **texture_parameters** is an array of float which contains UV textures' coordinates.This array is filled during mesh traversal as well.

Required or Option	**Data Type**	**Data Description**
Required	BinaryTextureData	binary_texture_data
Required	UnsignedInteger	reference_array_size
Required	ArrayOf[UnsignedIntegerWithVariableBitNumber]	`reference_array`
Required	Double	texture_parameters_tolerance reserved for future use. Should be set to 0.
Required	UnsignedInteger	texture_parameters_size

Required	ArrayOf[FloatAsBytes]	texture_parameters

7.8.9.9 BinaryTextureData

BinaryTextureData represents a bit field. It indicates during mesh traversal whether UV coordinates are referenced. (See previous chapter for more details). Then **last_integer_used_bit_number** bits are added to this array so that Texture_binary_data_size becomes a multiple of 32. Consequently, 0 <= **last_integer_used_bit_number** < 32. Then the unsigned integer array is written byte by byte : each unsigned integer leads to 4 bytes obtained from **MakePortable32BitsUnsigned**. (see chapter **MakePortable32BitsUnsigned**) ;

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Texture_binary_data_size / 32
Required	ArrayOf[bits(8)]	Texture_binary_data. This size array is equal to Texture_binary_data_size / 4.
Required	UnsignedInteger	last_integer_used_bit_number

7.9 Topology

7.9.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_TOPO	PRC_TYPE_ROOT + 140	
PRC_TYPE_TOPO_Context	PRC_TYPE_TOPO + 1	
PRC_TYPE_TOPO_Item	PRC_TYPE_TOPO + 2	
PRC_TYPE_TOPO_MultipleVertex	PRC_TYPE_TOPO + 3	nyi
PRC_TYPE_TOPO_UniqueVertex	PRC_TYPE_TOPO + 4	nyi
PRC_TYPE_TOPO_WireEdge	PRC_TYPE_TOPO + 5	nyi
PRC_TYPE_TOPO_Edge	PRC_TYPE_TOPO + 6	nyi
PRC_TYPE_TOPO_CoEdge	PRC_TYPE_TOPO + 7	
PRC_TYPE_TOPO_Loop	PRC_TYPE_TOPO + 8	nyi
PRC_TYPE_TOPO_Face	PRC_TYPE_TOPO + 9	yes
PRC_TYPE_TOPO_Shell	PRC_TYPE_TOPO + 10	nyi

PRC_TYPE_TOPO_Connex	PRC_TYPE_TOPO + 11	nyi
PRC_TYPE_TOPO_Body	PRC_TYPE_TOPO + 12	
PRC_TYPE_TOPO_SingelWireBody	PRC_TYPE_TOPO +1 3	
PRC_TYPE_TOPO_BrepData	PRC_TYPE_TOPO + 14	
PRC_TYPE_TOPO_SingleWireBodyCompress	PRC_TYPE_TOPO + 15	
PRC_TYPE_TOPO_BrepDataCompress	PRC_TYPE_TOPO + 16	
PRC_TYPE_TOPO_WIreBody	PRC_TYPE_TOPO + 17	

7.9.2 PRC_TYPE_TOPO

Abstract base class for topology.

7.9.3 PRC_TYPE_TOPO_Context

A topological context is a self-contained set of geometry and topology. Every geometrical and topological entity belongs to a single topological context. A topological context contains topological bodies represented as entry elements that point to topological items and geometry.

granularity represents the minimal size of an edge. This is a non-dimensional value.

tolerance represents the global base tolerance used in the context for topological elements. This is a non-dimensional value and can be superseded by looser local tolerances for particular topological elements. See Section 5.7.

smallest_thickness represents the smallest face thickness. It is used for loop algorithms, and its default value is 100 * granularity.

scale represents an optional scale that can be used to interpret the context data. This scale accommodates the different ranges of values of various CAD systems. The preceding values * scale yield dimensional values to be interpreted with the unit. For example, granularity * scale is dimensional granularity in part units.

The **behavior** field defines the behavior of **PRC_TYPE_TOPO_BrepData** bodies. It is a character of bits which define

- The order of outer loops within the list of loops on a face.
- Whether UV curves are clamped to the parameter domain boundaries for periodic surfaces or can extend past the boundary.
- Whether 3D edge curves (and faces) on closed or periodic surfaces are split along the seam or not.

The following table defines the bit values and behavior for this field:

Value	Type Name	Type Description
0x0001	PRC_CONTEXT_OuterLoopsFirst	Outer loops are first in the list of loops on a face.

0x0002	PRC_CONTEXT_NoClamp	UV curves can go beyond the domain of the bearing surface; this is used for interpreting UV curves on periodic-surfaces.
0x0004	PRC_CONTEXT_NoSplit	3d edge curves on closed or periodic surfaces are allowed to cross the seam of the surface.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TYPO_Context
Required	ContentPRCBase	
Required	Character	behavior
Required	Double	grandularity
Required	Double	tolerance
Required	Boolean	TRUE if smallest face thickness is present else FALSE
OPTION: TRUE	Double	Smallest face thickness
Required	Boolean	TRUE if the scale factor is present; else FALSE
OPTION: TRUE	Double	Scale
Required	UnsignedInteger	Number of bodies
Required	ArrayOf [PRC_TYPE_TOPO_Body]	Array of bodies

7.9.4 PRC_TYPE_TOPO_Item

Abstract root type for any topological entity (body or single item)

7.9.5 PRC_TYPE_TOPO_MultipleVertex

This represents a vertex whose position is the average of all edges‘ extremity positions which end at that vertex, that is,

```
Vertex_position = (points_for_vertex[0] + ... + point_for_vertex[number_of_points]) / number_of_points;
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_MultipleVertex
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	UnsignedInteger	Number of points
Required	ArrayOf [Vector3d]	Array of points for vertex

7.9.6 PRC_TYPE_TOPO_UniqueVertex

This represents a vertex whose position is specified by a 3D absolute postion and a tolerance. By default, the tolerance is the same as the tolerance of the topological context, but it can be over-ridden by a local one. The optional tolerance must be either 0.0 or greater than the tolerance of the topological context of the vertex.

The tolerance is used to define a sphere around the vertex within which the vertex may lie. It is used to determine if a position is the same (within tolerance) as this vertex. See Section 5.7.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_UniqueVetex
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	Vector3d	Position of vertex
Required	Bit	TRUE if there is an associated tolerance; else FALSE
OPTION: TRUE	Double	tolerance

7.9.7 PRC_TYPE_TOPO_WireEdge

A WireEdge may belong to either a wire body or single wire body. It is not bound by vertices.

The geometry of an wire edge is a 3D curve which has a optional trim interval to limit the geometric definition of the curve. The sense of the WireEdge is the same as the underlying curve.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_WireEdge
Required	ContentWireEdge	3D curve defining the wire edge and an optional interval to restrict the wire edge to a subset of the curve.

7.9.8 PRC_TYPE_TOPO_Edge

This class represents an edge which is a bounded segment of a curve where the segment is not coincident or self-intersecting except possibly at the end points of the edge. The geometry of an edge is provided by a wire edge which has an optional trim interval to limit the geometric definition of the curve. The sense of the edge is the same as the sense of the wire edge which is the same as the sense of the underlying curve.

An optional tolerance may be provided which is either zero or greater than the tolerance of the topological context the edge lies in. The tolerance is used to define a pipe centered on the edge within which the edge may lie. It is used to determine if a position lies on (within tolerance of) the edge. See Section 5.7.

A start and end vertex, of type PTR_TYPE_TOPO_UniqueVertex or PTR_TYPE_TOP_MultipleVertex, represent the start and end positions on the edge. The vertices and curve trim interval are related by the tolerances associated with the vertex and edge

```
Distance(Vertex, Edge_end)<= Vertex.Tolerance() + Edge.Tolerance()
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_Edge
Requried	ContentWireEdge	Curve providing the geometric definition of the edge along with trimming information
Required	PtrTopology	Start vertex
Required	PtrTopology	End vertex

Required	Boolean	Has tolerance
OPTION: TRUE	Double	Tolerance

7.9.9 PRC_TYPE_TOPO_CoEdge

A coedge represents the usage of an edge within a loop. The usage specifies the orientation of the coedge with respect to the edge:

- 0 Opposite direction
- 1 Same direction
- 2 Unknown

Normally, the orientation will be in the opposite or same direction. If the orientation is set to unknown, then PRC_CONTEXT_OuterLoopsFirst must be set to TRUE to assist in the computation of the proper orientation.

A coedge may have a UV curve which may be NULL or of type PRC_TYPE_CRV_NURBS. The UV curve maps R^1 (the interval of the UV curve) to R^2 (the domain space of the surface defining the face the loop of coedges lie in). As with an edge, the UV curve has an orientation (opposite, same, unknown) with respect to the orientation of the coedge within the loop.

If orientation_with_loop is equal to orientation_uv_with_loop, the 3D curve orientation is the same as 2D UV curve, that is, the start point of coedge (base_surface.evaluate(curve_uv.evaluate(curve_uv.param.min)) is the same as the start point of edge (within the tolerance of edge).

Required or Option	**Data Type**	**Data Description**
Required	UnsignedInteger	PRC_TYPE_TOPO_CoEdge
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	PtrTopology	This must be an edge (PRC_TYPE_TOPO_Edge) and must not be NULL
Required	PtrCurve	UV Curve in the domain of the face this coedge lies in; may be NULL
Required	Character	Orientation of edge with respect to the loop
Required	Character	Orientation of the UV curve with respect to the loop

7.9.10 PRC_TYPE_TOPO_Loop

7.9.10.1 General

A loop is an list of coedges bounding a portion of a face in a B-rep entity. The loop may define an outer boundary of a face or it may define a hole within the face.

A loop has the following properties:

- A loop is an ordered array of references to coedges which define the boundary of the loop.
- The list of coedges form a closed boundary for the portion of the face delimited by the loop. None of the references may be null and all references must be to PTR_TYPE_TOPO_CoEdge. The start vertex of one coedge must be the end vertex of the next coedge in the list.
- The loop of coedges is oriented with respect to the face normal using the rule of material to the left. That is, the cross-product of the tangent to the coedge at any position on the coedge with the face normal at that same position will point towards or opposite the material of the face within the loop. The orientation of the loop might be
    - o 0 Opposite direction
    - o 1 Same direction
    - o 2 Unknown

with respect to the normal of the face. If it is set to unknown, geometric tests must be performed to determine the correct orientation of the loop (same or opposite).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_Loop
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	Character	Orientation of loop with respect to surface normal
Required	UnsignedInteger	Number of coedges in the loop
Required	ArrayOf [CoedgesInLoop]	Coedges in loop

7.9.10.2 CoedgesInLoop

This represents a list of coedges around a loop. The PtrTopology must not be NULL and must point to a PTR_TYPE_TOPO_CoEdge.

Each coedge in the loop may index a neighboring coedge which shares the same edge but represents another usage of the edge in a boundary of a face, usually another face on another surface.

Required or Option	Data Type	Data Description
Required	PtrTopology	Next coedge in loop
Required	UnsignedInteger	Index of neighboring coedge (i.e. coedge which points to the same edge as this coedge) or 0 if there is no neighboring coedge

7.9.11 PRC_TYPE_TOPO_Face

A face is a bounded portion of a surface where the surface is not coincident or self intersecting except possibly at the boundary of the face

It is defined by

- A surface providing the geometric definition of the face. The face always has the same orientation as the underlying surface.
- An optional Domain may restrict the definition of the face to a portion of the surface. Otherwise the parameter domain of the face is the domain of the surface.
- Like a vertex and an edge, a face has an associated tolerance which is the topological context tolerance unless an optional tolerance is specified. If the optional tolerance is specified, it must be either 0.0 or greater than the topologial context tolerance. See Section 5.7.
- An unordered list of loops delimiting the bounded portion (interior) of the face.
- One of the loops represents the exterior boundary of the face and the other loops (if any) represent interior loops (holes) within the face. If PRC_CONTEXT_OuterLoopsFirst is set to TRUE in the topological context the face is contained in, the index of the outer loop must be defined.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_Face
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	PtrSurface	Surface geometry
Required	Bit	TRUE if the suface definition is to be trimmed to a specific domain; else FALSE
OPTION: TRUE	Domain	UV domain of trimmed surface
Required	Bit	TRUE if there is a tolerance associated with this face; else

		FALSE
OPTION:TRUE	Double	Tolerance
Required	UnsignedInteger	Number of loops in this face; must be 1 or more
Required	Integer	Index of outer loop; it must be set to -1 if it is not defined
Required	ArrayOf [PtrTopology]	Array of loops within this face; each pointer must be of type PRC_TYPE_TOPO_Loop

7.9.12 PRC_TYPE_TOPO_Shell

7.9.12.1 General

A shell is a collection of faces which form either a closed or open boundary.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_Shell
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	Boolean	TRUE if the shell is closed; else FALSE
Required	UnsignedInteger	Number of faces in shell
Required	ArrayOf [FacesInShell]	Faces within shell

7.9.12.2 FacesInShell

This represents a collection of faces within a shell. Each face is oriented with respect to the underlying surface so that the shell normal points outside the material of the shell if the shell is closed and is arbitrary otherwise.

The orientation of the surface with respect to the shell may be

- 0 Opposite direction
- 1 Same direction
- 2 Unknown

If the orientation is unknown, geometric tests must be performed to determine the correct orientation (within the shell) of the face (same or opposite) with respect to the surface.

Required or Option	Data Type	Data Description
Required	PtrTopology	Face within this shell; this must be non-NULL and of type PRC_TYPE_TOPO_Face
Required	Character	Orientation of face with repsect to the underlying surface

7.9.13 PRC_TYPE_TOPO_Connex

This represents a region of space delimited by one or more shells. The shells may be open or closed, may touch at a vertex, an edge, or a face, or may be contained within another shell if the interior shell represents a void within the exterior shell.

The region

- may represent a skin if the shells are open
- may represent a manifold solid if all of the shells are closed but not touching
- may represent a non-manifold solid where all of the shells are closed but some touch at a vertex, edge, or face

If the connex is delimiting material, it is manitory that it be bounded by one or more closed shells.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_Connex
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	UnsignedInteger	Number of shells in connex
Required	ArrayOf [PtrTopology]	Shells within this connex; each entry must be a shell, be non-NULL, and be of type PRC_TYPE_TOPO_Shell

7.9.14 PRC_TYPE_TOPO_Body

This represents an abstract type for any topological body

- PRC_TYPE_TOPO_SingleWireBody

- PRC_TYPE_TOPO_BrepData
- PRC_TYPE_TOPO_SingleWireBodyCompress
- PRC_TYPE_TOPO_BrepDataCompress

7.9.15 ContentBody

ContentBody provides additional infromation about base topological entities (PRC_TYPE_TOPO_SingleWireBody and PRC_TYPE_TOPO_BrepData) such as its name, attributes, and CAD identifier and how the bounding box for a PRC_TYPE_TOPO_BrepData has been calculated.

The following table shows the possible values for bounding box behavior:

Value	Type Name	Type Description
0x001	PRC_BODY_BBOX_Evaluation	Bounding box based on geometry
0x002	PRC_BODY_BBOX_Precise	Bounding box based on tessellation
0x004	PRC_BODY_BBOX_CADData	Bounding box given by a CAD data file

Required or Option	Data Type	Data Description
Required	BaseTopology	Optional topology information (name, attributes, CAD identifier)
Required	Character	Bounding box behavior; relevent only for PRC_TYPE_TOPO_BrepData; otherwise must be set to 0

7.9.16 ContentWireEdge

This represents the data defining a wire edge. It points to a 3D curve defining the geometrical shape of the edge. Any curve, including curves on a surface may be used. An optional interval may be used to limit the portion of the curve used to define the geometry of the edge. This interval must lie within the interval of the underlying curve. If no triming interval is specified, the edge is defined by the interval defining the curve.

Required or Option	Data Type	Data Description
Required	BaseTopology	Common topology data (name, attributes, CAD identifier)
Required	PtrCurve	3D curve defining the geometry of the wire edge

Required	Bit	TRUE if the wire edge restricts the 3D curve to a subset
OPTION: TRUE	Interval	Interval defining the subset of the 3D curve represented by the wire edge

7.9.17 PRC_TYPE_TOPO_SingleWireBody

PRC_TYPE_TOPO_SingleWireBody is the topological equivalent of a single curve.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_SingleWireBody
Required	ContentBody	Common data for PRC base entities
Required	PtrTopology	wire edge must be of type PRC_TYPE_TOPO_WireEdge or PRC_TYPE_TOPO_Edge

7.9.18 PRC_TYPE_TOPO_BrepData

This is the main representation of solid and surface topology (which is not highly compressed).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_BrepData
Required	ContentBody	Common data for PRC base entities
Required	UnsignedInteger	Number of connex entities in this B-rep
Required	ArrayOf [PtrTopology]	Array of connex entities in this B-rep; each entry must be non-NULLL and reference a PRC_TYPE_TOPO_Connex entity
OPTION:TRUE	BoundingBox	Optional bounding box; the required field ContentBody defines the Boolean flag indicating the presence of this bounding box

7.9.19 PRC_TYPE_TOPO_SingleWireBodyCompress

This represents a single wire body stored in compressed format.

Curve_tolerance is the tolerance used to approximate the curve of a single wire body. See Section 5.7.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_SingleWireBodyCompress
Required	ContentBody	Common data for PRC base entities
Required	Double	Curve_tolerance is the tolerance that has been used to approximate the curve of a single wire body.
Required	CompressedCurve	

7.9.20 PRC_TYPE_TOPO_BrepDataCompress

7.9.20.1 General

This represents manifold brep data stored in compressed format. In contrast to PRC_TYPE_TOPO_BrepData, geometrical and topological entities are not shared with other bodies even if they belong to the same topological context.

- **brep_data_compressed_tolerance** represents the tolerance used for the brep data approximation.
- **number_of_bits_to_store_reference** represents the number of bits written in the file for the following integers
- **number_vertex_references** represents the number of referenced verticies in the brep; see **CompressedVertex**
- **number_edge_references** represents the number of referenced edges in the brep; see **CompressedCurve**

The number of faces in the compressed brep is calculated as the number of faces in all of the shells in all of the connex entities.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_TOPO_BrepDataCompress
Required	Double	brep_data_compressed_tolerance
Required	NumberOfBitsThenUnsignedInteger	number_of_bits_to_store_reference
Required	UnsignedIntegerWithVariableBitNumber	number_vertex_references
Required	UnsignedIntegerWithVariableBitNumber	number_edge_references

Required	Boolean	TRUE if this brep consists of one connex entity with one shell
Option:TRUE	CompressedShell	Single compressed shell
Option: FALSE	MultipleCompressedConnex	Multiple compressed connex stored in file
Required	ArrayOf[BaseTopology]	Base topology data for each of the faces in the compressed brep data; the order of elements in this array corresponds to the order the faces are encountered in the scanning of connex/shell data within the compressed brep

7.9.20.2 MultipleCompressedConnex

This represents the data stored when the compressed brep data contains multiple connex entities or multiple shells within a single connex entity.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number of connex entities
Required	ArrayOf[CompressedConnex]	Array of compressed connex entities

7.9.20.3 CompressedConnex

This represents all of the shells within a compressed connex entity.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number of shell entities
Required	ArrayOf[CompressedShell]	Array of compressed shells within a connex entity

7.9.20.4 CompressedShell

This represents the compressed data for a single shell.

Required or Option	Data Type	Data Description
Required	Boolean	True if there is only a single face in the

		shell
Option: FALSE	NumberOfBitsThenUnsignedInteger	Number of faces in the shell if there is more than a single face
Required	ArrayOf[CompressedFace]	Array of faces within the shell. All iso faces are stored first followed by all of the non-iso faces.
Required	ArrayOf[Boolean]	Array of Boolean values indicating if the face is an iso face (TRUE) or not (FALSE). This is in the same order as the previous array of compressed faces.

7.9.20.5 Compressed Face

7.9.20.5.1 General

This represents the data for a single compressed face. There are two types of compressed faces: iso-faces and ana-face. An iso-face is a surface trimmed by four iso-parametric curves. If a face is not an iso-face, it is an ana-face.

The types of iso-faces include ISO_PLANE, ISO_CYLINDER, ISO_CONE, ISO_SPHERE, ISO_TORUS, and ISO_NURBS. Except for an ISO_NURBS, an iso-face is described using two curves. For example, an ISO_CYLINDER is described using the first line and the first circular arc. These data are sufficient to deduce the cylindrical surface and the two other trimming edge curves.

For both types of faces, all curves used to define or trim them are 3D. Therefore, surface parameterizations are not described and are set arbitrarily. For performance reasons, it is preferable that the surface parameterization is set so that the trimming edges do not cross seams on closed or periodic surfaces.

A curve is implicit if it is computed using iso-face properties. For example, the third and fourth curves in an ISO_CYLINDER are implicit curves. Other curves are explicit. Explicit curves are stored with the same orientation as the loop that references them first in the compressed B-rep data.

If the face belongs to a shell and the curve was already serialized by a neighbor's face, it is referenced with an index.

7.9.20.5.2 Enumeration of compressed entity types

The following lists the types for compressed entities.

Value	Name	Description
0	PRC_HCG_NewLoop	Intermediate trimming loop on an AnaFace
1	PRC_HCG_EndLoop	Last trimming loop on an AnaFace
2	PRC_HCG_IsoPlane	Plane trimmed by iso parametric curves
3	PRC_HCG_IsoCylinder	Cylinder trimmed by iso parametric curves
4	PRC_HCG_IsoTorus	Torus trimmed by iso parametric curves

5	PRC_HCG_IsoSphere	Sphere trimmed by iso parametric curves
6	PRC_HCG_IsoCone	Cone trimmed by iso parametric curves
7	PRC_HCG_IsoNurbs	Nurbs trimmed by iso parametric curves
8	PRC_HCG_AnaPlane	Plane trimmed with non-iso parametric curves
9	PRC_HCG_AnaCylinder	Cylinder trimmed with non-iso parametric curves
10	PRC_HCG_AnaTorus	Torus trimmed with non-iso parametric curves
11	PRC_HCG_AnaSphere	Sphere trimmed with non-iso parametric curves
12	PRC_HCG_AnaCone	Cone trimmed with non-iso parametric curves
13	PRC_HCG_AnaNurbs	Cone trimmed with non-iso parametric curves
14	PRC_HCG_AnaGenericFace	ana face lying on an uncompressed surface which can be of any type under PRC_TYPE_SURF
0	PRC_HCG_Line	Compressed line
1	PRC_HCG_Circle	Compressed circle
2	PRC_HCG_BsplineHermiteCurve	Compressed hermite bspline
12	PRC_HCG_Ellipse	Compressed ellipse; reserved for future use
13	PRC_HCG_CompositeCurve	Compressed composite

7.9.20.5.3 PRC_HCG_IsoPlane

The origin of the plane is the first vertex of the loop.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_IsoPlane
Required	Double	X coordinate of unit plane normal
Required	Double	Y coordinate of unit plane normal
Required	Boolean	TRUE if Z coordinate of unit plane normal is greater than 0.0; else FALSE
Required	ContentCompressedFace	Boundary of compressed face

7.9.20.5.4 PRC_HCG_IsoCylinder

A conforming PRC Reader should recognize accordingly lines and circles from ContentCompressedFace to reconstruct a cylinder surface.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_IsoCylinder
Required	ContentCompressedFace	Boundary of compressed face

7.9.20.5.5 PRC_HCG_IsoTorus

is_major_radius is true if the first serialized circle corresponds to the major radius, which is the largest one if the torus is a donut.

A conforming PRC Reader should recognize and classify circles from ContentCompressedFace to reconstruct a torus surface.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_IsoTorus
Required	Boolean	Is_major_radius
Required	ContentCompressedFace	Boundary of compressed face

7.9.20.5.6 PRC_HCG_IsoSphere

A conforming PRC Reader should recognize and classify circles from ContentCompressedFace to reconstruct a sphere surface.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_IsoSphere
Required	ContentCompressedFace	Boundary of compressed face

7.9.20.5.7 PRC_HCG_IsoCone

A conforming PRC Reader should recognize accordingly lines and circles from ContentCompressedFace to reconstruct a cone surface.

Required or	Data Type	Data Description

Option		
Required	CompressedEntityType	PRC_HCG_IsoCone
Required	ContentCompressedFace	Boundary of compressed face

7.9.20.5.8 PRC_HCG_AnaPlane

The origin of the plane is the first vertex of the loop.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_AnaPlane
Required	Double	X coordinate of unit plane normal
Required	Double	Y coordinate of unit plane normal
Required	Boolean	TRUE if Z coordinate of unit plane normal is greater than 0.0; else FALSE
Required	ContentCompressedFace	Boundary of compressed face

7.9.20.5.9 PRC_HCG_AnaCylinder

The analytic cylinder axis is defined from point_on_axis and cylinder_axis_direction. The cylinder radius is computed using loop vertices to obtain an average radius when projected onto the axis.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_AnaCylinder
Required	ContentCompressedFace	Boundary of compressed face
Required	CompressedPoint	Point on cylinder axis
Required	CompressedPoint	Direction of cylinder axis

7.9.20.5.10 PRC_HCG_AnaTorus

x_axis and y_axis define the torus placement.
x_axis length is equal to the major torus radius.
y_axis length is equal to the minor torus radius.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_AnaTorus
Required	ContentCompressedFace	Boundary of compressed face
Required	CompressedPoint	Torus center
Required	CompressedPoint	Torus x_axis
Required	CompressedPoint	Torus y_axis

7.9.20.5.11 PRC_HCG_AnaSphere

The radius of the sphere is computed using the first vertex of the loop and sphere_center.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_AnaSphere
Required	ContentCompressedFace	Boundary of compressed face
Required	CompressedPoint	Sphere center

7.9.20.5.12 PRC_HCG_AnaCone

axis_point and apex_point are used to compute the z-axis. The cone origin and the semi-angle correspond to the loop vertex that is furthest from the z-axis.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_AnaCone
Required	ContentCompressedFace	Boundary of compressed face
Required	CompressedPoint	Axis point
Required	CompressedPoint	Apex point

7.9.20.5.13 PRC_HCG_AnaGenericFace

This represents the data stored for any analytic face where the surface data of the face are not compressed. In this case, the trimming data for the face is saved and a regular surface description is saved, if there is one.

It cases where no surface data has been saved, the entity type PRC_TYPE_ROOT is stored in the file. This can happen if the original data in the PRC File are corrupted.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_AnaGenericFace
Required	ContentCompressedFace	Boundary of compressed face
Required	PRC_TYPE_SURF or PRC_TYPE_ROOT	Surface definition

7.9.20.5.14 PRC_HCG_IsoNurbs

7.9.20.5.14.1 General

An iso-NURBS surface is a face trimmed by four iso-parametric curves. This type of iso-face is special because it is not stored using the first two curves (see 7.9.20.5).

In this case, the surface is stored using the following information:

- **orientation_surface_with_shell** is defined in FacesInShell.
- **orientation_loop_with_surface** is defined in PRC_TYPE_TOPO_Loop.
- **sense_array** is the correspondence between the surface natural boundaries, as described in CompressedNurbs, and the trim curves.

The two first boolean values describe where the first curve is. Their possible values are:

- `false false` : the first curve is on `Umin`.
- `false true` : the first curve is on `Vmin`.
- `true false` : the first curve is on `Umax`.
- `true true` : the first curve is on `Vmax`.

If the last boolean value is false, the sense is the same as if the first curve is on Umin, the second curve is on Vmin, and the third curve is on Umax. If the last boolean value is true, the reverse sense is applied.

The four curve types are stored as reference (identifier), line, circle, or other (iso boundary of surface).

- **number_of_bits_to_store_reference** is described in PRC_TYPE_TOPO_BrepDataCompress.
- **reference_indice** is described in RefOrCompressedCurve.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_IsoNurbs

Required	Boolean	Orientation of surface with shell
Required	Boolean	Orientation of loop with surface
Required	ArrayOf[Boolean]	Three values of sense array
Required	PRC_HPG_Nurbs	Compressed Nurbs surface
Required	ArrayOf[IsoNurbsTrimCurve]	Trimming information for four boundary curves

7.9.20.5.14.2 IsoNurbsTrimCurve

Required or Option	Data Type	Data Description
Required	Boolean	Is referenced
Option:TRUE	UnsignedIntegerWithVariableBitNumber	Index of trim curve
Option:FALSE	IsoNurbsTrimCrv	Save trim curve

7.9.20.5.14.3 IsoNurbsTrimCrv

7.9.20.5.14.3.1 General

Save the actual trim curve data on an iso NURBS surface.

Required or Option	Data Type	Data Description
Option: Curve_type == PRC_HCG_Line	IsoNurbsTrimCrvLine	Save the data defining a trim curve which is of type PRC_HCG_Line
Option: Curve_type == PRC_HCG_Circle	IsoNurbsTrimCrvCircle	Save the data defining a trim curve which is of type PRC_HCG_Circle
Option: default	Boolean	Save the boolean flag with value TRUE; this means the trimming curve is one of the iso boundaries of the surface

7.9.20.5.14.3.2 IsoNurbsTrimCrvLine

Required or Option	Data Type	Data Description
Required	Boolean	Save FALSE
Required	Boolean	Save FALSE

7.9.20.5.14.3.3 IsoNurbsTrimCrvCircle

Required or Option	Data Type	Data Description
Required	Boolean	Save FALSE
Required	Boolean	Save TRUE
Required	CompressedCircle	Save compressed circle as trim curve

7.9.20.5.15 PRC_HCG_AnaNurbs

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_AnaNurbs
Required	ContentCompressedFace	Boundary of compressed face
Required	CompressedNurbs	Compressed nurbs surface

7.9.20.6 CompressedNurbs

7.9.20.6.1 General

This defines the storage of a compressed NURBS surface.

A compressed NURBS surface is defined by the following data.

See **CompressedControlPoints**, **CompressedKnotVector**, and **CompressedMultiplicities** for a description of converting from a NURBS surface to a compressed NURBS surface.

Name	Description
Number_ccpt_in_u	Number of control points in u
Number_ccpt_in_v	Number of control points in v
Number_knots_in_u	Number of knots in U

Number_knots_in_v	Number of knots in V
Is_closed_in_u	Boolean flag indicating a surface closed in u
Is_closed_in_v	Boolean flag indicating a surface closed in v
Ccpt	Two dimensional array of control points
Ccpt_type	Two dimensional array of integers defining the type of control point; see **type_param** for legal values
Cknot_u	Array of knots in U
Cknot_v	Array of knots in V
Mult_u	Array of multiplicities at the knots in U
Mult_v	Array of multiplicities at the knots in V
Is_rational	Boolean flag indicating if the surface is a rational surface (TRUE) and thus has an optional array of weights at the control points
Weight	Array of weights if the surface is rational

brep_data_compressed_tolerance is the tolerance for approximation as described in **PRC_TYPE_TOPO_BrepDataCompress**.

The following are used for stored compressed control points:

- **number_of_bits_for_isomin** is the number of bits used to store first row and column of control points
- **number_of_bits_for_rest** is the number of bits to store the remainder of the control points

The following are used for stored compressed knot values in U or V:

- **number_bit_parameter** is the number of bits used to store knots
- **tolerance_parameter** is the tolerance used to store knots

The following are used for stored weights of rational NURBS surfaces:

- **number_bit_weight** is the number of bits to store weights
- **weight_tolerance** is the tolerance used to store weights

A conforming PRC Writer must ensure that all these numbers and tolerances used to store control points, knots and weights are appropriately chosen so that the overall **brep_data_compressed_tolerance** is respected. The algorithms to ensure this are not part of this specification.

type_param can have one of the following values:

- `0` for uniform parameterization.
- `1` for non-uniform parameterization.

- 2 for pseudo-uniform parameterization, meaning that the parameterization is uniform except for extremities, mostly coming from a trim applied uniformly.

The following are used in the definition of a compressed nurbs surface:

Nurbs_tolerance = brep_data_compressed_tolerance / 5.0

Number_stored_knots_in_u = number_of_knots_in_u - 2

Number_stored_knots_in_v = number_of_knots_in_v – 2

Number_bits_u = (`degree_in_u ? ceil[ log( degree_in_u + 2 ) / log(2) ] : 2)`

Number_bits_v = (`degree_in_v ? ceil[ log( degree_in_v + 2 ) / log(2) ] : 2)`

tolerance_parameter `= 1./ 2^( number_bit_parameter -1)`

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_Nurbs
Required	UnsignedIntegerWithVariableBitNumber	Degree_in_u stored with 5 bits
Required	UnsignedIntegerWithVariableBitNumber	Degree_in_v stored with 5 bits
Required	UnsignedIntegerWithVariableBitNumber	Number_stored_knots_in_u stored using 16 bits; the integer represents the number of knots in u that are stored in the knot_u array
Required	ArrayOf[CompressedMultiplicities]	Array mult_u of data describing the knot multiplicities in U for Number_stored_ knots_in_u knots
Required	UnsignedIntegerWithVariableBitNumber	Number_stored_ knots_in_v stored using 16 bits; the integer represents the number of knots in u that are stored in the knot_u array
Required	ArrayOf[CompressedMultiplicities]	Array mult_v of data describing the knot multiplicities in V for Number_stored_ knots_in_v knots
Required	Boolean	Is_closed_in_u
Required	Boolean	Is_closed_in_v

Required	UnsignedIntegerWithVariableBitNumber	Number_of_bits_for_isomin stored using 20 bits
Required	UnsignedIntegerWithVariableBitNumber	Number_of_bits_for_rest stored using 20 bits
Required	CompressedControlPoints	Compressed control points for the surface
Required	CompressedKnotVector	Save type_param_u, number_knots_u, and knots_u
Required	CompressedKnotVector	Save type_param_v, number_knots_v, and knots_v
Required	Boolean	Is_rational
Option:TRUE	CompressedWeights	Save weights

7.9.20.6.2 CompressedMultiplicities

This defines an array of data stored to define the multiplicity of knots at each knot in the knot array for either U or V parameter.

number_stored_knots is either **Number_stored_knots_in_u** or **Number_stored_knots_in_v**

number_bits is either **number_bits_u** or **number_bits_v**.

Multiplicity is either **mult_u** or **mult_v**.

For each of the knots (0 <= i < number_stored_knots), the following data is stored

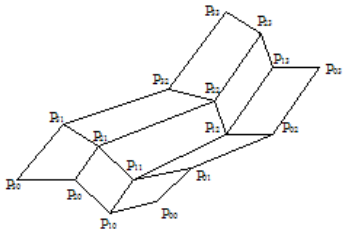
- A Boolean flag indicating if additional data is stored
    - **Multiplicity_is_stored** = (`multiplicity[i] == multiplicity[i-1] || !i)`
- Optionally store the multiplicity at this knot using **`number_bit`** `bits`

Required or Option	Data Type	Data Description
Required	Boolean	Multiplicity_is_stored
Option:TRUE	UnsignedIntegerWithVariableBitNumber	Multiplicity[i] stored using number_bits

7.9.20.6.3 CompressedControlPoints

7.9.20.6.3.1 General

nurbs_tolerance describes the tolerance used to approximate the original nurbs surface. It ensures that each point on the compressed nurbs surface is at a distance of the original surface less than nurbs_tolerance.



compressed_control_point P is a two dimension array containing double values that allow to recompute x, y, z values of consecutive points. At start, the surface is copied into a working structure which is updated step by step. P00 coordinates are pushed in compressed_control_point and into the working structure, thus P00_compressed = P00_working_structure = P00. Then P10 - P00 is computed and each vector component is approximated to the nearest multiple of the nurbs_tolerance. This truncated vector (P10 - P00_working_structure)_truncated is pushed into compressed_control_point. Then P10_working_structure is computed as follows : P10_ working_structure = (P10 - P00_working_structure)_truncated + P00 _working_structure. This point is re-injected into the working structure to avoid error propagation. In the same way, (P20-P10_working_structure)_truncated is pushed into compressed_control_point. P20_working_structure is computed and re-injected in the working structure. The same formula applies for each Pi0 and then P0i control points.

Internal compressed control points are also computed with previously stored points. for each i and for each j, Pij_compressed is computed

$$V = P_{j,i-1} - P_{j-1,i-1}$$

$$\vec{U} = P_{j-1,i} - P_{j-1,i-1}$$

$$\vec{N} = \vec{U} \wedge \vec{V}$$

$$P_{i,j(compressed)} = P_{i,j} - \left(P_{i-1,j-1} + \vec{V} + \vec{U}\right)$$

Four cases are considered:

- If Pij(compressed) length is less than nurbs_tolerance, control_point_type is set to zero and no value is pushed into compressed_control_point.

  $\left(P_{i-1,j-1} + \vec{V} + \vec{U}\right)$ is reinjected in the working structure to replace Pij.

  Otherwise, Pij(compressed) is evaluated in the following local coordinate system :

$$P_{i,j} = \left(P_{i,j(compressed)} \cdot \frac{\vec{\vec{U}}}{\left\|\vec{U}\right\|},\ P_{i,j(compressed)} \cdot \left(\frac{\vec{N}}{\left\|\vec{N}\right\|} \wedge \frac{\vec{U}}{\left\|\vec{U}\right\|} \right),\ P_{i,j(compressed)} \cdot \frac{\vec{\vec{N}}}{\left\|\vec{N}\right\|} \right)$$

- if $x^2 + y^2$ value is less than nurbs_tolerance^2, control_point_type is set to 1 and z is added in compressed_control_point.
- else if the z component length is less than nurbs_tolerance, control_point_type is set to 2 and the coordinates x and y are added in compressed_control_point.
- else x, y and z are stored in compressed_control_point. control_point_type is set to 3. If one of the vectors V, U, or N has a length less that 1e-12, this case is systematically used.

P is a two dimensional representation of the control points for a compressed Nurbs surface.

When linearized, the data is stored in the following order

- The corner point P[0][0] is stored as a Vector_3d (i.e. three doubles);
- The remainder of the first row of control points is stored as Point3DWithVarBitNumber
- The remainder of the first column of control points is stored as Point3DwithVarBitNumber
- The remainder of the matrix is stored by row (i.e. for a given i from 1 to number_ccpts_in_u, save the row of control points from j=1 to number_ccpts_in_v. For each point
    - Save the type of control point
    - If the type is 1, save the z coordinate of the control point
    - If the type is 2 save the x and y coordinates of the control point
    - If the type is 3 save the x, y, and z coordinates of the contol point

All Point3DWithVariableBitNumber data is written using Nurbs_tolerance and (number_of_bits_for_isomin + 1).

Required or Option	**Data Type**	**Data Description**
Required	Vector_3d	P[0][0]
Required	ArrayOf[Point3DwithVariableBitNumber]	P[0][j] 1 <= j < number_ccpt_in_v
Required	ArrayOf[Point3DwithVariableBitNumber]	P[i][0] 1 <= i < number_ccpt_in_u
Required	ArrayOf[InteriorCompressedControlPoints]	Array of data describing interior compressed control points. The points are saved using the order in the previous pseudo code

7.9.20.6.3.2 InteriorCompressedControlPoints

This represents the data stored for each interior compressed control point.

All DoubleWithVariableBitNumber data is written using Nurbs_tolerance and (number_of_bits_for_rest + 1).

The array of interior compressed control points is a two dimensional array written as

```
for ( i = 1; i < number_ccpt_in_u; i++) {
      for (j  = 1; j < number_ccpt_in_v; j++) {
            write P[i][j]
      }
}
```

Required or Option	Data Type	Data Description
Required	UnsignedIntegerWithVariableBitNumber	Ccpt_Type[i][j] Is written with 2 bits
Option: Type == 1	DoubleWithVariableBitNumber	P[i][j].z
Option: Type == 2	DoubleWithVariableBitNumber	P[i][j].x
Option: Type == 2	DoubleWithVariableBitNumber	P[i][j].y
Option: Type == 3	DoubleWithVariableBitNumber	P[i][j].x
Option: Type == 3	DoubleWithVariableBitNumber	P[i][j].y
Option:	DoubleWithVariableBitNumber	P[i][j].z

Type == 3		

7.9.20.6.4 CompressedKnotVector

7.9.20.6.4.1 General

The knot vectors are always between 0 and 1. The multiplicities are stored as described in the PRC File Format Specification. See 7.9.20.6.2**.** U knots are treated first, then V. Three types of knot parameterization are considered.

- If it is uniform, no parameter is stored in compressed_knot. This corresponds to type_param = 0.
- If it is pseudo uniform, the interval length between the two first parameters is computed and truncated using tolerance_parameter, and then stored into compressed_knot. Same for the two last parameters. This corresponds to type_param = 1.
- Otherwise, internal parameters are stored. For each internal parameter, a difference between the precedent compressed parameter is computed, truncated using tolerance_parameter and stored into compressed_knot. This corresponds to type_param = 2.

This represents the data stored for the knot vector of a compressed NURBS surface. The type_param and knot vector is saved for either the U or V parameter values.

Required or Option	**Data Type**	**Data Description**
Required	UnsignedIntegerWithVariableBitNumber	Number_bit_parameter is saved using 6 bits
Required	Boolean	`Type_param == 0`
Option: FALSE	CompressedKnots	

7.9.20.6.4.2 CompressedKnots

This represents saving the knots (either knots_u or knots_v) for a compressed NURBS.

Required or Option	**Data Type**	**Data Description**
Required	Boolean	Type_param == 1
Option: FALSE	Boolean	`Type_param == 2`

Required	ArrayOf[CompressedKnot]	Save the array of compressed knots

7.9.20.6.4.3 CompressedKnot

This represents a single entry in an array of compressed knots.

- The number of elements in the array is given by number_knots
- The DoubleWithVariableBitNumber is written using tolerance_parameter and number_bit_parameter
- The format of the data is either Double or DoubleWithVariableBitNumber in the file depending upon the Boolean test (number_bit_parameter > 30).

The array may be either an array of knot_u or knot_v defining the compressed NURBS surface.

Required or Option	Data Type	Data Description
Option: TRUE	Double	Knot[i]
Option:FALSE	DoubleWithVariableBitNumber	Knot[i]

7.9.20.6.5 CompressedWeights

This represents the data stored for the weights of a compressed NURBS surface.

- The number of entries to store is (number_ccpt_in_u * number_ccpt_in_v)
- The DoubleWithVariableBitNumber is written using weight_tolerance and number_bit_weight
- The format of the data is either Double or DoubleWithVariableBitNumber in the file depending upon the Boolean test (number_bit_weight > 30)

Required or Option	Data Type	Data Description
Required	UnsignedIntegerWithVariableBitNumber	Number_bit_weight save with 6 bits
Option:TRUE	ArrayOf[Double]	Save weight array (without compression)
Option:FALSE	Double	Weight tolerance
Option:FALSE	ArrayOf[DoubleWithVariableBitNumber]	Save weight array using compression

7.9.20.7 ContentCompressedFace

7.9.20.7.1 General

This represents the data for a compressed face. A compressed face is further classified by its trimming curves. An IsoFace is trimmed by four iso-parametric trimming curves. An AnaFace is trimmed by any other combination of trimming curves.

Vertex loops are used to represent a loop consisting of a single vertex, such as might exist on the apex of a cone, or a sphere touching a plane. They are represented by a degenerate line which has identical start and end vertices.

orientation_surface_with_shell describes the orientation of the surface normal with respect to the shell. See PRC_TYPE_TOPO_Shell.

orientation_loop_with_surface describes the orientation of a loop with respect to a surface normal. See PRC_TYPE_TOPO_Loop.

Required or Option	Data Type	Data Description
Required	Boolean	Orientation_surface_with_shell
Option: is_an_iso_face is TRUE	ContentCompressedIsoFace	Save data for a compressed iso face
Option: is_an_iso_face is FALSE	ContentCompressedAnaFace	Save data for a face trimmed by analytic curves

7.9.20.7.2 ContentCompressedIsoFace

In the case of an IsoFace, the first and second trim curves are stored explicitly, either by reference or actual data. Reference to the third or fourth trim curve is stored in case the actual data has been stored by an adjacent face. Otherwise, the third or fourth trim curve will have to be deduced from the first and second trim curves and a vertex representing the join between the third and fouth trim curves.

Required or Option	Data Type	Data Description
Required	Boolean	Orientation_loop_with_surface
Required	RefOrCompressedCurve	First trim curve on face
Required	RefOrCompressedCurve	Second trim curve on face
Required	Boolean	TRUE if third_trim_curve_is_not_yet_saved
Option:FALSE	NumberOfBitsThenUnsignedInteger	Index of the third trim curve

Required	Boolean	TRUE if fourth_trim_curve_is_not_yet_saved
Option:FALSE	NumberOfBitsThenUnsignedInteger	Index of the fourth trim curve
Option: third_trim_curve_is_not_yet_saved **AND** fourth_trim_curve_is_not_yet_saved	CompressedVertex	Save the common_third_fourth_vertex if third_trim_curve_is_not_yet_saved AND fourth_trim_curve_is_not_yet_saved

7.9.20.7.3 ContentCompressedAnaFace

7.9.20.7.3.1 General

If the AnaFace has trimming boundaries, all of the trimming loops are stored as an array of loops with the last loop in the array having type PRC_HCG_EndLoop and all other loops having type PRC_HCG_NewLoop.

If the AnaFace is defined by a torus and all of the trimming loop are vertex trimming loops (e.g. the loop consists of a single degenerate line), a point on the torus far from degeneracy is stored to indicate the interior of the face (this corresponds to a "lemon" side versus an "apple" side).

Required or Option	Data Type	Data Description
Required	Boolean	TRUE if surface is trimmed; else FALSE
Option: TRUE	ArrayOf[AnaFaceTrimLoop]	Array of trimming loops on the face. The last loop in the array is of type PRC_HCG_EndLoop; all others are of type HPC_HCG_TOPO_NewLoop
Option: all_loops_are_vertex_loops **AND** Surface_type is PRC_HCG_AnaTorus **AND** surface is trimmed	CompressedPoint	Point_on_torus

7.9.20.7.3.2 AnaFaceTrimLoop

This represents the trimming curves for a loop on a compressed AnaFace.

The last loop on the face is of type of PRC_HCG_EndLoop. Other loops on the face are of type PRC_HCG_NewLoop.

PRC_HCG_EndLoop and PRC_HCG_NewLoop are used to denote the end of loops with "particular" curve types that will signal the end of a loop or the end of all loops.

Required or Option	Data Type	Data Description
Required	Boolean	Orientation of loop with surface
Required	ArrayOf[RefOrCompressedCurve]	Array of curves in the trimming loop
Required	Boolean	Boolean value is always TRUE; this represents a boolean flag that means curve_is_NOT_already_stored is TRUE; it is used when reading a PRC File to signal the end of the curves in a loop and the end of all loops.
Required	CompressedEntityType	Will be PRC_HCG_EndLoop for last loop and PRC_HCG_NewLoop for all other loops

7.9.20.8 RefOrCompressedCurve

The flag **curve_is_NOT_already_stored i**ndicates if the trim curve has already been stored in the compressed brep data**.** If the curve has already been stored, the index of the curve is stored in the file; otherwise, a compressed version of the trim curve is stored.

The index is stored with a variable number of bits indicated by `number_of_bits_to_store_reference`. See `7.9.20` for a definition of this number.

Required or Option	Data Type	Data Description
Required	Boolean	Curve_is_NOT_already_stored
Option:FALSE	UnsignedIntegerWithVariableBitNumber	Index to the already stored compressed curve data
Option:TRUE	CompressedCurve	Store the compressed curve data.

7.9.20.9 CompressedCurve

7.9.20.9.1 General

A compressed curve is one of **PRC_HCG_Line**, **PRC_HCG_Circle, PRC_HCG_BsplineHermiteCurve or PRC_HCG_CompositeCurve.**

The curve type PRC_HCG_Ellipse is reserved for future use.

7.9.20.9.2 PRC_HCG_Line

The representation of a compressed line (PRC_HCG_Line) is context dependent.

If a compressed line is part of a compressed face, the compressed line is represented by a pair of start/end vertices; otherwise it is represented by a pair of start/end points.

Curve_trimming_face is TRUE if this compressed line is part of a PRC_TYPE_TOPO_BrepDataCompress; it is FALSE if this compressed line is a part of a PRC_TYPE_TOPO_SingleWireBodyCompress.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_Line
Required	StartEndData	Save the start/end trim data

7.9.20.9.3 PRC_HCG_Circle

7.9.20.9.3.1 General

The representation of a compressed circle (PRC_HCG_Circle) is context dependent.

The data stored for a compressed circle depends upon the context it is used in:

- curve_trimming_face is TRUE to indicate that the circle is used as part of the trimming data for a face (i.e as part of a PRC_TYPE_TOPO_BrepDataCompress) or is FALSE to indicate that the circle is not part of a face (i.e. it is used as part of PRC_TYPE_TOPO_SingleWireBodyCompress)

- compressed_iso_spline is TRUE if the circle is being used as the trim boundary of an PRC_HCG_IsoNurbs; otherwise it is FALSE

In addition, the data stored for a compressed circle depends on the geometry of the circle. A circular arc with angle 0.0, PI, or 2*PI will have different data. The particular_circle Boolean flag indicates this.

Required or Option	Data Type	Data Description
Option: compressed_iso_spline is FALSE	CompressedEntityType	PRC_HCG_Circle
Required	Boolean	Particular_circle
Option: Particular_circle is TRUE	ParticularCircle	Circular arc with angle 0, PI, or 2*PI
Option: Particular_circle is FALSE	GeneralCircle	Not a special circle

7.9.20.9.3.2 ParticularCircle

This is the data that is stored if the compressed circle is a special case (circular arc with angle 0, PI, or 2*PI).

Full_circle is TRUE if the start point and end point of the trim curve are identical (to within tolerance).

Required or Option	Data Type	Data Description

Required	Boolean	Full_circle
Option: compressed_iso_spline is FALSE	StartEndData	Save the start/end trim data
Option: Full_circle is TRUE	CompressedPoint	Center of circle
Option: Full_circle is TRUE	CompressedPoint	Normal to plane of circle
Option: Full_circle is FALSE	CompressedPoint	Middle point on circular arc

7.9.20.9.3.3 GeneralCircle

This is the data that is stored for a general circle or circular arc.

Required or Option	Data Type	Data Description
Required	StartEndData	Save the start/end trim data
Required	CompressedPoint	Center of circle
Required	Boolean	Circle_angle > PI

7.9.20.9.4 PRC_HCG_BsplineHermiteCurve

A compressed Hermite curve structure contains information to store a compact representation of a Bspline curve with degree 3.

A Hermite curve is defined by the following data:

- Compression_tolerance is the tolerance that was used to approximate the original nurbs curve; see 7.9.20 or 7.9.19.
- Compressed_points are the control points representing the Hermite curve
- Compressed_tangents describe how internal points computed from tangents are compressed
- Point_number_bits is the number of bits used to store each compressed control point coordinate if control points are stored using a variable number of bits; a conforming PRC Writer will obtain this number through the routine GetNumberOfBitUsedToStoreUnsignedInteger
- Tangent_number_bits is the number of bits used to store each compressed tangent coordinate if the tangents are stored using a variable number of bits

Start and end curve points are explicitly stored in the PRC File. Compressed_points (Ptc) and compressed_tangents (Tgtc) allow to compute the control polygon. Compressed_points contains points on the curve stored with difference :

$$P_0 = StartPt \quad P_{3i} = P_{3(i-1)} + \left(Pt_c[i\,x\,3], Pt_c[i\,x\,3+1], Pt_c[i\,x\,3+2]\right) \quad P_n = EndPt$$

compressed_tangent contains curves' tangents at each Ptc. It is used to determine two controls points between each point on curve (Pi)

$$P_1 = P_0 + \frac{(Tgt_c[0], Tgt_c[1], Tgt_c[2])}{\|P_3 - P_0\|}$$

$$P_2 = P_3 - \frac{(Tgt_c[3], Tgt_c[4], Tgt_c[5])}{\|P_3 - P_0\|}$$

$$P_4 = P_3 + \frac{(Tgt_c[3], Tgt_c[4], Tgt_c[5])}{\|P_6 - P_3\|}$$

The Bspline knot values are implicit and computed using control points.

$$U_0 = 0 \qquad U_i = U_{i-1} + \left\|\vec{P}_{3i} - \vec{P}_{3(i-1)}\right\|$$

Multiplicities are implicitly 4 for start and end knot and 3 for internal knots.

Required or Option	Data Type	Data Description
Required	CompressedEntityType	PRC_HCG_BsplineHermiteCurve
Required	StartEndData	Save the start and end trimming data as either vertices or points
Required	UnsignedIntegerWithVariable BitNumber	Number_bits is the number of bits used to store number_points; this number is stored with 4 bits
Required	UnsignedIntegerWithVariable BitNumber	Number_points is the number of compressed control points
Required	UnsignedIntegerWithVariable BitNumber	Point_number_bits is the number of bits used to store compressed points; this number is stored with 6 bits

Option: point_number_bits>30	ArrayOf[Vector3d]	Array of number_points compressed points
Option: Point_number_bits<=30	ArrayOf[Point3DWithVariableBitNumber]	Array of number_points compressed points with variable number of bits
Required	UnsignedIntegerWithVariable BitNumber	Tangent_number_bits is the number of bits used to store compressed tangents; this number is stored with 6 bits
Option: tangent_number_bits>30	ArrayOf[Vector3d]	Save number_points compressed tangents
Option: tangent_number_bits<=30	ArrayOf[Point3DWithVariableBitNumber]	Save number_points compressed tangents with variable number of bits

7.9.20.9.5 PRC_HCG_CompositeCurve

Required or Option	Data Type	Data Description
Option: ! compressed_iso_spline	CompressedEntityType	PRC_HCG_CompositeCurve
Required	StartEndData	Save the start and end trimming data as either vertices or points
Required	UnsignedInteger	Dimension of the compressed composite curve (either 2 or 3)
Required	Boolean	TRUE if the curve is closed; else FALSE
Required	UnsignedInteger	Number of curves in the composite
Required	ArrayOf[CompressedCurve]	Array of compressed curves which was written with the flag curve_trimming_face set to FALSE

7.9.20.9.6 StartEndData

This data defines the start and end verticies/positions of a trim curve. It is context dependent (see 7.9.20.9.2 or 7.9.20.9.3).

Option: Curve_trimming_face is TRUE	CompressedVertex	Start vertex
Option: Curve_trimming_face is TRUE	CompressedVertex	End vertex
Option: Curve_trimming_face is FALSE	CompressedPoint	Start point
Option: Curve_trimming_face is FALSE	CompressedPoint	End point

7.9.20.10 CompressedVertex

This represents a compressed vertex either as a compressed point or a reference to an already compressed point. Each compressed brep data serialization maintains an array of previously written vertices, starting at index 0.

number_of_bits_to_store_reference is the number of bits used to define a reference to compressed data and is described in PRC_TYPE_TOPO_BrepDataCompress.

Required or Option	**Data Type**	**Data Description**
Required	Boolean	TRUE if vertex is NOT already stored
Option:FALSE	UnsignedIntegerWithVariableBitNumber	Index to the already stored compressed point data
Option:TRUE	CompressedPoint	Compressed point data.

7.9.20.11 CompressedPoint

This represents a compressed point. The representation of the compressed point in the PRC File may be either as a Point3DwithVariableBitNumber or as a PRC_TYPE_TOPO_UniqueVertex depending upon how many bits are necessary to represent the data.

When a PRC File Writer writes a compressed point, the number of bits necessary to store the point is calculated using the formula

Double dTol = `brep_data_compressed_tolerance` / 100.0;

Unsigned int uMaxCoordinate = MAX(fabs(x), fabs(y), fabs(z)) / dTol + 1;

Unsigned int uNbBits = **GetNumberOfBitUsedToStoreUnsignedInteger**(uMaxCoordinate);

If uNbBits is greater than 30, the compressed point is stored as a PRC_TYPE_UniqueVertex; otherwise it is stored as a Point3DwithVariableBitNumber.

Required or Option	Data Type	Data Description
Required	UnsignedIntegerWithVariableBitNumber	uNbBits is stored using 6 bits
Option: uNbrBits <=30	Point3DWithVariableBitNumber	Compressed point data is stored as a 3D point with uNbBits bits and dTol tolerance
Option: uNbrBits >30	Vector3d	Compressed point data is stored as a unique vertex

7.9.21 PRC_TYPE_TOPO_WireBody

[Editor Note: Reserved for future use.]

7.9.22 References

7.9.22.1 General

Each curve, surface, or topological entity within an individual topological context is assigned an identifier which can be used to refer to it from other entities within the same topological context.

The first reference to an entity stores the actual data and generates an identifier for that entity. Subsequent references to that entity store only the identifier.

Note that when the actual data is stored, the first entry is an UnsignedInteger which indicates the type of data stored. **PRC_TYPE_ROOT (0)** is used to indicate that the entity corresponds to a NULL pointer and no additional data is saved. Otherwise, the integer will be one of the subtypes of curve, surface, or topology.

7.9.22.2 PtrCurve

If the Boolean flag is TRUE, the actual curve data is stored. Otherwise, the identifier of the curve is stored. The only legal curves are those represented by the base class PRC_TYPE_CRV.

Required or Option	Data Type	Data Description
Required	Boolean	If TRUE actual entity data stored; else FALSE (i.e. identifier of the entity is stored)

Option: TRUE	PRC_TYPE_CRV	The actual data for the stored entity.
Option: FALSE	UnsignedInteger	Identifier of stored entity

7.9.22.3 PtrSurface

If the Boolean flag is TRUE, the actual surface data is stored. Otherwise, the identifier of the surface is stored. The only legal surfaces are those represented by the base class PRC_TYPE_SURF.

Required or Option	Data Type	Data Description
Required	Boolean	If TRUE actual entity data stored; else FALSE (i.e. identifier of the entity is stored)
Option: TRUE	PRC_TYPE_SURF	The actual data for the stored entity.
Option: FALSE	UnsignedInteger	Identifier of stored entity

7.9.22.4 PtrTopology

If the Boolean flag is TRUE, the actual topological entity is stored. Otherwise, the identifier of the topological entity is stored. The only legal topological entities are those represented by the base class PRC_TYPE_TOPO.

Required or Option	Data Type	Data Description
Required	Boolean	If TRUE actual entity data stored; else FALSE (i.e. identifier of the entity is stored)
Option: TRUE	PRC_TYPE_TOPO	The actual data for the stored entity.
Option: FALSE	UnsignedInteger	Identifier of stored entity

7.10 Curve

7.10.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_CRV	PRC_TYPE_ROOT + 10	
PRC_TYPE_CRV_Base	PRC_TYPE_CRV + 1	
PRC_TYPE_CRV_Blend02Boundary	PRC_TYPE_CRV + 2	
PRC_TYPE_CRV_NURBS	PRC_TYPE_CRV + 3	
PRC_TYPE_CRV_Circle	PRC_TYPE_CRV + 4	
PRC_TYPE_CRV_Composite	PRC_TYPE_CRV + 5	
PRC_TYPE_CRV_OnSurf	PRC_TYPE_CRV + 6	
PRC_TYPE_CRV_Ellipse	PRC_TYPE_CRV + 7	
PRC_TYPE_CRV_Equation	PRC_TYPE_CRV + 8	
PRC_TYPE_CRV_Helix01	PRC_TYPE_CRV + 9	
PRC_TYPE_CRV_Hyperbola	PRC_TYPE_CRV + 10	
PRC_TYPE_CRV_Intersection	PRC_TYPE_CRV + 11	
PRC_TYPE_CRV_Line	PRC_TYPE_CRV + 12	
PRC_TYPE_CRV_Offset	PRC_TYPE_CRV +1 3	
PRC_TYPE_CRV_Parabola	PRC_TYPE_CRV + 14	
PRC_TYPE_CRV_PolyLine	PRC_TYPE_CRV + 15	
PRC_TYPE_CRV_Transform	PRC_TYPE_CRV + 16	

7.10.2 PRC_TYPE_CRV

Abstract type for curves.

7.10.3 PRC_TYPE_CRV_Base

7.10.3.1 General

Abstract type for all geometric curves. The following data is stored for all curve types.

7.10.3.2 ContentCurve

ContentCurve provides additional information about a curve such as its name and attributes, how it extends past its boundary (start and end points), and if it is a 2D or 3D curve.

Is_3D_flag: this flag is set toTRUE if the curve is a 3D curve; otherwise it is false. If a curve has a transformation, this flag is used to determine if it a 2D transformation or a 3D transformation (See 7.4.11)

Required or Option	Data Type	Data Description
Required	BaseGeometry	Optional geometric information
Required	UnsignedInteger	Indicates how the curve is extended; see 7.9.10
Required	Boolean	Is_3D flag; TRUE if the curve is 3D, otherwise FALSE

7.10.4 PRC_TYPE_CRV_Blend02Boundary

This entity represents a U iso-parametric curve of a Blend02 surface (along its center curve direction) at either the surfaces v minimum or v maximum value. The parameterization of the curve is inherited from the Blend02 surface.

blend represents the Blend02 surface and must not be NULL; it must be of type **PRC_TYPE_SURF_Blend02**.

bound indicates which blend U iso-parametric boundary is used:
- 0 represents the first blend bound (v minimum).
- 1 represents the second blend bound (v maximum).

Bounding_surface is the bounding surface that the Blend02Boundary curve lies on.
Sense_of_bounding_surface is equal to the sense of the bounding surface used in the intersection curve.

A Blend02Boundary curve is always a 3D curve (i.e. is_3d must be true).

A Transformation positions the curve in model space. This transformation is capable of translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization enables a reparameterization and trim of the curve.

A Blend02Boundary curve can also be considered to be the intersection between one of the bounding surfaces of the blend and a construction surface, which is an implicit surface intersecting the blend and is orthogonal to the blend surface along its bounding curve. Since the shape of this surface is not significant for the curve's geometry, it is not described.

In practice, a Blend02Boundary curve is created as an intersection curve but is interpreted as an iso curve on a Blend02 surface. The parameters of the intersection curve which are calculated during the construction process are saved are described below (see 7.10.13 for a description of the crossing points).

intersection_order is TRUE if the first intersection surface is the implicit surface and the second one is the bounding surface; otherwise it is FALSE.

number_of_crossing_points and **crossing_point_positions** give an approximation to the curve through an ordered set of spatial positions.

chordal_error is an estimate of the maximum distance between the curve and the set of segments given by the array of crossing points.

angular_error is the maximum angle between the tangents of two sequential crossing points.

bounding_surface is the adjacent surface.

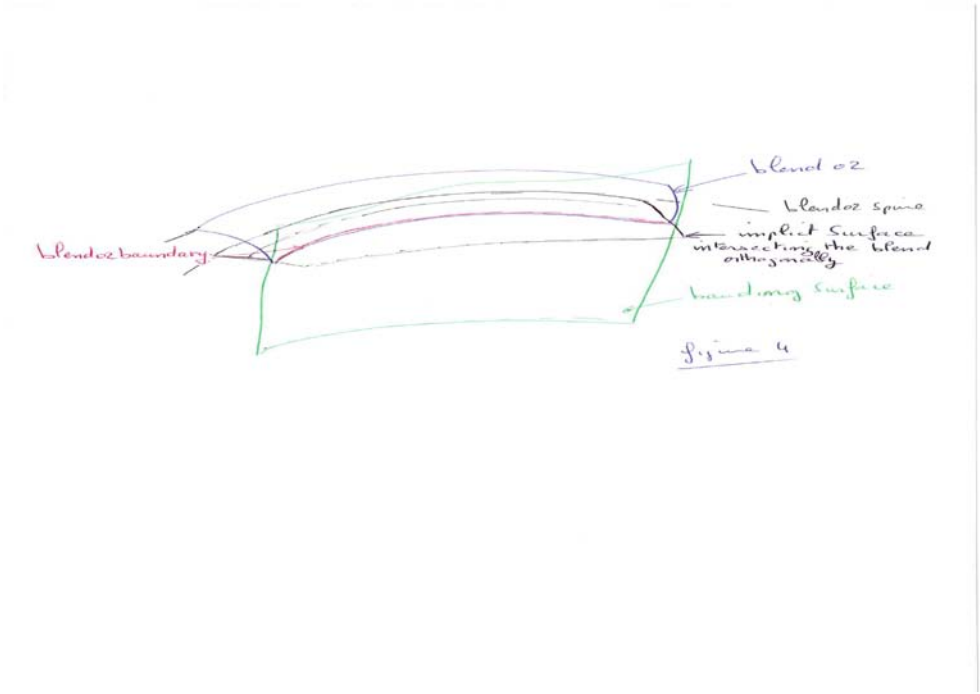
base_parameter is the parameter at the first crossing point.

base_scale is the scale at the first crossing point.

Start_limit_point and **end_limit_point** define the bounded portion of the curve; each point is further described by **start_limit_type** and **end_limit_type** (see 7.10.13.3).

The evaluation formula at a parameter value on a Blend02Boundary curve is

```
Calculate the implicit_parameter from the given parameter using this curves‘s Parameterization data.

If (bound == 0)

    v = minimum V value of the blend surface

Else

    v = maximum V value of the blend surface


XYZ = blend.evaluate( implicit_parameter, v )
```



Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Blend02Boundary
Required	ContentCurve	Common curve data
Required	Transformation	Transformation positioning curve in model coordinate system
Required	Parameterization	Redefine the parameterization
Required	PtrSurface	Blend surface must be of type PRC_TYPE_SURF_Blend02
Required	Integer	Bound
Required	UnsignedInteger	Number of crossing points

Required	ArrayOf [Vector3d]	Crossing point positions
Required	Double	Chordal error
Required	Double	Angular error
Required	PtrSurface	Bounding surface
Required	Boolean	Sense of bounding surface
Required	Boolean	Intersection order
Required	Boolean	Sense of intersection curve
Required	Double	Base parameter
Required	Double	Base scale
Required	Vector3d	Starting limit point
Required	UnsignedInteger	Starting limit type
Required	Vector3d	Ending limit point
Required	UnsignedInteger	Ending limit type

7.10.5 PRC_TYPE_CRV_NURBS

7.10.5.1 General

This class represents a non-uniform rational bspline curve. The curve may be either 2D or 3D and it may be either rational or non-rational.

A NURBS curve is defined by the following data:

- **d** is the degree of the curve and is restricted to the range 1 <= degree <= 25
- **P** is an array of control points.
- **Np** (the number of control points) = **highest_index_of_control_points** + 1
- **U** is the knot vector
  - the knots must be a non-decreasing sequence , that is, U[i] <= U[i+1]
  - The number of times a knot value u occurs in the knot vector is called its multiplicity; knot values are compared (to determine multiplicity) by : U[i+1] <= nextafter(U[i],DBL_MAX), nextafter being the IEEE-754 standard function returning the next representable neighbor of a double-precision floating point (see Bibliography).
  - multiple end knots are required; for non-periodic curves, the multiplicity of the end knots is degree+1.
  - Interior knots may have multiplicity up to degree+1. Thus, the interior of NURBS curves may be C0 or G1 for instance.
  - **knot_type** must be set in the **EPRCKnotType** range value

- **Nu (**number of knots in the knot vector) = **highest_index_of_knots** + 1; it must satisfy Nu = d + Np + 1.
- **Rational** is TRUE if the curve is rational and has an optional array of weights
- **W** is an optional weight at each control point; W(i) must be within [0.001, 1000]; all the coordinates x,y,z are weighted.
- **curve_form** must be set in the **EPRCBsplineCurveForm** range value.

The evaluation formula at a parameter value on a Nurbs curve is

The curve C(u) at a parameter value u is given by:

$$C(u) = \frac{\sum_{i=0}^{k} W_i P_i N_i(u)}{\sum_{i=0}^{k} W_i N_i(u)}$$

Where

$k + 1$ = number of control points,

P_i = control points,

W_i = weights,

d = degree.

N_i are the normalized B-spline basis functions of degree d defined on the knot set:

$U_{i-d},...,U_{i+1}$ $U_{j+1} >= U_j$ (i.e. non-decreasing).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_NURBS
Required	ContentCurve	Common curve data
Required	Boolean	**Rational** is TRUE if this is a rational NURBS curve; else FALSE
Required	UnsignedInteger	**d** is the degree of curve
Required	UnsignedInteger	**highest_index_of_control_points**
Required	UnsignedInteger	**highest_index_of_knots**
Required	ArrayOf	**P** is an array of control points

	[ControlPointsNurbsCrv]	defining curve.
Required	ArrayOf [Double]	**U** is an array of knots
Required	UnsignedInteger	**Knot_type** (EPRCKnotType)
Required	UnsignedInteger	**Curve_form** (EPRCBsplineCurveForm)

7.10.5.2 ControlPointsNurbsCrv

Required or Option	Data Type	Data Description
Required	Double	X coordinate of control point
Required	Double	Y coordinate of control point
OPTION: is_3d TRUE	Double	Z coordinate of control point; the Boolean flag is_3d comes from the ContentCurve data
OPTION: is_rational TRUE	Double	W coordinate of control point

7.10.5.3 EPRCKnotType

This enumeration is used to characterize a knot vector.

NOTE: this value is currently unused and should be set to KEPRCKnotTypeUnspecified.

Value	Type Name	Type Description
0	KEPRCKnotTypeUniformKnots	Uniform knot vector
1	KEPRCKnotTypeUnspecified	Unspecified knot type
2	KEPRCKnotTypeQuasiUniformKnots	Quasi-uniform knot vector
3	KEPRCKnotTypePiecewiseBezierKnots	Extrema with multiplicities of degree + 1

7.10.5.4 EPRCBsplineCurveForm

This enumerated type defines the possible NURBS curve forms.

NOTE: this value is currently not used and should be set to KEPRCBsplineCurveFormUnspecified.

Value	Type Name	Type Description
0	KEPRCBSplineCurveFormUnspecified	Unspecified curve form
1	KEPRCBSplineCurveFormPolylinel	Polygon
2	KEPRCBSplineCurveFormCircularArc	Circular arc
3	KEPRCBSplineCurveFormEllipticArc	Elliptical arc
4	KEPRCBSplineCurveFormParabolicArc	Parabolic arc
5	KEPRCBSplineCurveFormHyperbolicArc	Hyperbolic arc

7.10.5.5 EPRCExtendType

This enumerated type defines the possible methods for curve and surface extentions. Extensions may be either C or G continuous.

The first bit is reserved for future use and must be set to 0 in any variable representing this type

Value	Type Name	Type Description
0	KEPRCExtendTypeNone	Discontinuous position
2	KEPRCExtendTypeExt1	Same as KEPRCExtendTypeCInfinity
4	KEPRCExtendTypeExt2	Same as KEPRCExtendTypeG1R for surface and KEPRCExtendTypeG1 for curve
6	KEPRCExtendTypeG1	Continuous in direction but not magnitude of first derivative
8	KEPRCExtendTypeG1R	Surface extended with a ruled surface that connects with G1 continuity
10	KEPRCExtendTypeG1_G2	Extended by reflection, yielding a G2 continuous extension
12	KEPRCExtendTypeCInfinity	Unlimited continuity

7.10.6 PRC_TYPE_CRV_Circle

A cannonical circle is defined by its radius and lies on the XY plane centered at the origin. It is parameterized in radians on the interval [0.0, 2*Pi] where 0.0 lies on the x-axis and the mapping is defined by the right-hand rule relative to the z-axis normal to the plane of definition.

A Transformation positions the curve in model space. This transformation is capable of translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11)

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization enables a reparameterization and trim of the circle.

The evaluation formula at a parameter value on a circle is

Calculate the implicit_parameter from the given parameter using this circle's Parameterization data.

X = Radius * cos(implicit_parameter);

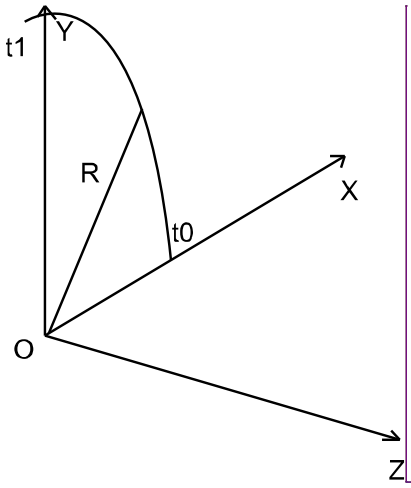
Y = Radius * sin(implicit_parameter);

Z = 0.0

The following examples illustrate possible uses of the Parameterization class:

- To specify the interval in radians, set Coeff_a to 1.0, Coeff_b to 0.0, and interval to [0.0,2.0* Pi]. These parameter values specify an identity conversion.
- To specify the interval in degrees, set Coeff_a to PI/180, Coeff_b to 0.0, and interval to [0.0, 360.0]. Coeff_a is the ratio of radians to degrees.
- To reparameterize the circle so the parameter values are in the interval [0.0, 1.0], set Coeff_a to 2.0 * Pi, Coeff_b to 0.0 and interval to [0.0, 1.0]

Example of a circular arc



Comment [dgh1]: This is a replacement drawing in vector format. It is the prototype for all future drawings. It can be scaled to any size.

Comment [vvz2]: Yes vectorial drawings are really better. As it is a prototype, we should take care of proportions and perspective effects. In the circle case, we can see that the radius R is more or less as long as the vector Y wheras along the X axis the radius seams to be more than 2 times shorter. (we assume that X,Y,Z are unit vectors.)

Comment [FC3]: Did you take this remark into account in subsequent drawings ?

Comment [dgh4]: Yes. This has been forwarded.

In the above example, the circular arc is in the XY plane (and therefore has an identity transformation), has radius **R**, and is restricted to the **[t0 , t1]** interval

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Circle
Required	ContentCurve	Common curve data
Required	Transformation	Transformation positioning circle in model coordinate system
Required	Parameterization	Redefine the parameterization
Required	Double	Radius

7.10.7 PRC_TYPE_CRV_Composite

This represents a composite curve consisting of one or more subcurves.

The following restrictions apply:

- Each subcurve must be of the same dimensionality as the composite (i.e. all 2D or all 3D).
- The subcurves must define a piecewise continous curve, that is, they must define a G0 continuous curve.

A Transformation positions the composite curve in model space. This transformation is capable of translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization enables a reparameterization and composite curve.

The implicit parameterization of the composite is the interval [0.0, NumberOfSubCurves]. The subinterval from [i, i+1] corresponds to the ith subcurve either in the same or opposite direction according to the sense of the composite curve.

To evaluate a composite curve at a parameter value:

Calculate the implicit_parameter from the given parameter using this composite curve's Parameterization data. The implicit_parameter must lie in the interval[0.0, NumberOfSubCurves].

Find the sub-interval that the implicit_parameter lies in. Say it lies in the ith subinterval [i, i+1], that is, i <= implicit_parameter <= i + 1.

Get the interval defining the ith SubCurve. Say it is the interval [a, b].

Calculate delta = implicit_parameter – i;

If the sense of the ith SubCurve is the same as the sense of the composite

    Parameter_OnSubCurve = a + delta * (b-a)

Else

    Parameter_OnSubCurve = b – delta * (b-a)

Position = SubCurve.evaluate(Parameter_OnSubCurve)

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Composite
Required	ContentCurve	Common curve data
Required	Transformation	Transformation positioning composite curve in model space
Required	Parameterization	Define parameterization
Required	UnsignedInteger	Number of subcurves
Required	ArrayOf [CompositeSubCurve]	Array of subcurves comprising the composite
Required	Boolean	TRUE if the composite curve is closed; else FALSE

7.10.7.1 CompositeSubCurve

A subcurve of a composite curve consists of a pointer to the definition of the subcurve and a flag indicating if the subcurve is in the same direction as the composite curve (TRUE) or in the opposite direcection (FALSE).

Required or Option	Data Type	Data Description
Required	PrtCurve	A subcurve in the composite
Required	Boolean	TRUE if the subcurve is in the same direction as the composite curve; FALSE if it is in the opposite direction

7.10.8 PRC_TYPE_CRV_OnSurf

This represents a 3D curve defined as a UV curve lying in the domain of a surface.

The specified domain is currently ignored and the underlying surface domain is used to define the domain of the surface that the UV curve must lie within.

A Transformation positions the curve in model space. This transformation is capable of translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization will enable this curve to be reparameterized and trimmed.

The tolerance is used internally but does not take part of the definition of the curve on surface. It indicates an appropriate tolerance that can be used to obtain a “representative” 3D NURBS approximation of the curve to aid in various operations. If not known it must be set to 0.0. The unit of this tolerance is the same as that used to store CrvOnSurf Data. See Section 5.7.

To evaluate this curve at a parameter value:

> Calculate the implicit_parameter from the given parameter using this CurveOnSurf's Parameterization data.
>
> UV_position = UV_curve.evaluate(implicit_parameter)
>
> XYZ = Surface.evaluate(UV_position)

Required or Option	Data Type	Data Description

Required	UnsignedInteger	PRC_TYPE_CRV_onSurf
Required	ContentCurve	Common curve data
Required	Transfomation	Postition curve in model space
Required	Parmeterization	Define parameterization and trimming information
Required	Double	Tolerance; default is 0.0
Required	PtrCurve	UV curve in parameter domain of surface; this must be a 2D curve in the UV space of the surface.
Required	PtrSurface	Surface the curve lies on
Required	Domain	UV domain on the surface; this is currently ignored and the surface domain is used

7.10.9 PRC_TYPE_CRV_Ellipse

A cannonical ellipse is centered at the origin with radius Rx along the x-axis and radius Ry along the y-axis and lies in the XY-plane. It is parameterized in radians on the interval [0.0, 2.0*Pi] with 0.0 on the positive x-axis and the mapping is defined by a right-hand rule about the z-axis normal to the plane of definition.

A Transformation positions the curve in model space. This transformation is capable of translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	**Type Name**	**Data Description**
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A parameterization will enable the ellipse to be reparameterized and trimmed.

The evaluation formula at a parameter value on an ellipse is

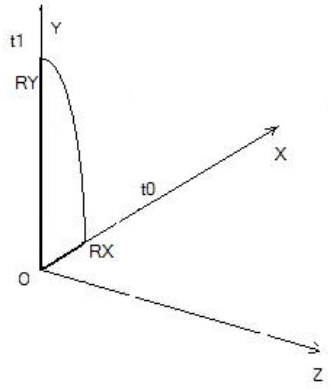
Calculate the implicit_parameter from the given parameter using this ellipse‘s Parameterization data.

X = Rx * cos(implicit_parameter);

Y = Ry * sin(implicit_parameter);

Z = 0.0

Example of an elliptic arc



In this example, the ellipse is in the XY plane (and therefore has an identity transformation), with radii **Rx** and **Ry** and is restricted to the **[t0 , t1]** interval. Assuming Coeff_a is 1.0 and Coeffb 0.0 (which indicates a parameterization in radians), then t0=0 and t1=Pi/2, t0 corresponds to the Cartesian coordinates (Rx,0,0) and t1 to (0,Ry,0).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Ellipse
Required	ContentCurve	Common curve data
Required	Transformation	Position ellipse into model space
Required	Parameterization	Define parameterization and trimming information
Required	Double	Radius along x axis
Required	Double	Radius along y axis

7.10.10 PRC_TYPE_CRV_Equation

This defines a curve by 1D mathematical functions in X, Y, and optionally, Z (see 7.12.3).

A Transformation positions the curve in model space. This transformation is capable of translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A parameterization will enable the curve to be reparameterized and trimmed.

The evaluation formula at a parameter value on this curve is

Calculate the implicit_parameter from the given parameter using this curve equation's Parameterization data.

X = X_Function.evaluate(implicit_parameter)

Y = Y_Function.evaluate(implicit_parameter)

If (is_3D)

    Z = Z_Function.evaluate(implicit_parameter)

Else

    Z = 0.0

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Equation
Required	ContentCurve	Common curve data
Required	Transformation	Position curve in model space

Required	Parameterization	Reparameterize and trim
Required	Interval	This interval should be set to the same interval as in the Parameterization data
Required	PRC_TYPE_MATH_FCT_1D	X function
Required	PRC_TYPE_MATH_FCT_1D	Y function
OPTION: is_3D	PRC_TYPE_MATH_FCT_1D	Z funtion if this is a 3D curve; the Boolean flag comes from the ContentCurve field

7.10.11 PRC_TYPE_CRV_Helix01

7.10.11.1 General

This curve type defines a helix defined on the inteval `[- infinite_param , infinite_param]`. A Helix is always a 3D curve.

A Transformation positions the helix in model space. This transformation is capable of translation, rotation, and scaling Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization will enable the helix to be reparameterized and trimmed.

A Type variable indicates which of two kinds of helix is defined by this curve:

- Constant pitch (type 0)
- Variable pitch (type 1)

Each has unique data and a unique evaluation formula.

The Start variable represents a 3D position which is used to define the starting position of the helix.

Required or Option	Data Type	Data Description

Required	UnsignedInteger	PRC_TYPE_CRV_Helix01
Required	ContentCurve	Common curve data
Required	Transformation	Position helix in model coordinate system
Required	Parameterization	Reparameterize and trim helix
Required	Character	Type of helix; must be 0 or 1
Required	Boolean	Trigonometric orientation (TRUE if helix turns in a clockwise direction and FALSE if it turns in a counter-clockwise direction
Required	Vector3d	start
OPTION: type = 0	Type0HelixData	Data for type 0 helix
OPTION: type = 1	Type1HelixData	Data for type 1 helix

7.10.11.2 Type0HelixData

A type 0 helix represents a constant radius helix.

The origin and direction define the axis of the helix The axis of the helix is denoted as the z-axis.The projection of the Start position onto the helix axis determines an origin_on_axis. The x-axis is then defined as the vector from the origin_on_axis to the start point. This defines a coordinate system orienting and defining the constant radius helix.

The pitch of the helix is the width of one complete helix turn measured along the helix axis.

The radius evolution is used to define a linear evolution of radius, such as a conic helix.

The following is the evaluation formula for a helix of type 0 at a parameter value:

```
origin = point_3d ( origin[0], origin[1], origin[2] );
z_axis = vector_3d ( direction[0], direction[1], direction[2]);
origin_on_axis = project_point( origin, z_axis, start );
x_axis = vector_3d( start - origin_on_axis);
radius = x_axis.length + param * radius_evolution;
if (trignometric_orientation) {
    tmp_point.x = radius * cos( param );
    tmp_point.y = radius * sin( param );
    tmp_point.z = pitch * param;
} else {
    tmp_point.x = radius * cos( - param );
```

Required or Option	Data Type	Data Description
Required	Double	Origin[0]
Required	Double	Direction[0]
Required	Double	Origin[1]
Required	Double	Direction[1]
Required	Double	Origin[2]
Required	Double	Direction[2]
Required	Double	Pitch
Required	Doubel	Radius_evolution

7.10.11.3 Type1HelixData

For a variable pitch helix (type 1) the following restrictions apply:

- The `radius_law` must be of type PRC_TYPE_MATH_FCT_1D_Polynom.
- The `theta_law` must be of type PRC_TYPE_MATH_FCT_1D_Polynom.
- The following values are reserved for future use but must be initialized to the specified values:
  - `reserved_double_0` must be set to `1`.
  - `reserved_double_1` must be set to `1`.
  - `reserved_double_2` must be set to `1`.
  - `reserved_double_3` must be set to `0`.

A coordinate system whose

- origin is at the start point
- z-axis is the unit_z vector
- x-axis is the unit_u vector

orients the helix.

The radius and theta laws are used to change the radius according to the angle around the helix.

The z law is used to change the pitch of the helix along its z-axis.

The following is the evaluation formula for a helix of type 1 at a parameter value:

```
r1 = radius_law.coeffient[0];
r2 = radius_law.coeffient[1];
t1 = theta_law.coeffient[0];
t2 = theta_law.coeffient[1];
param = (param / ( r1 + r2 )) * 2;
radius = r1 + ( param - t1 ) * ( r2 - r1 ) / ( t2 - t1 );

if (trigonometric_orientation) {
        tmp_point.x = radius * cos( param );
        tmp_point.y = radius * sin( param );
        tmp_point.z = z_law.evaluate( param );
} else {
        tmp_point.x = radius * cos( - param );
        tmp_point.y = radius * sin( - param );
        tmp_point.z = z_law.evaluate( param );
}

x_axis = vector_3d ( unit_u[0], unit_u[1], unit_u[2]);
z_axis = vector_3d ( unit_z[0], unit_z[1], unit_z[2]);
eval_point = transform_point( start, x_axis, z_axis, tmp_point );
```

Required or Option	Data Type	Data Description
Required	Double	Unit_z[0]
Required	Double	Unit_u[0]
Required	Double	Unit_z[1]
Required	Double	Unit_u[1]
Required	Double	Unit_z[2]

Required	Double	Unit_u[2]
Required	Double	Reserved_double_0; must be set to 1
Required	Double	Reserved_double_1; must be set to 1
Required	Double	Reserved_double_2; must be set to 1
Required	Double	Reserved_double_3; must be set to 0
Required	PRC_TYPE_MATH_FCT_1D	Radius law
Required	PRC_TYPE_MATH_FCT_1D	z law
Required	PRC_TYPE_MATH_FCT_1D	Theta law

7.10.12 PRC_TYPE_CRV_Hyperbola

A cannonical hyperbola is centered at the origin with semi_axis length along the x-axis and semi_image_axis length along the y-axis and lies in the XY-plane. It is parameterized on the interval [-infinite_param, infinite_param].

A Transformation positions the hyperboa in model space. This transformation is capable of translation, rotation, and scaling Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization will enable the hyperbola to be reparameterized and trimmed.

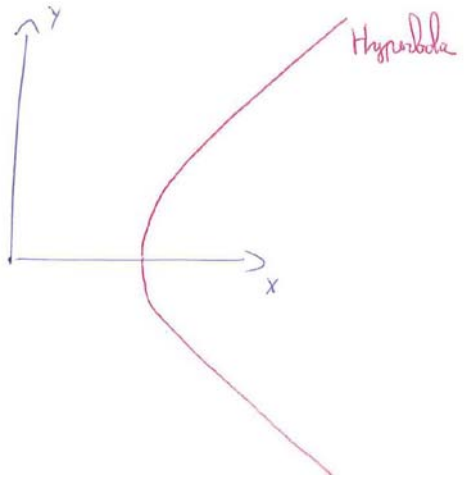
The type of hyperbola defines how the parameterization of the hyperbola is to be interpreted:

0	The parameterization must be changed so that param represents the value of the coordinate on the y-axis
1	The nominal parameterization formula applies based on cosh and sinh respectively for x and y;

The evaluation formula for a hyperbola at a parameter value is:

```
Calculate the implicit_parameter from the given parameter using this hyperbola's
Parameterization data.

If this is a type 0 hyperbola {

      Eval_point.x = semi_image_axis * sqrt(1.0 + pow(implicit_parameter/semi_axis, 2));

      Eval_point.y = implicit_parameter;

} else { // this is a type 1 hyperbola

      Eval_point.x = semi_image_axis * cosh(implicit_parameter);

      Eval_point.y = semi_axis * sinh(implicit_parameter);

}

Eval_point.z = 0.0;
```

Example of a hyperbola



Required or Option	Data Type	Data Description

Required	UnsignedInteger	PRC_TYPE_CRV_Hyperbola
Required	ContentCurve	Common curve data
Required	Transformation	Position hyperbola in model coordinate system
Required	Parameterization	Reparameterize and trim hyperbola
Required	Double	Semi_axis
Required	Double	Semi_axis_image
Required	Character	Type of hyperbola; must be 0 or 1

7.10.13 PRC_TYPE_CRV_Intersection

7.10.13.1 General

This represents a curve which is the exact intersection of two surfaces.

A Transformation positions the curve in model space. This transformation is capable of translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization will enable the curve to be reparameterized and trimmed.

A piecewise linear approximation to the true intersection curve is defined by a sequence of crossing points where each of the crossing points lie on the true intersection. At each crossing point the following in known

- the spacial position of the crossing point.
- the UV parameter value of the crossing point on each surface
- the unit tangent of the intersection curve at this point (defined as the cross product of the surface normals (surface 1 cross surface 2) or reversed (surface 2 cross surface 1) depending on the sense of the intersection curve with an optional sense applied to each surface normal);

- the parameter value (which should satisfy the parameterization requirements described in 7.10.13.2);
- a scale value (which should satisfy the parameterization requirements described in 7.10.13.2);

The NumberOfCrossingPoints must be sufficient so that evaluation of the curve at a parameter value results in a unique solution (see evaluation method below). The ChordalError and AngularError are used to indicate when more crossing points must be added to the definition to ensure a unique solution when evaluating an intersection curve at a parameter value (see below).

The intersection curve is limited by two points (start_limit_point and end_limit_point) each characterized with a type of limit (start_limit_type and end_limit_type) as described in 7.10.13.3.

In the case of KEPRCIntersectionLimitTypeTerminator, the limit position is present in the crossing points array (as the first or last point).

ChordalError is an estimate of the maximum distance between the curve and the set of segments given by the crossing points array.

AngularError is the maximum angle between the tangents of two sequential crossing points.

parameterization_definition_respected indicates whether the parameters of the crossing points array are compliant with the parameterization requirements (see 7.10.13.2).

This corresponds to a valid geometry in the sense of the crossing_point_flags, which should be set to true.

An intersection curve is always a 3D curve (i.e. is_3d must be true).

The evaluation formula for an intersection curve at a parameter value t is:

- If t matches a crossing point parameter, the crossing point position is the intersection curve point.
- If not, find two consecutive crossing points P1 and P2 such that the parameter t is included in the interval [t1, t2] (t1= parameter of P1 and t2 = parameter of P2]. The intersection curve point will be the intersection of surface 1, surface 2 and the plane defined by the origin O and the normal N where :

  $O = P1 + [(t-t1)/(t2-t1)] * (P2-P1)$

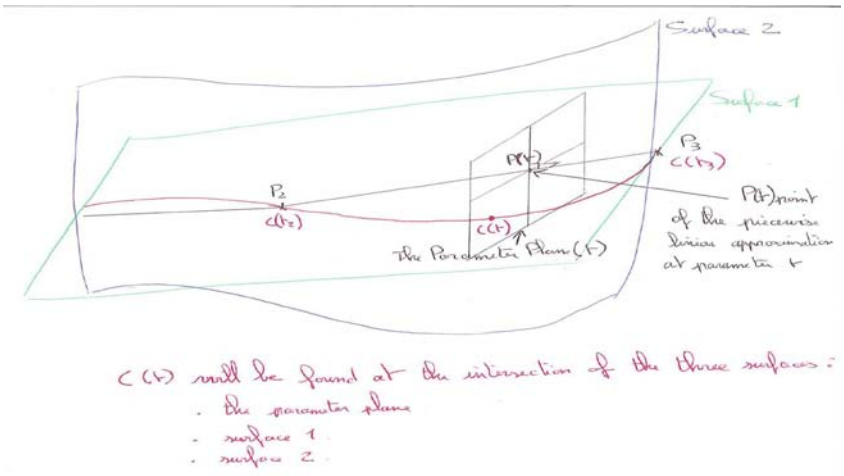
  $N = P2 – P1$

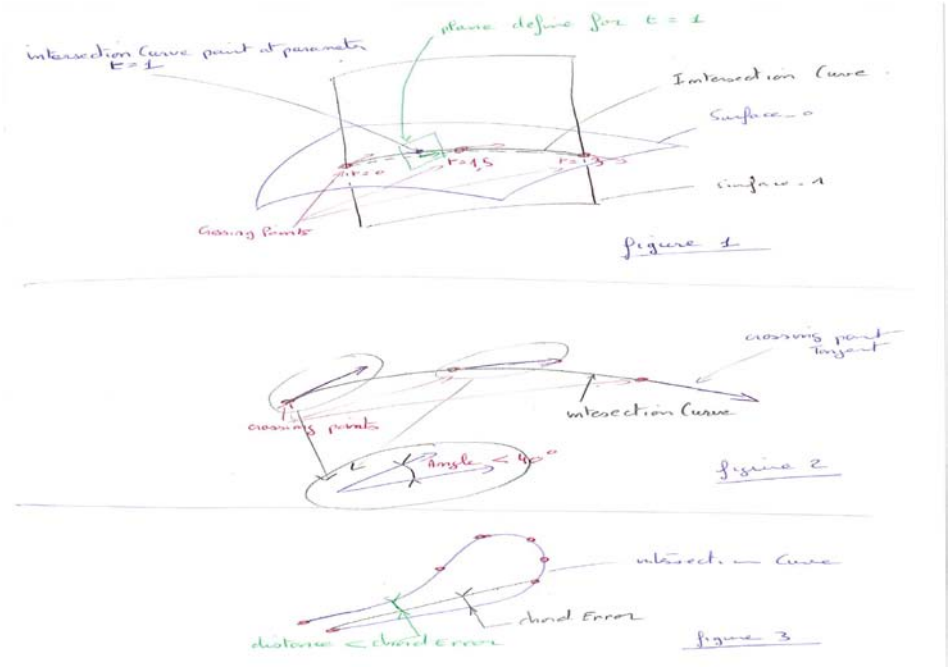
In fact, to evaluate a point at a given parameter, an iterative process is used to find the intersection of the three surfaces : surface 1, surface 2 and the plane defined above (the plane depends on parameter t).

Hints on how to ensure a good intersection curve definition:

To ensure that there is a unique solution, additional conditions have to be added on crossing point tangent definition. Theoretically, the tangents of two consecutive points must have angle smaller than 180 degree. In practice an angle smaller than 40 degree should be used to avoid numerical problems during the computation of the 3 surfaces intersection. This is the AngularError defined above and should be in radians.

Also to ensure that there is a unique solution, the plane defined above can't cross the intersection curve many times within the ChordalError associated with the intersection curve. The more crossing points there are, the smaller the ChordalError will be. Therefore in this case more crossing points have to be added in the intersection curve definition to avoid such situations.





Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Intersection
Required	ContentCurve	Common curve data
Required	Transformation	Positions curve in model coordinate system
Required	Parameterization	Reparameterize and trim curve
Required	PtrSurface	Surface 1
Required	PtrSurface	Surface 2
Required	Boolean	TRUE if sense is the same as surface 1; FALSE otherwise
Required	Boolean	TRUE if sense is the same as surface 2; FALSE otherwise

Required	Boolean	TRUE if the sense of the intersection sense is surface 1 cross surface 2; FALSE otherwise
Required	UnsignedInteger	Number of crossing points
Required	ArrayOf [CrossingPointsCrvIntersection]	Array of crossing points
Required	Vector3d	Start limit point
Required	UnsignedInteger	Start limit type; EPRCIntersectionLimitTypes
Required	Vector3d	End limit point
Required	UnsignedInteger	End limit type; EPRCIntersectionLimitTypes
Required	Double	Chordal error
Required	Double	Angular error
Required	Boolean	Parameterization definition respected

7.10.13.2 CrossingPointsCrvIntersection

Each crossing point is described by the following:

- The spatial position (crossing_point_position).
- The parametric position on surface_1 (crossing_point_uv_1).
- The parametric position on surface_2 (crossing_point_uv_2).
- The normalized tangent on the curve, crossing_point_tangent, is given by the cross product of two surface normals, taking into account the senses of surfaces surface_1_sense and surface_2_sense.
- Parameter value associated with the crossing point.
- Scale associated with the crossing point.
- The flag must be set to

  PRC_INTERSECTION_CROSS_POINT_SURFACE1 |
  PRC_INTERSECTION_CROSS_POINT_SURFACE2 |
  PRC_INTERSECTION_CROSS_POINT_INSIDE_CURVE_INTERVAL

  to indicate that

  - the uv position on surface 1 is filled
  - the uv position on surface 2 is filled
  - The crossing point is inside the curve interval.

At the i^{th} crossing point, the parameter and scale should adhere to the following **Parameterization Requirements**

$$Scale[i] = \frac{Tangent[i].(Position[i] - Position[i-1])}{Tangent[i].(Position[i+1] - Position[i])} Scale[i-1]$$

$$Parameter[i] = Parameter[i-1] + scale[i-1].\|Position[i] - Position[i-1]\|$$

parameter[0] is the parameter at first crossing point and the minimum parameter of the curve interval;

scale[0] should be set to 1.0 if not known.

This method is useful to get a more or less curvilinear parameterization without too much computation (function integration for example...).

The intersection curve has a boolean flag to indicate if these parameterization requirements are met

The . (period) in the scale formula is the dot product while in the parameter formula it is simple multiplication.

Between two crossing points, the parameter of an intersection curve point is given by its projection onto the previous crossing point tangent line.

.

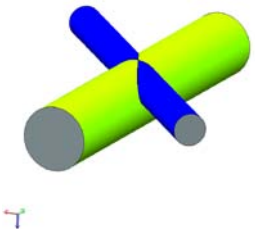
Required or Option	Data Type	Data Description
Required	Vector3d	Crossing point position
Required	Vector2d	Crossing point uv on surface 1
Required	Vector2d	Crossing point uv on surface 2
Required	Vector3d	Crossing point tangent
Required	Double	Crossing point parameter
Required	Double	Crossing point scale
Required	Character	Crossing point flags

7.10.13.3 EPRCIntersectionLimitType

This enumeration is used to classify an endpoint of a bounded portion (curve segment) of an intersection curve defined by the intesection of two surfaces (PRC_ TYPE_CRV_Intersection). This classification takes into consideration the nature and relationship of the surface normals of the point on each of the two surfaces as well as the shape of the intersection curve (finite/infinite, open/closed) .

The endpoint is classified **KEPRCIntersectionLimitTypeTerminator** if one (or both) of the surface normals is degenerate or if both of the surface normals are well defined but the normals are collinear.

Example: Consider the following intersection of two cylinders. The intersection point where the two surface normals become collinear will be a limit point of type Terminator which is used to define two separate intersection curves (i.e. each branch of the intersection results in an intersection curve).



The endpoint is classified **KEPRCIntersectionLimitTypeBoundary** if the intersection curve is used as a center curve of a blend02 surface that becomes degenerate but it is not relevant to the intersection curve.

Example Consider a blend02 surface with a radius that becomes equal to its center curve curvature radius (an intersection curve).

The endpoint is classified **KEPRCIntersectionLimitTypeLimit**, if it lies on an infinite intersection curve. In this case arbitrary endpoints are chosen to limit the curve segment to avoid having an infinite curve.

Example: Consider the intersection of two cylinders which result in two parallel lines. Two endpoints of limit type Limit are picked to define a finite line segment on each of the branches of the surface/surface intersection.

The endpoint is classified **KEPRCIntersectionLimitTypeHelp**, if it lies on a finite, closed intersection curve. In this case an arbitrary point is chosen to represent the end point of the curve segement.

Example: Consider the following intersection of a plane and cone which results in an elliptical intersection curve. Any point on the intersection curve may be used as the limit point with limit type Help.

Value	Type Name	Type Description
0	KEPRCIntersectionLimitTypeHelp	Arbitary limit on a closed intersection curve.
1	KEPRCIntersectionLimitTypeTerminator	Limit where one of the two intersection surface normals is degenerate or where they become colinear.
2	KEPRCIntersectionLimitTypeLimit	Artificial limit to avoid an infinite curve.
3	KEPRCIntersectionLimitTypeBoundary	Limit of the curve if a PRV_TYPE_SURF_Blend02 surface (that uses the intersection curve as its center curve) becomes degenerate.

7.10.14 PRC_TYPE_CRV_Line

The cannonical line is defined along the x-axis. The implicit parameterization is [-infinite_param, infinite_param] with 0.0 being the origin and positive values along the positive x-axis.

The Transformation can reposition the cannonical representation in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The Parameterization enables the line to be reparameterized and trimmed.

The evaluation formula for a line at a parameter value is:

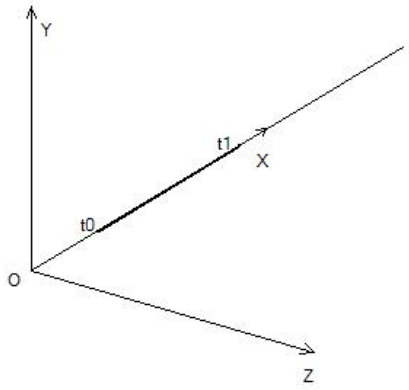
Calculate the implicit_parameter from the given parameter using this line‘s Parameterization data.

X = implicit_parameter;

Y = 0.0;

Z = 0.0;

Example of a line segment



In the above illustration, the line is restricted to **[t0 , t1]** interval on the X vector of its Cartesian transformation.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Line
Required	ContentCurve	Common curve data
Required	Transformation	Position line in model coordinate system
Required	Parameterization	Reparameterize and trim line

7.10.15 PRC_TYPE_CRV_Offset

This represents the offset of a 3D curve following the binormal defined by the tangent of the curve and the offset plane normal.

The curve must be 3D, that is, the Is_3d Boolean flag of the ContentCurve must be true.

The curve must not have a transformation, that is, the Has_transformation Boollean flag of the Transformation must be false.

Parameterization must have Coeff_a = 1.0, Coeff_b = 0.0, and the interval must lie within the base curve interval.

You can use an existing offset curve entity as the base curve used to create a new offset curve.

The evaluation formula for an offset curve at a parameter value is:

Calculate the implicit_parameter from the given parameter using this curves's Parameterization data.

base_point = base_curve.evaluate(implicit_param)

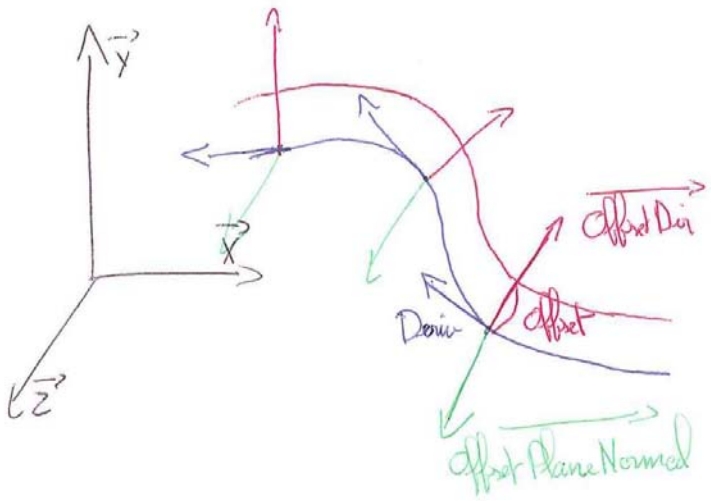
base_deriv = base_curve.evaluate_derivative(implicit_param)

offset_dir = normalize(base_deriv ^ offset_plane_normal)

Eval_point = base_point + offset_distance * offset_dir

Where X ^ Y is the cross product of the vectors X and Y

Example of a offset curve



In this example the base curve is offset by a 3D vector V which specifies the combination of the offset plane normal and the offset distance.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Offset
Required	ContentCurve	Common curve data
Required	Transformation	Postion offset curve in model space
Requried	Parmeterization	Reparameterize and trim curve
Required	PtrCurve	Base curve to offset
Required	Vector3d	Offset plane normal; this should be a unit vector
Required	Double	Offset distance

7.10.16 PRC_TYPE_CRV_Parabola

A cannonical parabola has its focus at (focal_length, 0, 0), its directrix at x = - focal_length and lies in the XY-plane. It is parameterized on the interval [-infinite_param, infinite_param].

A Transformation positions the parabola in model space. This transformation is capable of translation, rotation, and scaling Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization will enable the parabola to be reparameterized and trimmed.

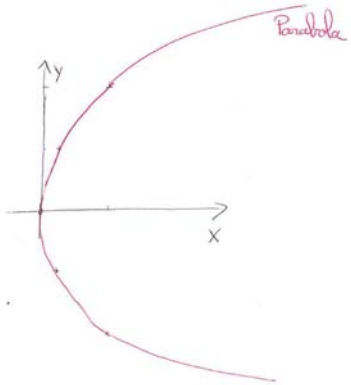
The type of parabola defines how the parameterization of the parabola is to be interpreted:

0	The parameter represents the value of the coordinate on the x-axis; the y-axis is the axis of the parabola
1	The nominal parameterization formula applies; the parameter is proportional to the value of the coordinate on the y axis; the x-axis is the axis of the parabola

The nominal evaluation formula for a parabola at param value is:

```
If type is 1 {

    Eval_point.x = focal_length * param * param;

    Eval_point.y = 2.0 * focal_length * param;

} else {

    param2 = param * param;
    p2 = param2/2.0 +
         sqrt(param2*param2/4.0 + 16.0*focal_length*focal_length*param2);
    param2 = (param < 0.0)sqrt(p2) : -sqrt(p2);
    if (param2 < 0)
    {
        Eval_point.x = - param2
        Eval_point.y = - 2.0 * focal_length * sqrt( - param2/ focal_length)
    }
    else
    {
        Eval_point.x = param2
        Eval_point.y = 2.0 * focal_length * sqrt( param2/ focal_length)
    }

}

Eval_point.z = 0.0
```

Example of a parabolic arc



Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Parabola
Required	ContentCurve	Common curve data
Required	Transformation	Position curve in model coordinate system
Required	Parameterization	Reparameterize and trim curve
Required	Double	Focal length
Required	Character	Parameterization type; must be 0 or 1

7.10.17 PRC_TYPE_CRV_PolyLine

7.10.17.1 General

This represents a PolyLine curve defined by a sequence of 2D or 3D points.

The implicit parameterization of a polyline is the interval [0.0, number of points]. The interval [i, i+1] corresponds to the segment between point[i] and point[i+1]. The curve between consective points is a straight Added transformation info line.

A Transformation positions the polyline in model space. This transformation is capable of translation, rotation, and scaling Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A Parameterization will enable the polyline to be reparameterized and trimmed.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_PolyLine
Required	ContentCurve	Common curve data
Required	Transformation	Postion curve in model space

Requried	Parmeterization	Reparameterize and trim curve
Required	UnsignedInteger	Number of points in polyline
Required	ArrayOf [PolyLinePoint]	Array of points defining polyline

7.10.17.2 PolyLinePoint

Required or Option	Data Type	Data Description
OPTION: is_3d TRUE	Vector3d	3D point; the is_3D Boolean flag comes from the ContentCurve
OPTION: is_3d FALSE	Vector2d	2D point

7.10.18 PRC_TYPE_CRV_Transform

A Transform curve represents a curve defined by applying a 3D mathematical function to a base curve.

Both the transform curve and the base_curve must be 3D curves.

The Transformation can reposition the curve in model space using a translation, rotation, and scaling. Only the following flags are acceptable (7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The Parameterization enables the curve to be reparameterized and trimmed.

The nominal evaluation formula for a transform curve at param value is:

```
Calculate the implicit_parameter from the given parameter using this transform curve's Parameterization data.

Tmp_point = base_curve.evaluate( implicit_parameter );

If (math_transformation != NULL)

    Eval_point =math_transformation.evaluate( tmp_point );

Else
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_CRV_Transform
Required	ContentCurve	Common curve data
Required	Transformation	Position curve in model space
Requried	Parmeterization	Reparameterize and trim curve
Required	PtrCurve	Base curve
Required	PRC_TYPE_MATH_FCT_3D	3D mathematical transformation to apply to base curve

7.11 Surface

7.11.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_SURF	PRC_TYPE_ROOT + 75	
PRC_TYPE_SURF_Base	PRC_TYPE_SURF + 1	
PRC_TYPE_SURF_Blend01	PRC_TYPE_SURF + 2	
PRC_TYPE_SURF_Blend02	PRC_TYPE_SURF + 3	
PRC_TYPE_SURF_Blend02	PRC_TYPE_SURF + 4	
PRC_TYPE_SURF_NURBS	PRC_TYPE_SURF + 5	
PRC_TYPE_SURF_Cone	PRC_TYPE_SURF + 6	
PRC_TYPE_SURF_Cylinder	PRC_TYPE_SURF + 7	
PRC_TYPE_SURF_Cyldrical	PRC_TYPE_SURF + 8	
PRC_TYPE_SURF_Offset	PRC_TYPE_SURF + 9	
PRC_TYPE_SURF_Pipe	PRC_TYPE_SURF + 10	
PRC_TYPE_SURF_Plane	PRC_TYPE_SURF + 11	
PRC_TYPE_SURF_Ruled	PRC_TYPE_SURF + 12	
PRC_TYPE_SURF_Sphere	PRC_TYPE_SURF + 13	

PRC_TYPE_SURF_Revolution	PRC_TYPE_SURF +14	
PRC_TYPE_SURF_Extrusion	PRC_TYPE_SURF + 15	
PRC_TYPE_SURF_FromCurves	PRC_TYPE_SURF + 16	
PRC_TYPE_SURF_Torus	PRC_TYPE_SURF + 17	
PRC_TYPE_SURF_Transform	PRC_TYPE_SURF + 18	
PRC_TYPE_SURF_Blend04	PRC_TYPE_SURF + 19	

7.11.2 PRC_TYPE_SURF

Abstract type for surfaces. If this appears in the documentation defining a field in the PRC File, it means that any surface type may be used.

7.11.3 PRC_TYPE_SURF_Base

7.11.3.1 General

Abstract type for surfaces. The following data is stored for all surface types.

7.11.3.2 ContentSurface

ContentSurface provides additional information about a surface such as its name and attributes and how it extends past its boundary.

Required or Option	Data Type	Data Description
Required	BaseGeometry	Optional geometric information
Required	UnsignedInteger	Indicates how the surface is extended; see **EPRCExtendType**

7.11.4 PRC_TYPE_SURF_Blend01

A Blend01 surface is defined by three curves, a center curve, an origin curve and an optional tangent curve, all defined over the same parameter interval. If the tangent curve is not defined (NULL), the first derivative of the origin curve is used instead. The implicit parameterization of a Blend01 surface is [0, 2*Pi] x [center_curve.interval.min, center_curve.interval.max].

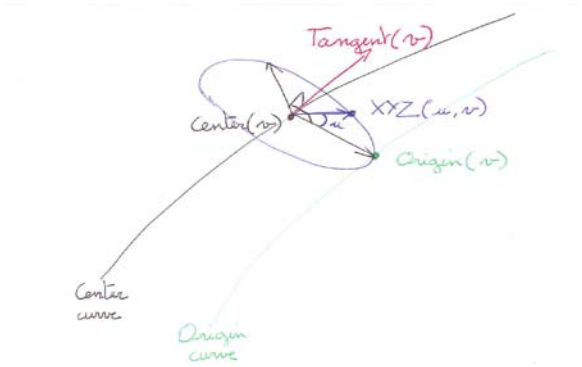
A Blend01 surface represents a variable radius pipe surface centered on the center curve with the origin curve defining both the radius and 0.0 location of the u parameter and the tangent curve defining the normal of the cross section plane of the Blend01 surface along the center curve.

A Transformation can reposition the surface in model space using a translation, rotation, and scalingOnly the following flags are acceptable (see 7.4.11**).**

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

A UVParameterization enables the surface to be reparameterized and trimmed.

The following diagram illustrates the relationship between the curves defining the Blend01 surface.



To evaluate a Blend01 surface at a parameter value:

Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.

R(implicit_param_v) = origin_curve(implicit_param_v) - center_curve(implicit_param_v)

XYZ = origin_curve(implicit_param_v) + cos(implicit_param_u) * R(implicit_param_v) + sin(implicit_param_u) * [tangent_curve(implicit_param_v) ^ R(implicit_param_v)]

(where ^ is the cross product)

If the tangent curve is NULL, use the unitized first derivative of origin curve instead of the tangent curve.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Blend01
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	PtrCurve	Center curve; must not be NULL
Required	PtrCurve	Origin curve; must not be NULL
Required	PtrCurve	Tangent curve; may be NULL in which case the first derivative of the center curve is used as the tangent curve.

7.11.5 PRC_TYPE_SURF_Blend02

A Blend02 surface is an exact rolling ball blend defined by rolling a ball of a constant radius along a center curve while maintaining tangential contact with two bounding surfaces (or curves in the case of a "cliff hanging" blend). A point on the center curve is projected onto the two bounding geometries and the Blend02 surface is defined by the circular arc between these two points (from point1 to point2).

The Blend02 surface is defined by

- A rolling ball center curve (center_curve) which defines the u parameter of the surface.
- A first bounding surface or a first bounding curve (bound_surface 1 or bound_curve 1). Either bound_surface 1 or bound_curve 1 may be given and the other must be NULL.
- A second bounding surface or a second bounding curve (bound_surface 2 or bound_curve 2).). Either bound_surface 2 or bound_curve 2 may be given and the other must be NULL.
- A radius and two senses. Radius 1 and radius 2 have the same absolute value but may have different signs. A positive sign indicates that the blend is on the side of the surface normal after taking into account the senses of the surfaces bound_surface_sense 1 and bound_surface_sense 2. A negative sign indicates that the blend is on the opposite side as the bounding surface normal.
- Two instances of cliff_supporting_surface. If one of the bounds is a curve, the surface is a cliff edge blend and its two supporting surfaces are the surfaces of the faces adjacent to the cliff edge (NULL otherwise).
- The type of parameterization (parameterization_type):
  - 0 indicates that the v parameter is zero at the first bound and one at the second bound.
  - 1 indicates that the v parameter is zero at the first bound and the angular value, in radians, at the second bound.
- The implicit parameterization is
  - [center_curve.interval.min, center_curve.interval.max] x [0, 1] if the parameter_type is 0
  - [center_curve.interval.min, center_curve.interval.max] x [0, 2*Pi] if the parameter_type is 1

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

A boundary surface may be replaced by a boundary curve, in that case, known as “cliff edge blending”, the center curve point must be projected onto this curve and not onto the missing boundary surface.

To evaluate a Blend02 surface at a parameter value:

Calculate the implicit_parameter from the given parameter using this surface‘s UVParameterization data.

Radius is the absolute value of radius 1 and radius 2

P1 = center(u) projected onto Bound Surface 1 (or Bound Curve 1)

P2 = center(u) projected onto Bound Surface 2 (or Bound Curve 2)

Where this is a perpendicular projection and the distance between Center(u) and P1 (and P2) must equal the blend radius

$X(u) = (P1 – center(u)) / \| P1 – center(u) \|$

$Y(u) = (P2 – center(u)) / \| P2 – center(u) \|$

A(u) = angle between X(u) and Y(u)

$Y_2(u) = ((X(u) \wedge Y(u)) \wedge X(u)) / \| ((X(u) \wedge Y(u)) \wedge X(u)) \|$

So that $X(u) . Y_2(u) = 0$ and $Y_2(u)$ is a unit vector

If (parameter_type == 0)

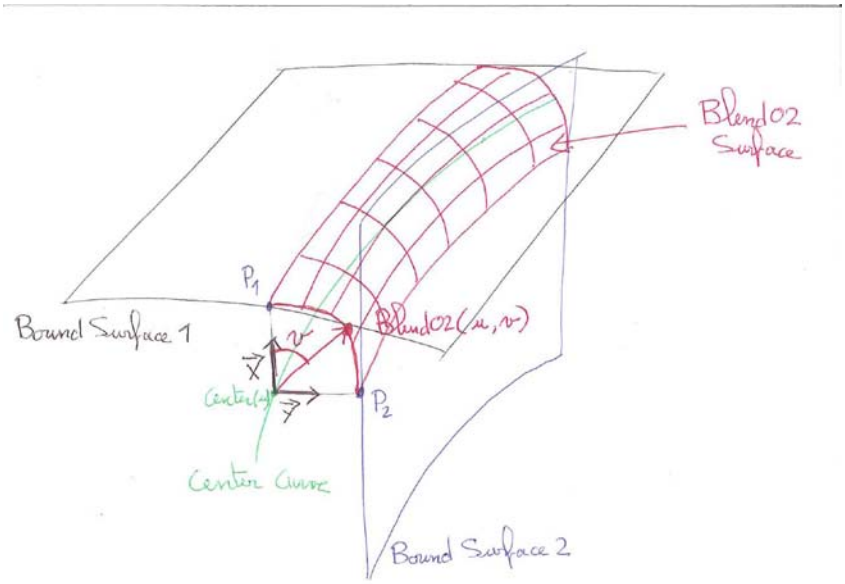
$XYZ = center(u) + Radius * (\cos(A(u) * v) * X(u) + \sin(A(u)*v) * Y_2(u))$

Else

$XYZ = Center(u) + Radius * (\cos(v).X(u) + \sin(v).Y_2(u))$

where

- u and v mean implicit_param_u and implicit_param_v in all descriptions



P_1 = Center (u) projected on Bound Surface 1

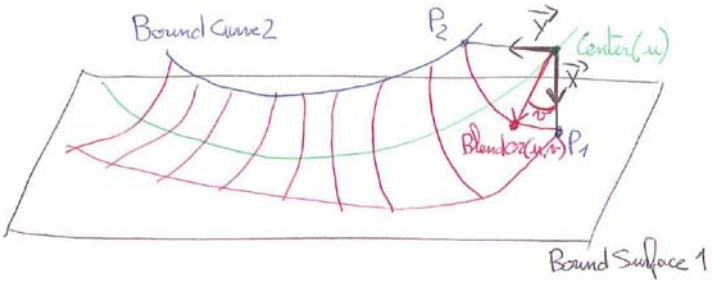
P_2 = Center (u) projected on Bound Surface 2

$$\vec{X}(u) = \frac{P_1 - \text{Center}(u)}{\| P_1 - \text{Center}(u) \|} \qquad \vec{Y}(u) = \frac{P_2 - \text{Center}(u)}{\| P_2 - \text{Center}(u) \|}$$

$A(u)$ = Angle between $\vec{X}(u)$ and $\vec{Y}(u)$

and $\vec{Y_2}(u) = \dfrac{[X(u) \wedge Y(u)] \wedge X(u)}{\| [X(u) \wedge Y(u)] \wedge X(u) \|}$ (so that $X(u) \cdot Y_2(u) = 0$ and $Y_2(u)$ is an unit vector)

so : $XYZ = \text{Center}(u) + \text{Radius} \cdot \left(\cos(A(u) \cdot v) \cdot \vec{X}(u) + \sin(A(u) \cdot v) \cdot \vec{Y_2}(u) \right)$



Idem for diff Blend
except : P_2 = Center(u) projected on BoundCurve2

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Blend02
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	PtrSurface	Bound Surface 0
Required	PtrCurve	Bound curve 0
Required	PtrSurface	Bound surface 1
Required	PtrCurve	Bound curve 1
Required	PtrCurve	Center curve
Required	Boolean	Center curve sense
Required	Boolean	Bound surface 0 sense
Required	Boolean	Bound surface 1 sense
Required	Double	Radius 0
Required	Double	Radius 1
Required	PtrSurface	Cliff supporting surface 0
Required	PtrSurface	Cliff supporting surface 1
Required	Character	Parameterization type

7.11.6 PRC_TYPE_SURF_Blend03

A Blend03 surface is a fillet surface defined by four curves: a center curve, two rail curves, and an angle curve which defines the V parameterization of the surface. All curves are quintic splines defined over the same nodal vector (Number_of_elements, Parameters and Multiplicities).

A center_curve, rail_curve_1, and rail_curve2 are defined using point, tangent, and second derivative data from Number_of_element entries in the Points, Tangents, and SecondDerivative arrays:

- center_curve uses data at indicies i*3 + 0;

- rail_curve_1 uses data at indicies i*3 + 1;
- and rail_curve_2 uses data at indicies i*3 + 2.

where 0 <= i <= Number_of_elements.

A rail2_anglesV_curve is defined using the data in the arrays Rail2AnglesV, Rail2DerivativesV, and Rail2SecondDerivativesV. This curve defines the V parameterization of the Blend03 surface by controlling the V parameterization along the isoparametric U curves.

Each isoparametric U curve is a circle defined on a plane centered on a point evaluated on the center_curve, where the point on rail_curve_1 evaluated at the same parameter gives the x-axis, and the point on rail_curve_2 gives the y-axis. For each circle, rail_curve_1 corresponds to Parameter[0], and the parameter corresponding to the point on rail_curve_2 is found using the angle curve function rail2_anglesV_curve. The same parameter is divided by Rail2ParameterV.

The implicit parameterization is [Parameter[0], Parameter[Number_of_elements - 1]] x [trim_v_min, trim_v_max].

If trim_v_max is less than trim_v_min, the V parameterization is set to [0, 1].

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section **Transformation**)

Value	**Type Name**	**Data Description**
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

To evaluate a Blend03 surface at a parameter value:

Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.

X(u) = (rail_curve_1(u) – center_curve(u)) / || rail_curve_1(u) – center_curve(u) || *(so that X(u) is a unit vector)*

Y(u) = rail_curve_2(u) – center_curve(u)

$Y_2(u)$ = [X(u)^Y(u)]^X(u) / || [X(u)^Y(u)]^X(u) || *(so that X(u).$Y_2(u)$=0 and $Y_2(u)$ is a unit vector)*

A(u) = rail2_anglesV_curve(u) / Rail2ParameterV

Radius(u) = || rail_curve_2(u) - center_curve(u) ||

XYZ = center _curve (u) + Radius(u) * (cos(A(u)*v).X(u) + sin(A(u)*v).$Y_2(u)$)

Where

- u and v mean implicit_param_u and implicit_param_v.
- || X || is the length of vector X
- X^Y is the cross product of X and Y vectors.

The following values are reserved for future use:

- reserved_int[0] should be set to 5.
- reserved_int[1] should be set to 0.
- reserved_int[2] should be set to 0.
- reserved_int[3] should be set to number_of_element.
- reserved_int[4] should be set to 0.
- reserved_int[5] should be set to 1.
- reserved_chars_0 should be set to 1.
- reserved_chars_1 should be set to 0.
- reserved_chars_2 should be set to 0.
- reserved_supplemental_doubles[i] and number_of_supplemental_doubles should be set to 0.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Blend03
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model

		space
Required	UVParameterization	Define parameterization and trimming information
Required	Integer	Number_of_elements
Required	ArrayOf [Double]	Parameters
Required	ArrayOf [Integer]	Multiplicities
Required	ArrayOf [Vector3d]	Array of Points
Required	ArrayOf [Double]	Array of Rail2AnglesV
Required	ArrayOf [Vector3d]	Array of Tangents
Required	ArrayOf[Double]	Array of Rail2DerivativesV
Required	ArrayOf [Vector3d]	Array of SecondDerivatives
Required	ArrayOf [Double]	Array of Rail2SecondDerivativesV
Required	Double	Rail2Parameter V
Required	Double	Trim v min
Required	Double	Trim v max
Required	Integer[6]	Reserved_int
Rquired	Character	Reserved_char_0
Rquired	Character	Reserved_char_1
Rquired	Character	Reserved_char_2
Required	Integer	Number of reserved supplimental doubles
Required	ArrayOf [Double]	Reserved supplimental doubles

7.11.7 PRC_TYPE_SURF_NURBS

7.11.7.1 General

This class represents a non-uniform rational bspline surface.

A NURBS surface is defined by the following data:

- **Du** is the degree of the surface in u and is restricted to the range 1 <= degree <= 25
- **Dv** is the degree of the surface in v and is restricted to the range 1 <= degree <= 25
- **P** is a two dimensional array of control points.
- **Npu** (number of control points in u) = **highest_index_of_control_points_in_u** + 1
- **Npv** (number of control points in v) = **highest_index_of_control_points_in_v** + 1
- **Ku** is the knot vector in u
  - the knots must be a non-decreasing sequence , that is, Ku[i] <= Ku[i+1]
  - multiple end knots are required; for non-periodic surfaces, the multiplicity of the end knots is Du+1.
  - Interior knots may have multiplicity up to Du+1.
- **Nku (**number of knots in the u knot vector) = **highest_index_of_knots_in_u** + 1; it must satisfy Nku = Du + Npu + 1.
- **Kv** is the knot vector in v
  - the knots must be a non-decreasing sequence , that is, Kv[i] <= Kv[i+1]
  - multiple end knots are required; for non-periodic surfaces, the multiplicity of the end knots is Dv+1.
  - Interior knots may have multiplicity up to Du+1.
- **Nkv (**number of knots in the v knot vector) = **highest_index_of_knots_in_v** + 1; it must satisfy Nkv = Dv + Npv + 1.
- **knot_type** must be set in the **EPRCKnotType** range value
- **Rational** is TRUE if the surface is rational and has an optional array of weights
- **W** is an optional weight at each control point; W(i,j) must be within [0.001, 1000]; all the coordinates x,y,z are weighted.
- **surface_form** must be set in the **EPRCSplineSurfaceForm** range value.

The evaluation formula at a parameter value on a Nurbs surface is

The surface S(u,v)) at a parameter value u and v is given by:

$$S(u,v) = \frac{\sum_{i=0}^{Npu}\sum_{j=0}^{Npv} W_{ij}P_{ij}N_i(u)N_j(v)}{\sum_{i=0}^{Npu}\sum_{j=0}^{Npv} W_{ij}N_i(u)N_j(v)}$$

Where

Npu = number of control points in u

Npv = number of control points in v

P_{ij} = control points,

W_{ij} = weights,

Du = degree in u

Dv = degree in v

N_i are the normalized B-spline basis functions of degree d defined on the knot set:

$U_{i-d},...,U_{i+1}$ $U_{j+1} >= U_j$ (i.e. non-decreasing).

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_NURBS
Required	ContentSurface	Common surface data
Required	Boolean	**Rational** is TRUE if this is a rational NURBS surface; else FALSE
Required	UnsignedInteger	**Du** is the degree of surface in u
Required	UnsignedInteger	**Dv** is the degree of surface in v
Required	UnsignedInteger	**highest_index_of_control_points_in_u**
Required	UnsignedInteger	**highest_index_of_control_points_in_v**
Required	UnsignedInteger	**highest_index_of_knots_in_u**
Required	UnsignedInteger	**highest_index_of_knots_in_u**
Required	2DimArrayOf [ControlPointsNurbsSurf]	**P** is a two dimensional array of control points defining surface.
Required	ArrayOf [Double]	**Ku** is an array of knots in u
Required	ArrayOf [Double]	**Kv** is an array of knots in v
Required	UnsignedInteger	**Knot_type** must be set to a value in EPRCKnotType
Required	UnsignedInteger	**Surface_form** must be set to a value in EPRCSplineSurfaceForm

7.11.7.2 ControlPointsNurbsSurf

An array of control points for a Nurbs surface are stored in a two dimensional array

```
For (i=0; i<= highest_index_of_control_points_in u; i++)
    For (j=0; j<=highest_index_of_control_points_in_v;  j++)
        Store the x, y, z and optional w value
```

Required or Option	Data Type	Data Description
Required	Double	X coordinate of control point
Required	Double	Y coordinate of control point
Required	Double	Z coordinate of control point
OPTION: is_rational TRUE	Double	W coordinate of control point

7.11.7.3 EPRCSplineSurfaceForm

This enumerated type defines the possible NURBS surface forms.

NOTE: this value is currently not used and should be set to KEPRCBsplineSurfaceFormUnspecified.

Value	Type Name	Type Description
0	KEPRCBSplineSurfaceFormPlane	Planar surface
1	KEPRCBSplineSurfaceFormCylindrical	Cylindrical surface
2	KEPRCBSplineSurfaceFormConical	Conical surface
3	KEPRCBSplineSurfaceFormSpherical	Spherical surface
4	KEPRCBSplineSurfaceFormRevolution	Surface of revolution
5	KEPRCBSplineSurfaceFormRuled	Ruled surface
6	KEPRCBSplineSurfaceFormGeneralizedCone	Cone
7	KEPRCBSplineSurfaceFormQuadric	Quadric surface
8	KEPRCBSplineSurfaceFormLinearExtrusion	Surface of extrusion
9	KEPRCBSplineSurfaceFormUnspecified	Unspecified surface
10	KEPRCBSplineSurfaceFormPolynomial	Polynomial surface

7.11.8 PRC_TYPE_SURF_Cone

This represents a cannonical definition of a conical surface where the axis of the cone lies along the z-axis. The x-axis represents the 0.0 value of the u parameter interval [0.0, 2* Pi] with positive values counter-clockwise about the z-axis using the right hand rule. The z-axis represents the v parameter interval [-infinite_param, infinite_param]. The 0.0 value of the v parameter is indicated by the bottom radius. The semi-angle is the half angle of the cone in radians.

The implicit parameterization of the cone is [0.0, 2*Pi] x [-infinite_param, infinite_param].

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

To evaluate this surface at a parameter value:

Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.

radius= bottom_radius + implicit_param_v * tan(semi-angle)
tmp_point.x = cos(implicit_param_u) * radius
tmp_point.y = sin(implicit_param_u) * radius
tmp_point.z = implicit_param_v

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Cone
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	Double	Bottom radius
Required	Double	Semi angle in radians

7.11.9 PRC_TYPE_SURF_Cylinder

This represents a cannonical definition of a cylinder where the axis of the cylinder lies along the z-axis. The x-axis represents the 0.0 value of the u parameter (radians) interval [0.0, 2* Pi] with positive values counter-clockwise about the z-axis using the right hand rule. The z-axis represents the v parameter interval [-infinite_param, infinite_param]. The 0.0 value of the v parameter is at the origin.

The implicit parameterization of the cylinder is [0.0, 2*Pi] x [-infinite_param, infinite_param].

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

To evaluate this surface at a parameter value:

Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.

tmp_point.x = cos(implicit_param_u) * radius
tmp_point.y = sin(implicit_param_u) * radius
tmp_point.z = implicit_param_v

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Cylinder
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space

Required	UVParameterization	Define parameterization and trimming information
Required	Double	Radius

7.11.10 PRC_TYPE_SURF_Cylindrical

This represents a cylindrical surface expressed in cylindrical coordinate system where (R, Theta, h). A base surface defines the mapping from UV to (R, Theta, h) = (x, y, z). The axis of the cylinder lies along the z-axis.

The implicit parameterization of the cylindrical surface is the same as the UV domain of the base surface.

The Transformation can reposition the surface in model space using a translation, rotation, and scalingOnly the following flags are acceptable (see section 7.4.11).

Value	**Type Name**	**Data Description**
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

The tolerance is used internally but does not take part of the definition of the surface. It indicates an appropriate tolerance that can be used to obtain a “representative” 3D NURBS approximation of the surface to aid in various operations. If not known it must be set to 0.0.

To evaluate this surface at a parameter value

Calculate the implicit_parameter from the given parameter using this surface‘s UVParameterization data.

```
base_point = base_surface.evaluate( implicit_param_u, implicit_param_v)
tmp_point.x = base_point .x * cos(base_point.y)
tmp_point.y = base_point .x * sin(base_point.y)
tmp_point.z = base_point.z
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Cylindrical
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	PtrSurface	Base surface
Required	Double	Tolerance

7.11.11 PRC_TYPE_SURF_Offset

This represents a surface defined by offsetting a given surface along its normal by a specified distance. The implicit parameterization is the same as the UV domain of the base surface.

The Transformation can reposition the surface in model space using a translation, rotation, and scaling.Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

To evaluate this surface at a parameter value

Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.

base_point = base_surface.evalaute(implicit_param_u, implicit_param_v)
base_normal = base_surface.evaluate_normal(implicit_param_u, implicit_param_v)
tmp_point = base_point + (offset_distance * base_normal)

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Offset
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	PtrSurface	Base surface
Required	Double	Offset distance

7.11.12 PRC_TYPE_SURF_Pipe

This surface type is currently not supported and reserved for future use.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Pipe
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	PtrCurve	Center curve
Required	PtrCurve	Origin curve
Required	Double	Radius of pipe

7.11.13 PRC_TYPE_SURF_Plane

The represents the cannonical definition of a planar surface which is defined as the XY plane

The cannonical representation of a plane has

- the x-axis set to (1, 0, 0)
- the y axis set to (0, 1, 0)
- the z axis set to (0,0,1)
- the implicit parameterization is the uv domain [-infinite_param, infinite_param] x [-infinite_param, infinite_param]

The implicit parameter value for a plane is calculated using

```
Implicit_param.u = u_parameter_coeff_a * param.u + u_parameter_coeff_b;

Implicit_param.v = v_parameter_coeff_a * param.v + v_parameter_coeff_b
```

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

To evaluate this surface at a parameter value

```
Calculate the implicit_parameter from the given parameter using this surface's
UVParameterization data.

tmp_point.x = implicit_param_u
tmp_point.y = implicit_param_v
tmp_point.z = 0
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Plane
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	Domain	Define parameterization and trimming information
Required	Double	U parameter coeff_a
Require	Double	V parameter coeff_a
Required	Double	U parameter coeff_b
Required	Double	V parameter coeff_b

7.11.14 PRC_TYPE_SURF_Ruled

This represents a ruled surface defined by connecting points on each of two curves by a straight line. It is required that both curves are defined over the same interval since points at equal parameter value along each curve are connected by a straight line.

The implicit parameterization of the ruled surface is [0, 1] x [first_curve.interval.min, first_curve.interval.max].

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

To evaluate this surface at a parameter value

Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.

```
base_point1 = first_curve.evaluate( implicit_param_v)
base_point2 = second_curve.evaluate( implicit_param_v)

tmp_point.x = (1.0-implicit_param_u) * base_point1.x + implicit_param_u * base_point2.x
tmp_point.y = (1.0-implicit_param_u) * base_point1.y + implicit_param_u * base_point2.y
tmp_point.z = (1.0-implicit_param_u) * base_point1.z + implicit_param_u * base_point2.z
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Ruled
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	PtrCurve	First curve
Required	PtrCurve	Second curve

7.11.15 PRC_TYPE_SURF_Sphere

This represents the cannonical definition of a spherical surface centered at the origin. The u parameter corresponds to a circle in the XY plane with 0.0 being the x-axis and positive angles measured around the z-axis using the right hand rule. The v parameter corresponds to a semi-circle in the plane defined by the u parameter and passing through the z-axis with the XY plane being 0.0 and positive angles above the XY plane and negative angles below the XY plane. The implicit parameterization of a sphere is [0, 2* Pi] x [-Pi/2, Pi/2].

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

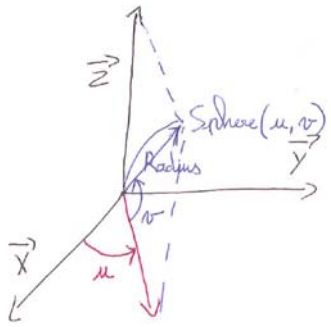
The UVParameterization enables the surface to be reparameterized and trimmed.

To evaluate this surface at a parameter value

Calculate the implicit_parameter from the given parameter using this surface‘s UVParameterization data.

tmp_point.x = radius * cos(implicit_param_v) * cos(implicit_param_u)
tmp_point.y = radius * cos(implicit_param_v) * sin(implicit_param_u)
tmp_point.z = radius * sin(implicit_param_v)

Example of a sphere



Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Sphere
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	Double	Radius

7.11.16 PRC_TYPE_SURF_Revolution

This represents a surface of revolution defined as revolving a base curve around an axis of revolution.

The implicit parameterization is [0, 2* Pi], [base_curve.interval.min, base_curve.interval.max]. The u value is in radians and the 0.0 value corresponds to a point on the base_curve. Positive angles are in the direction determined by the axis direction using the right hand rule.

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description

0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

The axis of revolution is defined by an origin and the cross product of x-axis and y-axis.

The tolerance is used internally but does not take part of the definition of the surface. It indicates an appropriate tolerance that can be used to determine if the base_curve lies in a plane passing by the axis of revolution. If not known it must be set to 0.0. See Section 5.7.

To evaluate this surface at a parameter value

Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.

base_point = Base_curve.evaluate(implicit_param_v)
point_on_axis = axis_of_revolution.project_on_DirectionZ(base_point)
tmp_axis_x = base_point – point_on_axis
tmp_axis_y = axis_of_revolution.DirectionZ ^ tmp_axis_x
tmp_point = point_on_axis + cos(implicit_param_u) * tmp_axis_x +sin(implicit_param_u) * tmp_axis_y

where ^ indicates the cross product

Required or Option	**Data Type**	**Data Description**
Required	UnsignedInteger	PRC_TYPE_SURF_Revolution
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	Double	Tolerance
Required	Vector3d	Origin

Required	Vector3d	X axis
Required	Vector3d	Y axis
Required	PtrCurve	Base curve

7.11.17 PRC_TYPE_SURF_Extrusion

This represents an extruded surface where a base curve is extruded along a sweep vector.

The implicit parameterization of the extruded surface is

[base_curve.interval.min, base_curve.interval.max] x [-infinite_param, infinite_param].

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

The evaluation at a paramter value param is

> Calculate the implicit_parameter from the given parameter using this surface's UVParameterization data.
>
> XYZ = base_curve.evaluate(implicit_param_u) + implicit_param_v * sweep_vector;

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Extrusion
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space

Required	UVParameterization	Define parameterization and trimming information
Required	Vector3d	Sweep vector must be a unit vector
Required	PtrCurve	base curve

7.11.18 PRC_TYPE_SURF_FromCurves

The implicit parameterization is [first_curve.interval.min, first_curve.interval.max] x [second_curve.interval.min, second_curve.interval.max].

The Transformation can reposition the surface in model space using a translation, rotation, and scaling. Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

The evaluation at a paramter value param is

```
Eval_point = first_curve.evaluate(param.u) + second_curve.evaluate(param.v) – origin;
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_FromCurves
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and

		trimming information
Required	Vector3d	Origin
Required	PtrCurve	First curve
Required	PtrCurve	Second curve

7.11.19 PRC_TYPE_SURF_Torus

This represents the cannonical definition of a torus centered at the origin with the major axis in the XY plane. The implicit parameterization is [0, 2* Pi] x [0, 2*Pi] where the u parameter is 0.0 corresponding to a circle on the XZ plane with radius = minor_radius and the v parameter corresponds to a circle on the XY plane with radius = major_radius + minor_radius.

The Transformation can reposition the surface in model space using a translation, rotation, and scaling.Only the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

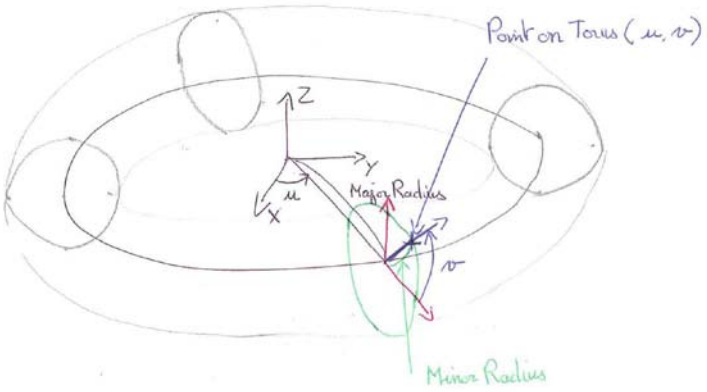
To evaluate this surface at a parameter value

Calculate the implicit_parameter from the given parameter using this surface‘s UVParameterization data.

radius = major_radius + minor_radius * cos(implicit_param_v);

tmp_point.x = radius * cos(implicit_param_u)
tmp_point.y = radius * sin(imiplicit_param_u)
tmp_point.z = minor_radius * sin(implicit_param_v)

Example of a torus



Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Torus
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	Double	Major radius
Required	Double	Minor radius

7.11.20 PRC_TYPE_SURF_Transform

A Transform surface is defined by applying a 3D mathematical transformation to a base surface.

The implicit parameterization is the same as the base surface.

The Transformation can reposition the surface in model space using a translation, rotation, and scalingOnly the following flags are acceptable (see section 7.4.11).

Value	Type Name	Data Description
0x00	PRC_TRANSFORMATION_Identity	Identity
0x01	PRC_TRANSFORMATION_Translate	Translation
0x02	PRC_TRANSFORMATION_Rotate	Rotation
0x08	PRC_TRANSFORMATION_Scale	Uniform scale

The UVParameterization enables the surface to be reparameterized and trimmed.

The mathematical transformation can be NULL.

The nominal evaluation formula for a transform curve at param value is:

```
Calculate the implicit_parameter from the given parameter using this transform surface's
Parameterization data.

Tmp_point = base_surface.evaluate( implicit_parameter );

If (math_transformation != NULL)

    Eval_point =math_transformation.evaluate( tmp_point );

Else

    Eval_point = tmp_point;
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_SURF_Transform
Required	ContentSurface	Common surface data
Required	Transformation	Position surface into model space
Required	UVParameterization	Define parameterization and trimming information
Required	PtrSurface	Base surface
Required	PRC_TYPE_MATH_FCT_3D	3D mathematical transformation

7.11.21 PRC_TYPE_SURF_Blend04

This type is currently not supported and is reserved for future use.

7.12 Mathematical Operator

7.12.1 Entity Types

Type Name	Type Value	Referenceable
PRC_TYPE_MATH	PRC_TYPE_ROOT + 900	

PRC_TYPE_MATH_FCT_1D	PRC_TYPE_MATH + 1	
PRC_TYPE_MATH_FCT_1D_Polynom	PRC_TYPE_MATH_FCT_1D + 1	
PRC_TYPE_MATH_FCT_1D_Trigonometric	PRC_TYPE_MATH_FCT_1D + 2	
PRC_TYPE_ MATH_FCT_1D_Fraction	PRC_TYPE_MATH_FCT_1D + 3	
PRC_TYPE_ MATH_FCT_1D_ArctanCos	PRC_TYPE_MATH_FCT_1D + 4	
PRC_TYPE_ MATH_FCT_1D_Combination	PRC_TYPE_MATH_FCT_1D + 5	
PRC_TYPE_MATH_FCT_3D	PRC_TYPE_MATH + 10	
PRC_TYPE_MATH_FCT_3D_Linear	PRC_TYPE_MATH_FCT_3D + 1	
PRC_TYPE_MATH_FCT_3D_NonLinear	PRC_TYPE_MATH_FCT_3D + 2	

7.12.2 PRC_TYPE_MATH

Abstract class for mathematical operators.

7.12.3 PRC_TYPE_MATH_FCT_1D

Base type for a equation of one variable. The following are legal types of equations;

- Polynomial equation PRC_TYPE_MATCH_FCT_1D_Polynom
- Cosine based equation PRC_TYPE_MATCH_FCT_1D_Trigonometric
- Fraction of two 1D equations PRC_TYPE_MATCH_FCT_1D_Fraction
- Specific equation PRC_TYPE_MATCH_FCT_1D_ArctanCos
- Combination of 1D equation PRC_TYPE_MATCH_FCT_1D_Combination

7.12.4 PRC_TYPE_MATH_FCT_1D_Polynom

This represents 1D polynomial equation.

The evaluation formula for a given parameter value is

```
output = 0.0
For (i=0; i<number_of_coefficients; i++) {
    output = output + coefficient[i] * pow(param, i);
}
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MATH_FCT_1D_Polynom
Required	UnsignedInteger	Number of coefficients in polynomial
Required	ArrayOf [Double]	Array of coefficients

7.12.5 PRC_TYPE_MATH_FCT_1D_Trigonometric

This represents a 1D trigonometric equation.

The evaluation formula for param value is

```
Output = dc_offset + amplitude * cos(param*freq – phase)
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MATH_1D_Trigonometric
Required	Double	Amplitude
Required	Double	Phase
Required	Double	Frequency
Required	Double	Dc_offset

7.12.6 PRC_TYPE_MATH_FCT_1D_Fraction

This represents a 1D equation that is a fraction of two 1D equations.

The evaluation formula for param value is

```
Output = Numerator / Denominator
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MATH_FCT_1D_Fraction
Required	PRC_TYPE_MTH_FCT_1D	Numerator
Required	PRC_TYPE_MTH_FCT_1D	Denominator

7.12.7 PRC_TYPE_MATH_FCT_1D_ArctanCos

This represents a 1D trignometric arcfunction.

The evaluation formula for param is

Output = atan((amplitude * cos((param * frequency) + phase))) * a

Where

- -Pi/2 < amplitude < Pi/2
- -263 < (param * frequency) < 263

E Reserved_double must be set to 0.0

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MATH_FCT_1D_ArctanCos
Required	Double	A
Required	Double	Amplitude
Required	Double	Frequency
Required	Double	Phase
Required	Double	E ; note that this is not used

7.12.8 PRC_TYPE_MATH_FCT_1D_Combination

7.12.8.1 General

This represents a function that is a combination of several 1D functions

The evaluation formula at a param value is

```
Output = 0.0
For (i=0; i<number_of_coefficients; i++) {
    Output = output + coefficient[i] * function[i];
}
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MATH_FCT_1D_Combination
Requiried	UnsignedInteger	Number of coefficients (or functions)
Required	ArrayOf [CombinationFunctions]	Array of coefficients and functions

7.12.8.2 CombinationFunctions

Required or Option	Data Type	Data Description
Required	Double	Coefficient
Required	PRC_TYPE_MATH_FCT_1D	Any 1D mathematical function

7.12.9 PRC_TYPE_MATH_FCT_3D

Abstract class for 3D mathematical functions.

The following are legal 3D mathematical functions:

- PRC_TYPE__MATH_FCT_3D_Linear
- PRC_TYPE_MATH_FCT_3D_NonLinear

7.12.10 PRC_TYPE_MATH_FCT_3D_Linear

The represents a 3D linear function.

The evaluation formula at a param value is

```
output.x  = mat[0] [0] * param.x + mat[1] [0] * param.y + mat[2][0] * param.z + vect[0]

output.y  = mat[0] [1] * param.x + mat[1] [1] * param.y + mat[2][1] * param.z + vect[1]

output.z  = mat[0] [2] * param.x + mat[1] [2] * param.y + mat[2][2] * param.z + vect[2]
```

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MATH_FCT_3D_Linear
Required	Double[3] [3]	Matrix[3][3] stored row by row
Required	Double[3]	Vector of 3 coordinates

7.12.11 PRC_TYPE_MATH_FCT_3D_NonLinear

This represents a 3D non-linear mathematical function.

The evaluation formula for param is as follows:

```
tmp_result = left_transformation.evaluate(param);

output.x = tmp_result.x * cos(tmp_result.y * d2);

output.y = tmp_result.x * sin(tmp_result.y *d2);

output.z = tmp_result.z;

output = right_transformation.evaluate(output);
```

Note that the following are reserved for future use but should be initialized to default values:

- Reserved_double must be set to Pi
- Reserved_int_1 must be set to 2
- Reserved_int_2 must be set to 2
- Reserved_int_3 must be set to 0

Required or Option	Data Type	Data Description
Required	UnsignedInteger	PRC_TYPE_MATH_FCT_3D_NonLinear

Required	PRC_TYPE__MATH_FCT_3D	Left 3D non-linear transformation
Required	PRC_TYPE__MATH_FCT_3D	Rignt 3D non-linear transformation
Required	Double	d2
Required	Double	Reserved_double (not used)
Required	Integer	Reserved_int_1 (not used)
Required	Integer	Reserved_int_2 (not used)
Required	Integer	Reserved_int_3 (not used)

8 Other Data Classes

8.1 Other data classes

Other Data Classes represent composite data fields within a Base Entity. A flag indicates the presense or absense of this data. This flag may be a Boolean (TRUE or FALSE), a value within an enumerated type, or an integer value. The flag may also be a character whose bits represent options where one or more of the options may be present.

8.2 Parameter Range

8.2.1 Infinite_param

This represents a real number 12345 which is used to define infinity, the largest parameter value.

8.2.2 Interval

An Interval is a subset of R^1. It is represented by two double precision numbers defining the minimum and maximum values of the interval. An interval is defined to be all values between the minimum and maximum

minimum <= t <= maximum

Infinite or semi-infinite intervals may be specified by using infinite_param to represent infinity. The following are examples of intervals:

- [0.0, 1.0]
- [-infinite_param, infinite_param]
- [0.0, infinite_param]

The primary use of an interval is to define the domain of a curve. For a non-periodic curve, whether an open or closed curve, the interval defines the domain of legal parameter values for the curve. For a periodic closed curve, such as a circle or periodic closed NURBS curve, the interval defines the period (i.e. the length of the interval) as well as the primary domain of the curve. Any value is a legal parameter when evaluated modulo the period. For a periodic open curve, such as a proper subset of a circle, the interval defines the valid subset of R^1 that is the domain of legal parameter values.

Required or Option	Data Type	Data Description
Required	Double	Minimum value
Required	Double	Maximum value

8.2.3 Parameterization

Parameterization data provide a way to reparameterize a curve and to define the domain of the curve. The domain of the curve define legal parameter values.

Two doubles, Coeff-a and Coeff-b, are used to reparameterize a curve towards its implicit parameterization.

The evaluation formula to calculate the implicit parameter from a parameter is:

Implicit_parameter = Coeff_a * parameter + Coeff_b

If there is no reparameterization of the curve, Coeff_a must be set to 1.0 and Coeff_b must be set to 0.0. In this case implicit_parameter equals the specified parameter.

Parameterization data also contain an interval used to define the domain of the curve. This interval restricts the curve before applying the reparameterization formula. The reparameterization formula can be used to calculate the implicit_interval from the given interval in the same way that it is used to calculate the implicit_parameter from a given parameter.

All curves must have an interval to define the legal parmeter values of the curve. In the case of base curves (curves not defined by reference to other curves), this interval defines the domain of definition for the curve. For curves which are defined in terms of other curves, the interval represents a subset of the curve which may be the entire curve or a portion of it. The interpretation of the inteval depends upon the curve being periodic or non-periodic (see the definition of Interval).

For instance, a circle has the implicit domain of [0.0, 2*Pi] and a line has the implicit domain of [-infinite_param, infinite_param]. A portion of the circle might be limited to [0.0, Pi].

Required or Option	Data Type	Data Description
Required	Interval	trim interval
Required	Double	Coeff_a
Required	Double	Coeff_b

As an example, consider a circle. It has an implicit parameterization on the interval [0.0, 2.0*PI]. For this case

- Coeff_a = 1.0
- Coeff_b = 0.0
- Interval = [0.0, 2.0* Pi]

To reparameterize the circle so the parameter values are in the interval [0.0, 1.0] would give

- Coeff_a = 2.0 * Pi
- Coeff_b = 0.0,
- Interval = [0.0, 1.0]

To define a semi circle

- Coeff_a = 1.0
- Coeff_b = 0.0
- Interval = [-Pi/2.0, Pi/2.0]

8.2.4 Domain

A Domain is a subset of R^2 that is used to define the domain of definition of a surface. It is represented by a 2D vector defining the minimum UV parameter values and a 2D vector defining the maximum UV parameters.

The domain of the surface is

minimum U <= u <= maximum U

minimum V <= v <= maximum V

The interval in U (or V) when used with a specific surface definition will define a surface that is open, closed, or periodic in that parameter. For a non-periodic surface, whether an open or closed surface, the interval defines the domain of legal parameter values for the surface. For a periodic closed surface, such as a cone or periodic closed NURBS surface, the interval defines the period (i.e. the length of the interval) as well as the primary domain of the surface. Any value is a legal parameter when evaluated modulo the period. For a periodic open surface, such as a proper subset of a cone, the interval defines the valid subset of R^2 that is the domain of legal parameter values.

Required or Option	Data Type	Data Description
Required	Vector2d	Minimum U and V
Required	Vector2d	Maximum U and V

8.2.5 UVParameterization

This describes a reparameterization of a surface towards implicit parameterization. Coeff-a must be set to 1.0 and Coeff-b must be set to 0.0 if there is no reparameterization.

The evaluation formula is:

If (!swap_uv) {

Implicit_param.u = U_param_coeff_a * param.u + U_param_coeff_b

```
    Implicit_param.v = V_param_coeff_a * param.v + V_param_coeff_b
} else {
    Implicit_param.u = V_param_coeff_a * param.v + V_param_coeff_b
    Implicit_param.v = U_param_coeff_a * param.u + U_param_coeff_b
}
```

The domain of the surface is specified in numbers before the previous reparameterization formula is aplied.

Required or Option	Data Type	Data Description
Required	Boolean	TRUE implies swap the uv param; FALSE implies do not swap
Required	Domain	Surface domain
Required	Double	U_param_coeff_a
Required	Double	V_param_coeff_a
Required	Double	U_param_coeff_b
Required	Double	V_param_coeff_b

8.3 BaseTopology

Base topology represents optional information for a topological entity. Such optional information is comprised of a name, attributes, and an identifier within the originating CAD system. The identifier is not used as an identifier within the PRC File, but is simply carried as information about the origin of this entity within the originating CAD system.

Required or Option	Data Type	Data Description
Required	Boolean	TRUE if base information is present; else FALSE
OPTION: TRUE	AttributeData	Attributes attached to the topological entity

OPTION: TRUE	Name	Name attached to the topological entity
OPTION: TRUE	UnsignedInteger	Identifier in originating CAD system; this may not be used to reference entity within PRC File;

8.4 BaseGeometry

Base geometry represents optional information for a geometric entity. Such optional information is comprised of a name, attributes, and an identifier within the originating CAD system. The identifier is not used as an identifier within the PRC File, but is simply carried as information about the origin of this entity within the originating CAD system.

Required or Option	Data Type	Data Description
Required	Boolean	TRUE if base information is present; else FALSE
OPTION: TRUE	AttributeData	Attributes attached to the geometric entity
OPTION: TRUE	Name	Name attached to the geometric entity
OPTION: TRUE	UnsignedInteger	Identifier in originating CAD system; this may not be used to reference entity within PRC File;

8.5 Basic types for geometry

8.5.1 Vector2d

Representation of a 2D vector. This type may be used to represent either 2D positions or vectors. The context must be used to determine if a position or vector is meant.

Required or Option	Data Type	Data Description
Required	Double	X value
Required	Double	Y value

8.5.2 Vector3d

Representation of a 3D vector. This type may be used to represent either 3D positions or vectors. The context must be used to determine if a position or vector is meant.

Required or Option	Data Type	Data Description
Required	Double	X value
Required	Double	Y value
Required	Double	Z value

8.5.3 BoundingBox

Define a bounding box with sides parallel to the XYZ coordinate planes using two diagonal corners of the box.

The minimum and maximum must satisfy

- Xmin < Xmax
- Ymin < Ymax
- Zmin < Zmax

Required or Option	Data Type	Data Description
Required	Vector3d	Minimum corner
Required	Vector3d	Maximum corner

8.6 UserData

Applications that write a PRC File may store an arbitrary bit stream of private data for the following data types :

- Subtypes of PRC_TYPE_ASM
- Subtypes of PRC_TYPE_RI
- Subtypes of PRC_TYPE_MKP
- Subtypes of PRC_TYPE_MISC

All those entities bear implicitely UserData at the end of their data definition.

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number of bits in the bit stream
OPTION: number bits > 0	UserDataStream	Arbitrary bit stream of the specified length

8.6.1 UserDataStream

Required or Option	Data Type	Data Description
Required	UnsignedInteger	Number of UserData sub sections in the bit stream
Required	ArrayOf[UserDataSubSection]	Array of sub-sections

8.6.2 UserDataSubSection

Required or Option	Data Type	Data Description
Required	Boolean	Same Application UUID than FileStructure
OPTION : FALSE	CompressedUniqueId	Application UUID
Required	UnsignedInteger	Number of bits in the bit stream for this user data subsection
OPTION: number bits > 0	BITS	Arbitrary bit stream of the specified length

9 Schema Definition

9.1 General

The PRC File Format Specification provides a mechanism for describing data stored within a PRC File. A schema definition language can describe changes between versions of the PRC File Format Specification. See 5.2 for a basic description of versioning in PRC.

This mechanism allows new entity types to be defined and information to be added for an existing entity type and written to a PRC File which is readable by PRC File Reader software written to conform to previous versions of the PRC File Format Specification.

New data for an existing entity type must be added at the end and before UserData (if any); this rule applies to all inheritance chain for this given type.

The schema definition language contains tokens to

- describe the primary data in an entity (Boolean, Integer, UnsignedInteger, Double, Character, String, Vector 2D, Vector3D, Interval, Domain, BoundingBox);
- define a block of data which may indicate a version number;
- describe conditional processing (if/then/else, relational tests such as less than, less than or equal to, greater than, greater than or equal to, equal to, not equal to);
- define variables which may be assigned values;
- perform simple binary mathemathical operations such as multiplication, division, addition, subtraction;
- test the type of geometry present in a file;
- read a curve or surface.

Using these schema tokens, new entity types or additions to an existing entity type may be described. This capability is used to describe changes between versions of the PRC File Format Specification.

The following describes processing the data in a PRC File from the point of view of a PRC File Writer and a PRC File Reader:

- The current data in the PRC File (indicated by the authoring_version) consists of base entity data (indicated by the mimimal_version_for_read) plus a delta describing new fields within base entities plus new entities (indicated by schema definitions in the PRC File).
- A PRC File Writer
    - Is based on a current_version of the PRC File Format Specification.
    - Writes minimal_version_for_read to indicate the structure and content of the base data that the Writer is based on.
    - Writes authoring_version to indicate the structure and content of the current file.
    - If minimal_version_for_read = authoring_version, the Writer does not have to write any extra data.
    - If mimimal_version_for_read < authoring_version, the PRC File Writer must include schema definitions in the PRC File describing the delta: the new fields of base entities and all new entities, but only if such entities actually exist in the file.
- A PRC File Reader
    - Is based on a current_version of the PRC File Format Specification.
    - If the current_version is less that the minimal_version_to_read, an error has occurred and the Reader should not continue to process the file. The Reader is built on a version of the specification that preceeds the (base) version of the specification that the Writer was built on.
    - If the current_version is the same as the authoring_version, the PRC File Reader may read all of the data in the file without reference to any schema definitions. Both the Writer and Reader have the same view of the data in the PRC File.
    - Otherwise, the Reader is built on a version of the specification that knows the structure of the base entities in the file and will use the schema definitions to process new data from the PRC File. When reading an entity

- If the entity type is a base entity type, the reader will read all of the base data for the entity from the file. It will then use the schema to skip new information in the file.
- If the entity type is a new entity, the Reader will use the schema to skip the information in the PRC File

9.2 Enumeration of Schema Tokens

The following table provides the value, name, and short description of the schema tokens that are used to describe a PRC entity. See 9.3 for a description of how these tokens are used.

Value	Token Name	Description
0	EPRCSchema_Data_Boolean	Read a Boolean
1	EPRCSchema_Data_Double	Read a Double
2	EPRCSchema_Data_Character	Read a Character
3	EPRCSchema_Data_Unsigned_Integer	Read an UnsignedInteger
4	EPRCSchema_Data_Integer	Read an Integer
5	EPRCSchema_Data_String	Read a String
6	EPRCSchema_Father_Type	Father type of the current object
7	EPRCSchema_Vector_2D	Read a Vector2d
8	EPRCSchema_Vector_3D	Read a Vector3d
9	EPRCSchema_Extent_1D	Read an Interval
10	EPRCSchema_Extent_2D	Read a Domain
11	EPRCSchema_Extent_3D	Read a BoundingBox
12	EPRCSchema_Ptr_Type	Read a specified typed object
13	EPRCSchema_Ptr_Surface	Read a surface
14	EPRCSchema_Ptr_Curve	Read a curve
15	EPRCSchema_For	Loop of instructions
16	EPRCSchema_SimpleFor	Loop of instructions
17	EPRCSchema_If	Condition block
18	EPRCSchema_Else	Condition block
18	EPRCSchema_Block_Start	Define an instruction block
20	EPRCSchema_Block_Version	Define a versioned instruction block
21	EPRCSchema_Block_End	End of a bloc
22	EPRCSchema_Value_Declare	Declare a global value
23	EPRCSchema_Value_Set	Set a global value
24	EPRCSchema_Value_DeclareAndSet	Declare and set a global value
25	EPRCSchema_Value	Access a global value
26	EPRCSchema_Value_Constant	Value constant
27	EPRCSchema_Value_For	Value of the for-loop
28	EPRCSchema_Value_CurveIs3D	Specific value (9.3)
29	EPRCSchema_Operator_MULT	* operator
30	EPRCSchema_Operator_DIV	/ operator
31	EPRCSchema_Operator_ADD	+ operator
32	EPRCSchema_Operator_SUB	– operator
33	EPRCSchema_Operator_LT	< operator
34	EPRCSchema_Operator_LE	<= operator
35	EPRCSchema_Operator_GT	> operator
36	EPRCSchema_Operator_GE	>= operator
37	EPRCSchema_Operator_EQ	== operator
38	EPRCSchema_Operator_NEQ	!= operator

9.3 Schema Processing

9.3.1 EPRCSchema_Data_Boolean

The next field in the file is a **Boolean**.

9.3.2 EPRCSchema_Data_Double

The next field in the file is a **Double**.

9.3.3 EPRCSchema_Data_Character

The next field in the file is a **Character**.

9.3.4 EPRCSchema_Data_Unsigned_Integer

The next field in the file is an **UnsignedInteger**.

9.3.5 EPRCSchema_Data_Integer

The next field in the file is an **Integer**.

9.3.6 EPRCSchema_Data_String

The next field in the file is a **String**.

9.3.7 EPRCSchema_Father_Type

This token indicates that the next token in the token array is the father type. Legal values of the father types are **PRC_TYPE_SURF** and **PRC_TYPE_CRV**.

Father type is used in case of inheritance from another object. There is no specific field in the PRC File corresponding to this type of token. In this case, the parent object (abstract curve or surface) is read before the object.

9.3.8 EPRCSchema_ Vector_2D

The next field in the file is a **Vector2d**.

9.3.9 EPRCSchema_ Vector_3D

The next field in the file is a **Vector3d**.

9.3.10 EPRCSchema_Extent_1D

The next field in the file is an **Interval**.

9.3.11 EPRCSchema_Extent_2D

The next field in the file is a **Domain**.

9.3.12 EPRCSchema_Extent_3D

The next field in the file is a **BoundingBox**.

9.3.13 EPRCSchema_Ptr_Type

This token indicates that the next token in the token array corresponds to the type of object that is next in the file. The legal types that may be read from the file are

- PRC_TYPE_SURF
- PRC_TYPE_CRV
- PRC_TYPE_TOPO
- PRC_TYPE_RI
- PRC_TYPE_MKP_Markup
- PRC_TYPE_MATH_FCT_1D
- PRC_TYPE_MATH_FCT_3D
- The integer number representing a new entity type defined in the schema sections of a PRC File.

9.3.14 EPRCSchema_Ptr_Surface

The next field in the file is a surface (**PRC_TYPE_SURF**). This is equivalent to a **EPRCSchema_Ptr_Type** with an implicit **PRC_TYPE_SURF** object type.

9.3.15 EPRCSchema_Ptr_Curve

The next field in the file is a curve (**PRC_TYPE_CRV**). This is equivalent to a **EPRCSchema_Ptr_Type** with an implicit **PRC_TYPE_CRV** object type.

9.3.16 EPRCSchema_For

An EPRCSchema_For defines a for loop as three tokens

- The token EPRCSchema_For indicating a for loop follows.
- A schema block defining the number of times to execute the block.
- A schema block defining the block to be executed.

Example:

Token Sequence	Pseudo Code

EPRCSchema_For
EPRCSchema_Operator_ADD
EPRCSchema_Data_Integer
EPRCSchema_Value_Constant
6
	int loop_end = read_integer() + 6;
	for (int i=0; i<loop_end; i++) {
	read_double();
	}

EPRCSchema_Data_Double	

9.3.17 EPRCSchema_SimpleFor

An EPRCSchema_SimpleFor defines a for loop as two tokens

- The token EPRCSchema_For indicating a for loop follows.
- The number of time to execute the block is not defined in the schema. For a SimpleFor loop the loop counter must be a simple integer in the PRC File and not defined by a more complex schema block.
- A schema block defining the block to be executed.

The next token is the loop counter number and the token after that is the token to execute counter number of times.

Example:

Token Sequence	Pseudo Code

EPRCSchema_SimpleFor
EPRCSchema_Block_Start
EPRCSchema_Data_Integer
EPRCSchema_Data_Double
EPRCSchema_If
EPRCSchema_Operator_EQ
EPRCSchema_Value_For
EPRCSchema_Value_Constant
0
EPRCSchema_Data_Double
EPRCSchema_Block_End
	int loop_end = read_unsigned_integer();
	for (int i=0; i<loop_end; i++)
	{
	read_integer();
	read_double()
	if (i == 0)
	read_double();
	}

9.3.18 EPRCSchema_If and EPRCSchema_Else

This token indicates the start of a conditional token clause. The next token instruction corresponds to the conditional value, and executes the following token instruction if the conditional value is true. If there is an EPRCSchema_Else token.

Example:

Token Sequence	Pseudo Code

EPRCSchema_If
  EPRCSchema_OperatorEQ
  EPRCSchema_Data_Integer
  EPRCSchema_Value_Constant
  3
  EPRCSchema_Vector_3D
EPRCSchema_Else
  EPRCSchema_Vector_2D
	tmp_integer = read_integer();
	If (tmp_integer == 3)
	  read_vector_3d();
	Else
	  read_vector_2d();

9.3.19 EPRCSchema_Block_Start

This token indicates a block of tokens terminated by an EPRCSchema_Block_End. The block of tokens is not versioned.

9.3.20 EPRCSchema_Block_Version

This token indicates a versioned block of tokens terminated by an EPRCSchema_Block_End. The token following the EPRCSchema_Block_Version is is the version number. If the version number is less than or equal to the current version, the block version is ignored.

Example:

Token Sequence	Pseudo Code

EPRCSchema_Block_Version
  150
  EPRCSchema_Data_Integer
  EPRCSchema_Vector_3D
EPRCSchema_Block_End
	If (current_version < 150) {
	  read_integer();
	  read_vector_3d();
	}

9.3.21 EPRCSchema_Block_End

This token indicates the end of a token block.

9.3.22 EPRCSchema_Value_Declare

This token declares and sets to zero a new value indexed by the next instruction value. All values are local to the block instruction.

Example:

Token Sequence	Pseudo Code
EPRCSchema_Value_Declare 5	int value_5 = 0;

9.3.23 EPRCSchema_Value_Set

Sets a value indexed by the next token instruction value. Before it can be used, a variable must be declared.

Example:

Token Sequence	Pseudo Code
EPRCSchema_Value_Declare 5 EPRCSchema_Value_Set 5 EPRC_Schema_Data_Integer	int value_5 = 0; value_5 = read_integer();

9.3.24 EPRCSchema_Value_DeclareAndSet

Declares and sets a new value indexed by the next instruction value.

Example:

Token Sequence	Pseudo Code
EPRCSchema_Value_DeclareAndSet 5 EPRC_Schema_Data_Integer	int value_5 = read_integer();

9.3.25 EPRCSchema_Value

Retrieves the content of the value indexed by the next instruction value. Before it can be used, a variable must be declared.

Example:

Token Sequence	Pseudo Code
EPRCSchema_Value_DeclareAndSet	int value_5 = 2;

5 EPRCSchema_Value_Constant 2 EPRCSchema_Value_DeclareAndSet 2 EPRCSchema_Data_Integer EPRCSchema_DeclareAndSet 6 EPRCSchema_Operator_ADD EPRCSchema_Value 5 EPRCSchema_Value 2	int value_2 = read_integer(); int value_6 = value_5 + value_2;

9.3.26 EPRCSchema_Value_Constant

Retrieves a constant value given by the next token.

Example:

Token Sequence	Pseudo Code
EPRCSchema_Value_DeclareAndSet 5 EPRC_Schema_Value_Constant 10	int value_5 = 10;

9.3.27 EPRCSchema_Value_For

Retrieves the value of the current for-loop.

Example:

See 9.3.17 example.

9.3.28 EPRCSchema_Value_CurveIs3D

If the current object is a curve, return `true` if it is a 3D curve, `false` if it is a 2D curve.

9.3.29 EPRCSchema_Operator_MULT

Binary operator for multiplication.

9.3.30 EPRCSchema_Operator_DIV

Binary operator for division.

9.3.31 EPRCSchema_Operator_ADD

Binary operator for addition.

9.3.32 EPRCSchema_Operator_SUB

Binary operator for subtraction.

9.3.33 EPRCSchema_Operator_LT

Relational operator for less than comparison.

Example:

Token Sequence	Pseudo Code

EPRCSchema_Value_DeclareAndSet
5
EPRCSchema_Data_Integer
EPRC_Schema_Value_DeclareAndSet
10
EPRCSchema_Data_Integer
EPRCSchema_If
EPRC_Schema_Operator_LT
EPRCSchema_Value
5
EPRCSchema_Value
10
EPRCSchema_Block_Start
.....
EPRCSchema_Block_End
	int value_5 = read_integer();
	int value_10 = read_integer();
	if (value_5 < value_10) {
	...
	}

9.3.34 EPRCSchema_Operator_LE

Relational operator for less than or equal to comparison.

9.3.35 EPRCSchema_Operator_GT

Relational operator for greater than comparison.

9.3.36 EPRCSchema_Operator_GE

Relational operator for greater than or equal to comparison.

9.3.37 EPRCSchema_Operator_EQ

Relational operator for equal to comparison.

9.3.38 EPRCSchema_Operator_NEQ

Relational operator for not equal to comparison.

9.4 Schema Examples

9.4.1 General

The following are some requirements of the schema definition language and its usage

1. All of the examples have an enclosing block which is not a version block. This is a requirement.
2. New curves (or surfaces) have either the PRC_TYPE_CRV or a particular curve entity type as the EPRCSchema_Father_Type. The Father_Type indicates what data will be inherited by the entity in the schema.
3. For example, for PRC_TYPE_CRV and PRC_TYPE_SURFACE, attributes, transform, and parametrization are inherited from the Father_Type.
4. The example to add a field to an existing entity did not define the previous data. All new data must be added to the end of an existing entity.

9.4.2 An existing entity

This is the schema definition of PRC_TYPE_MISC_GeneralTransformation which was defined in version 8137 of the PRC File Format Specification.

Data Type	Token Sequence	Comments
UnsignedInteger	PRC_TYPE_MISC_GeneralTransformation	Value indicating type of entity
UnsignedInteger	8	N umber of tokens that follow
UnsignedInteger	EPRCSchema_Block_Start	Start of all blocks
UnsignedInteger	EPRCSchema_Block_Version	Beginning of a versioned block

UnsignedInteger	8137	Version number
UnsignedInteger	EPRCSchema_SimpleFor	For loop to get the data
UnsignedInteger	16	Read 16
UnsignedInteger	EPRCSchema_Data_Double	Doubles in the transformation matrix
UnsignedInteger	EPRCSchema_Block_End	End of new versioned block
UnsignedInteger	EPRCSchema_Block_End	End of all blocks

9.4.3 Existing PRC_TYPE_CRV_Polyline

The following is the schema for the existing PRC_TYPE_CRV_Polyline

Data Type	Token Sequence	Comments
UnsignedInteger	PRC_TYPE_CRV_Polyline	Value indicating type of entity
UnsignedInteger	14	N umber of tokens that follow
UnsignedInteger	EPRCSchema_Block_Start	Start of all blocks
UnsignedInteger	EPRCSchema_Block_Version	Beginning of a versioned block
UnsignedInteger	Version_0	Version number
UnsignedInteger	EPRCSchema_Father_Type	The class of the parent entity
UnsignedInteger	PRC_TYPE_CRV	Is a curve
UnsignedInteger	EPRCSchema_SimpleFor	For loop to get the data
UnsignedInteger	Integer value	Number of points in the polyline
UnsignedInteger	EPRCSchema_If	if the
UnsignedInteger	EPRCSchema_Value_CurveIs3D	Curve is a 3D curve
UnsignedInteger	EPRCSchema_Vector_3D	The data field will be a 3D vector
UnsignedInteger	EPRCSchema_Else	otherwise
UnsignedInteger	EPRCSchema_Vector_3D	The data field will be a 2D vector
UnsignedInteger	EPRCSchema_Block_End	End of new versioned block
UnsignedInteger	EPRCSchema_Block_End	End of all blocks

9.4.4 Add a field to existing entity

In this example a string field is added to the existing curve PRC_TYPE_CRV_Line.

For an existing entity, new data must be added to the end.

Data Type	Token Sequence	Comments
UnsignedInteger	PRC_TYPE_CRV_Line	This is the entity type of the existing type of curve (line).
UnsignedInteger	6	Number of tokens to follow
UnsignedInteger	EPRCSchema_Block_Start	Start of all blocks
UnsignedInteger	EPRCSchema_Block_Version	Beginning of a versioned block
UnsignedInteger	9149	File version number 149th day of 2009
UnsignedInteger	EPRCSchema_Data_String	Data field
UnsignedInteger	EPRCSchema_Block_End	End of new versioned block
UnsignedInteger	EPRCSchema_Block_End	End of all blocks

9.4.5 Add a new curve

In this example a new curve type with 2 doubles and a pointer to a curve is added to the schema.

Data Type	Token Sequence	Comments
UnsignedInteger	PRC_TYPE_CRV_NewCurve	This is the entity type of the new curve.
UnsignedInteger	10	N umber of tokens to follow
UnsignedInteger	EPRCSchema_Block_Start	Start of all blocks
UnsignedInteger	EPRCSchema_Block_Version	Beginning of a versioned block
UnsignedInteger	Version_1	File version number
UnsignedInteger	EPRCSchema_Father_Type	The class of the parent entity
UnsignedInteger	PRC_TYPE_CRV	Is a curve
UnsignedInteger	EPRCSchema_Double	Data field is a double
UnsignedInteger	EPRCSchema_Double	Data field is a double
UnsignedInteger	EPRCSchema_Ptr_Curve	Data field is a curve
UnsignedInteger	EPRCSchema_Block_End	End of new versioned block

UnsignedInteger	EPRCSchema_Block_End	End of all blocks

9.4.6 Multiple revisions to an entity type

Data Type	Token Sequence	Comments
UnsignedInteger	PRC_TYPE_CRV_NewCurve	This is the entity type of the new curve.
UnsignedInteger	14	Number of tokens to follow
UnsignedInteger	EPRCSchema_Block_Start	Start of all blocks
UnsignedInteger	EPRCSchema_Block_Version	Start of
UnsignedInteger	Version_1	Version number of Version_1 block
UnsignedInteger	EPRCSchema_Father_Type	The father type
UnsignedInteger	PRC_TYPE_CRV_Line	Is a PRC_TYPE_CRV_Line
UnsignedInteger	EPRCSchema_Data_Double	Data field 1 is a double
UnsignedInteger	EPRCSchema_Data_Double	Data field 2 is a double
UnsignedInteger	EPRCSchema_Block_End	End of version 1 block
UnsignedInteger	EPRCSchema_Block_Version	Start of
UnsignedInteger	Version_2	Version number of Version_2 block
UnsignedInteger	EPRCSchema_SimpleFor	Simple for loop to read
UnsignedInteger	EPRCSchema_Data_Integer	An integer
UnsignedInteger	EPRCSchema_Block_End	End of version 2 block
UnsignedInteger	EPRCSchema_Block_End	End of all blocks

10 Data Types for Physical File

10.1 General

All data within a physical PRC File exists in a section which is a contiguous stream of bytes which starts and ends on a byte boundary. A section may be either uncompressed (header sections) or compressed (all other sections).

Data within an uncompressed section is written so that individual variables occupies a specific number of bytes.

Data within a compressed section has been written in a bit-by-bit mannor with no restrictions on individual variables crossing byte boundaries. At the end of the section, the last byte is padded with zero bits, the entire section is compressed using flate (see Bibliography) and the compressed section is written to the file. At this point, the current name and current graphics are reset.

Reading one section is independent from reading other sections and one can skip directly to a section since the offset (in bytes) of the section from the beginning of the file is known as it is also stored in the PRC File.

10.2 Uncompressed Types

10.2.1 General

The following types are used to define data contained within uncompressed sections of the PRC File.

10.2.2 UncompressedFiles

This type represents writing the data for multiple uncompressed file data.

Required or Option	Data Type	Data Description
Required	UncompressedUnsignedInteger	Number of uncompressed Files
Required	ArrayOf [UncompressedBlock]	Array of uncompressed file data

10.2.3 UncompressedBlock

The purpose of this function is to write the number of bytes being written followed by the specified number of bytes without the need to further interpret the content of the byte stream.

Required or Option	Data Type	Data Description
Required	UncompressedUnsignedInteger	size (bytes) of uncompressed block
Required	Byte stream of the specified size	Block of specified size

10.2.4 UncompressedUnsignedInteger

This type represents writing an unsigned integer. An unsigned integer is converted into a 4 byte array of unsigned characters which is independent of machine byte ordering using the algorithm **MakePortable32BitsUnsigned**.

Required or Option	Data Type	Data Description

Required	4 bytes	4 bytes representing unsigned integer

10.3 Compressed Types

10.3.1 General

The following types are used to define data contained within compressed sections of the PRC File. Each type may start or end at any bit position within a byte in the compressed section.

10.3.2 Bits

This requires a special algorithm. See 11.3.

10.3.3 Boolean

Required or Option	Data Type	Data Description
Required	Bits(1)	Single bit in a bit stream

10.3.4 Character

Required or Option	Data Type	Data Description
Required	Bits(8)	Single character in a bit stream

10.3.5 CharacterArray

This requires a special algorithm. See 11.6.

10.3.6 FloatAsBytes

This requires a special algorithm. See 11.5.

10.3.7 String

This is a UTF8-encoded string. The number of characters does not include a terminating null character.

Required or Option	Data Type	Data Description
Required	Boolean	TRUE if the string is not NULL; else FALSE
OPTION: TRUE	UnsignedInteger	Size of character array

OPTION: TRUE	ArrayOf[Character]	Array of characters in string

10.3.8 ShortArray

This requires a special algorithm. See 11.7.

10.3.9 Double

This requires a special algorithm. See 11.17.

10.3.10 DoubleWithVariableBitNumber

This requires a special algorithm. See 11.14.

10.3.11 Integer

This requires a special algorithm. See 11.11.

10.3.12 IntegerWithVariableBitNumber

This requires a special algorithm. See 11.12.

10.3.13 CompressedIntegerArray

This requires a special algorithm. See 11.8.

10.3.14 UnsignedInteger

This requires a special algorithm. See 11.10.

10.3.15 UnsignedIntegerWithVariableBitNumber

This requires a special algorithm. See 11.13.

10.3.16 CompressedIndiceArray

This requires a special algorithm. See 11.9.

10.3.17 NumberOfBitsThenUnsignedInteger

This requires a special algorithm. See 11.15.

10.3.18 CompressedEntityType

This requires a special algorithm. See 11.16.

11 I/O Algorithms

11.1 GetNumberOfBitsUsedToStoreUnsignedInteger

Computes the number of bits needed to serialize an unsigned integer.

The following code example shows how to compute the number of bits needed for an unsigned integer.

```
unsigned GetNumberOfBitsUsedToStoreUnsignedInteger(unsigned uValue)
{
   unsigned uNbBit = 1;
   unsigned uTemp = 1;
   while(uValue > uTemp)
   {
      uTemp*=2;
      uNbBit++;
   }
   return uNbBit;
}
```

11.2 MakePortable32BitsUnsigned

Builds an array of unsigned characters from an unsigned 32-bit integer. The final value is therefore independent of the machine byte-ordering.

The following code example shows how to make a portable array of unsigned character values.

```
void MakePortable32BitsUnsigned(unsigned uValue, unsigned char pcValue[4] )
{
   pcValue[0] = (unsigned char)(uValue & 0xFF);
   uValue >>= 8;
   pcValue[1] = (unsigned char)(uValue & 0xFF);
   uValue >>= 8;
   pcValue[2] = (unsigned char)(uValue & 0xFF);
   uValue >>= 8;
   pcValue[3] = (unsigned char)(uValue & 0xFF);
}
```

11.3 WriteBits

Writes bits from left to right.

The following code example shows how to write bits.

`iBitsCount` specifies the number of bits to be written.
The bits are added to a single byte (`uRemainder`), which is then flushed to the final device each time 8 bits are filled.

At the end of the serialization, `uRemainder` is padded with zeros before writing the last byte to the device.

```
void WriteBits( unsigned uValue, int iBitsCount )
{
   static unsigned uRemainder = 0;
   static int iRemainderBits = 0;
   while (iBitsCount > 0)
   {
      int iBitsDelta = iBitsCount + iRemainderBits - 8 ;
      if (iBitsDelta == 0)
      {
         uRemainder |= uValue;
         iRemainderBits = 0 ;
         iBitsCount = 0 ;
      }
      else if (iBitsDelta < 0)
```

```
      {
         uRemainder |= uValue << (-iBitsDelta);
         iRemainderBits += iBitsCount;
         iBitsCount = 0 ;
      }
      else
      {
         int loc = uValue >> iBitsDelta;
         uRemainder |= loc ;
         uValue -= loc << iBitsDelta;
         iBitsCount -= (8 - iRemainderBits) ;
         iRemainderBits = 0 ;
      }
      if (iRemainderBits == 0)
      {
         // writing 1 byte (uRemainder) to the device

      }
   }
}
```

11.4 WriteString

Writes a UTF8-encoded string. The number of characters does not include a terminating null character.
The following code example shows how to write a string.

```
void WriteString( const char* pcString )
{
   if (pcString == 0 )
      WriteBit(0);
   else
   {
      unsigned i, iNumberOfCharacters = strlen(pcString);
      WriteBit(1);
      WriteUnsignedInteger(iNumberOfCharacters);
      for (i=0;i<iNumberOfCharacters;i++)
         WriteCharacter(pcString[i]);
   }
}
```

11.5 WriteFloatAsBytes

Write a float as 4 bytes.

```
void WriteFloatAsBytes(float fValue)
{
   union
   {
      float fFloat;
      unsigned int uInt;
   }unionFloatunsignedInt ;

   unionFloatunsignedInt.fFloat = (float) fValue;
   unsigned uTemp = unionFloatunsignedInt.uInt;
   WriteBits(uTemp & 0xFF, 8 );
   uTemp >>= 8 ;
   WriteBits(uTemp & 0xFF, 8 );
```

```
   uTemp >>= 8 ;
   WriteBits(uTemp & 0xFF, 8 );
   uTemp >>= 8 ;
   WriteBits(uTemp & 0xFF, 8 );
}
```

11.6 WriteCharacterArray

This function writes an array of characters.

The following code example shows how to write a character array.

This function allows for both direct storage, or storage after Huffman compression (see section **Huffman Algorithm**), as denoted by variable `bIsCompressed`. The strategy whether or not compressing is left outside the scope of the standard and can vary between implementations to try to optimize size of output file.

For instance, compression can be skipped systematically when the array size is lower than 5, since compressed strategy leads to write at least one unsigned integer.

`bWriteCompressStrategy` is true by default except for calls from **WriteCompressedIndiceArray** (See 11.9) when writing Point_reference_array in **Entity description** (See 7.8.8.8).

```
void Huffman(
   char* pcArray,unsigned uCharArraySize,
   unsigned*& puHuffmanArray,unsigned& uHuffmanArraySize);

// gives Huffman compression strategy on case-by-case basis
bool HuffmanCompression();

void WriteCharacterArray (
   char* pcArray,unsigned uCharArraySize,
   unsigned uBitNumber,bool bWriteCompressStrategy=true)
{
   bool bIsCompressed = HuffmanCompression();
   if (bWriteCompressStrategy)
      WriteBoolean ( bIsCompressed );

   if( bIsCompressed )
   {
      // calling Huffman to create
      // puHuffmanArray and uHuffmanArraySize
      unsigned* puHuffmanArray;
      unsigned u,uHuffmanArraySize;
      Huffman(pcArray,uCharArraySize,
         puHuffmanArray,uHuffmanArraySize);

      WriteUnsignedInteger ( uHuffmanArraySize );
      for( u=0; u<uHuffmanArraySize; u++)
         WriteUnsignedInteger ( puHuffmanArray[u] );
   }
   else
   {
      WriteUnsignedInteger ( uCharArraySize );
      unsigned u;
      for(u=0 ; u<uCharArraySize; u++)
         WriteCharacter ( pcArray[u] );
   }
```

```
}
```

11.7 WriteShortArray

This function writes an array of shorts.

The following code example shows how to write a short array.

This function allows for both direct storage, or storage after Huffman compression (see section 12.2, **Huffman Algorithm**), as denoted by variable `bIsCompressed`. The strategy whether or not compressing is left outside the scope of the standard and can vary between implementations to try to optimize size of output file.

```
void Huffman(
   short* psArray,unsigned uShortArraySize,
   unsigned*& puHuffmanArray,unsigned& uHuffmanArraySize);

void WriteShortArray(
   short* psArray,unsigned uShortArraySize,
   unsigned uBitNumber,bool bIsCompressed)
{
   WriteBoolean ( bIsCompressed );
   if( bIsCompressed )
   {
      // calling Huffman to create
      // puHuffmanArray and uHuffmanArraySize
      unsigned* puHuffmanArray;
      unsigned u,uHuffmanArraySize;
      Huffman(psArray,uShortArraySize,
         puHuffmanArray,uHuffmanArraySize);
      WriteUnsignedInteger ( uHuffmanArraySize );
      for( u=0; u<uHuffmanArraySize; u++)
         WriteUnsignedInteger ( puHuffmanArray[u] );
   }
   else
   {
      WriteUnsignedInteger ( uShortArraySize );
      unsigned u;
      for(u=0 ; u<uShortArraySize; u++)
      {
         WriteCharacter ( psArray[u]&0x00ff );
         WriteCharacter ( (psArray[u]&0xff00)>>8 );
      }
   }
}
```

11.8 WriteCompressedIntegerArray

Writes an array of integers.

The following code example shows how to write a compressed integer array.

```
void WriteCompressedIntegerArray(
   int* piArray, unsigned uIntArraySize,bool bWriteCompressStrategy=true)
{
   unsigned u;
   char* pcArray=new char[uIntArraySize];

   for( u=1; u<uIntArraySize; u++)
```

```
      pcArray[u] = (char) GetNumberOfBitsUsedToStoreInteger(piArray[u]);
   WriteCharacterArray(pcArray,uIntArraySize, 6, bWriteCompressStrategy);

   for( u=0;u < uIntArraySize; u++)
      WriteIntegerWithVariableBitNumber(piArray[u], pcArray[u]);
   delete [] pcArray;
}
```

11.9 WriteCompressedIndiceArray

Writes an array of indices.
The following code example shows how to write an array of indices; the indices are always positive at input.

```
void WriteCompressedIndiceArray(
   int* piArray, unsigned uIntArraySize,bool bWriteCompressStrategy=true)
{
   unsigned u;
   char* pcArray=new char[uIntArraySize];
   pcArray[0] =
      (char)GetNumberOfBitsUsedToStoreUnsignedInteger(abs(piArray[0])+1);
   int iTemp;
   char cTemp;
   for( u=1; u<uIntArraySize; u++)
   {
      iTemp = piArray[u]-piArray[u-1];
      cTemp = (char)GetNumberOfBitsUsedToStoreUnsignedInteger(abs(iTemp))+1;
      pcArray[u] = cTemp - pcArray[u-1];
   }
   WriteCharacterArray(pcArray,uIntArraySize, 6, bWriteCompressStrategy);

   WriteIntegerWithVariableBitNumber(piArray[0],pcArray[0]);
   for( u=1;u < uIntArraySize; u++)
      WriteIntegerWithVariableBitNumber(
         piArray[u] - piArray[u-1], pcArray[u] + pcArray[u - 1] );
   delete [] pcArray;
}
```

11.10 WriteUnsignedInteger

Writes an unsigned integer.
The following code example shows how to write an unsigned integer.

```
void WriteUnsignedInteger(unsigned uValue)
{
   for(;;)
   {
      if (uValue == 0)
      {
         WriteBits(0, 1);
         return;
      }
      WriteBits(1, 1);
      WriteBits(uValue & 0xFF, 8);
      uValue >>= 8;
   }
}
```

11.11 WriteInteger

Writes an integer.

The following code example shows how to write an integer.

```
void WriteInteger(int iValue)
{
   if (iValue == 0)
   {
      WriteBits(0, 1) ;
      return ;
   }
   for(;;)
   {
      int loc = iValue & 0xFF;
      WriteBits(1, 1);
      WriteBits(loc, 8);
      iValue >>= 8 ;
      if (((iValue == 0) && ((loc & 0x80) == 0)) ||
         ((iValue == -1) && ((loc & 0x80) != 0)))
      {
         WriteBits (0, 1) ;
         return ;
      }
   }
}
```

11.12 WriteIntegerWithVariableBitNumber

Writes an integer using a specified number of bits.
The first bit describes the sign. It is set to `true` if the integer is positive and `false` if it is negative. The remaining bits describe the absolute value of the integer, beginning with the most significant bit.

The following code example shows how to write a signed integer with a variable number of bits.

```
void WriteIntegerWithVariableBitNumber(int iValue,unsigned uBitNumber)
{
   WriteBoolean (iValue < 0);
   WriteUnsignedIntegerWithVariableBitNumber(abs(iValue), uBitNumber - 1);
}
```

11.13 WriteUnsignedIntegerWithVariableBitNumber

Writes an unsigned integer using a specified number of bits.
The bits describe the value of the unsigned integer, beginning with the most significant bit.

The following code example shows how to write an unsigned integer with a variable number of bits.

```
void WriteUnsignedIntegerWithVariableBitNumber(
   unsigned uValue, unsigned uBitNumber)
{
   unsigned u;
   for(u=0; u<uBitNumber; u++)
   {
      if( uValue >= 1<<(uBitNumber - 1 - u) )
      {
         WriteBoolean (true);
         uValue -= 1<<(uBitNumber - 1 - u);
      }
```

```
      else
      {
         WriteBoolean(false);
      }
   }
}
```

11.14 WriteDoubleWithVariableBitNumber

Writes a double using a specified number of bits and a tolerance.

The following code example shows how to write a double with a variable number of bits.

`uBitNumber` must be less than 31.

```
void WriteDoubleWithVariableBitNumber(
   double dValue, double dTolerance, unsigned uBitNumber)
{
   WriteBoolean (dValue < 0 );

   // calling functions must ensure no overflow
   unsigned uTempValue = (unsigned) ( fabs(dValue) / dTolerance );
   if(fabs(dValue) / dTolerance - uTempValue  > 0.5)
      uTempValue++;

   unsigned u;
   for(u = 0; u < uBitNumber - 1; u++)
   {
      if(uTempValue>= 1<< (uBitNumber - 2 - u))
      {
         WriteBoolean(true);
         uTempValue -= 1<< (uBitNumber - 2 - u);
      }
      else
      {
         WriteBoolean(false);
      }
   }
}
```

11.15 WriteNumberOfBitsThenUnsignedInteger

Writes the number of bits followed by an unsigned integer whose size corresponds to that number of bits.

The following code example shows how to write the number of bits followed by an unsigned integer.

```
void WriteNumberOfBitsThenUnsignedInteger(unsigned uValue)
{
   unsigned uNbBit =
      GetNumberOfBitsUsedToStoreUnsignedInteger(uValue);
   WriteUnsignedIntegerWithVariableBitNumber(uNbBit,5);
   WriteUnsignedIntegerWithVariableBitNumber(uValue,uNbBit);
}
```

11.16 WriteCompressedEntityType

Writes a curve or surface type for compressed B-rep data.
The following code example shows how to write a curve or surface type.

```
void WriteCompressedEntityType(
   unsigned uEntityType, bool bIsACurve)
{
   WriteBoolean ( bIsACurve );
   if ( bIsACurve )
   {
      switch (uEntityType)
      {
      case PRC_HCG_Line:
         // writing 2 bits (00)
         return WriteUnsignedIntegerWithVariableBitNumber (0 , 2);

      case PRC_HCG_Circle:
         // writing 2 bits (01)
         return WriteUnsignedIntegerWithVariableBitNumber (1 , 2);

      case PRC_HCG_BSplineHermiteCurve :
         // writing 2 bits (10)
         return WriteUnsignedIntegerWithVariableBitNumber (2 , 2);

      case PRC_HCG_Ellipse :
         // writing 2 bits (11) then 2 other (00)
         return WriteUnsignedIntegerWithVariableBitNumber ((3 << 2) + 0 , 4);

      case PRC_HCG_CompositeCurve :
         // writing 2 bits (11) then 2 other (01)
         return WriteUnsignedIntegerWithVariableBitNumber ((3 << 2) + 1 , 4);
      }
   }
   else
   {
      return WriteUnsignedIntegerWithVariableBitNumber (uEntityType , 4);
   }
}
```

11.17 WriteDouble

11.17.1 General

Macros PRC_LITTLE_ENDIAN and PRC_BIG_ENDIAN are used to define the byte order for binary values (must be set depending on the compilation platform).

11.17.2 Data definition for double storage

Contains an array of **sCodageOfFrequentDoubleOrExponent** used in **Procedure for WriteDouble**. This array is sorted to allow the use of a binary search to find values.

```
#include <memory.h>
#include <stdlib.h>
#include <math.h>

void WriteBits(unsigned value, unsigned bits_count);

# define PRC_LITTLE_ENDIAN // For compilation on Windows

#define NUMBEROFELEMENTINACOFDOE (2077)

enum ValueType
{
   VT_double,
   VT_exponent
};

/* Coding of a frequent double or exponent value. */
typedef struct
{
   /* Value type (VT_double or VT_exponent) */
   short Type;

   /* Number of bits. */
   short NumberOfBits;

   /* Bit values. */
   unsigned Bits;

   /* Unsigned or double value. */
   union
   {
      unsigned ul[2]; /* Two unsigned values. */
      double Value; /* Double value. */
   } u2uod;
} sCodageOfFrequentDoubleOrExponent;

#if defined( PRC_BIG_ENDIAN )
#       define DOUBLEWITHTWODWORDINTREE(upper,lower)    {upper,lower}
#elif defined( PRC_LITTLE_ENDIAN )
#       define DOUBLEWITHTWODWORDINTREE(upper,lower)    {lower,upper}
#endif

sCodageOfFrequentDoubleOrExponent acofdoe[NUMBEROFELEMENTINACOFDOE]=
{
   {VT_double,2,0x1,DOUBLEWITHTWODWORDINTREE(0x00000000,0x00000000)},
   {VT_exponent,22,0xd1d32,DOUBLEWITHTWODWORDINTREE(0x00000000,0x00000000)},
   {VT_exponent,22,0xd1d33,DOUBLEWITHTWODWORDINTREE(0x00100000,0x00000000)},
```

```
  {VT_exponent,22,0xf78d8,DOUBLEWITHTWODWORDINTREE(0x00200000,0x00000000)},
  {VT_exponent,22,0xf78d9,DOUBLEWITHTWODWORDINTREE(0x00300000,0x00000000)},
  {VT_exponent,22,0xf78da,DOUBLEWITHTWODWORDINTREE(0x00400000,0x00000000)},
  {VT_exponent,22,0xf78db,DOUBLEWITHTWODWORDINTREE(0x00500000,0x00000000)},
  {VT_exponent,22,0xf78dc,DOUBLEWITHTWODWORDINTREE(0x00600000,0x00000000)},
  {VT_exponent,22,0xf78dd,DOUBLEWITHTWODWORDINTREE(0x00700000,0x00000000)},
  {VT_exponent,22,0xf78de,DOUBLEWITHTWODWORDINTREE(0x00800000,0x00000000)},
  {VT_exponent,22,0xf78df,DOUBLEWITHTWODWORDINTREE(0x00900000,0x00000000)},
  {VT_exponent,22,0xf78e0,DOUBLEWITHTWODWORDINTREE(0x00a00000,0x00000000)},
  {VT_exponent,22,0xf78e1,DOUBLEWITHTWODWORDINTREE(0x00b00000,0x00000000)},
  {VT_exponent,22,0xf78e2,DOUBLEWITHTWODWORDINTREE(0x00c00000,0x00000000)},
  {VT_exponent,22,0xf78e3,DOUBLEWITHTWODWORDINTREE(0x00d00000,0x00000000)},
  {VT_exponent,22,0xf78e4,DOUBLEWITHTWODWORDINTREE(0x00e00000,0x00000000)},
  {VT_exponent,22,0xf78e5,DOUBLEWITHTWODWORDINTREE(0x00f00000,0x00000000)},
  {VT_exponent,22,0xf78e6,DOUBLEWITHTWODWORDINTREE(0x01000000,0x00000000)},
  {VT_exponent,22,0xf78e7,DOUBLEWITHTWODWORDINTREE(0x01100000,0x00000000)},
  {VT_exponent,22,0xf78e8,DOUBLEWITHTWODWORDINTREE(0x01200000,0x00000000)},
  {VT_exponent,22,0xf78e9,DOUBLEWITHTWODWORDINTREE(0x01300000,0x00000000)},
  {VT_exponent,22,0xf78ea,DOUBLEWITHTWODWORDINTREE(0x01400000,0x00000000)},
  {VT_exponent,22,0xf78eb,DOUBLEWITHTWODWORDINTREE(0x01500000,0x00000000)},
  {VT_exponent,22,0xf78ec,DOUBLEWITHTWODWORDINTREE(0x01600000,0x00000000)},
  {VT_exponent,22,0xf78ed,DOUBLEWITHTWODWORDINTREE(0x01700000,0x00000000)},
  {VT_exponent,22,0xf78ee,DOUBLEWITHTWODWORDINTREE(0x01800000,0x00000000)},
  {VT_exponent,22,0xf78ef,DOUBLEWITHTWODWORDINTREE(0x01900000,0x00000000)},
  {VT_exponent,22,0xf78f0,DOUBLEWITHTWODWORDINTREE(0x01a00000,0x00000000)},
  {VT_exponent,22,0xf78f1,DOUBLEWITHTWODWORDINTREE(0x01b00000,0x00000000)},
  {VT_exponent,22,0xf78f2,DOUBLEWITHTWODWORDINTREE(0x01c00000,0x00000000)},
  {VT_exponent,22,0xf78f3,DOUBLEWITHTWODWORDINTREE(0x01d00000,0x00000000)},
  {VT_exponent,22,0xf78f4,DOUBLEWITHTWODWORDINTREE(0x01e00000,0x00000000)},
  {VT_exponent,22,0xf78f5,DOUBLEWITHTWODWORDINTREE(0x01f00000,0x00000000)},
  {VT_exponent,22,0xf78f6,DOUBLEWITHTWODWORDINTREE(0x02000000,0x00000000)},
  {VT_exponent,22,0xf78f7,DOUBLEWITHTWODWORDINTREE(0x02100000,0x00000000)},
  {VT_exponent,22,0xf78f8,DOUBLEWITHTWODWORDINTREE(0x02200000,0x00000000)},
  {VT_exponent,22,0xf78f9,DOUBLEWITHTWODWORDINTREE(0x02300000,0x00000000)},
  {VT_exponent,22,0xf78fa,DOUBLEWITHTWODWORDINTREE(0x02400000,0x00000000)},
  {VT_exponent,22,0xf78fb,DOUBLEWITHTWODWORDINTREE(0x02500000,0x00000000)},
  {VT_exponent,22,0xf78fc,DOUBLEWITHTWODWORDINTREE(0x02600000,0x00000000)},
  {VT_exponent,22,0xf78fd,DOUBLEWITHTWODWORDINTREE(0x02700000,0x00000000)},
  {VT_exponent,22,0xf78fe,DOUBLEWITHTWODWORDINTREE(0x02800000,0x00000000)},
  {VT_exponent,22,0xf78ff,DOUBLEWITHTWODWORDINTREE(0x02900000,0x00000000)},
  {VT_exponent,22,0x3a8300,DOUBLEWITHTWODWORDINTREE(0x02a00000,0x00000000)},
  {VT_exponent,22,0x3a8301,DOUBLEWITHTWODWORDINTREE(0x02b00000,0x00000000)},
  {VT_exponent,22,0x3a8302,DOUBLEWITHTWODWORDINTREE(0x02c00000,0x00000000)},
  {VT_exponent,22,0x3a8303,DOUBLEWITHTWODWORDINTREE(0x02d00000,0x00000000)},
  {VT_exponent,22,0x3a8304,DOUBLEWITHTWODWORDINTREE(0x02e00000,0x00000000)},
  {VT_exponent,22,0x3a8305,DOUBLEWITHTWODWORDINTREE(0x02f00000,0x00000000)},
  {VT_exponent,22,0x3a8306,DOUBLEWITHTWODWORDINTREE(0x03000000,0x00000000)},
  {VT_exponent,22,0x3a8307,DOUBLEWITHTWODWORDINTREE(0x03100000,0x00000000)},
  {VT_exponent,22,0x3a8308,DOUBLEWITHTWODWORDINTREE(0x03200000,0x00000000)},
  {VT_exponent,22,0x3a8309,DOUBLEWITHTWODWORDINTREE(0x03300000,0x00000000)},
  {VT_exponent,22,0x3a830a,DOUBLEWITHTWODWORDINTREE(0x03400000,0x00000000)},
  {VT_exponent,22,0x3a830b,DOUBLEWITHTWODWORDINTREE(0x03500000,0x00000000)},
  {VT_exponent,22,0x3a830c,DOUBLEWITHTWODWORDINTREE(0x03600000,0x00000000)},
  {VT_exponent,22,0x3a830d,DOUBLEWITHTWODWORDINTREE(0x03700000,0x00000000)},
  {VT_exponent,22,0x3a830e,DOUBLEWITHTWODWORDINTREE(0x03800000,0x00000000)},
  {VT_exponent,22,0x3a830f,DOUBLEWITHTWODWORDINTREE(0x03900000,0x00000000)},
  {VT_exponent,22,0x3a8310,DOUBLEWITHTWODWORDINTREE(0x03a00000,0x00000000)},
  {VT_exponent,22,0x3a8311,DOUBLEWITHTWODWORDINTREE(0x03b00000,0x00000000)},
  {VT_exponent,22,0x3a8312,DOUBLEWITHTWODWORDINTREE(0x03c00000,0x00000000)},
  {VT_exponent,22,0x3a8313,DOUBLEWITHTWODWORDINTREE(0x03d00000,0x00000000)},
  {VT_exponent,22,0x3a8314,DOUBLEWITHTWODWORDINTREE(0x03e00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8315,DOUBLEWITHTWODWORDINTREE(0x03f00000,0x00000000)},
  {VT_exponent,22,0x3a8316,DOUBLEWITHTWODWORDINTREE(0x04000000,0x00000000)},
  {VT_exponent,22,0x3a8317,DOUBLEWITHTWODWORDINTREE(0x04100000,0x00000000)},
  {VT_exponent,22,0x3a8318,DOUBLEWITHTWODWORDINTREE(0x04200000,0x00000000)},
  {VT_exponent,22,0x3a8319,DOUBLEWITHTWODWORDINTREE(0x04300000,0x00000000)},
  {VT_exponent,22,0x3a831a,DOUBLEWITHTWODWORDINTREE(0x04400000,0x00000000)},
  {VT_exponent,22,0x3a831b,DOUBLEWITHTWODWORDINTREE(0x04500000,0x00000000)},
  {VT_exponent,22,0x3a831c,DOUBLEWITHTWODWORDINTREE(0x04600000,0x00000000)},
  {VT_exponent,22,0x3a831d,DOUBLEWITHTWODWORDINTREE(0x04700000,0x00000000)},
  {VT_exponent,22,0x3a831e,DOUBLEWITHTWODWORDINTREE(0x04800000,0x00000000)},
  {VT_exponent,22,0x3a831f,DOUBLEWITHTWODWORDINTREE(0x04900000,0x00000000)},
  {VT_exponent,22,0x3a8320,DOUBLEWITHTWODWORDINTREE(0x04a00000,0x00000000)},
  {VT_exponent,22,0x3a8321,DOUBLEWITHTWODWORDINTREE(0x04b00000,0x00000000)},
  {VT_exponent,22,0x3a8322,DOUBLEWITHTWODWORDINTREE(0x04c00000,0x00000000)},
  {VT_exponent,22,0x3a8323,DOUBLEWITHTWODWORDINTREE(0x04d00000,0x00000000)},
  {VT_exponent,22,0x3a8324,DOUBLEWITHTWODWORDINTREE(0x04e00000,0x00000000)},
  {VT_exponent,22,0x3a8325,DOUBLEWITHTWODWORDINTREE(0x04f00000,0x00000000)},
  {VT_exponent,22,0x3a8326,DOUBLEWITHTWODWORDINTREE(0x05000000,0x00000000)},
  {VT_exponent,22,0x3a8327,DOUBLEWITHTWODWORDINTREE(0x05100000,0x00000000)},
  {VT_exponent,22,0x3a8328,DOUBLEWITHTWODWORDINTREE(0x05200000,0x00000000)},
  {VT_exponent,22,0x3a8329,DOUBLEWITHTWODWORDINTREE(0x05300000,0x00000000)},
  {VT_exponent,22,0x3a832a,DOUBLEWITHTWODWORDINTREE(0x05400000,0x00000000)},
  {VT_exponent,22,0x3a832b,DOUBLEWITHTWODWORDINTREE(0x05500000,0x00000000)},
  {VT_exponent,22,0x3a832c,DOUBLEWITHTWODWORDINTREE(0x05600000,0x00000000)},
  {VT_exponent,22,0x3a832d,DOUBLEWITHTWODWORDINTREE(0x05700000,0x00000000)},
  {VT_exponent,22,0x3a832e,DOUBLEWITHTWODWORDINTREE(0x05800000,0x00000000)},
  {VT_exponent,22,0x3a832f,DOUBLEWITHTWODWORDINTREE(0x05900000,0x00000000)},
  {VT_exponent,22,0x3a8330,DOUBLEWITHTWODWORDINTREE(0x05a00000,0x00000000)},
  {VT_exponent,22,0x3a8331,DOUBLEWITHTWODWORDINTREE(0x05b00000,0x00000000)},
  {VT_exponent,22,0x3a8332,DOUBLEWITHTWODWORDINTREE(0x05c00000,0x00000000)},
  {VT_exponent,22,0x3a8333,DOUBLEWITHTWODWORDINTREE(0x05d00000,0x00000000)},
  {VT_exponent,22,0x3a8334,DOUBLEWITHTWODWORDINTREE(0x05e00000,0x00000000)},
  {VT_exponent,22,0x3a8335,DOUBLEWITHTWODWORDINTREE(0x05f00000,0x00000000)},
  {VT_exponent,22,0x3a8336,DOUBLEWITHTWODWORDINTREE(0x06000000,0x00000000)},
  {VT_exponent,22,0x3a8337,DOUBLEWITHTWODWORDINTREE(0x06100000,0x00000000)},
  {VT_exponent,22,0x3a8338,DOUBLEWITHTWODWORDINTREE(0x06200000,0x00000000)},
  {VT_exponent,22,0x3a8339,DOUBLEWITHTWODWORDINTREE(0x06300000,0x00000000)},
  {VT_exponent,22,0x3a833a,DOUBLEWITHTWODWORDINTREE(0x06400000,0x00000000)},
  {VT_exponent,22,0x3a833b,DOUBLEWITHTWODWORDINTREE(0x06500000,0x00000000)},
  {VT_exponent,22,0x3a833c,DOUBLEWITHTWODWORDINTREE(0x06600000,0x00000000)},
  {VT_exponent,22,0x3a833d,DOUBLEWITHTWODWORDINTREE(0x06700000,0x00000000)},
  {VT_exponent,22,0x3a833e,DOUBLEWITHTWODWORDINTREE(0x06800000,0x00000000)},
  {VT_exponent,22,0x3a833f,DOUBLEWITHTWODWORDINTREE(0x06900000,0x00000000)},
  {VT_exponent,22,0x3a8340,DOUBLEWITHTWODWORDINTREE(0x06a00000,0x00000000)},
  {VT_exponent,22,0x3a8341,DOUBLEWITHTWODWORDINTREE(0x06b00000,0x00000000)},
  {VT_exponent,22,0x3a8342,DOUBLEWITHTWODWORDINTREE(0x06c00000,0x00000000)},
  {VT_exponent,22,0x3a8343,DOUBLEWITHTWODWORDINTREE(0x06d00000,0x00000000)},
  {VT_exponent,22,0x3a8344,DOUBLEWITHTWODWORDINTREE(0x06e00000,0x00000000)},
  {VT_exponent,22,0x3a8345,DOUBLEWITHTWODWORDINTREE(0x06f00000,0x00000000)},
  {VT_exponent,22,0x3a8346,DOUBLEWITHTWODWORDINTREE(0x07000000,0x00000000)},
  {VT_exponent,22,0x3a8347,DOUBLEWITHTWODWORDINTREE(0x07100000,0x00000000)},
  {VT_exponent,22,0x3a8348,DOUBLEWITHTWODWORDINTREE(0x07200000,0x00000000)},
  {VT_exponent,22,0x3a8349,DOUBLEWITHTWODWORDINTREE(0x07300000,0x00000000)},
  {VT_exponent,22,0x3a834a,DOUBLEWITHTWODWORDINTREE(0x07400000,0x00000000)},
  {VT_exponent,22,0x3a834b,DOUBLEWITHTWODWORDINTREE(0x07500000,0x00000000)},
  {VT_exponent,22,0x3a834c,DOUBLEWITHTWODWORDINTREE(0x07600000,0x00000000)},
  {VT_exponent,22,0x3a834d,DOUBLEWITHTWODWORDINTREE(0x07700000,0x00000000)},
  {VT_exponent,22,0x3a834e,DOUBLEWITHTWODWORDINTREE(0x07800000,0x00000000)},
  {VT_exponent,22,0x3a834f,DOUBLEWITHTWODWORDINTREE(0x07900000,0x00000000)},
  {VT_exponent,22,0x3a8350,DOUBLEWITHTWODWORDINTREE(0x07a00000,0x00000000)},
  {VT_exponent,22,0x3a8351,DOUBLEWITHTWODWORDINTREE(0x07b00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8352,DOUBLEWITHTWODWORDINTREE(0x07c00000,0x00000000)},
  {VT_exponent,22,0x3a8353,DOUBLEWITHTWODWORDINTREE(0x07d00000,0x00000000)},
  {VT_exponent,22,0x3a8354,DOUBLEWITHTWODWORDINTREE(0x07e00000,0x00000000)},
  {VT_exponent,22,0x3a8355,DOUBLEWITHTWODWORDINTREE(0x07f00000,0x00000000)},
  {VT_exponent,22,0x3a8356,DOUBLEWITHTWODWORDINTREE(0x08000000,0x00000000)},
  {VT_exponent,22,0x3a8357,DOUBLEWITHTWODWORDINTREE(0x08100000,0x00000000)},
  {VT_exponent,22,0x3a8358,DOUBLEWITHTWODWORDINTREE(0x08200000,0x00000000)},
  {VT_exponent,22,0x3a8359,DOUBLEWITHTWODWORDINTREE(0x08300000,0x00000000)},
  {VT_exponent,22,0x3a835a,DOUBLEWITHTWODWORDINTREE(0x08400000,0x00000000)},
  {VT_exponent,22,0x3a835b,DOUBLEWITHTWODWORDINTREE(0x08500000,0x00000000)},
  {VT_exponent,22,0x3a835c,DOUBLEWITHTWODWORDINTREE(0x08600000,0x00000000)},
  {VT_exponent,22,0x3a835d,DOUBLEWITHTWODWORDINTREE(0x08700000,0x00000000)},
  {VT_exponent,22,0x3a835e,DOUBLEWITHTWODWORDINTREE(0x08800000,0x00000000)},
  {VT_exponent,22,0x3a835f,DOUBLEWITHTWODWORDINTREE(0x08900000,0x00000000)},
  {VT_exponent,22,0x3a8360,DOUBLEWITHTWODWORDINTREE(0x08a00000,0x00000000)},
  {VT_exponent,22,0x3a8361,DOUBLEWITHTWODWORDINTREE(0x08b00000,0x00000000)},
  {VT_exponent,22,0x3a8362,DOUBLEWITHTWODWORDINTREE(0x08c00000,0x00000000)},
  {VT_exponent,22,0x3a8363,DOUBLEWITHTWODWORDINTREE(0x08d00000,0x00000000)},
  {VT_exponent,22,0x3a8364,DOUBLEWITHTWODWORDINTREE(0x08e00000,0x00000000)},
  {VT_exponent,22,0x3a8365,DOUBLEWITHTWODWORDINTREE(0x08f00000,0x00000000)},
  {VT_exponent,22,0x3a8366,DOUBLEWITHTWODWORDINTREE(0x09000000,0x00000000)},
  {VT_exponent,22,0x3a8367,DOUBLEWITHTWODWORDINTREE(0x09100000,0x00000000)},
  {VT_exponent,22,0x3a8368,DOUBLEWITHTWODWORDINTREE(0x09200000,0x00000000)},
  {VT_exponent,22,0x3a8369,DOUBLEWITHTWODWORDINTREE(0x09300000,0x00000000)},
  {VT_exponent,22,0x3a836a,DOUBLEWITHTWODWORDINTREE(0x09400000,0x00000000)},
  {VT_exponent,22,0x3a836b,DOUBLEWITHTWODWORDINTREE(0x09500000,0x00000000)},
  {VT_exponent,22,0x3a836c,DOUBLEWITHTWODWORDINTREE(0x09600000,0x00000000)},
  {VT_exponent,22,0x3a836d,DOUBLEWITHTWODWORDINTREE(0x09700000,0x00000000)},
  {VT_exponent,22,0x3a836e,DOUBLEWITHTWODWORDINTREE(0x09800000,0x00000000)},
  {VT_exponent,22,0x3a836f,DOUBLEWITHTWODWORDINTREE(0x09900000,0x00000000)},
  {VT_exponent,22,0x3a8370,DOUBLEWITHTWODWORDINTREE(0x09a00000,0x00000000)},
  {VT_exponent,22,0x3a8371,DOUBLEWITHTWODWORDINTREE(0x09b00000,0x00000000)},
  {VT_exponent,22,0x3a8372,DOUBLEWITHTWODWORDINTREE(0x09c00000,0x00000000)},
  {VT_exponent,22,0x3a8373,DOUBLEWITHTWODWORDINTREE(0x09d00000,0x00000000)},
  {VT_exponent,22,0x3a8374,DOUBLEWITHTWODWORDINTREE(0x09e00000,0x00000000)},
  {VT_exponent,22,0x3a8375,DOUBLEWITHTWODWORDINTREE(0x09f00000,0x00000000)},
  {VT_exponent,22,0x3a8376,DOUBLEWITHTWODWORDINTREE(0x0a000000,0x00000000)},
  {VT_exponent,22,0x3a8377,DOUBLEWITHTWODWORDINTREE(0x0a100000,0x00000000)},
  {VT_exponent,22,0x3a8378,DOUBLEWITHTWODWORDINTREE(0x0a200000,0x00000000)},
  {VT_exponent,22,0x3a8379,DOUBLEWITHTWODWORDINTREE(0x0a300000,0x00000000)},
  {VT_exponent,22,0x3a837a,DOUBLEWITHTWODWORDINTREE(0x0a400000,0x00000000)},
  {VT_exponent,22,0x3a837b,DOUBLEWITHTWODWORDINTREE(0x0a500000,0x00000000)},
  {VT_exponent,22,0x3a837c,DOUBLEWITHTWODWORDINTREE(0x0a600000,0x00000000)},
  {VT_exponent,22,0x3a837d,DOUBLEWITHTWODWORDINTREE(0x0a700000,0x00000000)},
  {VT_exponent,22,0x3a837e,DOUBLEWITHTWODWORDINTREE(0x0a800000,0x00000000)},
  {VT_exponent,22,0x3a837f,DOUBLEWITHTWODWORDINTREE(0x0a900000,0x00000000)},
  {VT_exponent,22,0x3a8380,DOUBLEWITHTWODWORDINTREE(0x0aa00000,0x00000000)},
  {VT_exponent,22,0x3a8381,DOUBLEWITHTWODWORDINTREE(0x0ab00000,0x00000000)},
  {VT_exponent,22,0x3a8382,DOUBLEWITHTWODWORDINTREE(0x0ac00000,0x00000000)},
  {VT_exponent,22,0x3a8383,DOUBLEWITHTWODWORDINTREE(0x0ad00000,0x00000000)},
  {VT_exponent,22,0x3a8384,DOUBLEWITHTWODWORDINTREE(0x0ae00000,0x00000000)},
  {VT_exponent,22,0x3a8385,DOUBLEWITHTWODWORDINTREE(0x0af00000,0x00000000)},
  {VT_exponent,22,0x3a8386,DOUBLEWITHTWODWORDINTREE(0x0b000000,0x00000000)},
  {VT_exponent,22,0x3a8387,DOUBLEWITHTWODWORDINTREE(0x0b100000,0x00000000)},
  {VT_exponent,22,0x3a8388,DOUBLEWITHTWODWORDINTREE(0x0b200000,0x00000000)},
  {VT_exponent,22,0x3a8389,DOUBLEWITHTWODWORDINTREE(0x0b300000,0x00000000)},
  {VT_exponent,22,0x3a838a,DOUBLEWITHTWODWORDINTREE(0x0b400000,0x00000000)},
  {VT_exponent,22,0x3a838b,DOUBLEWITHTWODWORDINTREE(0x0b500000,0x00000000)},
  {VT_exponent,22,0x3a838c,DOUBLEWITHTWODWORDINTREE(0x0b600000,0x00000000)},
  {VT_exponent,22,0x3a838d,DOUBLEWITHTWODWORDINTREE(0x0b700000,0x00000000)},
  {VT_exponent,22,0x3a838e,DOUBLEWITHTWODWORDINTREE(0x0b800000,0x00000000)},
```

```
  {VT_exponent,22,0x3a838f,DOUBLEWITHTWODWORDINTREE(0x0b900000,0x00000000)},
  {VT_exponent,22,0x3a8390,DOUBLEWITHTWODWORDINTREE(0x0ba00000,0x00000000)},
  {VT_exponent,22,0x3a8391,DOUBLEWITHTWODWORDINTREE(0x0bb00000,0x00000000)},
  {VT_exponent,22,0x3a8392,DOUBLEWITHTWODWORDINTREE(0x0bc00000,0x00000000)},
  {VT_exponent,22,0x3a8393,DOUBLEWITHTWODWORDINTREE(0x0bd00000,0x00000000)},
  {VT_exponent,22,0x3a8394,DOUBLEWITHTWODWORDINTREE(0x0be00000,0x00000000)},
  {VT_exponent,22,0x3a8395,DOUBLEWITHTWODWORDINTREE(0x0bf00000,0x00000000)},
  {VT_exponent,22,0x3a8396,DOUBLEWITHTWODWORDINTREE(0x0c000000,0x00000000)},
  {VT_exponent,22,0x3a8397,DOUBLEWITHTWODWORDINTREE(0x0c100000,0x00000000)},
  {VT_exponent,22,0x3a8398,DOUBLEWITHTWODWORDINTREE(0x0c200000,0x00000000)},
  {VT_exponent,22,0x3a8399,DOUBLEWITHTWODWORDINTREE(0x0c300000,0x00000000)},
  {VT_exponent,22,0x3a839a,DOUBLEWITHTWODWORDINTREE(0x0c400000,0x00000000)},
  {VT_exponent,22,0x3a839b,DOUBLEWITHTWODWORDINTREE(0x0c500000,0x00000000)},
  {VT_exponent,22,0x3a839c,DOUBLEWITHTWODWORDINTREE(0x0c600000,0x00000000)},
  {VT_exponent,22,0x3a839d,DOUBLEWITHTWODWORDINTREE(0x0c700000,0x00000000)},
  {VT_exponent,22,0x3a839e,DOUBLEWITHTWODWORDINTREE(0x0c800000,0x00000000)},
  {VT_exponent,22,0x3a839f,DOUBLEWITHTWODWORDINTREE(0x0c900000,0x00000000)},
  {VT_exponent,22,0x3a83a0,DOUBLEWITHTWODWORDINTREE(0x0ca00000,0x00000000)},
  {VT_exponent,22,0x3a83a1,DOUBLEWITHTWODWORDINTREE(0x0cb00000,0x00000000)},
  {VT_exponent,22,0x3a83a2,DOUBLEWITHTWODWORDINTREE(0x0cc00000,0x00000000)},
  {VT_exponent,22,0x3a83a3,DOUBLEWITHTWODWORDINTREE(0x0cd00000,0x00000000)},
  {VT_exponent,22,0x3a83a4,DOUBLEWITHTWODWORDINTREE(0x0ce00000,0x00000000)},
  {VT_exponent,22,0x3a83a5,DOUBLEWITHTWODWORDINTREE(0x0cf00000,0x00000000)},
  {VT_exponent,22,0x3a83a6,DOUBLEWITHTWODWORDINTREE(0x0d000000,0x00000000)},
  {VT_exponent,22,0x3a83a7,DOUBLEWITHTWODWORDINTREE(0x0d100000,0x00000000)},
  {VT_exponent,22,0x3a83a8,DOUBLEWITHTWODWORDINTREE(0x0d200000,0x00000000)},
  {VT_exponent,22,0x3a83a9,DOUBLEWITHTWODWORDINTREE(0x0d300000,0x00000000)},
  {VT_exponent,22,0x3a83aa,DOUBLEWITHTWODWORDINTREE(0x0d400000,0x00000000)},
  {VT_exponent,22,0x3a83ab,DOUBLEWITHTWODWORDINTREE(0x0d500000,0x00000000)},
  {VT_exponent,22,0x3a83ac,DOUBLEWITHTWODWORDINTREE(0x0d600000,0x00000000)},
  {VT_exponent,22,0x3a83ad,DOUBLEWITHTWODWORDINTREE(0x0d700000,0x00000000)},
  {VT_exponent,22,0x3a83ae,DOUBLEWITHTWODWORDINTREE(0x0d800000,0x00000000)},
  {VT_exponent,22,0x3a83af,DOUBLEWITHTWODWORDINTREE(0x0d900000,0x00000000)},
  {VT_exponent,22,0x3a83b0,DOUBLEWITHTWODWORDINTREE(0x0da00000,0x00000000)},
  {VT_exponent,22,0x3a83b1,DOUBLEWITHTWODWORDINTREE(0x0db00000,0x00000000)},
  {VT_exponent,22,0x3a83b2,DOUBLEWITHTWODWORDINTREE(0x0dc00000,0x00000000)},
  {VT_exponent,22,0x3a83b3,DOUBLEWITHTWODWORDINTREE(0x0dd00000,0x00000000)},
  {VT_exponent,22,0x3a83b4,DOUBLEWITHTWODWORDINTREE(0x0de00000,0x00000000)},
  {VT_exponent,22,0x3a83b5,DOUBLEWITHTWODWORDINTREE(0x0df00000,0x00000000)},
  {VT_exponent,22,0x3a83b6,DOUBLEWITHTWODWORDINTREE(0x0e000000,0x00000000)},
  {VT_exponent,22,0x3a83b7,DOUBLEWITHTWODWORDINTREE(0x0e100000,0x00000000)},
  {VT_exponent,22,0x3a83b8,DOUBLEWITHTWODWORDINTREE(0x0e200000,0x00000000)},
  {VT_exponent,22,0x3a83b9,DOUBLEWITHTWODWORDINTREE(0x0e300000,0x00000000)},
  {VT_exponent,22,0x3a83ba,DOUBLEWITHTWODWORDINTREE(0x0e400000,0x00000000)},
  {VT_exponent,22,0x3a83bb,DOUBLEWITHTWODWORDINTREE(0x0e500000,0x00000000)},
  {VT_exponent,22,0x3a83bc,DOUBLEWITHTWODWORDINTREE(0x0e600000,0x00000000)},
  {VT_exponent,22,0x3a83bd,DOUBLEWITHTWODWORDINTREE(0x0e700000,0x00000000)},
  {VT_exponent,22,0x3a83be,DOUBLEWITHTWODWORDINTREE(0x0e800000,0x00000000)},
  {VT_exponent,22,0x3a83bf,DOUBLEWITHTWODWORDINTREE(0x0e900000,0x00000000)},
  {VT_exponent,22,0x3a83c0,DOUBLEWITHTWODWORDINTREE(0x0ea00000,0x00000000)},
  {VT_exponent,22,0x3a83c1,DOUBLEWITHTWODWORDINTREE(0x0eb00000,0x00000000)},
  {VT_exponent,22,0x3a83c2,DOUBLEWITHTWODWORDINTREE(0x0ec00000,0x00000000)},
  {VT_exponent,22,0x3a83c3,DOUBLEWITHTWODWORDINTREE(0x0ed00000,0x00000000)},
  {VT_exponent,22,0x3a83c4,DOUBLEWITHTWODWORDINTREE(0x0ee00000,0x00000000)},
  {VT_exponent,22,0x3a83c5,DOUBLEWITHTWODWORDINTREE(0x0ef00000,0x00000000)},
  {VT_exponent,22,0x3a83c6,DOUBLEWITHTWODWORDINTREE(0x0f000000,0x00000000)},
  {VT_exponent,22,0x3a83c7,DOUBLEWITHTWODWORDINTREE(0x0f100000,0x00000000)},
  {VT_exponent,22,0x3a83c8,DOUBLEWITHTWODWORDINTREE(0x0f200000,0x00000000)},
  {VT_exponent,22,0x3a83c9,DOUBLEWITHTWODWORDINTREE(0x0f300000,0x00000000)},
  {VT_exponent,22,0x3a83ca,DOUBLEWITHTWODWORDINTREE(0x0f400000,0x00000000)},
  {VT_exponent,22,0x3a83cb,DOUBLEWITHTWODWORDINTREE(0x0f500000,0x00000000)},
```

```
  {VT_exponent,22,0x3a83cc,DOUBLEWITHTWODWORDINTREE(0x0f600000,0x00000000)},
  {VT_exponent,22,0x3a83cd,DOUBLEWITHTWODWORDINTREE(0x0f700000,0x00000000)},
  {VT_exponent,22,0x3a83ce,DOUBLEWITHTWODWORDINTREE(0x0f800000,0x00000000)},
  {VT_exponent,22,0x3a83cf,DOUBLEWITHTWODWORDINTREE(0x0f900000,0x00000000)},
  {VT_exponent,22,0x3a83d0,DOUBLEWITHTWODWORDINTREE(0x0fa00000,0x00000000)},
  {VT_exponent,22,0x3a83d1,DOUBLEWITHTWODWORDINTREE(0x0fb00000,0x00000000)},
  {VT_exponent,22,0x3a83d2,DOUBLEWITHTWODWORDINTREE(0x0fc00000,0x00000000)},
  {VT_exponent,22,0x3a83d3,DOUBLEWITHTWODWORDINTREE(0x0fd00000,0x00000000)},
  {VT_exponent,22,0x3a83d4,DOUBLEWITHTWODWORDINTREE(0x0fe00000,0x00000000)},
  {VT_exponent,22,0x3a83d5,DOUBLEWITHTWODWORDINTREE(0x0ff00000,0x00000000)},
  {VT_exponent,22,0x3a83d6,DOUBLEWITHTWODWORDINTREE(0x10000000,0x00000000)},
  {VT_exponent,22,0x3a83d7,DOUBLEWITHTWODWORDINTREE(0x10100000,0x00000000)},
  {VT_exponent,22,0x3a83d8,DOUBLEWITHTWODWORDINTREE(0x10200000,0x00000000)},
  {VT_exponent,22,0x3a83d9,DOUBLEWITHTWODWORDINTREE(0x10300000,0x00000000)},
  {VT_exponent,22,0x3a83da,DOUBLEWITHTWODWORDINTREE(0x10400000,0x00000000)},
  {VT_exponent,22,0x3a83db,DOUBLEWITHTWODWORDINTREE(0x10500000,0x00000000)},
  {VT_exponent,22,0x3a83dc,DOUBLEWITHTWODWORDINTREE(0x10600000,0x00000000)},
  {VT_exponent,22,0x3a83dd,DOUBLEWITHTWODWORDINTREE(0x10700000,0x00000000)},
  {VT_exponent,22,0x3a83de,DOUBLEWITHTWODWORDINTREE(0x10800000,0x00000000)},
  {VT_exponent,22,0x3a83df,DOUBLEWITHTWODWORDINTREE(0x10900000,0x00000000)},
  {VT_exponent,22,0x3a83e0,DOUBLEWITHTWODWORDINTREE(0x10a00000,0x00000000)},
  {VT_exponent,22,0x3a83e1,DOUBLEWITHTWODWORDINTREE(0x10b00000,0x00000000)},
  {VT_exponent,22,0x3a83e2,DOUBLEWITHTWODWORDINTREE(0x10c00000,0x00000000)},
  {VT_exponent,22,0x3a83e3,DOUBLEWITHTWODWORDINTREE(0x10d00000,0x00000000)},
  {VT_exponent,22,0x3a83e4,DOUBLEWITHTWODWORDINTREE(0x10e00000,0x00000000)},
  {VT_exponent,22,0x3a83e5,DOUBLEWITHTWODWORDINTREE(0x10f00000,0x00000000)},
  {VT_exponent,22,0x3a83e6,DOUBLEWITHTWODWORDINTREE(0x11000000,0x00000000)},
  {VT_exponent,22,0x3a83e7,DOUBLEWITHTWODWORDINTREE(0x11100000,0x00000000)},
  {VT_exponent,22,0x3a83e8,DOUBLEWITHTWODWORDINTREE(0x11200000,0x00000000)},
  {VT_exponent,22,0x3a83e9,DOUBLEWITHTWODWORDINTREE(0x11300000,0x00000000)},
  {VT_exponent,22,0x3a83ea,DOUBLEWITHTWODWORDINTREE(0x11400000,0x00000000)},
  {VT_exponent,22,0x3a83eb,DOUBLEWITHTWODWORDINTREE(0x11500000,0x00000000)},
  {VT_exponent,22,0x3a83ec,DOUBLEWITHTWODWORDINTREE(0x11600000,0x00000000)},
  {VT_exponent,22,0x3a83ed,DOUBLEWITHTWODWORDINTREE(0x11700000,0x00000000)},
  {VT_exponent,22,0x3a83ee,DOUBLEWITHTWODWORDINTREE(0x11800000,0x00000000)},
  {VT_exponent,22,0x3a83ef,DOUBLEWITHTWODWORDINTREE(0x11900000,0x00000000)},
  {VT_exponent,22,0x3a83f0,DOUBLEWITHTWODWORDINTREE(0x11a00000,0x00000000)},
  {VT_exponent,22,0x3a83f1,DOUBLEWITHTWODWORDINTREE(0x11b00000,0x00000000)},
  {VT_exponent,22,0x3a83f2,DOUBLEWITHTWODWORDINTREE(0x11c00000,0x00000000)},
  {VT_exponent,22,0x3a83f3,DOUBLEWITHTWODWORDINTREE(0x11d00000,0x00000000)},
  {VT_exponent,22,0x3a83f4,DOUBLEWITHTWODWORDINTREE(0x11e00000,0x00000000)},
  {VT_exponent,22,0x3a83f5,DOUBLEWITHTWODWORDINTREE(0x11f00000,0x00000000)},
  {VT_exponent,22,0x3a83f6,DOUBLEWITHTWODWORDINTREE(0x12000000,0x00000000)},
  {VT_exponent,22,0x3a83f7,DOUBLEWITHTWODWORDINTREE(0x12100000,0x00000000)},
  {VT_exponent,22,0x3a83f8,DOUBLEWITHTWODWORDINTREE(0x12200000,0x00000000)},
  {VT_exponent,22,0x3a83f9,DOUBLEWITHTWODWORDINTREE(0x12300000,0x00000000)},
  {VT_exponent,22,0x3a83fa,DOUBLEWITHTWODWORDINTREE(0x12400000,0x00000000)},
  {VT_exponent,22,0x3a83fb,DOUBLEWITHTWODWORDINTREE(0x12500000,0x00000000)},
  {VT_exponent,22,0x3a83fc,DOUBLEWITHTWODWORDINTREE(0x12600000,0x00000000)},
  {VT_exponent,22,0x3a83fd,DOUBLEWITHTWODWORDINTREE(0x12700000,0x00000000)},
  {VT_exponent,22,0x3a83fe,DOUBLEWITHTWODWORDINTREE(0x12800000,0x00000000)},
  {VT_exponent,22,0x3a83ff,DOUBLEWITHTWODWORDINTREE(0x12900000,0x00000000)},
  {VT_exponent,22,0x3a8400,DOUBLEWITHTWODWORDINTREE(0x12a00000,0x00000000)},
  {VT_exponent,22,0x3a8401,DOUBLEWITHTWODWORDINTREE(0x12b00000,0x00000000)},
  {VT_exponent,22,0x3a8402,DOUBLEWITHTWODWORDINTREE(0x12c00000,0x00000000)},
  {VT_exponent,22,0x3a8403,DOUBLEWITHTWODWORDINTREE(0x12d00000,0x00000000)},
  {VT_exponent,22,0x3a8404,DOUBLEWITHTWODWORDINTREE(0x12e00000,0x00000000)},
  {VT_exponent,22,0x3a8405,DOUBLEWITHTWODWORDINTREE(0x12f00000,0x00000000)},
  {VT_exponent,22,0x3a8406,DOUBLEWITHTWODWORDINTREE(0x13000000,0x00000000)},
  {VT_exponent,22,0x3a8407,DOUBLEWITHTWODWORDINTREE(0x13100000,0x00000000)},
  {VT_exponent,22,0x3a8408,DOUBLEWITHTWODWORDINTREE(0x13200000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8409,DOUBLEWITHTWODWORDINTREE(0x13300000,0x00000000)},
  {VT_exponent,22,0x3a840a,DOUBLEWITHTWODWORDINTREE(0x13400000,0x00000000)},
  {VT_exponent,22,0x3a840b,DOUBLEWITHTWODWORDINTREE(0x13500000,0x00000000)},
  {VT_exponent,22,0x3a840c,DOUBLEWITHTWODWORDINTREE(0x13600000,0x00000000)},
  {VT_exponent,22,0x3a840d,DOUBLEWITHTWODWORDINTREE(0x13700000,0x00000000)},
  {VT_exponent,22,0x3a840e,DOUBLEWITHTWODWORDINTREE(0x13800000,0x00000000)},
  {VT_exponent,22,0x3a840f,DOUBLEWITHTWODWORDINTREE(0x13900000,0x00000000)},
  {VT_exponent,22,0x3a8410,DOUBLEWITHTWODWORDINTREE(0x13a00000,0x00000000)},
  {VT_exponent,22,0x3a8411,DOUBLEWITHTWODWORDINTREE(0x13b00000,0x00000000)},
  {VT_exponent,22,0x3a8412,DOUBLEWITHTWODWORDINTREE(0x13c00000,0x00000000)},
  {VT_exponent,22,0x3a8413,DOUBLEWITHTWODWORDINTREE(0x13d00000,0x00000000)},
  {VT_exponent,22,0x3a8414,DOUBLEWITHTWODWORDINTREE(0x13e00000,0x00000000)},
  {VT_exponent,22,0x3a8415,DOUBLEWITHTWODWORDINTREE(0x13f00000,0x00000000)},
  {VT_exponent,22,0x3a8416,DOUBLEWITHTWODWORDINTREE(0x14000000,0x00000000)},
  {VT_exponent,22,0x3a8417,DOUBLEWITHTWODWORDINTREE(0x14100000,0x00000000)},
  {VT_exponent,22,0x3a8418,DOUBLEWITHTWODWORDINTREE(0x14200000,0x00000000)},
  {VT_exponent,22,0x3a8419,DOUBLEWITHTWODWORDINTREE(0x14300000,0x00000000)},
  {VT_exponent,22,0x3a841a,DOUBLEWITHTWODWORDINTREE(0x14400000,0x00000000)},
  {VT_exponent,22,0x3a841b,DOUBLEWITHTWODWORDINTREE(0x14500000,0x00000000)},
  {VT_exponent,22,0x3a841c,DOUBLEWITHTWODWORDINTREE(0x14600000,0x00000000)},
  {VT_exponent,22,0x3a841d,DOUBLEWITHTWODWORDINTREE(0x14700000,0x00000000)},
  {VT_exponent,22,0x3a841e,DOUBLEWITHTWODWORDINTREE(0x14800000,0x00000000)},
  {VT_exponent,22,0x3a841f,DOUBLEWITHTWODWORDINTREE(0x14900000,0x00000000)},
  {VT_exponent,22,0x3a8420,DOUBLEWITHTWODWORDINTREE(0x14a00000,0x00000000)},
  {VT_exponent,22,0x3a8421,DOUBLEWITHTWODWORDINTREE(0x14b00000,0x00000000)},
  {VT_exponent,22,0x3a8422,DOUBLEWITHTWODWORDINTREE(0x14c00000,0x00000000)},
  {VT_exponent,22,0x3a8423,DOUBLEWITHTWODWORDINTREE(0x14d00000,0x00000000)},
  {VT_exponent,22,0x3a8424,DOUBLEWITHTWODWORDINTREE(0x14e00000,0x00000000)},
  {VT_exponent,22,0x3a8425,DOUBLEWITHTWODWORDINTREE(0x14f00000,0x00000000)},
  {VT_exponent,22,0x3a8426,DOUBLEWITHTWODWORDINTREE(0x15000000,0x00000000)},
  {VT_exponent,22,0x3a8427,DOUBLEWITHTWODWORDINTREE(0x15100000,0x00000000)},
  {VT_exponent,22,0x3a8428,DOUBLEWITHTWODWORDINTREE(0x15200000,0x00000000)},
  {VT_exponent,22,0x3a8429,DOUBLEWITHTWODWORDINTREE(0x15300000,0x00000000)},
  {VT_exponent,22,0x3a842a,DOUBLEWITHTWODWORDINTREE(0x15400000,0x00000000)},
  {VT_exponent,22,0x3a842b,DOUBLEWITHTWODWORDINTREE(0x15500000,0x00000000)},
  {VT_exponent,22,0x3a842c,DOUBLEWITHTWODWORDINTREE(0x15600000,0x00000000)},
  {VT_exponent,22,0x3a842d,DOUBLEWITHTWODWORDINTREE(0x15700000,0x00000000)},
  {VT_exponent,22,0x3a842e,DOUBLEWITHTWODWORDINTREE(0x15800000,0x00000000)},
  {VT_exponent,22,0x3a842f,DOUBLEWITHTWODWORDINTREE(0x15900000,0x00000000)},
  {VT_exponent,22,0x3a8430,DOUBLEWITHTWODWORDINTREE(0x15a00000,0x00000000)},
  {VT_exponent,22,0x3a8431,DOUBLEWITHTWODWORDINTREE(0x15b00000,0x00000000)},
  {VT_exponent,22,0x3a8432,DOUBLEWITHTWODWORDINTREE(0x15c00000,0x00000000)},
  {VT_exponent,22,0x3a8433,DOUBLEWITHTWODWORDINTREE(0x15d00000,0x00000000)},
  {VT_exponent,22,0x3a8434,DOUBLEWITHTWODWORDINTREE(0x15e00000,0x00000000)},
  {VT_exponent,22,0x3a8435,DOUBLEWITHTWODWORDINTREE(0x15f00000,0x00000000)},
  {VT_exponent,22,0x3a8436,DOUBLEWITHTWODWORDINTREE(0x16000000,0x00000000)},
  {VT_exponent,22,0x3a8437,DOUBLEWITHTWODWORDINTREE(0x16100000,0x00000000)},
  {VT_exponent,22,0x3a8438,DOUBLEWITHTWODWORDINTREE(0x16200000,0x00000000)},
  {VT_exponent,22,0x3a8439,DOUBLEWITHTWODWORDINTREE(0x16300000,0x00000000)},
  {VT_exponent,22,0x3a843a,DOUBLEWITHTWODWORDINTREE(0x16400000,0x00000000)},
  {VT_exponent,22,0x3a843b,DOUBLEWITHTWODWORDINTREE(0x16500000,0x00000000)},
  {VT_exponent,22,0x3a843c,DOUBLEWITHTWODWORDINTREE(0x16600000,0x00000000)},
  {VT_exponent,22,0x3a843d,DOUBLEWITHTWODWORDINTREE(0x16700000,0x00000000)},
  {VT_exponent,22,0x3a843e,DOUBLEWITHTWODWORDINTREE(0x16800000,0x00000000)},
  {VT_exponent,22,0x3a843f,DOUBLEWITHTWODWORDINTREE(0x16900000,0x00000000)},
  {VT_exponent,22,0x3a8440,DOUBLEWITHTWODWORDINTREE(0x16a00000,0x00000000)},
  {VT_exponent,22,0x3a8441,DOUBLEWITHTWODWORDINTREE(0x16b00000,0x00000000)},
  {VT_exponent,22,0x3a8442,DOUBLEWITHTWODWORDINTREE(0x16c00000,0x00000000)},
  {VT_exponent,22,0x3a8443,DOUBLEWITHTWODWORDINTREE(0x16d00000,0x00000000)},
  {VT_exponent,22,0x3a8444,DOUBLEWITHTWODWORDINTREE(0x16e00000,0x00000000)},
  {VT_exponent,22,0x3a8445,DOUBLEWITHTWODWORDINTREE(0x16f00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8446,DOUBLEWITHTWODWORDINTREE(0x17000000,0x00000000)},
  {VT_exponent,22,0x3a8447,DOUBLEWITHTWODWORDINTREE(0x17100000,0x00000000)},
  {VT_exponent,22,0x3a8448,DOUBLEWITHTWODWORDINTREE(0x17200000,0x00000000)},
  {VT_exponent,22,0x3a8449,DOUBLEWITHTWODWORDINTREE(0x17300000,0x00000000)},
  {VT_exponent,22,0x3a844a,DOUBLEWITHTWODWORDINTREE(0x17400000,0x00000000)},
  {VT_exponent,22,0x3a844b,DOUBLEWITHTWODWORDINTREE(0x17500000,0x00000000)},
  {VT_exponent,22,0x3a844c,DOUBLEWITHTWODWORDINTREE(0x17600000,0x00000000)},
  {VT_exponent,22,0x3a844d,DOUBLEWITHTWODWORDINTREE(0x17700000,0x00000000)},
  {VT_exponent,22,0x3a844e,DOUBLEWITHTWODWORDINTREE(0x17800000,0x00000000)},
  {VT_exponent,22,0x3a844f,DOUBLEWITHTWODWORDINTREE(0x17900000,0x00000000)},
  {VT_exponent,22,0x3a8450,DOUBLEWITHTWODWORDINTREE(0x17a00000,0x00000000)},
  {VT_exponent,22,0x3a8451,DOUBLEWITHTWODWORDINTREE(0x17b00000,0x00000000)},
  {VT_exponent,22,0x3a8452,DOUBLEWITHTWODWORDINTREE(0x17c00000,0x00000000)},
  {VT_exponent,22,0x3a8453,DOUBLEWITHTWODWORDINTREE(0x17d00000,0x00000000)},
  {VT_exponent,22,0x3a8454,DOUBLEWITHTWODWORDINTREE(0x17e00000,0x00000000)},
  {VT_exponent,22,0x3a8455,DOUBLEWITHTWODWORDINTREE(0x17f00000,0x00000000)},
  {VT_exponent,22,0x3a8456,DOUBLEWITHTWODWORDINTREE(0x18000000,0x00000000)},
  {VT_exponent,22,0x3a8457,DOUBLEWITHTWODWORDINTREE(0x18100000,0x00000000)},
  {VT_exponent,22,0x3a8458,DOUBLEWITHTWODWORDINTREE(0x18200000,0x00000000)},
  {VT_exponent,22,0x3a8459,DOUBLEWITHTWODWORDINTREE(0x18300000,0x00000000)},
  {VT_exponent,22,0x3a845a,DOUBLEWITHTWODWORDINTREE(0x18400000,0x00000000)},
  {VT_exponent,22,0x3a845b,DOUBLEWITHTWODWORDINTREE(0x18500000,0x00000000)},
  {VT_exponent,22,0x3a845c,DOUBLEWITHTWODWORDINTREE(0x18600000,0x00000000)},
  {VT_exponent,22,0x3a845d,DOUBLEWITHTWODWORDINTREE(0x18700000,0x00000000)},
  {VT_exponent,22,0x3a845e,DOUBLEWITHTWODWORDINTREE(0x18800000,0x00000000)},
  {VT_exponent,22,0x3a845f,DOUBLEWITHTWODWORDINTREE(0x18900000,0x00000000)},
  {VT_exponent,22,0x3a8460,DOUBLEWITHTWODWORDINTREE(0x18a00000,0x00000000)},
  {VT_exponent,22,0x3a8461,DOUBLEWITHTWODWORDINTREE(0x18b00000,0x00000000)},
  {VT_exponent,22,0x3a8462,DOUBLEWITHTWODWORDINTREE(0x18c00000,0x00000000)},
  {VT_exponent,22,0x3a8463,DOUBLEWITHTWODWORDINTREE(0x18d00000,0x00000000)},
  {VT_exponent,22,0x3a8464,DOUBLEWITHTWODWORDINTREE(0x18e00000,0x00000000)},
  {VT_exponent,22,0x3a8465,DOUBLEWITHTWODWORDINTREE(0x18f00000,0x00000000)},
  {VT_exponent,22,0x3a8466,DOUBLEWITHTWODWORDINTREE(0x19000000,0x00000000)},
  {VT_exponent,22,0x3a8467,DOUBLEWITHTWODWORDINTREE(0x19100000,0x00000000)},
  {VT_exponent,22,0x3a8468,DOUBLEWITHTWODWORDINTREE(0x19200000,0x00000000)},
  {VT_exponent,22,0x3a8469,DOUBLEWITHTWODWORDINTREE(0x19300000,0x00000000)},
  {VT_exponent,22,0x3a846a,DOUBLEWITHTWODWORDINTREE(0x19400000,0x00000000)},
  {VT_exponent,22,0x3a846b,DOUBLEWITHTWODWORDINTREE(0x19500000,0x00000000)},
  {VT_exponent,22,0x3a846c,DOUBLEWITHTWODWORDINTREE(0x19600000,0x00000000)},
  {VT_exponent,22,0x3a846d,DOUBLEWITHTWODWORDINTREE(0x19700000,0x00000000)},
  {VT_exponent,22,0x3a846e,DOUBLEWITHTWODWORDINTREE(0x19800000,0x00000000)},
  {VT_exponent,22,0x3a846f,DOUBLEWITHTWODWORDINTREE(0x19900000,0x00000000)},
  {VT_exponent,22,0x3a8470,DOUBLEWITHTWODWORDINTREE(0x19a00000,0x00000000)},
  {VT_exponent,22,0x3a8471,DOUBLEWITHTWODWORDINTREE(0x19b00000,0x00000000)},
  {VT_exponent,22,0x3a8472,DOUBLEWITHTWODWORDINTREE(0x19c00000,0x00000000)},
  {VT_exponent,22,0x3a8473,DOUBLEWITHTWODWORDINTREE(0x19d00000,0x00000000)},
  {VT_exponent,22,0x3a8474,DOUBLEWITHTWODWORDINTREE(0x19e00000,0x00000000)},
  {VT_exponent,22,0x3a8475,DOUBLEWITHTWODWORDINTREE(0x19f00000,0x00000000)},
  {VT_exponent,22,0x3a8476,DOUBLEWITHTWODWORDINTREE(0x1a000000,0x00000000)},
  {VT_exponent,22,0x3a8477,DOUBLEWITHTWODWORDINTREE(0x1a100000,0x00000000)},
  {VT_exponent,22,0x3a8478,DOUBLEWITHTWODWORDINTREE(0x1a200000,0x00000000)},
  {VT_exponent,22,0x3a8479,DOUBLEWITHTWODWORDINTREE(0x1a300000,0x00000000)},
  {VT_exponent,22,0x3a847a,DOUBLEWITHTWODWORDINTREE(0x1a400000,0x00000000)},
  {VT_exponent,22,0x3a847b,DOUBLEWITHTWODWORDINTREE(0x1a500000,0x00000000)},
  {VT_exponent,22,0x3a847c,DOUBLEWITHTWODWORDINTREE(0x1a600000,0x00000000)},
  {VT_exponent,22,0x3a847d,DOUBLEWITHTWODWORDINTREE(0x1a700000,0x00000000)},
  {VT_exponent,22,0x3a847e,DOUBLEWITHTWODWORDINTREE(0x1a800000,0x00000000)},
  {VT_exponent,22,0x3a847f,DOUBLEWITHTWODWORDINTREE(0x1a900000,0x00000000)},
  {VT_exponent,22,0x3a8480,DOUBLEWITHTWODWORDINTREE(0x1aa00000,0x00000000)},
  {VT_exponent,22,0x3a8481,DOUBLEWITHTWODWORDINTREE(0x1ab00000,0x00000000)},
  {VT_exponent,22,0x3a8482,DOUBLEWITHTWODWORDINTREE(0x1ac00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8483,DOUBLEWITHTWODWORDINTREE(0x1ad00000,0x00000000)},
  {VT_exponent,22,0x3a8484,DOUBLEWITHTWODWORDINTREE(0x1ae00000,0x00000000)},
  {VT_exponent,22,0x3a8485,DOUBLEWITHTWODWORDINTREE(0x1af00000,0x00000000)},
  {VT_exponent,22,0x3a8486,DOUBLEWITHTWODWORDINTREE(0x1b000000,0x00000000)},
  {VT_exponent,22,0x3a8487,DOUBLEWITHTWODWORDINTREE(0x1b100000,0x00000000)},
  {VT_exponent,22,0x3a8488,DOUBLEWITHTWODWORDINTREE(0x1b200000,0x00000000)},
  {VT_exponent,22,0x3a8489,DOUBLEWITHTWODWORDINTREE(0x1b300000,0x00000000)},
  {VT_exponent,22,0x3a848a,DOUBLEWITHTWODWORDINTREE(0x1b400000,0x00000000)},
  {VT_exponent,22,0x3a848b,DOUBLEWITHTWODWORDINTREE(0x1b500000,0x00000000)},
  {VT_exponent,22,0x3a848c,DOUBLEWITHTWODWORDINTREE(0x1b600000,0x00000000)},
  {VT_exponent,22,0x3a848d,DOUBLEWITHTWODWORDINTREE(0x1b700000,0x00000000)},
  {VT_exponent,22,0x3a848e,DOUBLEWITHTWODWORDINTREE(0x1b800000,0x00000000)},
  {VT_exponent,22,0x3a848f,DOUBLEWITHTWODWORDINTREE(0x1b900000,0x00000000)},
  {VT_exponent,22,0x3a8490,DOUBLEWITHTWODWORDINTREE(0x1ba00000,0x00000000)},
  {VT_exponent,22,0x3a8491,DOUBLEWITHTWODWORDINTREE(0x1bb00000,0x00000000)},
  {VT_exponent,22,0x3a8492,DOUBLEWITHTWODWORDINTREE(0x1bc00000,0x00000000)},
  {VT_exponent,22,0x3a8493,DOUBLEWITHTWODWORDINTREE(0x1bd00000,0x00000000)},
  {VT_exponent,22,0x3a8494,DOUBLEWITHTWODWORDINTREE(0x1be00000,0x00000000)},
  {VT_exponent,22,0x3a8495,DOUBLEWITHTWODWORDINTREE(0x1bf00000,0x00000000)},
  {VT_exponent,22,0x3a8496,DOUBLEWITHTWODWORDINTREE(0x1c000000,0x00000000)},
  {VT_exponent,22,0x3a8497,DOUBLEWITHTWODWORDINTREE(0x1c100000,0x00000000)},
  {VT_exponent,22,0x3a8498,DOUBLEWITHTWODWORDINTREE(0x1c200000,0x00000000)},
  {VT_exponent,22,0x3a8499,DOUBLEWITHTWODWORDINTREE(0x1c300000,0x00000000)},
  {VT_exponent,22,0x3a849a,DOUBLEWITHTWODWORDINTREE(0x1c400000,0x00000000)},
  {VT_exponent,22,0x3a849b,DOUBLEWITHTWODWORDINTREE(0x1c500000,0x00000000)},
  {VT_exponent,22,0x3a849c,DOUBLEWITHTWODWORDINTREE(0x1c600000,0x00000000)},
  {VT_exponent,22,0x3a849d,DOUBLEWITHTWODWORDINTREE(0x1c700000,0x00000000)},
  {VT_exponent,22,0x3a849e,DOUBLEWITHTWODWORDINTREE(0x1c800000,0x00000000)},
  {VT_exponent,22,0x3a849f,DOUBLEWITHTWODWORDINTREE(0x1c900000,0x00000000)},
  {VT_exponent,22,0x3a84a0,DOUBLEWITHTWODWORDINTREE(0x1ca00000,0x00000000)},
  {VT_exponent,22,0x3a84a1,DOUBLEWITHTWODWORDINTREE(0x1cb00000,0x00000000)},
  {VT_exponent,22,0x3a84a2,DOUBLEWITHTWODWORDINTREE(0x1cc00000,0x00000000)},
  {VT_exponent,22,0x3a84a3,DOUBLEWITHTWODWORDINTREE(0x1cd00000,0x00000000)},
  {VT_exponent,22,0x3a84a4,DOUBLEWITHTWODWORDINTREE(0x1ce00000,0x00000000)},
  {VT_exponent,22,0x3a84a5,DOUBLEWITHTWODWORDINTREE(0x1cf00000,0x00000000)},
  {VT_exponent,22,0x3a84a6,DOUBLEWITHTWODWORDINTREE(0x1d000000,0x00000000)},
  {VT_exponent,22,0x3a84a7,DOUBLEWITHTWODWORDINTREE(0x1d100000,0x00000000)},
  {VT_exponent,22,0x3a84a8,DOUBLEWITHTWODWORDINTREE(0x1d200000,0x00000000)},
  {VT_exponent,22,0x3a84a9,DOUBLEWITHTWODWORDINTREE(0x1d300000,0x00000000)},
  {VT_exponent,22,0x3a84aa,DOUBLEWITHTWODWORDINTREE(0x1d400000,0x00000000)},
  {VT_exponent,22,0x3a84ab,DOUBLEWITHTWODWORDINTREE(0x1d500000,0x00000000)},
  {VT_exponent,22,0x3a84ac,DOUBLEWITHTWODWORDINTREE(0x1d600000,0x00000000)},
  {VT_exponent,22,0x3a84ad,DOUBLEWITHTWODWORDINTREE(0x1d700000,0x00000000)},
  {VT_exponent,22,0x3a84ae,DOUBLEWITHTWODWORDINTREE(0x1d800000,0x00000000)},
  {VT_exponent,22,0x3a84af,DOUBLEWITHTWODWORDINTREE(0x1d900000,0x00000000)},
  {VT_exponent,22,0x3a84b0,DOUBLEWITHTWODWORDINTREE(0x1da00000,0x00000000)},
  {VT_exponent,22,0x3a84b1,DOUBLEWITHTWODWORDINTREE(0x1db00000,0x00000000)},
  {VT_exponent,22,0x3a84b2,DOUBLEWITHTWODWORDINTREE(0x1dc00000,0x00000000)},
  {VT_exponent,22,0x3a84b3,DOUBLEWITHTWODWORDINTREE(0x1dd00000,0x00000000)},
  {VT_exponent,22,0x3a84b4,DOUBLEWITHTWODWORDINTREE(0x1de00000,0x00000000)},
  {VT_exponent,22,0x3a84b5,DOUBLEWITHTWODWORDINTREE(0x1df00000,0x00000000)},
  {VT_exponent,22,0x3a84b6,DOUBLEWITHTWODWORDINTREE(0x1e000000,0x00000000)},
  {VT_exponent,22,0x3a84b7,DOUBLEWITHTWODWORDINTREE(0x1e100000,0x00000000)},
  {VT_exponent,22,0x3a84b8,DOUBLEWITHTWODWORDINTREE(0x1e200000,0x00000000)},
  {VT_exponent,22,0x3a84b9,DOUBLEWITHTWODWORDINTREE(0x1e300000,0x00000000)},
  {VT_exponent,22,0x3a84ba,DOUBLEWITHTWODWORDINTREE(0x1e400000,0x00000000)},
  {VT_exponent,22,0x3a84bb,DOUBLEWITHTWODWORDINTREE(0x1e500000,0x00000000)},
  {VT_exponent,22,0x3a84bc,DOUBLEWITHTWODWORDINTREE(0x1e600000,0x00000000)},
  {VT_exponent,22,0x3a84bd,DOUBLEWITHTWODWORDINTREE(0x1e700000,0x00000000)},
  {VT_exponent,22,0x3a84be,DOUBLEWITHTWODWORDINTREE(0x1e800000,0x00000000)},
  {VT_exponent,22,0x3a84bf,DOUBLEWITHTWODWORDINTREE(0x1e900000,0x00000000)},
```

```
  {VT_exponent,22,0x3a84c0,DOUBLEWITHTWODWORDINTREE(0x1ea00000,0x00000000)},
  {VT_exponent,22,0x3a84c1,DOUBLEWITHTWODWORDINTREE(0x1eb00000,0x00000000)},
  {VT_exponent,22,0x3a84c2,DOUBLEWITHTWODWORDINTREE(0x1ec00000,0x00000000)},
  {VT_exponent,22,0x3a84c3,DOUBLEWITHTWODWORDINTREE(0x1ed00000,0x00000000)},
  {VT_exponent,22,0x3a84c4,DOUBLEWITHTWODWORDINTREE(0x1ee00000,0x00000000)},
  {VT_exponent,22,0x3a84c5,DOUBLEWITHTWODWORDINTREE(0x1ef00000,0x00000000)},
  {VT_exponent,22,0x3a84c6,DOUBLEWITHTWODWORDINTREE(0x1f000000,0x00000000)},
  {VT_exponent,22,0x3a84c7,DOUBLEWITHTWODWORDINTREE(0x1f100000,0x00000000)},
  {VT_exponent,22,0x3a84c8,DOUBLEWITHTWODWORDINTREE(0x1f200000,0x00000000)},
  {VT_exponent,22,0x3a84c9,DOUBLEWITHTWODWORDINTREE(0x1f300000,0x00000000)},
  {VT_exponent,22,0x3a84ca,DOUBLEWITHTWODWORDINTREE(0x1f400000,0x00000000)},
  {VT_exponent,22,0x3a84cb,DOUBLEWITHTWODWORDINTREE(0x1f500000,0x00000000)},
  {VT_exponent,22,0x3a84cc,DOUBLEWITHTWODWORDINTREE(0x1f600000,0x00000000)},
  {VT_exponent,22,0x3a84cd,DOUBLEWITHTWODWORDINTREE(0x1f700000,0x00000000)},
  {VT_exponent,22,0x3a84ce,DOUBLEWITHTWODWORDINTREE(0x1f800000,0x00000000)},
  {VT_exponent,22,0x3a84cf,DOUBLEWITHTWODWORDINTREE(0x1f900000,0x00000000)},
  {VT_exponent,22,0x3a84d0,DOUBLEWITHTWODWORDINTREE(0x1fa00000,0x00000000)},
  {VT_exponent,22,0x3a84d1,DOUBLEWITHTWODWORDINTREE(0x1fb00000,0x00000000)},
  {VT_exponent,22,0x3a84d2,DOUBLEWITHTWODWORDINTREE(0x1fc00000,0x00000000)},
  {VT_exponent,22,0x3a84d3,DOUBLEWITHTWODWORDINTREE(0x1fd00000,0x00000000)},
  {VT_exponent,22,0x3a84d4,DOUBLEWITHTWODWORDINTREE(0x1fe00000,0x00000000)},
  {VT_exponent,22,0x3a84d5,DOUBLEWITHTWODWORDINTREE(0x1ff00000,0x00000000)},
  {VT_exponent,22,0x3a84d6,DOUBLEWITHTWODWORDINTREE(0x20000000,0x00000000)},
  {VT_exponent,22,0x3a84d7,DOUBLEWITHTWODWORDINTREE(0x20100000,0x00000000)},
  {VT_exponent,22,0x3a84d8,DOUBLEWITHTWODWORDINTREE(0x20200000,0x00000000)},
  {VT_exponent,22,0x3a84d9,DOUBLEWITHTWODWORDINTREE(0x20300000,0x00000000)},
  {VT_exponent,22,0x3a84da,DOUBLEWITHTWODWORDINTREE(0x20400000,0x00000000)},
  {VT_exponent,22,0x3a84db,DOUBLEWITHTWODWORDINTREE(0x20500000,0x00000000)},
  {VT_exponent,22,0x3a84dc,DOUBLEWITHTWODWORDINTREE(0x20600000,0x00000000)},
  {VT_exponent,22,0x3a84dd,DOUBLEWITHTWODWORDINTREE(0x20700000,0x00000000)},
  {VT_exponent,22,0x3a84de,DOUBLEWITHTWODWORDINTREE(0x20800000,0x00000000)},
  {VT_exponent,22,0x3a84df,DOUBLEWITHTWODWORDINTREE(0x20900000,0x00000000)},
  {VT_exponent,22,0x3a84e0,DOUBLEWITHTWODWORDINTREE(0x20a00000,0x00000000)},
  {VT_exponent,22,0x3a84e1,DOUBLEWITHTWODWORDINTREE(0x20b00000,0x00000000)},
  {VT_exponent,22,0x3a84e2,DOUBLEWITHTWODWORDINTREE(0x20c00000,0x00000000)},
  {VT_exponent,22,0x3a84e3,DOUBLEWITHTWODWORDINTREE(0x20d00000,0x00000000)},
  {VT_exponent,22,0x3a84e4,DOUBLEWITHTWODWORDINTREE(0x20e00000,0x00000000)},
  {VT_exponent,22,0x3a84e5,DOUBLEWITHTWODWORDINTREE(0x20f00000,0x00000000)},
  {VT_exponent,22,0x3a84e6,DOUBLEWITHTWODWORDINTREE(0x21000000,0x00000000)},
  {VT_exponent,22,0x3a84e7,DOUBLEWITHTWODWORDINTREE(0x21100000,0x00000000)},
  {VT_exponent,22,0x3a84e8,DOUBLEWITHTWODWORDINTREE(0x21200000,0x00000000)},
  {VT_exponent,22,0x3a84e9,DOUBLEWITHTWODWORDINTREE(0x21300000,0x00000000)},
  {VT_exponent,22,0x3a84ea,DOUBLEWITHTWODWORDINTREE(0x21400000,0x00000000)},
  {VT_exponent,22,0x3a84eb,DOUBLEWITHTWODWORDINTREE(0x21500000,0x00000000)},
  {VT_exponent,22,0x3a84ec,DOUBLEWITHTWODWORDINTREE(0x21600000,0x00000000)},
  {VT_exponent,22,0x3a84ed,DOUBLEWITHTWODWORDINTREE(0x21700000,0x00000000)},
  {VT_exponent,22,0x3a84ee,DOUBLEWITHTWODWORDINTREE(0x21800000,0x00000000)},
  {VT_exponent,22,0x3a84ef,DOUBLEWITHTWODWORDINTREE(0x21900000,0x00000000)},
  {VT_exponent,22,0x3a84f0,DOUBLEWITHTWODWORDINTREE(0x21a00000,0x00000000)},
  {VT_exponent,22,0x3a84f1,DOUBLEWITHTWODWORDINTREE(0x21b00000,0x00000000)},
  {VT_exponent,22,0x3a84f2,DOUBLEWITHTWODWORDINTREE(0x21c00000,0x00000000)},
  {VT_exponent,22,0x3a84f3,DOUBLEWITHTWODWORDINTREE(0x21d00000,0x00000000)},
  {VT_exponent,22,0x3a84f4,DOUBLEWITHTWODWORDINTREE(0x21e00000,0x00000000)},
  {VT_exponent,22,0x3a84f5,DOUBLEWITHTWODWORDINTREE(0x21f00000,0x00000000)},
  {VT_exponent,22,0x3a84f6,DOUBLEWITHTWODWORDINTREE(0x22000000,0x00000000)},
  {VT_exponent,22,0x3a84f7,DOUBLEWITHTWODWORDINTREE(0x22100000,0x00000000)},
  {VT_exponent,22,0x3a84f8,DOUBLEWITHTWODWORDINTREE(0x22200000,0x00000000)},
  {VT_exponent,22,0x3a84f9,DOUBLEWITHTWODWORDINTREE(0x22300000,0x00000000)},
  {VT_exponent,22,0x3a84fa,DOUBLEWITHTWODWORDINTREE(0x22400000,0x00000000)},
  {VT_exponent,22,0x3a84fb,DOUBLEWITHTWODWORDINTREE(0x22500000,0x00000000)},
  {VT_exponent,22,0x3a84fc,DOUBLEWITHTWODWORDINTREE(0x22600000,0x00000000)},
```

```
  {VT_exponent,22,0x3a84fd,DOUBLEWITHTWODWORDINTREE(0x22700000,0x00000000)},
  {VT_exponent,22,0x3a84fe,DOUBLEWITHTWODWORDINTREE(0x22800000,0x00000000)},
  {VT_exponent,22,0x3a84ff,DOUBLEWITHTWODWORDINTREE(0x22900000,0x00000000)},
  {VT_exponent,22,0x3a8500,DOUBLEWITHTWODWORDINTREE(0x22a00000,0x00000000)},
  {VT_exponent,22,0x3a8501,DOUBLEWITHTWODWORDINTREE(0x22b00000,0x00000000)},
  {VT_exponent,22,0x3a8502,DOUBLEWITHTWODWORDINTREE(0x22c00000,0x00000000)},
  {VT_exponent,22,0x3a8503,DOUBLEWITHTWODWORDINTREE(0x22d00000,0x00000000)},
  {VT_exponent,22,0x3a8504,DOUBLEWITHTWODWORDINTREE(0x22e00000,0x00000000)},
  {VT_exponent,22,0x3a8505,DOUBLEWITHTWODWORDINTREE(0x22f00000,0x00000000)},
  {VT_exponent,22,0x3a8506,DOUBLEWITHTWODWORDINTREE(0x23000000,0x00000000)},
  {VT_exponent,22,0x3a8507,DOUBLEWITHTWODWORDINTREE(0x23100000,0x00000000)},
  {VT_exponent,22,0x3a8508,DOUBLEWITHTWODWORDINTREE(0x23200000,0x00000000)},
  {VT_exponent,22,0x3a8509,DOUBLEWITHTWODWORDINTREE(0x23300000,0x00000000)},
  {VT_exponent,22,0x3a850a,DOUBLEWITHTWODWORDINTREE(0x23400000,0x00000000)},
  {VT_exponent,22,0x3a850b,DOUBLEWITHTWODWORDINTREE(0x23500000,0x00000000)},
  {VT_exponent,22,0x3a850c,DOUBLEWITHTWODWORDINTREE(0x23600000,0x00000000)},
  {VT_exponent,22,0x3a850d,DOUBLEWITHTWODWORDINTREE(0x23700000,0x00000000)},
  {VT_exponent,22,0x3a850e,DOUBLEWITHTWODWORDINTREE(0x23800000,0x00000000)},
  {VT_exponent,22,0x3a850f,DOUBLEWITHTWODWORDINTREE(0x23900000,0x00000000)},
  {VT_exponent,22,0x3a8510,DOUBLEWITHTWODWORDINTREE(0x23a00000,0x00000000)},
  {VT_exponent,22,0x3a8511,DOUBLEWITHTWODWORDINTREE(0x23b00000,0x00000000)},
  {VT_exponent,22,0x3a8512,DOUBLEWITHTWODWORDINTREE(0x23c00000,0x00000000)},
  {VT_exponent,22,0x3a8513,DOUBLEWITHTWODWORDINTREE(0x23d00000,0x00000000)},
  {VT_exponent,22,0x3a8514,DOUBLEWITHTWODWORDINTREE(0x23e00000,0x00000000)},
  {VT_exponent,22,0x3a8515,DOUBLEWITHTWODWORDINTREE(0x23f00000,0x00000000)},
  {VT_exponent,22,0x3a8516,DOUBLEWITHTWODWORDINTREE(0x24000000,0x00000000)},
  {VT_exponent,22,0x3a8517,DOUBLEWITHTWODWORDINTREE(0x24100000,0x00000000)},
  {VT_exponent,22,0x3a8518,DOUBLEWITHTWODWORDINTREE(0x24200000,0x00000000)},
  {VT_exponent,22,0x3a8519,DOUBLEWITHTWODWORDINTREE(0x24300000,0x00000000)},
  {VT_exponent,22,0x3a851a,DOUBLEWITHTWODWORDINTREE(0x24400000,0x00000000)},
  {VT_exponent,22,0x3a851b,DOUBLEWITHTWODWORDINTREE(0x24500000,0x00000000)},
  {VT_exponent,22,0x3a851c,DOUBLEWITHTWODWORDINTREE(0x24600000,0x00000000)},
  {VT_exponent,22,0x3a851d,DOUBLEWITHTWODWORDINTREE(0x24700000,0x00000000)},
  {VT_exponent,22,0x3a851e,DOUBLEWITHTWODWORDINTREE(0x24800000,0x00000000)},
  {VT_exponent,22,0x3a851f,DOUBLEWITHTWODWORDINTREE(0x24900000,0x00000000)},
  {VT_exponent,22,0x3a8520,DOUBLEWITHTWODWORDINTREE(0x24a00000,0x00000000)},
  {VT_exponent,22,0x3a8521,DOUBLEWITHTWODWORDINTREE(0x24b00000,0x00000000)},
  {VT_exponent,22,0x3a8522,DOUBLEWITHTWODWORDINTREE(0x24c00000,0x00000000)},
  {VT_exponent,22,0x3a8523,DOUBLEWITHTWODWORDINTREE(0x24d00000,0x00000000)},
  {VT_exponent,22,0x3a8524,DOUBLEWITHTWODWORDINTREE(0x24e00000,0x00000000)},
  {VT_exponent,22,0x3a8525,DOUBLEWITHTWODWORDINTREE(0x24f00000,0x00000000)},
  {VT_exponent,22,0x3a8526,DOUBLEWITHTWODWORDINTREE(0x25000000,0x00000000)},
  {VT_exponent,22,0x3a8527,DOUBLEWITHTWODWORDINTREE(0x25100000,0x00000000)},
  {VT_exponent,22,0x3a8528,DOUBLEWITHTWODWORDINTREE(0x25200000,0x00000000)},
  {VT_exponent,22,0x3a8529,DOUBLEWITHTWODWORDINTREE(0x25300000,0x00000000)},
  {VT_exponent,22,0x3a852a,DOUBLEWITHTWODWORDINTREE(0x25400000,0x00000000)},
  {VT_exponent,22,0x3a852b,DOUBLEWITHTWODWORDINTREE(0x25500000,0x00000000)},
  {VT_exponent,22,0x3a852c,DOUBLEWITHTWODWORDINTREE(0x25600000,0x00000000)},
  {VT_exponent,22,0x3a852d,DOUBLEWITHTWODWORDINTREE(0x25700000,0x00000000)},
  {VT_exponent,22,0x3a852e,DOUBLEWITHTWODWORDINTREE(0x25800000,0x00000000)},
  {VT_exponent,22,0x3a852f,DOUBLEWITHTWODWORDINTREE(0x25900000,0x00000000)},
  {VT_exponent,22,0x3a8530,DOUBLEWITHTWODWORDINTREE(0x25a00000,0x00000000)},
  {VT_exponent,22,0x3a8531,DOUBLEWITHTWODWORDINTREE(0x25b00000,0x00000000)},
  {VT_exponent,22,0x3a8532,DOUBLEWITHTWODWORDINTREE(0x25c00000,0x00000000)},
  {VT_exponent,22,0x3a8533,DOUBLEWITHTWODWORDINTREE(0x25d00000,0x00000000)},
  {VT_exponent,22,0x3a8534,DOUBLEWITHTWODWORDINTREE(0x25e00000,0x00000000)},
  {VT_exponent,22,0x3a8535,DOUBLEWITHTWODWORDINTREE(0x25f00000,0x00000000)},
  {VT_exponent,22,0x3a8536,DOUBLEWITHTWODWORDINTREE(0x26000000,0x00000000)},
  {VT_exponent,22,0x3a8537,DOUBLEWITHTWODWORDINTREE(0x26100000,0x00000000)},
  {VT_exponent,22,0x3a8538,DOUBLEWITHTWODWORDINTREE(0x26200000,0x00000000)},
  {VT_exponent,22,0x3a8539,DOUBLEWITHTWODWORDINTREE(0x26300000,0x00000000)},
```

```
  {VT_exponent,22,0x3a853a,DOUBLEWITHTWODWORDINTREE(0x26400000,0x00000000)},
  {VT_exponent,22,0x3a853b,DOUBLEWITHTWODWORDINTREE(0x26500000,0x00000000)},
  {VT_exponent,22,0x3a853c,DOUBLEWITHTWODWORDINTREE(0x26600000,0x00000000)},
  {VT_exponent,22,0x3a853d,DOUBLEWITHTWODWORDINTREE(0x26700000,0x00000000)},
  {VT_exponent,22,0x3a853e,DOUBLEWITHTWODWORDINTREE(0x26800000,0x00000000)},
  {VT_exponent,22,0x3a853f,DOUBLEWITHTWODWORDINTREE(0x26900000,0x00000000)},
  {VT_exponent,22,0x3a8540,DOUBLEWITHTWODWORDINTREE(0x26a00000,0x00000000)},
  {VT_exponent,22,0x3a8541,DOUBLEWITHTWODWORDINTREE(0x26b00000,0x00000000)},
  {VT_exponent,22,0x3a8542,DOUBLEWITHTWODWORDINTREE(0x26c00000,0x00000000)},
  {VT_exponent,22,0x3a8543,DOUBLEWITHTWODWORDINTREE(0x26d00000,0x00000000)},
  {VT_exponent,22,0x3a8544,DOUBLEWITHTWODWORDINTREE(0x26e00000,0x00000000)},
  {VT_exponent,22,0x3a8545,DOUBLEWITHTWODWORDINTREE(0x26f00000,0x00000000)},
  {VT_exponent,22,0x3a8546,DOUBLEWITHTWODWORDINTREE(0x27000000,0x00000000)},
  {VT_exponent,22,0x3a8547,DOUBLEWITHTWODWORDINTREE(0x27100000,0x00000000)},
  {VT_exponent,22,0x3a8548,DOUBLEWITHTWODWORDINTREE(0x27200000,0x00000000)},
  {VT_exponent,22,0x3a8549,DOUBLEWITHTWODWORDINTREE(0x27300000,0x00000000)},
  {VT_exponent,22,0x3a854a,DOUBLEWITHTWODWORDINTREE(0x27400000,0x00000000)},
  {VT_exponent,22,0x3a854b,DOUBLEWITHTWODWORDINTREE(0x27500000,0x00000000)},
  {VT_exponent,22,0x3a854c,DOUBLEWITHTWODWORDINTREE(0x27600000,0x00000000)},
  {VT_exponent,22,0x3a854d,DOUBLEWITHTWODWORDINTREE(0x27700000,0x00000000)},
  {VT_exponent,22,0x3a854e,DOUBLEWITHTWODWORDINTREE(0x27800000,0x00000000)},
  {VT_exponent,22,0x3a854f,DOUBLEWITHTWODWORDINTREE(0x27900000,0x00000000)},
  {VT_exponent,22,0x3a8550,DOUBLEWITHTWODWORDINTREE(0x27a00000,0x00000000)},
  {VT_exponent,22,0x3a8551,DOUBLEWITHTWODWORDINTREE(0x27b00000,0x00000000)},
  {VT_exponent,22,0x3a8552,DOUBLEWITHTWODWORDINTREE(0x27c00000,0x00000000)},
  {VT_exponent,22,0x3a8553,DOUBLEWITHTWODWORDINTREE(0x27d00000,0x00000000)},
  {VT_exponent,22,0x3a8554,DOUBLEWITHTWODWORDINTREE(0x27e00000,0x00000000)},
  {VT_exponent,22,0x3a8555,DOUBLEWITHTWODWORDINTREE(0x27f00000,0x00000000)},
  {VT_exponent,22,0x3a8556,DOUBLEWITHTWODWORDINTREE(0x28000000,0x00000000)},
  {VT_exponent,22,0x3a8557,DOUBLEWITHTWODWORDINTREE(0x28100000,0x00000000)},
  {VT_exponent,22,0x3a8558,DOUBLEWITHTWODWORDINTREE(0x28200000,0x00000000)},
  {VT_exponent,22,0x3a8559,DOUBLEWITHTWODWORDINTREE(0x28300000,0x00000000)},
  {VT_exponent,22,0x3a855a,DOUBLEWITHTWODWORDINTREE(0x28400000,0x00000000)},
  {VT_exponent,22,0x3a855b,DOUBLEWITHTWODWORDINTREE(0x28500000,0x00000000)},
  {VT_exponent,22,0x3a855c,DOUBLEWITHTWODWORDINTREE(0x28600000,0x00000000)},
  {VT_exponent,22,0x3a855d,DOUBLEWITHTWODWORDINTREE(0x28700000,0x00000000)},
  {VT_exponent,22,0x3a855e,DOUBLEWITHTWODWORDINTREE(0x28800000,0x00000000)},
  {VT_exponent,22,0x3a855f,DOUBLEWITHTWODWORDINTREE(0x28900000,0x00000000)},
  {VT_exponent,22,0x3a8560,DOUBLEWITHTWODWORDINTREE(0x28a00000,0x00000000)},
  {VT_exponent,22,0x3a8561,DOUBLEWITHTWODWORDINTREE(0x28b00000,0x00000000)},
  {VT_exponent,22,0x3a8562,DOUBLEWITHTWODWORDINTREE(0x28c00000,0x00000000)},
  {VT_exponent,22,0x3a8563,DOUBLEWITHTWODWORDINTREE(0x28d00000,0x00000000)},
  {VT_exponent,22,0x3a8564,DOUBLEWITHTWODWORDINTREE(0x28e00000,0x00000000)},
  {VT_exponent,22,0x3a8565,DOUBLEWITHTWODWORDINTREE(0x28f00000,0x00000000)},
  {VT_exponent,22,0x3a8566,DOUBLEWITHTWODWORDINTREE(0x29000000,0x00000000)},
  {VT_exponent,22,0x3a8567,DOUBLEWITHTWODWORDINTREE(0x29100000,0x00000000)},
  {VT_exponent,22,0x3a8568,DOUBLEWITHTWODWORDINTREE(0x29200000,0x00000000)},
  {VT_exponent,22,0x3a8569,DOUBLEWITHTWODWORDINTREE(0x29300000,0x00000000)},
  {VT_exponent,22,0x3a856a,DOUBLEWITHTWODWORDINTREE(0x29400000,0x00000000)},
  {VT_exponent,22,0x3a856b,DOUBLEWITHTWODWORDINTREE(0x29500000,0x00000000)},
  {VT_exponent,22,0x3a856c,DOUBLEWITHTWODWORDINTREE(0x29600000,0x00000000)},
  {VT_exponent,22,0x3a856d,DOUBLEWITHTWODWORDINTREE(0x29700000,0x00000000)},
  {VT_exponent,22,0x3a856e,DOUBLEWITHTWODWORDINTREE(0x29800000,0x00000000)},
  {VT_exponent,22,0x3a856f,DOUBLEWITHTWODWORDINTREE(0x29900000,0x00000000)},
  {VT_exponent,22,0x3a8570,DOUBLEWITHTWODWORDINTREE(0x29a00000,0x00000000)},
  {VT_exponent,22,0x3a8571,DOUBLEWITHTWODWORDINTREE(0x29b00000,0x00000000)},
  {VT_exponent,22,0x3a8572,DOUBLEWITHTWODWORDINTREE(0x29c00000,0x00000000)},
  {VT_exponent,22,0x3a8573,DOUBLEWITHTWODWORDINTREE(0x29d00000,0x00000000)},
  {VT_exponent,22,0x3a8574,DOUBLEWITHTWODWORDINTREE(0x29e00000,0x00000000)},
  {VT_exponent,22,0x3a8575,DOUBLEWITHTWODWORDINTREE(0x29f00000,0x00000000)},
  {VT_exponent,22,0x3a8576,DOUBLEWITHTWODWORDINTREE(0x2a000000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8577,DOUBLEWITHTWODWORDINTREE(0x2a100000,0x00000000)},
  {VT_exponent,22,0x3a8578,DOUBLEWITHTWODWORDINTREE(0x2a200000,0x00000000)},
  {VT_exponent,22,0x3a8579,DOUBLEWITHTWODWORDINTREE(0x2a300000,0x00000000)},
  {VT_exponent,22,0x3a857a,DOUBLEWITHTWODWORDINTREE(0x2a400000,0x00000000)},
  {VT_exponent,22,0x3a857b,DOUBLEWITHTWODWORDINTREE(0x2a500000,0x00000000)},
  {VT_exponent,22,0x3a857c,DOUBLEWITHTWODWORDINTREE(0x2a600000,0x00000000)},
  {VT_exponent,22,0x3a857d,DOUBLEWITHTWODWORDINTREE(0x2a700000,0x00000000)},
  {VT_exponent,22,0x3a857e,DOUBLEWITHTWODWORDINTREE(0x2a800000,0x00000000)},
  {VT_exponent,22,0x3a857f,DOUBLEWITHTWODWORDINTREE(0x2a900000,0x00000000)},
  {VT_exponent,22,0x3a8580,DOUBLEWITHTWODWORDINTREE(0x2aa00000,0x00000000)},
  {VT_exponent,22,0x3a8581,DOUBLEWITHTWODWORDINTREE(0x2ab00000,0x00000000)},
  {VT_exponent,22,0x3a8582,DOUBLEWITHTWODWORDINTREE(0x2ac00000,0x00000000)},
  {VT_exponent,22,0x3a8583,DOUBLEWITHTWODWORDINTREE(0x2ad00000,0x00000000)},
  {VT_exponent,22,0x3a8584,DOUBLEWITHTWODWORDINTREE(0x2ae00000,0x00000000)},
  {VT_exponent,22,0x3a8585,DOUBLEWITHTWODWORDINTREE(0x2af00000,0x00000000)},
  {VT_exponent,22,0x3a8586,DOUBLEWITHTWODWORDINTREE(0x2b000000,0x00000000)},
  {VT_exponent,22,0x3a8587,DOUBLEWITHTWODWORDINTREE(0x2b100000,0x00000000)},
  {VT_exponent,22,0x3a8588,DOUBLEWITHTWODWORDINTREE(0x2b200000,0x00000000)},
  {VT_exponent,22,0x3a8589,DOUBLEWITHTWODWORDINTREE(0x2b300000,0x00000000)},
  {VT_exponent,22,0x3a858a,DOUBLEWITHTWODWORDINTREE(0x2b400000,0x00000000)},
  {VT_exponent,22,0x3a858b,DOUBLEWITHTWODWORDINTREE(0x2b500000,0x00000000)},
  {VT_exponent,22,0x3a858c,DOUBLEWITHTWODWORDINTREE(0x2b600000,0x00000000)},
  {VT_exponent,22,0x3a858d,DOUBLEWITHTWODWORDINTREE(0x2b700000,0x00000000)},
  {VT_exponent,22,0x3a858e,DOUBLEWITHTWODWORDINTREE(0x2b800000,0x00000000)},
  {VT_exponent,22,0x3a858f,DOUBLEWITHTWODWORDINTREE(0x2b900000,0x00000000)},
  {VT_exponent,22,0x3a8590,DOUBLEWITHTWODWORDINTREE(0x2ba00000,0x00000000)},
  {VT_exponent,22,0x3a8591,DOUBLEWITHTWODWORDINTREE(0x2bb00000,0x00000000)},
  {VT_exponent,22,0x3a8592,DOUBLEWITHTWODWORDINTREE(0x2bc00000,0x00000000)},
  {VT_exponent,22,0x3a8593,DOUBLEWITHTWODWORDINTREE(0x2bd00000,0x00000000)},
  {VT_exponent,22,0x3a8594,DOUBLEWITHTWODWORDINTREE(0x2be00000,0x00000000)},
  {VT_exponent,22,0x3a8595,DOUBLEWITHTWODWORDINTREE(0x2bf00000,0x00000000)},
  {VT_exponent,22,0x3a8596,DOUBLEWITHTWODWORDINTREE(0x2c000000,0x00000000)},
  {VT_exponent,22,0x3a8597,DOUBLEWITHTWODWORDINTREE(0x2c100000,0x00000000)},
  {VT_exponent,22,0x3a8598,DOUBLEWITHTWODWORDINTREE(0x2c200000,0x00000000)},
  {VT_exponent,22,0x3a8599,DOUBLEWITHTWODWORDINTREE(0x2c300000,0x00000000)},
  {VT_exponent,22,0x3a859a,DOUBLEWITHTWODWORDINTREE(0x2c400000,0x00000000)},
  {VT_exponent,22,0x3a859b,DOUBLEWITHTWODWORDINTREE(0x2c500000,0x00000000)},
  {VT_exponent,22,0x3a859c,DOUBLEWITHTWODWORDINTREE(0x2c600000,0x00000000)},
  {VT_exponent,22,0x3a859d,DOUBLEWITHTWODWORDINTREE(0x2c700000,0x00000000)},
  {VT_exponent,22,0x3a859e,DOUBLEWITHTWODWORDINTREE(0x2c800000,0x00000000)},
  {VT_exponent,22,0x3a859f,DOUBLEWITHTWODWORDINTREE(0x2c900000,0x00000000)},
  {VT_exponent,22,0x3a85a0,DOUBLEWITHTWODWORDINTREE(0x2ca00000,0x00000000)},
  {VT_exponent,22,0x3a85a1,DOUBLEWITHTWODWORDINTREE(0x2cb00000,0x00000000)},
  {VT_exponent,22,0x3a85a2,DOUBLEWITHTWODWORDINTREE(0x2cc00000,0x00000000)},
  {VT_exponent,22,0x3a85a3,DOUBLEWITHTWODWORDINTREE(0x2cd00000,0x00000000)},
  {VT_exponent,22,0x3a85a4,DOUBLEWITHTWODWORDINTREE(0x2ce00000,0x00000000)},
  {VT_exponent,22,0x3a85a5,DOUBLEWITHTWODWORDINTREE(0x2cf00000,0x00000000)},
  {VT_exponent,22,0x3a85a6,DOUBLEWITHTWODWORDINTREE(0x2d000000,0x00000000)},
  {VT_exponent,22,0x3a85a7,DOUBLEWITHTWODWORDINTREE(0x2d100000,0x00000000)},
  {VT_exponent,22,0x3a85a8,DOUBLEWITHTWODWORDINTREE(0x2d200000,0x00000000)},
  {VT_exponent,22,0x3a85a9,DOUBLEWITHTWODWORDINTREE(0x2d300000,0x00000000)},
  {VT_exponent,22,0x3a85aa,DOUBLEWITHTWODWORDINTREE(0x2d400000,0x00000000)},
  {VT_exponent,22,0x3a85ab,DOUBLEWITHTWODWORDINTREE(0x2d500000,0x00000000)},
  {VT_exponent,22,0x3a85ac,DOUBLEWITHTWODWORDINTREE(0x2d600000,0x00000000)},
  {VT_exponent,22,0x3a85ad,DOUBLEWITHTWODWORDINTREE(0x2d700000,0x00000000)},
  {VT_exponent,22,0x3a85ae,DOUBLEWITHTWODWORDINTREE(0x2d800000,0x00000000)},
  {VT_exponent,22,0x3a85af,DOUBLEWITHTWODWORDINTREE(0x2d900000,0x00000000)},
  {VT_exponent,22,0x3a85b0,DOUBLEWITHTWODWORDINTREE(0x2da00000,0x00000000)},
  {VT_exponent,22,0x3a85b1,DOUBLEWITHTWODWORDINTREE(0x2db00000,0x00000000)},
  {VT_exponent,22,0x3a85b2,DOUBLEWITHTWODWORDINTREE(0x2dc00000,0x00000000)},
  {VT_exponent,22,0x3a85b3,DOUBLEWITHTWODWORDINTREE(0x2dd00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a85b4,DOUBLEWITHTWODWORDINTREE(0x2de00000,0x00000000)},
  {VT_exponent,22,0x3a85b5,DOUBLEWITHTWODWORDINTREE(0x2df00000,0x00000000)},
  {VT_exponent,22,0x3a85b6,DOUBLEWITHTWODWORDINTREE(0x2e000000,0x00000000)},
  {VT_exponent,22,0x3a85b7,DOUBLEWITHTWODWORDINTREE(0x2e100000,0x00000000)},
  {VT_exponent,22,0x3a85b8,DOUBLEWITHTWODWORDINTREE(0x2e200000,0x00000000)},
  {VT_exponent,22,0x3a85b9,DOUBLEWITHTWODWORDINTREE(0x2e300000,0x00000000)},
  {VT_exponent,22,0x3a85ba,DOUBLEWITHTWODWORDINTREE(0x2e400000,0x00000000)},
  {VT_exponent,22,0x3a85bb,DOUBLEWITHTWODWORDINTREE(0x2e500000,0x00000000)},
  {VT_exponent,22,0x3a85bc,DOUBLEWITHTWODWORDINTREE(0x2e600000,0x00000000)},
  {VT_exponent,22,0x3a85bd,DOUBLEWITHTWODWORDINTREE(0x2e700000,0x00000000)},
  {VT_exponent,22,0x3a85be,DOUBLEWITHTWODWORDINTREE(0x2e800000,0x00000000)},
  {VT_exponent,22,0x3a85bf,DOUBLEWITHTWODWORDINTREE(0x2e900000,0x00000000)},
  {VT_exponent,22,0x3a85c0,DOUBLEWITHTWODWORDINTREE(0x2ea00000,0x00000000)},
  {VT_exponent,22,0x3a85c1,DOUBLEWITHTWODWORDINTREE(0x2eb00000,0x00000000)},
  {VT_exponent,22,0x3a85c2,DOUBLEWITHTWODWORDINTREE(0x2ec00000,0x00000000)},
  {VT_exponent,22,0x3a85c3,DOUBLEWITHTWODWORDINTREE(0x2ed00000,0x00000000)},
  {VT_exponent,22,0x3a85c4,DOUBLEWITHTWODWORDINTREE(0x2ee00000,0x00000000)},
  {VT_exponent,22,0x3a85c5,DOUBLEWITHTWODWORDINTREE(0x2ef00000,0x00000000)},
  {VT_exponent,22,0x3a85c6,DOUBLEWITHTWODWORDINTREE(0x2f000000,0x00000000)},
  {VT_exponent,22,0x3a85c7,DOUBLEWITHTWODWORDINTREE(0x2f100000,0x00000000)},
  {VT_exponent,22,0x3a85c8,DOUBLEWITHTWODWORDINTREE(0x2f200000,0x00000000)},
  {VT_exponent,22,0x3a85c9,DOUBLEWITHTWODWORDINTREE(0x2f300000,0x00000000)},
  {VT_exponent,22,0x3a85ca,DOUBLEWITHTWODWORDINTREE(0x2f400000,0x00000000)},
  {VT_exponent,22,0x3a85cb,DOUBLEWITHTWODWORDINTREE(0x2f500000,0x00000000)},
  {VT_exponent,22,0x3a85cc,DOUBLEWITHTWODWORDINTREE(0x2f600000,0x00000000)},
  {VT_exponent,22,0x3a85cd,DOUBLEWITHTWODWORDINTREE(0x2f700000,0x00000000)},
  {VT_exponent,22,0x3a85ce,DOUBLEWITHTWODWORDINTREE(0x2f800000,0x00000000)},
  {VT_exponent,22,0x3a85cf,DOUBLEWITHTWODWORDINTREE(0x2f900000,0x00000000)},
  {VT_exponent,22,0x3a85d0,DOUBLEWITHTWODWORDINTREE(0x2fa00000,0x00000000)},
  {VT_exponent,22,0x3a85d1,DOUBLEWITHTWODWORDINTREE(0x2fb00000,0x00000000)},
  {VT_exponent,22,0x3a85d2,DOUBLEWITHTWODWORDINTREE(0x2fc00000,0x00000000)},
  {VT_exponent,22,0x3a85d3,DOUBLEWITHTWODWORDINTREE(0x2fd00000,0x00000000)},
  {VT_exponent,22,0x3a85d4,DOUBLEWITHTWODWORDINTREE(0x2fe00000,0x00000000)},
  {VT_exponent,22,0x3a85d5,DOUBLEWITHTWODWORDINTREE(0x2ff00000,0x00000000)},
  {VT_exponent,22,0x3a85d6,DOUBLEWITHTWODWORDINTREE(0x30000000,0x00000000)},
  {VT_exponent,22,0x3a85d7,DOUBLEWITHTWODWORDINTREE(0x30100000,0x00000000)},
  {VT_exponent,22,0x3a85d8,DOUBLEWITHTWODWORDINTREE(0x30200000,0x00000000)},
  {VT_exponent,22,0x3a85d9,DOUBLEWITHTWODWORDINTREE(0x30300000,0x00000000)},
  {VT_exponent,22,0x3a85da,DOUBLEWITHTWODWORDINTREE(0x30400000,0x00000000)},
  {VT_exponent,22,0x3a85db,DOUBLEWITHTWODWORDINTREE(0x30500000,0x00000000)},
  {VT_exponent,22,0x3a85dc,DOUBLEWITHTWODWORDINTREE(0x30600000,0x00000000)},
  {VT_exponent,22,0x3a85dd,DOUBLEWITHTWODWORDINTREE(0x30700000,0x00000000)},
  {VT_exponent,22,0x3a85de,DOUBLEWITHTWODWORDINTREE(0x30800000,0x00000000)},
  {VT_exponent,22,0x3a85df,DOUBLEWITHTWODWORDINTREE(0x30900000,0x00000000)},
  {VT_exponent,22,0x3a85e0,DOUBLEWITHTWODWORDINTREE(0x30a00000,0x00000000)},
  {VT_exponent,22,0x3a85e1,DOUBLEWITHTWODWORDINTREE(0x30b00000,0x00000000)},
  {VT_exponent,22,0x3a85e2,DOUBLEWITHTWODWORDINTREE(0x30c00000,0x00000000)},
  {VT_exponent,22,0x3a85e3,DOUBLEWITHTWODWORDINTREE(0x30d00000,0x00000000)},
  {VT_exponent,22,0x3a85e4,DOUBLEWITHTWODWORDINTREE(0x30e00000,0x00000000)},
  {VT_exponent,22,0x3a85e5,DOUBLEWITHTWODWORDINTREE(0x30f00000,0x00000000)},
  {VT_exponent,22,0x3a85e6,DOUBLEWITHTWODWORDINTREE(0x31000000,0x00000000)},
  {VT_exponent,22,0x3a85e7,DOUBLEWITHTWODWORDINTREE(0x31100000,0x00000000)},
  {VT_exponent,22,0x3a85e8,DOUBLEWITHTWODWORDINTREE(0x31200000,0x00000000)},
  {VT_exponent,22,0x3a85e9,DOUBLEWITHTWODWORDINTREE(0x31300000,0x00000000)},
  {VT_exponent,22,0x3a85ea,DOUBLEWITHTWODWORDINTREE(0x31400000,0x00000000)},
  {VT_exponent,22,0x3a85eb,DOUBLEWITHTWODWORDINTREE(0x31500000,0x00000000)},
  {VT_exponent,22,0x3a85ec,DOUBLEWITHTWODWORDINTREE(0x31600000,0x00000000)},
  {VT_exponent,22,0x3a85ed,DOUBLEWITHTWODWORDINTREE(0x31700000,0x00000000)},
  {VT_exponent,22,0x3a85ee,DOUBLEWITHTWODWORDINTREE(0x31800000,0x00000000)},
  {VT_exponent,22,0x3a85ef,DOUBLEWITHTWODWORDINTREE(0x31900000,0x00000000)},
  {VT_exponent,22,0x3a85f0,DOUBLEWITHTWODWORDINTREE(0x31a00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a85f1,DOUBLEWITHTWODWORDINTREE(0x31b00000,0x00000000)},
  {VT_exponent,22,0x3a85f2,DOUBLEWITHTWODWORDINTREE(0x31c00000,0x00000000)},
  {VT_exponent,22,0x3a85f3,DOUBLEWITHTWODWORDINTREE(0x31d00000,0x00000000)},
  {VT_exponent,22,0x3a85f4,DOUBLEWITHTWODWORDINTREE(0x31e00000,0x00000000)},
  {VT_exponent,22,0x3a85f5,DOUBLEWITHTWODWORDINTREE(0x31f00000,0x00000000)},
  {VT_exponent,22,0x3a85f6,DOUBLEWITHTWODWORDINTREE(0x32000000,0x00000000)},
  {VT_exponent,22,0x3a85f7,DOUBLEWITHTWODWORDINTREE(0x32100000,0x00000000)},
  {VT_exponent,22,0x3a85f8,DOUBLEWITHTWODWORDINTREE(0x32200000,0x00000000)},
  {VT_exponent,22,0x3a85f9,DOUBLEWITHTWODWORDINTREE(0x32300000,0x00000000)},
  {VT_exponent,22,0x3a85fa,DOUBLEWITHTWODWORDINTREE(0x32400000,0x00000000)},
  {VT_exponent,22,0x3a85fb,DOUBLEWITHTWODWORDINTREE(0x32500000,0x00000000)},
  {VT_exponent,22,0x3a85fc,DOUBLEWITHTWODWORDINTREE(0x32600000,0x00000000)},
  {VT_exponent,22,0x3a85fd,DOUBLEWITHTWODWORDINTREE(0x32700000,0x00000000)},
  {VT_exponent,22,0x3a85fe,DOUBLEWITHTWODWORDINTREE(0x32800000,0x00000000)},
  {VT_exponent,22,0x3a85ff,DOUBLEWITHTWODWORDINTREE(0x32900000,0x00000000)},
  {VT_exponent,22,0x3a8600,DOUBLEWITHTWODWORDINTREE(0x32a00000,0x00000000)},
  {VT_exponent,22,0x3a8601,DOUBLEWITHTWODWORDINTREE(0x32b00000,0x00000000)},
  {VT_exponent,22,0x3a8602,DOUBLEWITHTWODWORDINTREE(0x32c00000,0x00000000)},
  {VT_exponent,22,0x3a8603,DOUBLEWITHTWODWORDINTREE(0x32d00000,0x00000000)},
  {VT_exponent,22,0x3a8604,DOUBLEWITHTWODWORDINTREE(0x32e00000,0x00000000)},
  {VT_exponent,22,0x3a8605,DOUBLEWITHTWODWORDINTREE(0x32f00000,0x00000000)},
  {VT_exponent,22,0x3a8606,DOUBLEWITHTWODWORDINTREE(0x33000000,0x00000000)},
  {VT_exponent,22,0x3a8607,DOUBLEWITHTWODWORDINTREE(0x33100000,0x00000000)},
  {VT_exponent,22,0x3a8608,DOUBLEWITHTWODWORDINTREE(0x33200000,0x00000000)},
  {VT_exponent,22,0x3a8609,DOUBLEWITHTWODWORDINTREE(0x33300000,0x00000000)},
  {VT_exponent,22,0x3a860a,DOUBLEWITHTWODWORDINTREE(0x33400000,0x00000000)},
  {VT_exponent,22,0x3a860b,DOUBLEWITHTWODWORDINTREE(0x33500000,0x00000000)},
  {VT_exponent,22,0x3a860c,DOUBLEWITHTWODWORDINTREE(0x33600000,0x00000000)},
  {VT_exponent,22,0x3a860d,DOUBLEWITHTWODWORDINTREE(0x33700000,0x00000000)},
  {VT_exponent,22,0x3a860e,DOUBLEWITHTWODWORDINTREE(0x33800000,0x00000000)},
  {VT_exponent,22,0x3a860f,DOUBLEWITHTWODWORDINTREE(0x33900000,0x00000000)},
  {VT_exponent,22,0x3a8610,DOUBLEWITHTWODWORDINTREE(0x33a00000,0x00000000)},
  {VT_exponent,22,0x3a8611,DOUBLEWITHTWODWORDINTREE(0x33b00000,0x00000000)},
  {VT_exponent,22,0x3a8612,DOUBLEWITHTWODWORDINTREE(0x33c00000,0x00000000)},
  {VT_exponent,22,0x3a8613,DOUBLEWITHTWODWORDINTREE(0x33d00000,0x00000000)},
  {VT_exponent,22,0x3a8614,DOUBLEWITHTWODWORDINTREE(0x33e00000,0x00000000)},
  {VT_exponent,22,0x3a8615,DOUBLEWITHTWODWORDINTREE(0x33f00000,0x00000000)},
  {VT_exponent,22,0x3a8616,DOUBLEWITHTWODWORDINTREE(0x34000000,0x00000000)},
  {VT_exponent,22,0x3a8617,DOUBLEWITHTWODWORDINTREE(0x34100000,0x00000000)},
  {VT_exponent,22,0x3a8618,DOUBLEWITHTWODWORDINTREE(0x34200000,0x00000000)},
  {VT_exponent,22,0x3a8619,DOUBLEWITHTWODWORDINTREE(0x34300000,0x00000000)},
  {VT_exponent,22,0x3a861a,DOUBLEWITHTWODWORDINTREE(0x34400000,0x00000000)},
  {VT_exponent,22,0x3a861b,DOUBLEWITHTWODWORDINTREE(0x34500000,0x00000000)},
  {VT_exponent,22,0x3a861c,DOUBLEWITHTWODWORDINTREE(0x34600000,0x00000000)},
  {VT_exponent,22,0x3a861d,DOUBLEWITHTWODWORDINTREE(0x34700000,0x00000000)},
  {VT_exponent,22,0x3a861e,DOUBLEWITHTWODWORDINTREE(0x34800000,0x00000000)},
  {VT_exponent,22,0x3a861f,DOUBLEWITHTWODWORDINTREE(0x34900000,0x00000000)},
  {VT_exponent,22,0x3a8620,DOUBLEWITHTWODWORDINTREE(0x34a00000,0x00000000)},
  {VT_exponent,22,0x3a8621,DOUBLEWITHTWODWORDINTREE(0x34b00000,0x00000000)},
  {VT_exponent,22,0x3a8622,DOUBLEWITHTWODWORDINTREE(0x34c00000,0x00000000)},
  {VT_exponent,22,0x3a8623,DOUBLEWITHTWODWORDINTREE(0x34d00000,0x00000000)},
  {VT_exponent,22,0x3a8624,DOUBLEWITHTWODWORDINTREE(0x34e00000,0x00000000)},
  {VT_exponent,22,0x3a8625,DOUBLEWITHTWODWORDINTREE(0x34f00000,0x00000000)},
  {VT_exponent,22,0x3a8626,DOUBLEWITHTWODWORDINTREE(0x35000000,0x00000000)},
  {VT_exponent,22,0x3a8627,DOUBLEWITHTWODWORDINTREE(0x35100000,0x00000000)},
  {VT_exponent,22,0x3a8628,DOUBLEWITHTWODWORDINTREE(0x35200000,0x00000000)},
  {VT_exponent,22,0x3a8629,DOUBLEWITHTWODWORDINTREE(0x35300000,0x00000000)},
  {VT_exponent,18,0xf787,DOUBLEWITHTWODWORDINTREE(0x35400000,0x00000000)},
  {VT_exponent,18,0xd1d4,DOUBLEWITHTWODWORDINTREE(0x35500000,0x00000000)},
  {VT_exponent,19,0x77f9f,DOUBLEWITHTWODWORDINTREE(0x35600000,0x00000000)},
  {VT_exponent,18,0x3b8fb,DOUBLEWITHTWODWORDINTREE(0x35700000,0x00000000)},
```

```
  {VT_exponent,19,0x1ef1a,DOUBLEWITHTWODWORDINTREE(0x35800000,0x00000000)},
  {VT_exponent,21,0x1d4315,DOUBLEWITHTWODWORDINTREE(0x35900000,0x00000000)},
  {VT_exponent,16,0x3de2,DOUBLEWITHTWODWORDINTREE(0x35a00000,0x00000000)},
  {VT_exponent,22,0x3a862c,DOUBLEWITHTWODWORDINTREE(0x35b00000,0x00000000)},
  {VT_exponent,22,0x3a862d,DOUBLEWITHTWODWORDINTREE(0x35c00000,0x00000000)},
  {VT_exponent,22,0x3a862e,DOUBLEWITHTWODWORDINTREE(0x35d00000,0x00000000)},
  {VT_exponent,22,0x3a862f,DOUBLEWITHTWODWORDINTREE(0x35e00000,0x00000000)},
  {VT_exponent,22,0x3a8630,DOUBLEWITHTWODWORDINTREE(0x35f00000,0x00000000)},
  {VT_exponent,22,0x3a8631,DOUBLEWITHTWODWORDINTREE(0x36000000,0x00000000)},
  {VT_exponent,22,0x3a8632,DOUBLEWITHTWODWORDINTREE(0x36100000,0x00000000)},
  {VT_exponent,22,0x3a8633,DOUBLEWITHTWODWORDINTREE(0x36200000,0x00000000)},
  {VT_exponent,22,0x3a8634,DOUBLEWITHTWODWORDINTREE(0x36300000,0x00000000)},
  {VT_exponent,22,0x3a8635,DOUBLEWITHTWODWORDINTREE(0x36400000,0x00000000)},
  {VT_exponent,22,0x3a8636,DOUBLEWITHTWODWORDINTREE(0x36500000,0x00000000)},
  {VT_exponent,22,0x3a8637,DOUBLEWITHTWODWORDINTREE(0x36600000,0x00000000)},
  {VT_exponent,22,0x3a8638,DOUBLEWITHTWODWORDINTREE(0x36700000,0x00000000)},
  {VT_exponent,21,0x1d431d,DOUBLEWITHTWODWORDINTREE(0x36800000,0x00000000)},
  {VT_exponent,22,0x3a8639,DOUBLEWITHTWODWORDINTREE(0x36900000,0x00000000)},
  {VT_exponent,22,0x3a863c,DOUBLEWITHTWODWORDINTREE(0x36a00000,0x00000000)},
  {VT_exponent,16,0x3de0,DOUBLEWITHTWODWORDINTREE(0x36b00000,0x00000000)},
  {VT_exponent,18,0x3a95e,DOUBLEWITHTWODWORDINTREE(0x36c00000,0x00000000)},
  {VT_exponent,21,0x1d431f,DOUBLEWITHTWODWORDINTREE(0x36d00000,0x00000000)},
  {VT_exponent,22,0x3a863d,DOUBLEWITHTWODWORDINTREE(0x36e00000,0x00000000)},
  {VT_exponent,22,0x3a8640,DOUBLEWITHTWODWORDINTREE(0x36f00000,0x00000000)},
  {VT_exponent,20,0x34749,DOUBLEWITHTWODWORDINTREE(0x37000000,0x00000000)},
  {VT_exponent,20,0x3474d,DOUBLEWITHTWODWORDINTREE(0x37100000,0x00000000)},
  {VT_exponent,22,0x3a8641,DOUBLEWITHTWODWORDINTREE(0x37200000,0x00000000)},
  {VT_exponent,22,0x3a8642,DOUBLEWITHTWODWORDINTREE(0x37300000,0x00000000)},
  {VT_exponent,22,0x3a8643,DOUBLEWITHTWODWORDINTREE(0x37400000,0x00000000)},
  {VT_exponent,22,0x3a8644,DOUBLEWITHTWODWORDINTREE(0x37500000,0x00000000)},
  {VT_exponent,22,0x3a8645,DOUBLEWITHTWODWORDINTREE(0x37600000,0x00000000)},
  {VT_exponent,22,0x3a8646,DOUBLEWITHTWODWORDINTREE(0x37700000,0x00000000)},
  {VT_exponent,22,0x3a8647,DOUBLEWITHTWODWORDINTREE(0x37800000,0x00000000)},
  {VT_exponent,22,0x3a8648,DOUBLEWITHTWODWORDINTREE(0x37900000,0x00000000)},
  {VT_exponent,22,0x3a8649,DOUBLEWITHTWODWORDINTREE(0x37a00000,0x00000000)},
  {VT_exponent,22,0x3a864a,DOUBLEWITHTWODWORDINTREE(0x37b00000,0x00000000)},
  {VT_exponent,22,0x3a864b,DOUBLEWITHTWODWORDINTREE(0x37c00000,0x00000000)},
  {VT_exponent,22,0x3a864c,DOUBLEWITHTWODWORDINTREE(0x37d00000,0x00000000)},
  {VT_exponent,22,0x3a864d,DOUBLEWITHTWODWORDINTREE(0x37e00000,0x00000000)},
  {VT_exponent,22,0x3a864e,DOUBLEWITHTWODWORDINTREE(0x37f00000,0x00000000)},
  {VT_exponent,22,0x3a864f,DOUBLEWITHTWODWORDINTREE(0x38000000,0x00000000)},
  {VT_exponent,22,0x3a8650,DOUBLEWITHTWODWORDINTREE(0x38100000,0x00000000)},
  {VT_exponent,22,0x3a8651,DOUBLEWITHTWODWORDINTREE(0x38200000,0x00000000)},
  {VT_exponent,22,0x3a8652,DOUBLEWITHTWODWORDINTREE(0x38300000,0x00000000)},
  {VT_exponent,22,0x3a8653,DOUBLEWITHTWODWORDINTREE(0x38400000,0x00000000)},
  {VT_exponent,22,0x3a8654,DOUBLEWITHTWODWORDINTREE(0x38500000,0x00000000)},
  {VT_exponent,22,0x3a8655,DOUBLEWITHTWODWORDINTREE(0x38600000,0x00000000)},
  {VT_exponent,22,0x3a8656,DOUBLEWITHTWODWORDINTREE(0x38700000,0x00000000)},
  {VT_exponent,22,0x3a8657,DOUBLEWITHTWODWORDINTREE(0x38800000,0x00000000)},
  {VT_exponent,22,0x3a8658,DOUBLEWITHTWODWORDINTREE(0x38900000,0x00000000)},
  {VT_exponent,22,0x3a8659,DOUBLEWITHTWODWORDINTREE(0x38a00000,0x00000000)},
  {VT_exponent,22,0x3a865a,DOUBLEWITHTWODWORDINTREE(0x38b00000,0x00000000)},
  {VT_exponent,22,0x3a865b,DOUBLEWITHTWODWORDINTREE(0x38c00000,0x00000000)},
  {VT_exponent,22,0x3a865c,DOUBLEWITHTWODWORDINTREE(0x38d00000,0x00000000)},
  {VT_exponent,22,0x3a865d,DOUBLEWITHTWODWORDINTREE(0x38e00000,0x00000000)},
  {VT_exponent,21,0x1d432f,DOUBLEWITHTWODWORDINTREE(0x38f00000,0x00000000)},
  {VT_exponent,22,0x3a8660,DOUBLEWITHTWODWORDINTREE(0x39000000,0x00000000)},
  {VT_exponent,22,0x3a8661,DOUBLEWITHTWODWORDINTREE(0x39100000,0x00000000)},
  {VT_exponent,22,0x3a8662,DOUBLEWITHTWODWORDINTREE(0x39200000,0x00000000)},
  {VT_exponent,21,0x1d4332,DOUBLEWITHTWODWORDINTREE(0x39300000,0x00000000)},
  {VT_exponent,18,0xd1d5,DOUBLEWITHTWODWORDINTREE(0x39400000,0x00000000)},
```

```
  {VT_exponent,18,0x3bfda,DOUBLEWITHTWODWORDINTREE(0x39500000,0x00000000)},
  {VT_exponent,15,0x7528,DOUBLEWITHTWODWORDINTREE(0x39600000,0x00000000)},
  {VT_exponent,15,0x7529,DOUBLEWITHTWODWORDINTREE(0x39700000,0x00000000)},
  {VT_exponent,18,0x3a95f,DOUBLEWITHTWODWORDINTREE(0x39800000,0x00000000)},
  {VT_exponent,17,0x1d434,DOUBLEWITHTWODWORDINTREE(0x39900000,0x00000000)},
  {VT_exponent,19,0x750cd,DOUBLEWITHTWODWORDINTREE(0x39a00000,0x00000000)},
  {VT_exponent,18,0x3a867,DOUBLEWITHTWODWORDINTREE(0x39b00000,0x00000000)},
  {VT_exponent,19,0x771fd,DOUBLEWITHTWODWORDINTREE(0x39c00000,0x00000000)},
  {VT_exponent,15,0x7764,DOUBLEWITHTWODWORDINTREE(0x39d00000,0x00000000)},
  {VT_exponent,20,0xee3f9,DOUBLEWITHTWODWORDINTREE(0x39e00000,0x00000000)},
  {VT_exponent,18,0x3bfdb,DOUBLEWITHTWODWORDINTREE(0x39f00000,0x00000000)},
  {VT_exponent,16,0xeff7,DOUBLEWITHTWODWORDINTREE(0x3a000000,0x00000000)},
  {VT_exponent,18,0xd1d6,DOUBLEWITHTWODWORDINTREE(0x3a100000,0x00000000)},
  {VT_exponent,22,0x3a8663,DOUBLEWITHTWODWORDINTREE(0x3a200000,0x00000000)},
  {VT_exponent,22,0x3a8666,DOUBLEWITHTWODWORDINTREE(0x3a300000,0x00000000)},
  {VT_exponent,18,0x3b8fa,DOUBLEWITHTWODWORDINTREE(0x3a400000,0x00000000)},
  {VT_exponent,17,0x1d435,DOUBLEWITHTWODWORDINTREE(0x3a500000,0x00000000)},
  {VT_exponent,17,0x1dfe4,DOUBLEWITHTWODWORDINTREE(0x3a600000,0x00000000)},
  {VT_exponent,19,0x750d8,DOUBLEWITHTWODWORDINTREE(0x3a700000,0x00000000)},
  {VT_exponent,18,0x3a95b,DOUBLEWITHTWODWORDINTREE(0x3a800000,0x00000000)},
  {VT_exponent,19,0x77f9e,DOUBLEWITHTWODWORDINTREE(0x3a900000,0x00000000)},
  {VT_exponent,19,0x750d9,DOUBLEWITHTWODWORDINTREE(0x3aa00000,0x00000000)},
  {VT_exponent,18,0xd1d7,DOUBLEWITHTWODWORDINTREE(0x3ab00000,0x00000000)},
  {VT_exponent,18,0x3b8ff,DOUBLEWITHTWODWORDINTREE(0x3ac00000,0x00000000)},
  {VT_exponent,17,0x1dc7c,DOUBLEWITHTWODWORDINTREE(0x3ad00000,0x00000000)},
  {VT_exponent,19,0x750da,DOUBLEWITHTWODWORDINTREE(0x3ae00000,0x00000000)},
  {VT_exponent,17,0x7bc2,DOUBLEWITHTWODWORDINTREE(0x3af00000,0x00000000)},
  {VT_exponent,18,0x3bfca,DOUBLEWITHTWODWORDINTREE(0x3b000000,0x00000000)},
  {VT_exponent,19,0x1a3a5,DOUBLEWITHTWODWORDINTREE(0x3b100000,0x00000000)},
  {VT_exponent,17,0x1d4ac,DOUBLEWITHTWODWORDINTREE(0x3b200000,0x00000000)},
  {VT_exponent,18,0x3a86e,DOUBLEWITHTWODWORDINTREE(0x3b300000,0x00000000)},
  {VT_exponent,17,0x1d438,DOUBLEWITHTWODWORDINTREE(0x3b400000,0x00000000)},
  {VT_exponent,18,0x3bfcb,DOUBLEWITHTWODWORDINTREE(0x3b500000,0x00000000)},
  {VT_exponent,19,0x1a3a7,DOUBLEWITHTWODWORDINTREE(0x3b600000,0x00000000)},
  {VT_exponent,17,0x1d4ae,DOUBLEWITHTWODWORDINTREE(0x3b700000,0x00000000)},
  {VT_exponent,18,0x3bfce,DOUBLEWITHTWODWORDINTREE(0x3b800000,0x00000000)},
  {VT_exponent,18,0xf78c,DOUBLEWITHTWODWORDINTREE(0x3b900000,0x00000000)},
  {VT_exponent,17,0x1dfec,DOUBLEWITHTWODWORDINTREE(0x3ba00000,0x00000000)},
  {VT_exponent,17,0x1d439,DOUBLEWITHTWODWORDINTREE(0x3bb00000,0x00000000)},
  {VT_exponent,17,0x68e8,DOUBLEWITHTWODWORDINTREE(0x3bc00000,0x00000000)},
  {VT_exponent,18,0xf786,DOUBLEWITHTWODWORDINTREE(0x3bd00000,0x00000000)},
  {VT_exponent,15,0x771e,DOUBLEWITHTWODWORDINTREE(0x3be00000,0x00000000)},
  {VT_exponent,17,0x1dfe6,DOUBLEWITHTWODWORDINTREE(0x3bf00000,0x00000000)},
  {VT_exponent,15,0x77f8,DOUBLEWITHTWODWORDINTREE(0x3c000000,0x00000000)},
  {VT_exponent,14,0x3bb3,DOUBLEWITHTWODWORDINTREE(0x3c100000,0x00000000)},
  {VT_exponent,14,0x3b8e,DOUBLEWITHTWODWORDINTREE(0x3c200000,0x00000000)},
  {VT_exponent,14,0x3a82,DOUBLEWITHTWODWORDINTREE(0x3c300000,0x00000000)},
  {VT_exponent,14,0x3b96,DOUBLEWITHTWODWORDINTREE(0x3c400000,0x00000000)},
  {VT_exponent,13,0x68f,DOUBLEWITHTWODWORDINTREE(0x3c500000,0x00000000)},
  {VT_exponent,12,0x3d4,DOUBLEWITHTWODWORDINTREE(0x3c600000,0x00000000)},
  {VT_exponent,13,0x1dca,DOUBLEWITHTWODWORDINTREE(0x3c700000,0x00000000)},
  {VT_exponent,12,0x346,DOUBLEWITHTWODWORDINTREE(0x3c800000,0x00000000)},
  {VT_exponent,12,0xee7,DOUBLEWITHTWODWORDINTREE(0x3c900000,0x00000000)},
  {VT_exponent,12,0xeea,DOUBLEWITHTWODWORDINTREE(0x3ca00000,0x00000000)},
  {VT_exponent,11,0x1ed,DOUBLEWITHTWODWORDINTREE(0x3cb00000,0x00000000)},
  {VT_exponent,12,0x3df,DOUBLEWITHTWODWORDINTREE(0x3cc00000,0x00000000)},
  {VT_exponent,11,0x1a2,DOUBLEWITHTWODWORDINTREE(0x3cd00000,0x00000000)},
  {VT_exponent,11,0x56f,DOUBLEWITHTWODWORDINTREE(0x3ce00000,0x00000000)},
  {VT_exponent,11,0xb9,DOUBLEWITHTWODWORDINTREE(0x3cf00000,0x00000000)},
  {VT_exponent,13,0x7b2,DOUBLEWITHTWODWORDINTREE(0x3d000000,0x00000000)},
  {VT_exponent,13,0x1dd8,DOUBLEWITHTWODWORDINTREE(0x3d100000,0x00000000)},
```

```
  {VT_exponent,13,0x15ba,DOUBLEWITHTWODWORDINTREE(0x3d200000,0x00000000)},
  {VT_exponent,12,0xee6,DOUBLEWITHTWODWORDINTREE(0x3d300000,0x00000000)},
  {VT_exponent,13,0x15b8,DOUBLEWITHTWODWORDINTREE(0x3d400000,0x00000000)},
  {VT_exponent,14,0xf79,DOUBLEWITHTWODWORDINTREE(0x3d500000,0x00000000)},
  {VT_exponent,14,0x3a81,DOUBLEWITHTWODWORDINTREE(0x3d600000,0x00000000)},
  {VT_exponent,14,0xd1c,DOUBLEWITHTWODWORDINTREE(0x3d700000,0x00000000)},
  {VT_exponent,15,0x7765,DOUBLEWITHTWODWORDINTREE(0x3d800000,0x00000000)},
  {VT_exponent,14,0xf54,DOUBLEWITHTWODWORDINTREE(0x3d900000,0x00000000)},
  {VT_exponent,13,0x15b9,DOUBLEWITHTWODWORDINTREE(0x3da00000,0x00000000)},
  {VT_exponent,13,0x7ab,DOUBLEWITHTWODWORDINTREE(0x3db00000,0x00000000)},
  {VT_exponent,15,0x7500,DOUBLEWITHTWODWORDINTREE(0x3dc00000,0x00000000)},
  {VT_exponent,15,0x1eaa,DOUBLEWITHTWODWORDINTREE(0x3dd00000,0x00000000)},
  {VT_exponent,15,0x7501,DOUBLEWITHTWODWORDINTREE(0x3de00000,0x00000000)},
  {VT_exponent,15,0x1eab,DOUBLEWITHTWODWORDINTREE(0x3df00000,0x00000000)},
  {VT_exponent,14,0x3b97,DOUBLEWITHTWODWORDINTREE(0x3e000000,0x00000000)},
  {VT_exponent,15,0x752a,DOUBLEWITHTWODWORDINTREE(0x3e100000,0x00000000)},
  {VT_exponent,15,0x77fa,DOUBLEWITHTWODWORDINTREE(0x3e200000,0x00000000)},
  {VT_double,10,0xf4,DOUBLEWITHTWODWORDINTREE(0x3e35798e,0xe2308c3a)},
  {VT_exponent,14,0x3a93,DOUBLEWITHTWODWORDINTREE(0x3e300000,0x00000000)},
  {VT_double,11,0x77c,DOUBLEWITHTWODWORDINTREE(0x3e45798e,0xe2308c3a)},
  {VT_exponent,13,0x1dc6,DOUBLEWITHTWODWORDINTREE(0x3e400000,0x00000000)},
  {VT_exponent,13,0x7bd,DOUBLEWITHTWODWORDINTREE(0x3e500000,0x00000000)},
  {VT_exponent,13,0x1dff,DOUBLEWITHTWODWORDINTREE(0x3e600000,0x00000000)},
  {VT_exponent,12,0xefe,DOUBLEWITHTWODWORDINTREE(0x3e700000,0x00000000)},
  {VT_double,8,0xaf,DOUBLEWITHTWODWORDINTREE(0x3e8ad7f2,0x9abcaf4a)},
  {VT_exponent,12,0xeed,DOUBLEWITHTWODWORDINTREE(0x3e800000,0x00000000)},
  {VT_exponent,11,0xb8,DOUBLEWITHTWODWORDINTREE(0x3e900000,0x00000000)},
  {VT_exponent,12,0x3d8,DOUBLEWITHTWODWORDINTREE(0x3ea00000,0x00000000)},
  {VT_exponent,11,0x1eb,DOUBLEWITHTWODWORDINTREE(0x3eb00000,0x00000000)},
  {VT_double,9,0x1d2,DOUBLEWITHTWODWORDINTREE(0x3ec0c6f7,0xa0b5ed8e)},
  {VT_exponent,13,0x1d4b,DOUBLEWITHTWODWORDINTREE(0x3ec00000,0x00000000)},
  {VT_exponent,13,0x7b3,DOUBLEWITHTWODWORDINTREE(0x3ed00000,0x00000000)},
  {VT_exponent,10,0x5d,DOUBLEWITHTWODWORDINTREE(0x3ee00000,0x00000000)},
  {VT_exponent,12,0xeeb,DOUBLEWITHTWODWORDINTREE(0x3ef00000,0x00000000)},
  {VT_exponent,11,0x1ee,DOUBLEWITHTWODWORDINTREE(0x3f000000,0x00000000)},
  {VT_exponent,10,0x5f,DOUBLEWITHTWODWORDINTREE(0x3f100000,0x00000000)},
  {VT_exponent,10,0x2b6,DOUBLEWITHTWODWORDINTREE(0x3f200000,0x00000000)},
  {VT_exponent,9,0x1de,DOUBLEWITHTWODWORDINTREE(0x3f300000,0x00000000)},
  {VT_double,10,0xd0,DOUBLEWITHTWODWORDINTREE(0x3f454c98,0x5f06f694)},
  {VT_double,6,0x9,DOUBLEWITHTWODWORDINTREE(0x3f4a36e2,0xeb1c432d)},
  {VT_exponent,8,0xe8,DOUBLEWITHTWODWORDINTREE(0x3f400000,0x00000000)},
  {VT_double,4,0xf,DOUBLEWITHTWODWORDINTREE(0x3f50624d,0xd2f1a9fc)},
  {VT_exponent,8,0xae,DOUBLEWITHTWODWORDINTREE(0x3f500000,0x00000000)},
  {VT_double,5,0x16,DOUBLEWITHTWODWORDINTREE(0x3f60624d,0xd2f1a9fc)},
  {VT_exponent,7,0x1b,DOUBLEWITHTWODWORDINTREE(0x3f600000,0x00000000)},
  {VT_exponent,7,0x76,DOUBLEWITHTWODWORDINTREE(0x3f700000,0x00000000)},
  {VT_exponent,7,0xa,DOUBLEWITHTWODWORDINTREE(0x3f800000,0x00000000)},
  {VT_exponent,6,0x8,DOUBLEWITHTWODWORDINTREE(0x3f900000,0x00000000)},
  {VT_exponent,6,0xe,DOUBLEWITHTWODWORDINTREE(0x3fa00000,0x00000000)},
  {VT_double,11,0x751,DOUBLEWITHTWODWORDINTREE(0x3fbe69ad,0x42c3c9ee)},
  {VT_exponent,6,0x4,DOUBLEWITHTWODWORDINTREE(0x3fb00000,0x00000000)},
  {VT_exponent,6,0xc,DOUBLEWITHTWODWORDINTREE(0x3fc00000,0x00000000)},
  {VT_exponent,5,0x3,DOUBLEWITHTWODWORDINTREE(0x3fd00000,0x00000000)},
  {VT_double,11,0x777,DOUBLEWITHTWODWORDINTREE(0x3fe00000,0x00000000)},
  {VT_double,9,0x1d6,DOUBLEWITHTWODWORDINTREE(0x3feffffe,0xf8000002)},
  {VT_exponent,4,0x8,DOUBLEWITHTWODWORDINTREE(0x3fe00000,0x00000000)},
  {VT_double,4,0x0,DOUBLEWITHTWODWORDINTREE(0x3ff00000,0x00000000)},
  {VT_exponent,5,0x13,DOUBLEWITHTWODWORDINTREE(0x3ff00000,0x00000000)},
  {VT_exponent,5,0x1b,DOUBLEWITHTWODWORDINTREE(0x40000000,0x00000000)},
  {VT_double,9,0x15a,DOUBLEWITHTWODWORDINTREE(0x401921fb,0x54442d18)},
  {VT_exponent,5,0x17,DOUBLEWITHTWODWORDINTREE(0x40100000,0x00000000)},
```

```
  {VT_exponent,5,0x12,DOUBLEWITHTWODWORDINTREE(0x40200000,0x00000000)},
  {VT_double,11,0x774,DOUBLEWITHTWODWORDINTREE(0x4035ee14,0x80000000)},
  {VT_exponent,5,0x19,DOUBLEWITHTWODWORDINTREE(0x40300000,0x00000000)},
  {VT_double,9,0x1d3,DOUBLEWITHTWODWORDINTREE(0x404ca5dc,0x1a63c1f8)},
  {VT_exponent,5,0x1a,DOUBLEWITHTWODWORDINTREE(0x40400000,0x00000000)},
  {VT_double,11,0x77e,DOUBLEWITHTWODWORDINTREE(0x405bb32f,0xe0000000)},
  {VT_double,10,0x5e,DOUBLEWITHTWODWORDINTREE(0x405c332f,0xe0000000)},
  {VT_exponent,5,0x18,DOUBLEWITHTWODWORDINTREE(0x40500000,0x00000000)},
  {VT_double,9,0x1d7,DOUBLEWITHTWODWORDINTREE(0x40668000,0x00000000)},
  {VT_exponent,5,0x1c,DOUBLEWITHTWODWORDINTREE(0x40600000,0x00000000)},
  {VT_double,9,0x1d5,DOUBLEWITHTWODWORDINTREE(0x40768000,0x00000000)},
  {VT_exponent,5,0x14,DOUBLEWITHTWODWORDINTREE(0x40700000,0x00000000)},
  {VT_double,11,0x77d,DOUBLEWITHTWODWORDINTREE(0x408f4000,0x00000000)},
  {VT_exponent,5,0x5,DOUBLEWITHTWODWORDINTREE(0x40800000,0x00000000)},
  {VT_double,10,0xd2,DOUBLEWITHTWODWORDINTREE(0x409233ff,0xffffffff)},
  {VT_double,8,0x3c,DOUBLEWITHTWODWORDINTREE(0x40923400,0x00000000)},
  {VT_double,11,0x753,DOUBLEWITHTWODWORDINTREE(0x40923400,0x00000001)},
  {VT_double,10,0xd3,DOUBLEWITHTWODWORDINTREE(0x4092abff,0xffffffff)},
  {VT_double,8,0x35,DOUBLEWITHTWODWORDINTREE(0x4092ac00,0x00000000)},
  {VT_double,11,0x770,DOUBLEWITHTWODWORDINTREE(0x4092ac00,0x00000001)},
  {VT_exponent,8,0x16,DOUBLEWITHTWODWORDINTREE(0x40900000,0x00000000)},
  {VT_exponent,12,0xee2,DOUBLEWITHTWODWORDINTREE(0x40a00000,0x00000000)},
  {VT_exponent,12,0xee4,DOUBLEWITHTWODWORDINTREE(0x40b00000,0x00000000)},
  {VT_double,7,0x1f,DOUBLEWITHTWODWORDINTREE(0x40c81c80,0x00000000)},
  {VT_exponent,8,0xac,DOUBLEWITHTWODWORDINTREE(0x40c00000,0x00000000)},
  {VT_exponent,13,0x15bb,DOUBLEWITHTWODWORDINTREE(0x40d00000,0x00000000)},
  {VT_exponent,22,0x3a8667,DOUBLEWITHTWODWORDINTREE(0x40e00000,0x00000000)},
  {VT_exponent,22,0x3a86d8,DOUBLEWITHTWODWORDINTREE(0x40f00000,0x00000000)},
  {VT_exponent,22,0x3a86d9,DOUBLEWITHTWODWORDINTREE(0x41000000,0x00000000)},
  {VT_exponent,22,0x3a86da,DOUBLEWITHTWODWORDINTREE(0x41100000,0x00000000)},
  {VT_exponent,17,0x1dc7e,DOUBLEWITHTWODWORDINTREE(0x41200000,0x00000000)},
  {VT_exponent,22,0x3a86db,DOUBLEWITHTWODWORDINTREE(0x41300000,0x00000000)},
  {VT_exponent,22,0x3a86dc,DOUBLEWITHTWODWORDINTREE(0x41400000,0x00000000)},
  {VT_exponent,22,0x3a86dd,DOUBLEWITHTWODWORDINTREE(0x41500000,0x00000000)},
  {VT_exponent,22,0x3a86de,DOUBLEWITHTWODWORDINTREE(0x41600000,0x00000000)},
  {VT_exponent,22,0x3a86df,DOUBLEWITHTWODWORDINTREE(0x41700000,0x00000000)},
  {VT_exponent,22,0x3a86f0,DOUBLEWITHTWODWORDINTREE(0x41800000,0x00000000)},
  {VT_exponent,22,0x3a86f1,DOUBLEWITHTWODWORDINTREE(0x41900000,0x00000000)},
  {VT_exponent,22,0x3a86f2,DOUBLEWITHTWODWORDINTREE(0x41a00000,0x00000000)},
  {VT_exponent,22,0x3a86f3,DOUBLEWITHTWODWORDINTREE(0x41b00000,0x00000000)},
  {VT_double,6,0x2a,DOUBLEWITHTWODWORDINTREE(0x41cdcd64,0xff800000)},
  {VT_exponent,22,0x3a86f4,DOUBLEWITHTWODWORDINTREE(0x41c00000,0x00000000)},
  {VT_exponent,22,0x3a86f5,DOUBLEWITHTWODWORDINTREE(0x41d00000,0x00000000)},
  {VT_exponent,22,0x3a86f6,DOUBLEWITHTWODWORDINTREE(0x41e00000,0x00000000)},
  {VT_exponent,22,0x3a86f7,DOUBLEWITHTWODWORDINTREE(0x41f00000,0x00000000)},
  {VT_exponent,22,0x3a86f8,DOUBLEWITHTWODWORDINTREE(0x42000000,0x00000000)},
  {VT_exponent,22,0x3a86f9,DOUBLEWITHTWODWORDINTREE(0x42100000,0x00000000)},
  {VT_exponent,22,0x3a86fa,DOUBLEWITHTWODWORDINTREE(0x42200000,0x00000000)},
  {VT_exponent,22,0x3a86fb,DOUBLEWITHTWODWORDINTREE(0x42300000,0x00000000)},
  {VT_exponent,22,0x3a86fc,DOUBLEWITHTWODWORDINTREE(0x42400000,0x00000000)},
  {VT_exponent,22,0x3a86fd,DOUBLEWITHTWODWORDINTREE(0x42500000,0x00000000)},
  {VT_exponent,22,0x3a86fe,DOUBLEWITHTWODWORDINTREE(0x42600000,0x00000000)},
  {VT_exponent,22,0x3a86ff,DOUBLEWITHTWODWORDINTREE(0x42700000,0x00000000)},
  {VT_exponent,22,0x3a8740,DOUBLEWITHTWODWORDINTREE(0x42800000,0x00000000)},
  {VT_exponent,22,0x3a8741,DOUBLEWITHTWODWORDINTREE(0x42900000,0x00000000)},
  {VT_exponent,22,0x3a8742,DOUBLEWITHTWODWORDINTREE(0x42a00000,0x00000000)},
  {VT_exponent,22,0x3a8743,DOUBLEWITHTWODWORDINTREE(0x42b00000,0x00000000)},
  {VT_exponent,22,0x3a8744,DOUBLEWITHTWODWORDINTREE(0x42c00000,0x00000000)},
  {VT_exponent,22,0x3a8745,DOUBLEWITHTWODWORDINTREE(0x42d00000,0x00000000)},
  {VT_exponent,22,0x3a8746,DOUBLEWITHTWODWORDINTREE(0x42e00000,0x00000000)},
  {VT_exponent,22,0x3a8747,DOUBLEWITHTWODWORDINTREE(0x42f00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8748,DOUBLEWITHTWODWORDINTREE(0x43000000,0x00000000)},
  {VT_exponent,22,0x3a8749,DOUBLEWITHTWODWORDINTREE(0x43100000,0x00000000)},
  {VT_exponent,22,0x3a874a,DOUBLEWITHTWODWORDINTREE(0x43200000,0x00000000)},
  {VT_exponent,22,0x3a874b,DOUBLEWITHTWODWORDINTREE(0x43300000,0x00000000)},
  {VT_exponent,22,0x3a874c,DOUBLEWITHTWODWORDINTREE(0x43400000,0x00000000)},
  {VT_exponent,22,0x3a874d,DOUBLEWITHTWODWORDINTREE(0x43500000,0x00000000)},
  {VT_exponent,22,0x3a874e,DOUBLEWITHTWODWORDINTREE(0x43600000,0x00000000)},
  {VT_exponent,22,0x3a874f,DOUBLEWITHTWODWORDINTREE(0x43700000,0x00000000)},
  {VT_exponent,22,0x3a8750,DOUBLEWITHTWODWORDINTREE(0x43800000,0x00000000)},
  {VT_exponent,22,0x3a8751,DOUBLEWITHTWODWORDINTREE(0x43900000,0x00000000)},
  {VT_exponent,22,0x3a8752,DOUBLEWITHTWODWORDINTREE(0x43a00000,0x00000000)},
  {VT_exponent,22,0x3a8753,DOUBLEWITHTWODWORDINTREE(0x43b00000,0x00000000)},
  {VT_exponent,22,0x3a8754,DOUBLEWITHTWODWORDINTREE(0x43c00000,0x00000000)},
  {VT_exponent,22,0x3a8755,DOUBLEWITHTWODWORDINTREE(0x43d00000,0x00000000)},
  {VT_exponent,22,0x3a8756,DOUBLEWITHTWODWORDINTREE(0x43e00000,0x00000000)},
  {VT_exponent,22,0x3a8757,DOUBLEWITHTWODWORDINTREE(0x43f00000,0x00000000)},
  {VT_exponent,22,0x3a8758,DOUBLEWITHTWODWORDINTREE(0x44000000,0x00000000)},
  {VT_exponent,15,0x1a3b,DOUBLEWITHTWODWORDINTREE(0x44100000,0x00000000)},
  {VT_exponent,22,0x3a8759,DOUBLEWITHTWODWORDINTREE(0x44200000,0x00000000)},
  {VT_exponent,22,0x3a875a,DOUBLEWITHTWODWORDINTREE(0x44300000,0x00000000)},
  {VT_exponent,22,0x3a875b,DOUBLEWITHTWODWORDINTREE(0x44400000,0x00000000)},
  {VT_exponent,22,0x3a875c,DOUBLEWITHTWODWORDINTREE(0x44500000,0x00000000)},
  {VT_exponent,22,0x3a875d,DOUBLEWITHTWODWORDINTREE(0x44600000,0x00000000)},
  {VT_exponent,22,0x3a875e,DOUBLEWITHTWODWORDINTREE(0x44700000,0x00000000)},
  {VT_exponent,22,0x3a875f,DOUBLEWITHTWODWORDINTREE(0x44800000,0x00000000)},
  {VT_exponent,22,0x3a8760,DOUBLEWITHTWODWORDINTREE(0x44900000,0x00000000)},
  {VT_exponent,22,0x3a8761,DOUBLEWITHTWODWORDINTREE(0x44a00000,0x00000000)},
  {VT_exponent,22,0x3a8762,DOUBLEWITHTWODWORDINTREE(0x44b00000,0x00000000)},
  {VT_exponent,22,0x3a8763,DOUBLEWITHTWODWORDINTREE(0x44c00000,0x00000000)},
  {VT_exponent,22,0x3a8764,DOUBLEWITHTWODWORDINTREE(0x44d00000,0x00000000)},
  {VT_exponent,22,0x3a8765,DOUBLEWITHTWODWORDINTREE(0x44e00000,0x00000000)},
  {VT_exponent,22,0x3a8766,DOUBLEWITHTWODWORDINTREE(0x44f00000,0x00000000)},
  {VT_exponent,22,0x3a8767,DOUBLEWITHTWODWORDINTREE(0x45000000,0x00000000)},
  {VT_exponent,22,0x3a8768,DOUBLEWITHTWODWORDINTREE(0x45100000,0x00000000)},
  {VT_exponent,22,0x3a8769,DOUBLEWITHTWODWORDINTREE(0x45200000,0x00000000)},
  {VT_exponent,22,0x3a876a,DOUBLEWITHTWODWORDINTREE(0x45300000,0x00000000)},
  {VT_exponent,22,0x3a876b,DOUBLEWITHTWODWORDINTREE(0x45400000,0x00000000)},
  {VT_exponent,22,0x3a876c,DOUBLEWITHTWODWORDINTREE(0x45500000,0x00000000)},
  {VT_exponent,22,0x3a876d,DOUBLEWITHTWODWORDINTREE(0x45600000,0x00000000)},
  {VT_exponent,22,0x3a876e,DOUBLEWITHTWODWORDINTREE(0x45700000,0x00000000)},
  {VT_exponent,22,0x3a876f,DOUBLEWITHTWODWORDINTREE(0x45800000,0x00000000)},
  {VT_exponent,22,0x3a8770,DOUBLEWITHTWODWORDINTREE(0x45900000,0x00000000)},
  {VT_exponent,22,0x3a8771,DOUBLEWITHTWODWORDINTREE(0x45a00000,0x00000000)},
  {VT_exponent,22,0x3a8772,DOUBLEWITHTWODWORDINTREE(0x45b00000,0x00000000)},
  {VT_exponent,22,0x3a8773,DOUBLEWITHTWODWORDINTREE(0x45c00000,0x00000000)},
  {VT_exponent,22,0x3a8774,DOUBLEWITHTWODWORDINTREE(0x45d00000,0x00000000)},
  {VT_exponent,22,0x3a8775,DOUBLEWITHTWODWORDINTREE(0x45e00000,0x00000000)},
  {VT_exponent,22,0x3a8776,DOUBLEWITHTWODWORDINTREE(0x45f00000,0x00000000)},
  {VT_exponent,22,0x3a8777,DOUBLEWITHTWODWORDINTREE(0x46000000,0x00000000)},
  {VT_exponent,22,0x3a8778,DOUBLEWITHTWODWORDINTREE(0x46100000,0x00000000)},
  {VT_exponent,22,0x3a8779,DOUBLEWITHTWODWORDINTREE(0x46200000,0x00000000)},
  {VT_exponent,22,0x3a877a,DOUBLEWITHTWODWORDINTREE(0x46300000,0x00000000)},
  {VT_exponent,22,0x3a877b,DOUBLEWITHTWODWORDINTREE(0x46400000,0x00000000)},
  {VT_exponent,22,0x3a877c,DOUBLEWITHTWODWORDINTREE(0x46500000,0x00000000)},
  {VT_exponent,22,0x3a877d,DOUBLEWITHTWODWORDINTREE(0x46600000,0x00000000)},
  {VT_exponent,22,0x3a877e,DOUBLEWITHTWODWORDINTREE(0x46700000,0x00000000)},
  {VT_exponent,22,0x3a877f,DOUBLEWITHTWODWORDINTREE(0x46800000,0x00000000)},
  {VT_exponent,22,0x3a8780,DOUBLEWITHTWODWORDINTREE(0x46900000,0x00000000)},
  {VT_exponent,22,0x3a8781,DOUBLEWITHTWODWORDINTREE(0x46a00000,0x00000000)},
  {VT_exponent,22,0x3a8782,DOUBLEWITHTWODWORDINTREE(0x46b00000,0x00000000)},
  {VT_exponent,22,0x3a8783,DOUBLEWITHTWODWORDINTREE(0x46c00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a8784,DOUBLEWITHTWODWORDINTREE(0x46d00000,0x00000000)},
  {VT_exponent,22,0x3a8785,DOUBLEWITHTWODWORDINTREE(0x46e00000,0x00000000)},
  {VT_exponent,22,0x3a8786,DOUBLEWITHTWODWORDINTREE(0x46f00000,0x00000000)},
  {VT_exponent,22,0x3a8787,DOUBLEWITHTWODWORDINTREE(0x47000000,0x00000000)},
  {VT_exponent,22,0x3a8788,DOUBLEWITHTWODWORDINTREE(0x47100000,0x00000000)},
  {VT_exponent,22,0x3a8789,DOUBLEWITHTWODWORDINTREE(0x47200000,0x00000000)},
  {VT_exponent,22,0x3a878a,DOUBLEWITHTWODWORDINTREE(0x47300000,0x00000000)},
  {VT_exponent,22,0x3a878b,DOUBLEWITHTWODWORDINTREE(0x47400000,0x00000000)},
  {VT_exponent,22,0x3a878c,DOUBLEWITHTWODWORDINTREE(0x47500000,0x00000000)},
  {VT_exponent,22,0x3a878d,DOUBLEWITHTWODWORDINTREE(0x47600000,0x00000000)},
  {VT_exponent,22,0x3a878e,DOUBLEWITHTWODWORDINTREE(0x47700000,0x00000000)},
  {VT_exponent,22,0x3a878f,DOUBLEWITHTWODWORDINTREE(0x47800000,0x00000000)},
  {VT_exponent,22,0x3a8790,DOUBLEWITHTWODWORDINTREE(0x47900000,0x00000000)},
  {VT_exponent,22,0x3a8791,DOUBLEWITHTWODWORDINTREE(0x47a00000,0x00000000)},
  {VT_exponent,22,0x3a8792,DOUBLEWITHTWODWORDINTREE(0x47b00000,0x00000000)},
  {VT_exponent,22,0x3a8793,DOUBLEWITHTWODWORDINTREE(0x47c00000,0x00000000)},
  {VT_exponent,22,0x3a8794,DOUBLEWITHTWODWORDINTREE(0x47d00000,0x00000000)},
  {VT_exponent,22,0x3a8795,DOUBLEWITHTWODWORDINTREE(0x47e00000,0x00000000)},
  {VT_exponent,22,0x3a8796,DOUBLEWITHTWODWORDINTREE(0x47f00000,0x00000000)},
  {VT_exponent,22,0x3a8797,DOUBLEWITHTWODWORDINTREE(0x48000000,0x00000000)},
  {VT_exponent,22,0x3a8798,DOUBLEWITHTWODWORDINTREE(0x48100000,0x00000000)},
  {VT_exponent,22,0x3a8799,DOUBLEWITHTWODWORDINTREE(0x48200000,0x00000000)},
  {VT_exponent,22,0x3a879a,DOUBLEWITHTWODWORDINTREE(0x48300000,0x00000000)},
  {VT_exponent,22,0x3a879b,DOUBLEWITHTWODWORDINTREE(0x48400000,0x00000000)},
  {VT_exponent,22,0x3a879c,DOUBLEWITHTWODWORDINTREE(0x48500000,0x00000000)},
  {VT_exponent,22,0x3a879d,DOUBLEWITHTWODWORDINTREE(0x48600000,0x00000000)},
  {VT_exponent,22,0x3a879e,DOUBLEWITHTWODWORDINTREE(0x48700000,0x00000000)},
  {VT_exponent,22,0x3a879f,DOUBLEWITHTWODWORDINTREE(0x48800000,0x00000000)},
  {VT_exponent,22,0x3a87a0,DOUBLEWITHTWODWORDINTREE(0x48900000,0x00000000)},
  {VT_exponent,22,0x3a87a1,DOUBLEWITHTWODWORDINTREE(0x48a00000,0x00000000)},
  {VT_exponent,22,0x3a87a2,DOUBLEWITHTWODWORDINTREE(0x48b00000,0x00000000)},
  {VT_exponent,22,0x3a87a3,DOUBLEWITHTWODWORDINTREE(0x48c00000,0x00000000)},
  {VT_exponent,22,0x3a87a4,DOUBLEWITHTWODWORDINTREE(0x48d00000,0x00000000)},
  {VT_exponent,22,0x3a87a5,DOUBLEWITHTWODWORDINTREE(0x48e00000,0x00000000)},
  {VT_exponent,22,0x3a87a6,DOUBLEWITHTWODWORDINTREE(0x48f00000,0x00000000)},
  {VT_exponent,22,0x3a87a7,DOUBLEWITHTWODWORDINTREE(0x49000000,0x00000000)},
  {VT_exponent,22,0x3a87a8,DOUBLEWITHTWODWORDINTREE(0x49100000,0x00000000)},
  {VT_exponent,22,0x3a87a9,DOUBLEWITHTWODWORDINTREE(0x49200000,0x00000000)},
  {VT_exponent,22,0x3a87aa,DOUBLEWITHTWODWORDINTREE(0x49300000,0x00000000)},
  {VT_exponent,22,0x3a87ab,DOUBLEWITHTWODWORDINTREE(0x49400000,0x00000000)},
  {VT_exponent,22,0x3a87ac,DOUBLEWITHTWODWORDINTREE(0x49500000,0x00000000)},
  {VT_exponent,22,0x3a87ad,DOUBLEWITHTWODWORDINTREE(0x49600000,0x00000000)},
  {VT_exponent,22,0x3a87ae,DOUBLEWITHTWODWORDINTREE(0x49700000,0x00000000)},
  {VT_exponent,22,0x3a87af,DOUBLEWITHTWODWORDINTREE(0x49800000,0x00000000)},
  {VT_exponent,22,0x3a87b0,DOUBLEWITHTWODWORDINTREE(0x49900000,0x00000000)},
  {VT_exponent,22,0x3a87b1,DOUBLEWITHTWODWORDINTREE(0x49a00000,0x00000000)},
  {VT_exponent,22,0x3a87b2,DOUBLEWITHTWODWORDINTREE(0x49b00000,0x00000000)},
  {VT_exponent,22,0x3a87b3,DOUBLEWITHTWODWORDINTREE(0x49c00000,0x00000000)},
  {VT_exponent,22,0x3a87b4,DOUBLEWITHTWODWORDINTREE(0x49d00000,0x00000000)},
  {VT_exponent,22,0x3a87b5,DOUBLEWITHTWODWORDINTREE(0x49e00000,0x00000000)},
  {VT_exponent,22,0x3a87b6,DOUBLEWITHTWODWORDINTREE(0x49f00000,0x00000000)},
  {VT_exponent,22,0x3a87b7,DOUBLEWITHTWODWORDINTREE(0x4a000000,0x00000000)},
  {VT_exponent,22,0x3a87b8,DOUBLEWITHTWODWORDINTREE(0x4a100000,0x00000000)},
  {VT_exponent,22,0x3a87b9,DOUBLEWITHTWODWORDINTREE(0x4a200000,0x00000000)},
  {VT_exponent,22,0x3a87ba,DOUBLEWITHTWODWORDINTREE(0x4a300000,0x00000000)},
  {VT_exponent,22,0x3a87bb,DOUBLEWITHTWODWORDINTREE(0x4a400000,0x00000000)},
  {VT_exponent,22,0x3a87bc,DOUBLEWITHTWODWORDINTREE(0x4a500000,0x00000000)},
  {VT_exponent,22,0x3a87bd,DOUBLEWITHTWODWORDINTREE(0x4a600000,0x00000000)},
  {VT_exponent,22,0x3a87be,DOUBLEWITHTWODWORDINTREE(0x4a700000,0x00000000)},
  {VT_exponent,22,0x3a87bf,DOUBLEWITHTWODWORDINTREE(0x4a800000,0x00000000)},
  {VT_exponent,22,0x3a87c0,DOUBLEWITHTWODWORDINTREE(0x4a900000,0x00000000)},
```

```
  {VT_exponent,22,0x3a87c1,DOUBLEWITHTWODWORDINTREE(0x4aa00000,0x00000000)},
  {VT_exponent,22,0x3a87c2,DOUBLEWITHTWODWORDINTREE(0x4ab00000,0x00000000)},
  {VT_exponent,22,0x3a87c3,DOUBLEWITHTWODWORDINTREE(0x4ac00000,0x00000000)},
  {VT_exponent,22,0x3a87c4,DOUBLEWITHTWODWORDINTREE(0x4ad00000,0x00000000)},
  {VT_exponent,22,0x3a87c5,DOUBLEWITHTWODWORDINTREE(0x4ae00000,0x00000000)},
  {VT_exponent,22,0x3a87c6,DOUBLEWITHTWODWORDINTREE(0x4af00000,0x00000000)},
  {VT_exponent,22,0x3a87c7,DOUBLEWITHTWODWORDINTREE(0x4b000000,0x00000000)},
  {VT_exponent,22,0x3a87c8,DOUBLEWITHTWODWORDINTREE(0x4b100000,0x00000000)},
  {VT_exponent,22,0x3a87c9,DOUBLEWITHTWODWORDINTREE(0x4b200000,0x00000000)},
  {VT_exponent,22,0x3a87ca,DOUBLEWITHTWODWORDINTREE(0x4b300000,0x00000000)},
  {VT_exponent,22,0x3a87cb,DOUBLEWITHTWODWORDINTREE(0x4b400000,0x00000000)},
  {VT_exponent,22,0x3a87cc,DOUBLEWITHTWODWORDINTREE(0x4b500000,0x00000000)},
  {VT_exponent,22,0x3a87cd,DOUBLEWITHTWODWORDINTREE(0x4b600000,0x00000000)},
  {VT_exponent,22,0x3a87ce,DOUBLEWITHTWODWORDINTREE(0x4b700000,0x00000000)},
  {VT_exponent,22,0x3a87cf,DOUBLEWITHTWODWORDINTREE(0x4b800000,0x00000000)},
  {VT_exponent,22,0x3a87d0,DOUBLEWITHTWODWORDINTREE(0x4b900000,0x00000000)},
  {VT_exponent,22,0x3a87d1,DOUBLEWITHTWODWORDINTREE(0x4ba00000,0x00000000)},
  {VT_exponent,22,0x3a87d2,DOUBLEWITHTWODWORDINTREE(0x4bb00000,0x00000000)},
  {VT_exponent,22,0x3a87d3,DOUBLEWITHTWODWORDINTREE(0x4bc00000,0x00000000)},
  {VT_exponent,22,0x3a87d4,DOUBLEWITHTWODWORDINTREE(0x4bd00000,0x00000000)},
  {VT_exponent,22,0x3a87d5,DOUBLEWITHTWODWORDINTREE(0x4be00000,0x00000000)},
  {VT_exponent,22,0x3a87d6,DOUBLEWITHTWODWORDINTREE(0x4bf00000,0x00000000)},
  {VT_exponent,22,0x3a87d7,DOUBLEWITHTWODWORDINTREE(0x4c000000,0x00000000)},
  {VT_exponent,22,0x3a87d8,DOUBLEWITHTWODWORDINTREE(0x4c100000,0x00000000)},
  {VT_exponent,22,0x3a87d9,DOUBLEWITHTWODWORDINTREE(0x4c200000,0x00000000)},
  {VT_exponent,22,0x3a87da,DOUBLEWITHTWODWORDINTREE(0x4c300000,0x00000000)},
  {VT_exponent,22,0x3a87db,DOUBLEWITHTWODWORDINTREE(0x4c400000,0x00000000)},
  {VT_exponent,22,0x3a87dc,DOUBLEWITHTWODWORDINTREE(0x4c500000,0x00000000)},
  {VT_exponent,22,0x3a87dd,DOUBLEWITHTWODWORDINTREE(0x4c600000,0x00000000)},
  {VT_exponent,22,0x3a87de,DOUBLEWITHTWODWORDINTREE(0x4c700000,0x00000000)},
  {VT_exponent,22,0x3a87df,DOUBLEWITHTWODWORDINTREE(0x4c800000,0x00000000)},
  {VT_exponent,22,0x3a87e0,DOUBLEWITHTWODWORDINTREE(0x4c900000,0x00000000)},
  {VT_exponent,22,0x3a87e1,DOUBLEWITHTWODWORDINTREE(0x4ca00000,0x00000000)},
  {VT_exponent,22,0x3a87e2,DOUBLEWITHTWODWORDINTREE(0x4cb00000,0x00000000)},
  {VT_exponent,22,0x3a87e3,DOUBLEWITHTWODWORDINTREE(0x4cc00000,0x00000000)},
  {VT_exponent,22,0x3a87e4,DOUBLEWITHTWODWORDINTREE(0x4cd00000,0x00000000)},
  {VT_exponent,22,0x3a87e5,DOUBLEWITHTWODWORDINTREE(0x4ce00000,0x00000000)},
  {VT_exponent,22,0x3a87e6,DOUBLEWITHTWODWORDINTREE(0x4cf00000,0x00000000)},
  {VT_exponent,22,0x3a87e7,DOUBLEWITHTWODWORDINTREE(0x4d000000,0x00000000)},
  {VT_exponent,22,0x3a87e8,DOUBLEWITHTWODWORDINTREE(0x4d100000,0x00000000)},
  {VT_exponent,22,0x3a87e9,DOUBLEWITHTWODWORDINTREE(0x4d200000,0x00000000)},
  {VT_exponent,22,0x3a87ea,DOUBLEWITHTWODWORDINTREE(0x4d300000,0x00000000)},
  {VT_exponent,22,0x3a87eb,DOUBLEWITHTWODWORDINTREE(0x4d400000,0x00000000)},
  {VT_exponent,22,0x3a87ec,DOUBLEWITHTWODWORDINTREE(0x4d500000,0x00000000)},
  {VT_exponent,22,0x3a87ed,DOUBLEWITHTWODWORDINTREE(0x4d600000,0x00000000)},
  {VT_exponent,22,0x3a87ee,DOUBLEWITHTWODWORDINTREE(0x4d700000,0x00000000)},
  {VT_exponent,22,0x3a87ef,DOUBLEWITHTWODWORDINTREE(0x4d800000,0x00000000)},
  {VT_exponent,22,0x3a87f0,DOUBLEWITHTWODWORDINTREE(0x4d900000,0x00000000)},
  {VT_exponent,22,0x3a87f1,DOUBLEWITHTWODWORDINTREE(0x4da00000,0x00000000)},
  {VT_exponent,22,0x3a87f2,DOUBLEWITHTWODWORDINTREE(0x4db00000,0x00000000)},
  {VT_exponent,22,0x3a87f3,DOUBLEWITHTWODWORDINTREE(0x4dc00000,0x00000000)},
  {VT_exponent,22,0x3a87f4,DOUBLEWITHTWODWORDINTREE(0x4dd00000,0x00000000)},
  {VT_exponent,22,0x3a87f5,DOUBLEWITHTWODWORDINTREE(0x4de00000,0x00000000)},
  {VT_exponent,22,0x3a87f6,DOUBLEWITHTWODWORDINTREE(0x4df00000,0x00000000)},
  {VT_exponent,22,0x3a87f7,DOUBLEWITHTWODWORDINTREE(0x4e000000,0x00000000)},
  {VT_exponent,22,0x3a87f8,DOUBLEWITHTWODWORDINTREE(0x4e100000,0x00000000)},
  {VT_exponent,22,0x3a87f9,DOUBLEWITHTWODWORDINTREE(0x4e200000,0x00000000)},
  {VT_exponent,22,0x3a87fa,DOUBLEWITHTWODWORDINTREE(0x4e300000,0x00000000)},
  {VT_exponent,22,0x3a87fb,DOUBLEWITHTWODWORDINTREE(0x4e400000,0x00000000)},
  {VT_exponent,22,0x3a87fc,DOUBLEWITHTWODWORDINTREE(0x4e500000,0x00000000)},
  {VT_exponent,22,0x3a87fd,DOUBLEWITHTWODWORDINTREE(0x4e600000,0x00000000)},
```

```
  {VT_exponent,22,0x3a87fe,DOUBLEWITHTWODWORDINTREE(0x4e700000,0x00000000)},
  {VT_exponent,22,0x3a87ff,DOUBLEWITHTWODWORDINTREE(0x4e800000,0x00000000)},
  {VT_exponent,22,0x3a9000,DOUBLEWITHTWODWORDINTREE(0x4e900000,0x00000000)},
  {VT_exponent,22,0x3a9001,DOUBLEWITHTWODWORDINTREE(0x4ea00000,0x00000000)},
  {VT_exponent,22,0x3a9002,DOUBLEWITHTWODWORDINTREE(0x4eb00000,0x00000000)},
  {VT_exponent,22,0x3a9003,DOUBLEWITHTWODWORDINTREE(0x4ec00000,0x00000000)},
  {VT_exponent,22,0x3a9004,DOUBLEWITHTWODWORDINTREE(0x4ed00000,0x00000000)},
  {VT_exponent,22,0x3a9005,DOUBLEWITHTWODWORDINTREE(0x4ee00000,0x00000000)},
  {VT_exponent,22,0x3a9006,DOUBLEWITHTWODWORDINTREE(0x4ef00000,0x00000000)},
  {VT_exponent,22,0x3a9007,DOUBLEWITHTWODWORDINTREE(0x4f000000,0x00000000)},
  {VT_exponent,22,0x3a9008,DOUBLEWITHTWODWORDINTREE(0x4f100000,0x00000000)},
  {VT_exponent,22,0x3a9009,DOUBLEWITHTWODWORDINTREE(0x4f200000,0x00000000)},
  {VT_exponent,22,0x3a900a,DOUBLEWITHTWODWORDINTREE(0x4f300000,0x00000000)},
  {VT_exponent,22,0x3a900b,DOUBLEWITHTWODWORDINTREE(0x4f400000,0x00000000)},
  {VT_exponent,22,0x3a900c,DOUBLEWITHTWODWORDINTREE(0x4f500000,0x00000000)},
  {VT_exponent,22,0x3a900d,DOUBLEWITHTWODWORDINTREE(0x4f600000,0x00000000)},
  {VT_exponent,22,0x3a900e,DOUBLEWITHTWODWORDINTREE(0x4f700000,0x00000000)},
  {VT_exponent,22,0x3a900f,DOUBLEWITHTWODWORDINTREE(0x4f800000,0x00000000)},
  {VT_exponent,22,0x3a9010,DOUBLEWITHTWODWORDINTREE(0x4f900000,0x00000000)},
  {VT_exponent,22,0x3a9011,DOUBLEWITHTWODWORDINTREE(0x4fa00000,0x00000000)},
  {VT_exponent,22,0x3a9012,DOUBLEWITHTWODWORDINTREE(0x4fb00000,0x00000000)},
  {VT_exponent,22,0x3a9013,DOUBLEWITHTWODWORDINTREE(0x4fc00000,0x00000000)},
  {VT_exponent,22,0x3a9014,DOUBLEWITHTWODWORDINTREE(0x4fd00000,0x00000000)},
  {VT_exponent,22,0x3a9015,DOUBLEWITHTWODWORDINTREE(0x4fe00000,0x00000000)},
  {VT_exponent,22,0x3a9016,DOUBLEWITHTWODWORDINTREE(0x4ff00000,0x00000000)},
  {VT_exponent,22,0x3a9017,DOUBLEWITHTWODWORDINTREE(0x50000000,0x00000000)},
  {VT_exponent,22,0x3a9018,DOUBLEWITHTWODWORDINTREE(0x50100000,0x00000000)},
  {VT_exponent,22,0x3a9019,DOUBLEWITHTWODWORDINTREE(0x50200000,0x00000000)},
  {VT_exponent,22,0x3a901a,DOUBLEWITHTWODWORDINTREE(0x50300000,0x00000000)},
  {VT_exponent,22,0x3a901b,DOUBLEWITHTWODWORDINTREE(0x50400000,0x00000000)},
  {VT_exponent,22,0x3a901c,DOUBLEWITHTWODWORDINTREE(0x50500000,0x00000000)},
  {VT_exponent,22,0x3a901d,DOUBLEWITHTWODWORDINTREE(0x50600000,0x00000000)},
  {VT_exponent,22,0x3a901e,DOUBLEWITHTWODWORDINTREE(0x50700000,0x00000000)},
  {VT_exponent,22,0x3a901f,DOUBLEWITHTWODWORDINTREE(0x50800000,0x00000000)},
  {VT_exponent,22,0x3a9020,DOUBLEWITHTWODWORDINTREE(0x50900000,0x00000000)},
  {VT_exponent,22,0x3a9021,DOUBLEWITHTWODWORDINTREE(0x50a00000,0x00000000)},
  {VT_exponent,22,0x3a9022,DOUBLEWITHTWODWORDINTREE(0x50b00000,0x00000000)},
  {VT_exponent,22,0x3a9023,DOUBLEWITHTWODWORDINTREE(0x50c00000,0x00000000)},
  {VT_exponent,22,0x3a9024,DOUBLEWITHTWODWORDINTREE(0x50d00000,0x00000000)},
  {VT_exponent,22,0x3a9025,DOUBLEWITHTWODWORDINTREE(0x50e00000,0x00000000)},
  {VT_exponent,22,0x3a9026,DOUBLEWITHTWODWORDINTREE(0x50f00000,0x00000000)},
  {VT_exponent,22,0x3a9027,DOUBLEWITHTWODWORDINTREE(0x51000000,0x00000000)},
  {VT_exponent,22,0x3a9028,DOUBLEWITHTWODWORDINTREE(0x51100000,0x00000000)},
  {VT_exponent,22,0x3a9029,DOUBLEWITHTWODWORDINTREE(0x51200000,0x00000000)},
  {VT_exponent,22,0x3a902a,DOUBLEWITHTWODWORDINTREE(0x51300000,0x00000000)},
  {VT_exponent,22,0x3a902b,DOUBLEWITHTWODWORDINTREE(0x51400000,0x00000000)},
  {VT_exponent,22,0x3a902c,DOUBLEWITHTWODWORDINTREE(0x51500000,0x00000000)},
  {VT_exponent,22,0x3a902d,DOUBLEWITHTWODWORDINTREE(0x51600000,0x00000000)},
  {VT_exponent,22,0x3a902e,DOUBLEWITHTWODWORDINTREE(0x51700000,0x00000000)},
  {VT_exponent,22,0x3a902f,DOUBLEWITHTWODWORDINTREE(0x51800000,0x00000000)},
  {VT_exponent,22,0x3a9030,DOUBLEWITHTWODWORDINTREE(0x51900000,0x00000000)},
  {VT_exponent,22,0x3a9031,DOUBLEWITHTWODWORDINTREE(0x51a00000,0x00000000)},
  {VT_exponent,22,0x3a9032,DOUBLEWITHTWODWORDINTREE(0x51b00000,0x00000000)},
  {VT_exponent,22,0x3a9033,DOUBLEWITHTWODWORDINTREE(0x51c00000,0x00000000)},
  {VT_exponent,22,0x3a9034,DOUBLEWITHTWODWORDINTREE(0x51d00000,0x00000000)},
  {VT_exponent,22,0x3a9035,DOUBLEWITHTWODWORDINTREE(0x51e00000,0x00000000)},
  {VT_exponent,22,0x3a9036,DOUBLEWITHTWODWORDINTREE(0x51f00000,0x00000000)},
  {VT_exponent,22,0x3a9037,DOUBLEWITHTWODWORDINTREE(0x52000000,0x00000000)},
  {VT_exponent,22,0x3a9038,DOUBLEWITHTWODWORDINTREE(0x52100000,0x00000000)},
  {VT_exponent,22,0x3a9039,DOUBLEWITHTWODWORDINTREE(0x52200000,0x00000000)},
  {VT_exponent,22,0x3a903a,DOUBLEWITHTWODWORDINTREE(0x52300000,0x00000000)},
```

```
  {VT_exponent,22,0x3a903b,DOUBLEWITHTWODWORDINTREE(0x52400000,0x00000000)},
  {VT_exponent,22,0x3a903c,DOUBLEWITHTWODWORDINTREE(0x52500000,0x00000000)},
  {VT_exponent,22,0x3a903d,DOUBLEWITHTWODWORDINTREE(0x52600000,0x00000000)},
  {VT_exponent,22,0x3a903e,DOUBLEWITHTWODWORDINTREE(0x52700000,0x00000000)},
  {VT_exponent,22,0x3a903f,DOUBLEWITHTWODWORDINTREE(0x52800000,0x00000000)},
  {VT_exponent,22,0x3a9040,DOUBLEWITHTWODWORDINTREE(0x52900000,0x00000000)},
  {VT_exponent,22,0x3a9041,DOUBLEWITHTWODWORDINTREE(0x52a00000,0x00000000)},
  {VT_exponent,22,0x3a9042,DOUBLEWITHTWODWORDINTREE(0x52b00000,0x00000000)},
  {VT_exponent,22,0x3a9043,DOUBLEWITHTWODWORDINTREE(0x52c00000,0x00000000)},
  {VT_exponent,22,0x3a9044,DOUBLEWITHTWODWORDINTREE(0x52d00000,0x00000000)},
  {VT_exponent,22,0x3a9045,DOUBLEWITHTWODWORDINTREE(0x52e00000,0x00000000)},
  {VT_exponent,22,0x3a9046,DOUBLEWITHTWODWORDINTREE(0x52f00000,0x00000000)},
  {VT_exponent,22,0x3a9047,DOUBLEWITHTWODWORDINTREE(0x53000000,0x00000000)},
  {VT_exponent,22,0x3a9048,DOUBLEWITHTWODWORDINTREE(0x53100000,0x00000000)},
  {VT_exponent,22,0x3a9049,DOUBLEWITHTWODWORDINTREE(0x53200000,0x00000000)},
  {VT_exponent,22,0x3a904a,DOUBLEWITHTWODWORDINTREE(0x53300000,0x00000000)},
  {VT_exponent,22,0x3a904b,DOUBLEWITHTWODWORDINTREE(0x53400000,0x00000000)},
  {VT_exponent,22,0x3a904c,DOUBLEWITHTWODWORDINTREE(0x53500000,0x00000000)},
  {VT_exponent,22,0x3a904d,DOUBLEWITHTWODWORDINTREE(0x53600000,0x00000000)},
  {VT_exponent,22,0x3a904e,DOUBLEWITHTWODWORDINTREE(0x53700000,0x00000000)},
  {VT_exponent,22,0x3a904f,DOUBLEWITHTWODWORDINTREE(0x53800000,0x00000000)},
  {VT_exponent,22,0x3a9050,DOUBLEWITHTWODWORDINTREE(0x53900000,0x00000000)},
  {VT_exponent,22,0x3a9051,DOUBLEWITHTWODWORDINTREE(0x53a00000,0x00000000)},
  {VT_exponent,22,0x3a9052,DOUBLEWITHTWODWORDINTREE(0x53b00000,0x00000000)},
  {VT_exponent,22,0x3a9053,DOUBLEWITHTWODWORDINTREE(0x53c00000,0x00000000)},
  {VT_exponent,22,0x3a9054,DOUBLEWITHTWODWORDINTREE(0x53d00000,0x00000000)},
  {VT_exponent,22,0x3a9055,DOUBLEWITHTWODWORDINTREE(0x53e00000,0x00000000)},
  {VT_exponent,22,0x3a9056,DOUBLEWITHTWODWORDINTREE(0x53f00000,0x00000000)},
  {VT_exponent,22,0x3a9057,DOUBLEWITHTWODWORDINTREE(0x54000000,0x00000000)},
  {VT_exponent,22,0x3a9058,DOUBLEWITHTWODWORDINTREE(0x54100000,0x00000000)},
  {VT_exponent,22,0x3a9059,DOUBLEWITHTWODWORDINTREE(0x54200000,0x00000000)},
  {VT_exponent,22,0x3a905a,DOUBLEWITHTWODWORDINTREE(0x54300000,0x00000000)},
  {VT_exponent,22,0x3a905b,DOUBLEWITHTWODWORDINTREE(0x54400000,0x00000000)},
  {VT_exponent,22,0x3a905c,DOUBLEWITHTWODWORDINTREE(0x54500000,0x00000000)},
  {VT_exponent,22,0x3a905d,DOUBLEWITHTWODWORDINTREE(0x54600000,0x00000000)},
  {VT_exponent,22,0x3a905e,DOUBLEWITHTWODWORDINTREE(0x54700000,0x00000000)},
  {VT_exponent,22,0x3a905f,DOUBLEWITHTWODWORDINTREE(0x54800000,0x00000000)},
  {VT_exponent,22,0x3a9060,DOUBLEWITHTWODWORDINTREE(0x54900000,0x00000000)},
  {VT_exponent,22,0x3a9061,DOUBLEWITHTWODWORDINTREE(0x54a00000,0x00000000)},
  {VT_exponent,22,0x3a9062,DOUBLEWITHTWODWORDINTREE(0x54b00000,0x00000000)},
  {VT_exponent,22,0x3a9063,DOUBLEWITHTWODWORDINTREE(0x54c00000,0x00000000)},
  {VT_exponent,22,0x3a9064,DOUBLEWITHTWODWORDINTREE(0x54d00000,0x00000000)},
  {VT_exponent,22,0x3a9065,DOUBLEWITHTWODWORDINTREE(0x54e00000,0x00000000)},
  {VT_exponent,22,0x3a9066,DOUBLEWITHTWODWORDINTREE(0x54f00000,0x00000000)},
  {VT_exponent,22,0x3a9067,DOUBLEWITHTWODWORDINTREE(0x55000000,0x00000000)},
  {VT_exponent,22,0x3a9068,DOUBLEWITHTWODWORDINTREE(0x55100000,0x00000000)},
  {VT_exponent,22,0x3a9069,DOUBLEWITHTWODWORDINTREE(0x55200000,0x00000000)},
  {VT_exponent,22,0x3a906a,DOUBLEWITHTWODWORDINTREE(0x55300000,0x00000000)},
  {VT_exponent,22,0x3a906b,DOUBLEWITHTWODWORDINTREE(0x55400000,0x00000000)},
  {VT_exponent,22,0x3a906c,DOUBLEWITHTWODWORDINTREE(0x55500000,0x00000000)},
  {VT_exponent,22,0x3a906d,DOUBLEWITHTWODWORDINTREE(0x55600000,0x00000000)},
  {VT_exponent,22,0x3a906e,DOUBLEWITHTWODWORDINTREE(0x55700000,0x00000000)},
  {VT_exponent,22,0x3a906f,DOUBLEWITHTWODWORDINTREE(0x55800000,0x00000000)},
  {VT_exponent,22,0x3a9070,DOUBLEWITHTWODWORDINTREE(0x55900000,0x00000000)},
  {VT_exponent,22,0x3a9071,DOUBLEWITHTWODWORDINTREE(0x55a00000,0x00000000)},
  {VT_exponent,22,0x3a9072,DOUBLEWITHTWODWORDINTREE(0x55b00000,0x00000000)},
  {VT_exponent,22,0x3a9073,DOUBLEWITHTWODWORDINTREE(0x55c00000,0x00000000)},
  {VT_exponent,22,0x3a9074,DOUBLEWITHTWODWORDINTREE(0x55d00000,0x00000000)},
  {VT_exponent,22,0x3a9075,DOUBLEWITHTWODWORDINTREE(0x55e00000,0x00000000)},
  {VT_exponent,22,0x3a9076,DOUBLEWITHTWODWORDINTREE(0x55f00000,0x00000000)},
  {VT_exponent,22,0x3a9077,DOUBLEWITHTWODWORDINTREE(0x56000000,0x00000000)},
```

```
  {VT_exponent,22,0x3a9078,DOUBLEWITHTWODWORDINTREE(0x56100000,0x00000000)},
  {VT_exponent,22,0x3a9079,DOUBLEWITHTWODWORDINTREE(0x56200000,0x00000000)},
  {VT_exponent,22,0x3a907a,DOUBLEWITHTWODWORDINTREE(0x56300000,0x00000000)},
  {VT_exponent,22,0x3a907b,DOUBLEWITHTWODWORDINTREE(0x56400000,0x00000000)},
  {VT_exponent,22,0x3a907c,DOUBLEWITHTWODWORDINTREE(0x56500000,0x00000000)},
  {VT_exponent,22,0x3a907d,DOUBLEWITHTWODWORDINTREE(0x56600000,0x00000000)},
  {VT_exponent,22,0x3a907e,DOUBLEWITHTWODWORDINTREE(0x56700000,0x00000000)},
  {VT_exponent,22,0x3a907f,DOUBLEWITHTWODWORDINTREE(0x56800000,0x00000000)},
  {VT_exponent,22,0x3a9080,DOUBLEWITHTWODWORDINTREE(0x56900000,0x00000000)},
  {VT_exponent,22,0x3a9081,DOUBLEWITHTWODWORDINTREE(0x56a00000,0x00000000)},
  {VT_exponent,22,0x3a9082,DOUBLEWITHTWODWORDINTREE(0x56b00000,0x00000000)},
  {VT_exponent,22,0x3a9083,DOUBLEWITHTWODWORDINTREE(0x56c00000,0x00000000)},
  {VT_exponent,22,0x3a9084,DOUBLEWITHTWODWORDINTREE(0x56d00000,0x00000000)},
  {VT_exponent,22,0x3a9085,DOUBLEWITHTWODWORDINTREE(0x56e00000,0x00000000)},
  {VT_exponent,22,0x3a9086,DOUBLEWITHTWODWORDINTREE(0x56f00000,0x00000000)},
  {VT_exponent,22,0x3a9087,DOUBLEWITHTWODWORDINTREE(0x57000000,0x00000000)},
  {VT_exponent,22,0x3a9088,DOUBLEWITHTWODWORDINTREE(0x57100000,0x00000000)},
  {VT_exponent,22,0x3a9089,DOUBLEWITHTWODWORDINTREE(0x57200000,0x00000000)},
  {VT_exponent,22,0x3a908a,DOUBLEWITHTWODWORDINTREE(0x57300000,0x00000000)},
  {VT_exponent,22,0x3a908b,DOUBLEWITHTWODWORDINTREE(0x57400000,0x00000000)},
  {VT_exponent,22,0x3a908c,DOUBLEWITHTWODWORDINTREE(0x57500000,0x00000000)},
  {VT_exponent,22,0x3a908d,DOUBLEWITHTWODWORDINTREE(0x57600000,0x00000000)},
  {VT_exponent,22,0x3a908e,DOUBLEWITHTWODWORDINTREE(0x57700000,0x00000000)},
  {VT_exponent,22,0x3a908f,DOUBLEWITHTWODWORDINTREE(0x57800000,0x00000000)},
  {VT_exponent,22,0x3a9090,DOUBLEWITHTWODWORDINTREE(0x57900000,0x00000000)},
  {VT_exponent,22,0x3a9091,DOUBLEWITHTWODWORDINTREE(0x57a00000,0x00000000)},
  {VT_exponent,22,0x3a9092,DOUBLEWITHTWODWORDINTREE(0x57b00000,0x00000000)},
  {VT_exponent,22,0x3a9093,DOUBLEWITHTWODWORDINTREE(0x57c00000,0x00000000)},
  {VT_exponent,22,0x3a9094,DOUBLEWITHTWODWORDINTREE(0x57d00000,0x00000000)},
  {VT_exponent,22,0x3a9095,DOUBLEWITHTWODWORDINTREE(0x57e00000,0x00000000)},
  {VT_exponent,22,0x3a9096,DOUBLEWITHTWODWORDINTREE(0x57f00000,0x00000000)},
  {VT_exponent,22,0x3a9097,DOUBLEWITHTWODWORDINTREE(0x58000000,0x00000000)},
  {VT_exponent,22,0x3a9098,DOUBLEWITHTWODWORDINTREE(0x58100000,0x00000000)},
  {VT_exponent,22,0x3a9099,DOUBLEWITHTWODWORDINTREE(0x58200000,0x00000000)},
  {VT_exponent,22,0x3a909a,DOUBLEWITHTWODWORDINTREE(0x58300000,0x00000000)},
  {VT_exponent,22,0x3a909b,DOUBLEWITHTWODWORDINTREE(0x58400000,0x00000000)},
  {VT_exponent,22,0x3a909c,DOUBLEWITHTWODWORDINTREE(0x58500000,0x00000000)},
  {VT_exponent,22,0x3a909d,DOUBLEWITHTWODWORDINTREE(0x58600000,0x00000000)},
  {VT_exponent,22,0x3a909e,DOUBLEWITHTWODWORDINTREE(0x58700000,0x00000000)},
  {VT_exponent,22,0x3a909f,DOUBLEWITHTWODWORDINTREE(0x58800000,0x00000000)},
  {VT_exponent,22,0x3a90a0,DOUBLEWITHTWODWORDINTREE(0x58900000,0x00000000)},
  {VT_exponent,22,0x3a90a1,DOUBLEWITHTWODWORDINTREE(0x58a00000,0x00000000)},
  {VT_exponent,22,0x3a90a2,DOUBLEWITHTWODWORDINTREE(0x58b00000,0x00000000)},
  {VT_exponent,22,0x3a90a3,DOUBLEWITHTWODWORDINTREE(0x58c00000,0x00000000)},
  {VT_exponent,22,0x3a90a4,DOUBLEWITHTWODWORDINTREE(0x58d00000,0x00000000)},
  {VT_exponent,22,0x3a90a5,DOUBLEWITHTWODWORDINTREE(0x58e00000,0x00000000)},
  {VT_exponent,22,0x3a90a6,DOUBLEWITHTWODWORDINTREE(0x58f00000,0x00000000)},
  {VT_exponent,22,0x3a90a7,DOUBLEWITHTWODWORDINTREE(0x59000000,0x00000000)},
  {VT_exponent,22,0x3a90a8,DOUBLEWITHTWODWORDINTREE(0x59100000,0x00000000)},
  {VT_exponent,22,0x3a90a9,DOUBLEWITHTWODWORDINTREE(0x59200000,0x00000000)},
  {VT_exponent,22,0x3a90aa,DOUBLEWITHTWODWORDINTREE(0x59300000,0x00000000)},
  {VT_exponent,22,0x3a90ab,DOUBLEWITHTWODWORDINTREE(0x59400000,0x00000000)},
  {VT_exponent,22,0x3a90ac,DOUBLEWITHTWODWORDINTREE(0x59500000,0x00000000)},
  {VT_exponent,22,0x3a90ad,DOUBLEWITHTWODWORDINTREE(0x59600000,0x00000000)},
  {VT_exponent,22,0x3a90ae,DOUBLEWITHTWODWORDINTREE(0x59700000,0x00000000)},
  {VT_exponent,22,0x3a90af,DOUBLEWITHTWODWORDINTREE(0x59800000,0x00000000)},
  {VT_exponent,22,0x3a90b0,DOUBLEWITHTWODWORDINTREE(0x59900000,0x00000000)},
  {VT_exponent,22,0x3a90b1,DOUBLEWITHTWODWORDINTREE(0x59a00000,0x00000000)},
  {VT_exponent,22,0x3a90b2,DOUBLEWITHTWODWORDINTREE(0x59b00000,0x00000000)},
  {VT_exponent,22,0x3a90b3,DOUBLEWITHTWODWORDINTREE(0x59c00000,0x00000000)},
  {VT_exponent,22,0x3a90b4,DOUBLEWITHTWODWORDINTREE(0x59d00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a90b5,DOUBLEWITHTWODWORDINTREE(0x59e00000,0x00000000)},
  {VT_exponent,22,0x3a90b6,DOUBLEWITHTWODWORDINTREE(0x59f00000,0x00000000)},
  {VT_exponent,22,0x3a90b7,DOUBLEWITHTWODWORDINTREE(0x5a000000,0x00000000)},
  {VT_exponent,22,0x3a90b8,DOUBLEWITHTWODWORDINTREE(0x5a100000,0x00000000)},
  {VT_exponent,22,0x3a90b9,DOUBLEWITHTWODWORDINTREE(0x5a200000,0x00000000)},
  {VT_exponent,22,0x3a90ba,DOUBLEWITHTWODWORDINTREE(0x5a300000,0x00000000)},
  {VT_exponent,22,0x3a90bb,DOUBLEWITHTWODWORDINTREE(0x5a400000,0x00000000)},
  {VT_exponent,22,0x3a90bc,DOUBLEWITHTWODWORDINTREE(0x5a500000,0x00000000)},
  {VT_exponent,22,0x3a90bd,DOUBLEWITHTWODWORDINTREE(0x5a600000,0x00000000)},
  {VT_exponent,22,0x3a90be,DOUBLEWITHTWODWORDINTREE(0x5a700000,0x00000000)},
  {VT_exponent,22,0x3a90bf,DOUBLEWITHTWODWORDINTREE(0x5a800000,0x00000000)},
  {VT_exponent,22,0x3a90c0,DOUBLEWITHTWODWORDINTREE(0x5a900000,0x00000000)},
  {VT_exponent,22,0x3a90c1,DOUBLEWITHTWODWORDINTREE(0x5aa00000,0x00000000)},
  {VT_exponent,22,0x3a90c2,DOUBLEWITHTWODWORDINTREE(0x5ab00000,0x00000000)},
  {VT_exponent,22,0x3a90c3,DOUBLEWITHTWODWORDINTREE(0x5ac00000,0x00000000)},
  {VT_exponent,22,0x3a90c4,DOUBLEWITHTWODWORDINTREE(0x5ad00000,0x00000000)},
  {VT_exponent,22,0x3a90c5,DOUBLEWITHTWODWORDINTREE(0x5ae00000,0x00000000)},
  {VT_exponent,22,0x3a90c6,DOUBLEWITHTWODWORDINTREE(0x5af00000,0x00000000)},
  {VT_exponent,22,0x3a90c7,DOUBLEWITHTWODWORDINTREE(0x5b000000,0x00000000)},
  {VT_exponent,22,0x3a90c8,DOUBLEWITHTWODWORDINTREE(0x5b100000,0x00000000)},
  {VT_exponent,22,0x3a90c9,DOUBLEWITHTWODWORDINTREE(0x5b200000,0x00000000)},
  {VT_exponent,22,0x3a90ca,DOUBLEWITHTWODWORDINTREE(0x5b300000,0x00000000)},
  {VT_exponent,22,0x3a90cb,DOUBLEWITHTWODWORDINTREE(0x5b400000,0x00000000)},
  {VT_exponent,22,0x3a90cc,DOUBLEWITHTWODWORDINTREE(0x5b500000,0x00000000)},
  {VT_exponent,22,0x3a90cd,DOUBLEWITHTWODWORDINTREE(0x5b600000,0x00000000)},
  {VT_exponent,22,0x3a90ce,DOUBLEWITHTWODWORDINTREE(0x5b700000,0x00000000)},
  {VT_exponent,22,0x3a90cf,DOUBLEWITHTWODWORDINTREE(0x5b800000,0x00000000)},
  {VT_exponent,22,0x3a90d0,DOUBLEWITHTWODWORDINTREE(0x5b900000,0x00000000)},
  {VT_exponent,22,0x3a90d1,DOUBLEWITHTWODWORDINTREE(0x5ba00000,0x00000000)},
  {VT_exponent,22,0x3a90d2,DOUBLEWITHTWODWORDINTREE(0x5bb00000,0x00000000)},
  {VT_exponent,22,0x3a90d3,DOUBLEWITHTWODWORDINTREE(0x5bc00000,0x00000000)},
  {VT_exponent,22,0x3a90d4,DOUBLEWITHTWODWORDINTREE(0x5bd00000,0x00000000)},
  {VT_exponent,22,0x3a90d5,DOUBLEWITHTWODWORDINTREE(0x5be00000,0x00000000)},
  {VT_exponent,22,0x3a90d6,DOUBLEWITHTWODWORDINTREE(0x5bf00000,0x00000000)},
  {VT_exponent,22,0x3a90d7,DOUBLEWITHTWODWORDINTREE(0x5c000000,0x00000000)},
  {VT_exponent,22,0x3a90d8,DOUBLEWITHTWODWORDINTREE(0x5c100000,0x00000000)},
  {VT_exponent,22,0x3a90d9,DOUBLEWITHTWODWORDINTREE(0x5c200000,0x00000000)},
  {VT_exponent,22,0x3a90da,DOUBLEWITHTWODWORDINTREE(0x5c300000,0x00000000)},
  {VT_exponent,22,0x3a90db,DOUBLEWITHTWODWORDINTREE(0x5c400000,0x00000000)},
  {VT_exponent,22,0x3a90dc,DOUBLEWITHTWODWORDINTREE(0x5c500000,0x00000000)},
  {VT_exponent,22,0x3a90dd,DOUBLEWITHTWODWORDINTREE(0x5c600000,0x00000000)},
  {VT_exponent,22,0x3a90de,DOUBLEWITHTWODWORDINTREE(0x5c700000,0x00000000)},
  {VT_exponent,22,0x3a90df,DOUBLEWITHTWODWORDINTREE(0x5c800000,0x00000000)},
  {VT_exponent,22,0x3a90e0,DOUBLEWITHTWODWORDINTREE(0x5c900000,0x00000000)},
  {VT_exponent,22,0x3a90e1,DOUBLEWITHTWODWORDINTREE(0x5ca00000,0x00000000)},
  {VT_exponent,22,0x3a90e2,DOUBLEWITHTWODWORDINTREE(0x5cb00000,0x00000000)},
  {VT_exponent,22,0x3a90e3,DOUBLEWITHTWODWORDINTREE(0x5cc00000,0x00000000)},
  {VT_exponent,22,0x3a90e4,DOUBLEWITHTWODWORDINTREE(0x5cd00000,0x00000000)},
  {VT_exponent,22,0x3a90e5,DOUBLEWITHTWODWORDINTREE(0x5ce00000,0x00000000)},
  {VT_exponent,22,0x3a90e6,DOUBLEWITHTWODWORDINTREE(0x5cf00000,0x00000000)},
  {VT_exponent,22,0x3a90e7,DOUBLEWITHTWODWORDINTREE(0x5d000000,0x00000000)},
  {VT_exponent,22,0x3a90e8,DOUBLEWITHTWODWORDINTREE(0x5d100000,0x00000000)},
  {VT_exponent,22,0x3a90e9,DOUBLEWITHTWODWORDINTREE(0x5d200000,0x00000000)},
  {VT_exponent,22,0x3a90ea,DOUBLEWITHTWODWORDINTREE(0x5d300000,0x00000000)},
  {VT_exponent,22,0x3a90eb,DOUBLEWITHTWODWORDINTREE(0x5d400000,0x00000000)},
  {VT_exponent,22,0x3a90ec,DOUBLEWITHTWODWORDINTREE(0x5d500000,0x00000000)},
  {VT_exponent,22,0x3a90ed,DOUBLEWITHTWODWORDINTREE(0x5d600000,0x00000000)},
  {VT_exponent,22,0x3a90ee,DOUBLEWITHTWODWORDINTREE(0x5d700000,0x00000000)},
  {VT_exponent,22,0x3a90ef,DOUBLEWITHTWODWORDINTREE(0x5d800000,0x00000000)},
  {VT_exponent,22,0x3a90f0,DOUBLEWITHTWODWORDINTREE(0x5d900000,0x00000000)},
  {VT_exponent,22,0x3a90f1,DOUBLEWITHTWODWORDINTREE(0x5da00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a90f2,DOUBLEWITHTWODWORDINTREE(0x5db00000,0x00000000)},
  {VT_exponent,22,0x3a90f3,DOUBLEWITHTWODWORDINTREE(0x5dc00000,0x00000000)},
  {VT_exponent,22,0x3a90f4,DOUBLEWITHTWODWORDINTREE(0x5dd00000,0x00000000)},
  {VT_exponent,22,0x3a90f5,DOUBLEWITHTWODWORDINTREE(0x5de00000,0x00000000)},
  {VT_exponent,22,0x3a90f6,DOUBLEWITHTWODWORDINTREE(0x5df00000,0x00000000)},
  {VT_exponent,22,0x3a90f7,DOUBLEWITHTWODWORDINTREE(0x5e000000,0x00000000)},
  {VT_exponent,22,0x3a90f8,DOUBLEWITHTWODWORDINTREE(0x5e100000,0x00000000)},
  {VT_exponent,22,0x3a90f9,DOUBLEWITHTWODWORDINTREE(0x5e200000,0x00000000)},
  {VT_exponent,22,0x3a90fa,DOUBLEWITHTWODWORDINTREE(0x5e300000,0x00000000)},
  {VT_exponent,22,0x3a90fb,DOUBLEWITHTWODWORDINTREE(0x5e400000,0x00000000)},
  {VT_exponent,22,0x3a90fc,DOUBLEWITHTWODWORDINTREE(0x5e500000,0x00000000)},
  {VT_exponent,22,0x3a90fd,DOUBLEWITHTWODWORDINTREE(0x5e600000,0x00000000)},
  {VT_exponent,22,0x3a90fe,DOUBLEWITHTWODWORDINTREE(0x5e700000,0x00000000)},
  {VT_exponent,22,0x3a90ff,DOUBLEWITHTWODWORDINTREE(0x5e800000,0x00000000)},
  {VT_exponent,22,0x3a9100,DOUBLEWITHTWODWORDINTREE(0x5e900000,0x00000000)},
  {VT_exponent,22,0x3a9101,DOUBLEWITHTWODWORDINTREE(0x5ea00000,0x00000000)},
  {VT_exponent,22,0x3a9102,DOUBLEWITHTWODWORDINTREE(0x5eb00000,0x00000000)},
  {VT_exponent,22,0x3a9103,DOUBLEWITHTWODWORDINTREE(0x5ec00000,0x00000000)},
  {VT_exponent,22,0x3a9104,DOUBLEWITHTWODWORDINTREE(0x5ed00000,0x00000000)},
  {VT_exponent,22,0x3a9105,DOUBLEWITHTWODWORDINTREE(0x5ee00000,0x00000000)},
  {VT_exponent,22,0x3a9106,DOUBLEWITHTWODWORDINTREE(0x5ef00000,0x00000000)},
  {VT_exponent,22,0x3a9107,DOUBLEWITHTWODWORDINTREE(0x5f000000,0x00000000)},
  {VT_exponent,22,0x3a9108,DOUBLEWITHTWODWORDINTREE(0x5f100000,0x00000000)},
  {VT_exponent,22,0x3a9109,DOUBLEWITHTWODWORDINTREE(0x5f200000,0x00000000)},
  {VT_exponent,22,0x3a910a,DOUBLEWITHTWODWORDINTREE(0x5f300000,0x00000000)},
  {VT_exponent,22,0x3a910b,DOUBLEWITHTWODWORDINTREE(0x5f400000,0x00000000)},
  {VT_exponent,22,0x3a910c,DOUBLEWITHTWODWORDINTREE(0x5f500000,0x00000000)},
  {VT_exponent,22,0x3a910d,DOUBLEWITHTWODWORDINTREE(0x5f600000,0x00000000)},
  {VT_exponent,22,0x3a910e,DOUBLEWITHTWODWORDINTREE(0x5f700000,0x00000000)},
  {VT_exponent,22,0x3a910f,DOUBLEWITHTWODWORDINTREE(0x5f800000,0x00000000)},
  {VT_exponent,22,0x3a9110,DOUBLEWITHTWODWORDINTREE(0x5f900000,0x00000000)},
  {VT_exponent,22,0x3a9111,DOUBLEWITHTWODWORDINTREE(0x5fa00000,0x00000000)},
  {VT_exponent,22,0x3a9112,DOUBLEWITHTWODWORDINTREE(0x5fb00000,0x00000000)},
  {VT_exponent,22,0x3a9113,DOUBLEWITHTWODWORDINTREE(0x5fc00000,0x00000000)},
  {VT_exponent,22,0x3a9114,DOUBLEWITHTWODWORDINTREE(0x5fd00000,0x00000000)},
  {VT_exponent,22,0x3a9115,DOUBLEWITHTWODWORDINTREE(0x5fe00000,0x00000000)},
  {VT_exponent,22,0x3a9116,DOUBLEWITHTWODWORDINTREE(0x5ff00000,0x00000000)},
  {VT_exponent,22,0x3a9117,DOUBLEWITHTWODWORDINTREE(0x60000000,0x00000000)},
  {VT_exponent,22,0x3a9118,DOUBLEWITHTWODWORDINTREE(0x60100000,0x00000000)},
  {VT_exponent,22,0x3a9119,DOUBLEWITHTWODWORDINTREE(0x60200000,0x00000000)},
  {VT_exponent,22,0x3a911a,DOUBLEWITHTWODWORDINTREE(0x60300000,0x00000000)},
  {VT_exponent,22,0x3a911b,DOUBLEWITHTWODWORDINTREE(0x60400000,0x00000000)},
  {VT_exponent,22,0x3a911c,DOUBLEWITHTWODWORDINTREE(0x60500000,0x00000000)},
  {VT_exponent,22,0x3a911d,DOUBLEWITHTWODWORDINTREE(0x60600000,0x00000000)},
  {VT_exponent,22,0x3a911e,DOUBLEWITHTWODWORDINTREE(0x60700000,0x00000000)},
  {VT_exponent,22,0x3a911f,DOUBLEWITHTWODWORDINTREE(0x60800000,0x00000000)},
  {VT_exponent,22,0x3a9120,DOUBLEWITHTWODWORDINTREE(0x60900000,0x00000000)},
  {VT_exponent,22,0x3a9121,DOUBLEWITHTWODWORDINTREE(0x60a00000,0x00000000)},
  {VT_exponent,22,0x3a9122,DOUBLEWITHTWODWORDINTREE(0x60b00000,0x00000000)},
  {VT_exponent,22,0x3a9123,DOUBLEWITHTWODWORDINTREE(0x60c00000,0x00000000)},
  {VT_exponent,22,0x3a9124,DOUBLEWITHTWODWORDINTREE(0x60d00000,0x00000000)},
  {VT_exponent,22,0x3a9125,DOUBLEWITHTWODWORDINTREE(0x60e00000,0x00000000)},
  {VT_exponent,22,0x3a9126,DOUBLEWITHTWODWORDINTREE(0x60f00000,0x00000000)},
  {VT_exponent,22,0x3a9127,DOUBLEWITHTWODWORDINTREE(0x61000000,0x00000000)},
  {VT_exponent,22,0x3a9128,DOUBLEWITHTWODWORDINTREE(0x61100000,0x00000000)},
  {VT_exponent,22,0x3a9129,DOUBLEWITHTWODWORDINTREE(0x61200000,0x00000000)},
  {VT_exponent,22,0x3a912a,DOUBLEWITHTWODWORDINTREE(0x61300000,0x00000000)},
  {VT_exponent,22,0x3a912b,DOUBLEWITHTWODWORDINTREE(0x61400000,0x00000000)},
  {VT_exponent,22,0x3a912c,DOUBLEWITHTWODWORDINTREE(0x61500000,0x00000000)},
  {VT_exponent,22,0x3a912d,DOUBLEWITHTWODWORDINTREE(0x61600000,0x00000000)},
  {VT_exponent,22,0x3a912e,DOUBLEWITHTWODWORDINTREE(0x61700000,0x00000000)},
```

```
  {VT_exponent,22,0x3a912f,DOUBLEWITHTWODWORDINTREE(0x61800000,0x00000000)},
  {VT_exponent,22,0x3a9130,DOUBLEWITHTWODWORDINTREE(0x61900000,0x00000000)},
  {VT_exponent,22,0x3a9131,DOUBLEWITHTWODWORDINTREE(0x61a00000,0x00000000)},
  {VT_exponent,22,0x3a9132,DOUBLEWITHTWODWORDINTREE(0x61b00000,0x00000000)},
  {VT_exponent,22,0x3a9133,DOUBLEWITHTWODWORDINTREE(0x61c00000,0x00000000)},
  {VT_exponent,22,0x3a9134,DOUBLEWITHTWODWORDINTREE(0x61d00000,0x00000000)},
  {VT_exponent,22,0x3a9135,DOUBLEWITHTWODWORDINTREE(0x61e00000,0x00000000)},
  {VT_exponent,22,0x3a9136,DOUBLEWITHTWODWORDINTREE(0x61f00000,0x00000000)},
  {VT_exponent,22,0x3a9137,DOUBLEWITHTWODWORDINTREE(0x62000000,0x00000000)},
  {VT_exponent,22,0x3a9138,DOUBLEWITHTWODWORDINTREE(0x62100000,0x00000000)},
  {VT_exponent,22,0x3a9139,DOUBLEWITHTWODWORDINTREE(0x62200000,0x00000000)},
  {VT_exponent,22,0x3a913a,DOUBLEWITHTWODWORDINTREE(0x62300000,0x00000000)},
  {VT_exponent,22,0x3a913b,DOUBLEWITHTWODWORDINTREE(0x62400000,0x00000000)},
  {VT_exponent,22,0x3a913c,DOUBLEWITHTWODWORDINTREE(0x62500000,0x00000000)},
  {VT_exponent,22,0x3a913d,DOUBLEWITHTWODWORDINTREE(0x62600000,0x00000000)},
  {VT_exponent,22,0x3a913e,DOUBLEWITHTWODWORDINTREE(0x62700000,0x00000000)},
  {VT_exponent,22,0x3a913f,DOUBLEWITHTWODWORDINTREE(0x62800000,0x00000000)},
  {VT_exponent,22,0x3a9140,DOUBLEWITHTWODWORDINTREE(0x62900000,0x00000000)},
  {VT_exponent,22,0x3a9141,DOUBLEWITHTWODWORDINTREE(0x62a00000,0x00000000)},
  {VT_exponent,22,0x3a9142,DOUBLEWITHTWODWORDINTREE(0x62b00000,0x00000000)},
  {VT_exponent,22,0x3a9143,DOUBLEWITHTWODWORDINTREE(0x62c00000,0x00000000)},
  {VT_exponent,22,0x3a9144,DOUBLEWITHTWODWORDINTREE(0x62d00000,0x00000000)},
  {VT_exponent,22,0x3a9145,DOUBLEWITHTWODWORDINTREE(0x62e00000,0x00000000)},
  {VT_exponent,22,0x3a9146,DOUBLEWITHTWODWORDINTREE(0x62f00000,0x00000000)},
  {VT_exponent,22,0x3a9147,DOUBLEWITHTWODWORDINTREE(0x63000000,0x00000000)},
  {VT_exponent,22,0x3a9148,DOUBLEWITHTWODWORDINTREE(0x63100000,0x00000000)},
  {VT_exponent,22,0x3a9149,DOUBLEWITHTWODWORDINTREE(0x63200000,0x00000000)},
  {VT_exponent,22,0x3a914a,DOUBLEWITHTWODWORDINTREE(0x63300000,0x00000000)},
  {VT_exponent,22,0x3a914b,DOUBLEWITHTWODWORDINTREE(0x63400000,0x00000000)},
  {VT_exponent,22,0x3a914c,DOUBLEWITHTWODWORDINTREE(0x63500000,0x00000000)},
  {VT_exponent,22,0x3a914d,DOUBLEWITHTWODWORDINTREE(0x63600000,0x00000000)},
  {VT_exponent,22,0x3a914e,DOUBLEWITHTWODWORDINTREE(0x63700000,0x00000000)},
  {VT_exponent,22,0x3a914f,DOUBLEWITHTWODWORDINTREE(0x63800000,0x00000000)},
  {VT_exponent,22,0x3a9150,DOUBLEWITHTWODWORDINTREE(0x63900000,0x00000000)},
  {VT_exponent,22,0x3a9151,DOUBLEWITHTWODWORDINTREE(0x63a00000,0x00000000)},
  {VT_exponent,22,0x3a9152,DOUBLEWITHTWODWORDINTREE(0x63b00000,0x00000000)},
  {VT_exponent,22,0x3a9153,DOUBLEWITHTWODWORDINTREE(0x63c00000,0x00000000)},
  {VT_exponent,22,0x3a9154,DOUBLEWITHTWODWORDINTREE(0x63d00000,0x00000000)},
  {VT_exponent,22,0x3a9155,DOUBLEWITHTWODWORDINTREE(0x63e00000,0x00000000)},
  {VT_exponent,22,0x3a9156,DOUBLEWITHTWODWORDINTREE(0x63f00000,0x00000000)},
  {VT_exponent,22,0x3a9157,DOUBLEWITHTWODWORDINTREE(0x64000000,0x00000000)},
  {VT_exponent,22,0x3a9158,DOUBLEWITHTWODWORDINTREE(0x64100000,0x00000000)},
  {VT_exponent,22,0x3a9159,DOUBLEWITHTWODWORDINTREE(0x64200000,0x00000000)},
  {VT_exponent,22,0x3a915a,DOUBLEWITHTWODWORDINTREE(0x64300000,0x00000000)},
  {VT_exponent,22,0x3a915b,DOUBLEWITHTWODWORDINTREE(0x64400000,0x00000000)},
  {VT_exponent,22,0x3a915c,DOUBLEWITHTWODWORDINTREE(0x64500000,0x00000000)},
  {VT_exponent,22,0x3a915d,DOUBLEWITHTWODWORDINTREE(0x64600000,0x00000000)},
  {VT_exponent,22,0x3a915e,DOUBLEWITHTWODWORDINTREE(0x64700000,0x00000000)},
  {VT_exponent,22,0x3a915f,DOUBLEWITHTWODWORDINTREE(0x64800000,0x00000000)},
  {VT_exponent,22,0x3a9160,DOUBLEWITHTWODWORDINTREE(0x64900000,0x00000000)},
  {VT_exponent,22,0x3a9161,DOUBLEWITHTWODWORDINTREE(0x64a00000,0x00000000)},
  {VT_exponent,22,0x3a9162,DOUBLEWITHTWODWORDINTREE(0x64b00000,0x00000000)},
  {VT_exponent,22,0x3a9163,DOUBLEWITHTWODWORDINTREE(0x64c00000,0x00000000)},
  {VT_exponent,22,0x3a9164,DOUBLEWITHTWODWORDINTREE(0x64d00000,0x00000000)},
  {VT_exponent,22,0x3a9165,DOUBLEWITHTWODWORDINTREE(0x64e00000,0x00000000)},
  {VT_exponent,22,0x3a9166,DOUBLEWITHTWODWORDINTREE(0x64f00000,0x00000000)},
  {VT_exponent,22,0x3a9167,DOUBLEWITHTWODWORDINTREE(0x65000000,0x00000000)},
  {VT_exponent,22,0x3a9168,DOUBLEWITHTWODWORDINTREE(0x65100000,0x00000000)},
  {VT_exponent,22,0x3a9169,DOUBLEWITHTWODWORDINTREE(0x65200000,0x00000000)},
  {VT_exponent,22,0x3a916a,DOUBLEWITHTWODWORDINTREE(0x65300000,0x00000000)},
  {VT_exponent,22,0x3a916b,DOUBLEWITHTWODWORDINTREE(0x65400000,0x00000000)},
```

```
  {VT_exponent,22,0x3a916c,DOUBLEWITHTWODWORDINTREE(0x65500000,0x00000000)},
  {VT_exponent,22,0x3a916d,DOUBLEWITHTWODWORDINTREE(0x65600000,0x00000000)},
  {VT_exponent,22,0x3a916e,DOUBLEWITHTWODWORDINTREE(0x65700000,0x00000000)},
  {VT_exponent,22,0x3a916f,DOUBLEWITHTWODWORDINTREE(0x65800000,0x00000000)},
  {VT_exponent,22,0x3a9170,DOUBLEWITHTWODWORDINTREE(0x65900000,0x00000000)},
  {VT_exponent,22,0x3a9171,DOUBLEWITHTWODWORDINTREE(0x65a00000,0x00000000)},
  {VT_exponent,22,0x3a9172,DOUBLEWITHTWODWORDINTREE(0x65b00000,0x00000000)},
  {VT_exponent,22,0x3a9173,DOUBLEWITHTWODWORDINTREE(0x65c00000,0x00000000)},
  {VT_exponent,22,0x3a9174,DOUBLEWITHTWODWORDINTREE(0x65d00000,0x00000000)},
  {VT_exponent,22,0x3a9175,DOUBLEWITHTWODWORDINTREE(0x65e00000,0x00000000)},
  {VT_exponent,22,0x3a9176,DOUBLEWITHTWODWORDINTREE(0x65f00000,0x00000000)},
  {VT_exponent,22,0x3a9177,DOUBLEWITHTWODWORDINTREE(0x66000000,0x00000000)},
  {VT_exponent,22,0x3a9178,DOUBLEWITHTWODWORDINTREE(0x66100000,0x00000000)},
  {VT_exponent,22,0x3a9179,DOUBLEWITHTWODWORDINTREE(0x66200000,0x00000000)},
  {VT_exponent,22,0x3a917a,DOUBLEWITHTWODWORDINTREE(0x66300000,0x00000000)},
  {VT_exponent,22,0x3a917b,DOUBLEWITHTWODWORDINTREE(0x66400000,0x00000000)},
  {VT_exponent,22,0x3a917c,DOUBLEWITHTWODWORDINTREE(0x66500000,0x00000000)},
  {VT_exponent,22,0x3a917d,DOUBLEWITHTWODWORDINTREE(0x66600000,0x00000000)},
  {VT_exponent,22,0x3a917e,DOUBLEWITHTWODWORDINTREE(0x66700000,0x00000000)},
  {VT_exponent,22,0x3a917f,DOUBLEWITHTWODWORDINTREE(0x66800000,0x00000000)},
  {VT_exponent,22,0x3a9180,DOUBLEWITHTWODWORDINTREE(0x66900000,0x00000000)},
  {VT_exponent,22,0x3a9181,DOUBLEWITHTWODWORDINTREE(0x66a00000,0x00000000)},
  {VT_exponent,22,0x3a9182,DOUBLEWITHTWODWORDINTREE(0x66b00000,0x00000000)},
  {VT_exponent,22,0x3a9183,DOUBLEWITHTWODWORDINTREE(0x66c00000,0x00000000)},
  {VT_exponent,22,0x3a9184,DOUBLEWITHTWODWORDINTREE(0x66d00000,0x00000000)},
  {VT_exponent,22,0x3a9185,DOUBLEWITHTWODWORDINTREE(0x66e00000,0x00000000)},
  {VT_exponent,22,0x3a9186,DOUBLEWITHTWODWORDINTREE(0x66f00000,0x00000000)},
  {VT_exponent,22,0x3a9187,DOUBLEWITHTWODWORDINTREE(0x67000000,0x00000000)},
  {VT_exponent,22,0x3a9188,DOUBLEWITHTWODWORDINTREE(0x67100000,0x00000000)},
  {VT_exponent,22,0x3a9189,DOUBLEWITHTWODWORDINTREE(0x67200000,0x00000000)},
  {VT_exponent,22,0x3a918a,DOUBLEWITHTWODWORDINTREE(0x67300000,0x00000000)},
  {VT_exponent,22,0x3a918b,DOUBLEWITHTWODWORDINTREE(0x67400000,0x00000000)},
  {VT_exponent,22,0x3a918c,DOUBLEWITHTWODWORDINTREE(0x67500000,0x00000000)},
  {VT_exponent,22,0x3a918d,DOUBLEWITHTWODWORDINTREE(0x67600000,0x00000000)},
  {VT_exponent,22,0x3a918e,DOUBLEWITHTWODWORDINTREE(0x67700000,0x00000000)},
  {VT_exponent,22,0x3a918f,DOUBLEWITHTWODWORDINTREE(0x67800000,0x00000000)},
  {VT_exponent,22,0x3a9190,DOUBLEWITHTWODWORDINTREE(0x67900000,0x00000000)},
  {VT_exponent,22,0x3a9191,DOUBLEWITHTWODWORDINTREE(0x67a00000,0x00000000)},
  {VT_exponent,22,0x3a9192,DOUBLEWITHTWODWORDINTREE(0x67b00000,0x00000000)},
  {VT_exponent,22,0x3a9193,DOUBLEWITHTWODWORDINTREE(0x67c00000,0x00000000)},
  {VT_exponent,22,0x3a9194,DOUBLEWITHTWODWORDINTREE(0x67d00000,0x00000000)},
  {VT_exponent,22,0x3a9195,DOUBLEWITHTWODWORDINTREE(0x67e00000,0x00000000)},
  {VT_exponent,22,0x3a9196,DOUBLEWITHTWODWORDINTREE(0x67f00000,0x00000000)},
  {VT_exponent,22,0x3a9197,DOUBLEWITHTWODWORDINTREE(0x68000000,0x00000000)},
  {VT_exponent,22,0x3a9198,DOUBLEWITHTWODWORDINTREE(0x68100000,0x00000000)},
  {VT_exponent,22,0x3a9199,DOUBLEWITHTWODWORDINTREE(0x68200000,0x00000000)},
  {VT_exponent,22,0x3a919a,DOUBLEWITHTWODWORDINTREE(0x68300000,0x00000000)},
  {VT_exponent,22,0x3a919b,DOUBLEWITHTWODWORDINTREE(0x68400000,0x00000000)},
  {VT_exponent,22,0x3a919c,DOUBLEWITHTWODWORDINTREE(0x68500000,0x00000000)},
  {VT_exponent,22,0x3a919d,DOUBLEWITHTWODWORDINTREE(0x68600000,0x00000000)},
  {VT_exponent,22,0x3a919e,DOUBLEWITHTWODWORDINTREE(0x68700000,0x00000000)},
  {VT_exponent,22,0x3a919f,DOUBLEWITHTWODWORDINTREE(0x68800000,0x00000000)},
  {VT_exponent,22,0x3a91a0,DOUBLEWITHTWODWORDINTREE(0x68900000,0x00000000)},
  {VT_exponent,22,0x3a91a1,DOUBLEWITHTWODWORDINTREE(0x68a00000,0x00000000)},
  {VT_exponent,22,0x3a91a2,DOUBLEWITHTWODWORDINTREE(0x68b00000,0x00000000)},
  {VT_exponent,22,0x3a91a3,DOUBLEWITHTWODWORDINTREE(0x68c00000,0x00000000)},
  {VT_exponent,22,0x3a91a4,DOUBLEWITHTWODWORDINTREE(0x68d00000,0x00000000)},
  {VT_exponent,22,0x3a91a5,DOUBLEWITHTWODWORDINTREE(0x68e00000,0x00000000)},
  {VT_exponent,22,0x3a91a6,DOUBLEWITHTWODWORDINTREE(0x68f00000,0x00000000)},
  {VT_exponent,22,0x3a91a7,DOUBLEWITHTWODWORDINTREE(0x69000000,0x00000000)},
  {VT_exponent,22,0x3a91a8,DOUBLEWITHTWODWORDINTREE(0x69100000,0x00000000)},
```

```
  {VT_exponent,22,0x3a91a9,DOUBLEWITHTWODWORDINTREE(0x69200000,0x00000000)},
  {VT_exponent,22,0x3a91aa,DOUBLEWITHTWODWORDINTREE(0x69300000,0x00000000)},
  {VT_exponent,22,0x3a91ab,DOUBLEWITHTWODWORDINTREE(0x69400000,0x00000000)},
  {VT_exponent,22,0x3a91ac,DOUBLEWITHTWODWORDINTREE(0x69500000,0x00000000)},
  {VT_exponent,22,0x3a91ad,DOUBLEWITHTWODWORDINTREE(0x69600000,0x00000000)},
  {VT_exponent,22,0x3a91ae,DOUBLEWITHTWODWORDINTREE(0x69700000,0x00000000)},
  {VT_exponent,22,0x3a91af,DOUBLEWITHTWODWORDINTREE(0x69800000,0x00000000)},
  {VT_exponent,22,0x3a91b0,DOUBLEWITHTWODWORDINTREE(0x69900000,0x00000000)},
  {VT_exponent,22,0x3a91b1,DOUBLEWITHTWODWORDINTREE(0x69a00000,0x00000000)},
  {VT_exponent,22,0x3a91b2,DOUBLEWITHTWODWORDINTREE(0x69b00000,0x00000000)},
  {VT_exponent,22,0x3a91b3,DOUBLEWITHTWODWORDINTREE(0x69c00000,0x00000000)},
  {VT_exponent,22,0x3a91b4,DOUBLEWITHTWODWORDINTREE(0x69d00000,0x00000000)},
  {VT_exponent,22,0x3a91b5,DOUBLEWITHTWODWORDINTREE(0x69e00000,0x00000000)},
  {VT_exponent,22,0x3a91b6,DOUBLEWITHTWODWORDINTREE(0x69f00000,0x00000000)},
  {VT_exponent,22,0x3a91b7,DOUBLEWITHTWODWORDINTREE(0x6a000000,0x00000000)},
  {VT_exponent,22,0x3a91b8,DOUBLEWITHTWODWORDINTREE(0x6a100000,0x00000000)},
  {VT_exponent,22,0x3a91b9,DOUBLEWITHTWODWORDINTREE(0x6a200000,0x00000000)},
  {VT_exponent,22,0x3a91ba,DOUBLEWITHTWODWORDINTREE(0x6a300000,0x00000000)},
  {VT_exponent,22,0x3a91bb,DOUBLEWITHTWODWORDINTREE(0x6a400000,0x00000000)},
  {VT_exponent,22,0x3a91bc,DOUBLEWITHTWODWORDINTREE(0x6a500000,0x00000000)},
  {VT_exponent,22,0x3a91bd,DOUBLEWITHTWODWORDINTREE(0x6a600000,0x00000000)},
  {VT_exponent,22,0x3a91be,DOUBLEWITHTWODWORDINTREE(0x6a700000,0x00000000)},
  {VT_exponent,22,0x3a91bf,DOUBLEWITHTWODWORDINTREE(0x6a800000,0x00000000)},
  {VT_exponent,22,0x3a91c0,DOUBLEWITHTWODWORDINTREE(0x6a900000,0x00000000)},
  {VT_exponent,22,0x3a91c1,DOUBLEWITHTWODWORDINTREE(0x6aa00000,0x00000000)},
  {VT_exponent,22,0x3a91c2,DOUBLEWITHTWODWORDINTREE(0x6ab00000,0x00000000)},
  {VT_exponent,22,0x3a91c3,DOUBLEWITHTWODWORDINTREE(0x6ac00000,0x00000000)},
  {VT_exponent,22,0x3a91c4,DOUBLEWITHTWODWORDINTREE(0x6ad00000,0x00000000)},
  {VT_exponent,22,0x3a91c5,DOUBLEWITHTWODWORDINTREE(0x6ae00000,0x00000000)},
  {VT_exponent,22,0x3a91c6,DOUBLEWITHTWODWORDINTREE(0x6af00000,0x00000000)},
  {VT_exponent,22,0x3a91c7,DOUBLEWITHTWODWORDINTREE(0x6b000000,0x00000000)},
  {VT_exponent,22,0x3a91c8,DOUBLEWITHTWODWORDINTREE(0x6b100000,0x00000000)},
  {VT_exponent,22,0x3a91c9,DOUBLEWITHTWODWORDINTREE(0x6b200000,0x00000000)},
  {VT_exponent,22,0x3a91ca,DOUBLEWITHTWODWORDINTREE(0x6b300000,0x00000000)},
  {VT_exponent,22,0x3a91cb,DOUBLEWITHTWODWORDINTREE(0x6b400000,0x00000000)},
  {VT_exponent,22,0x3a91cc,DOUBLEWITHTWODWORDINTREE(0x6b500000,0x00000000)},
  {VT_exponent,22,0x3a91cd,DOUBLEWITHTWODWORDINTREE(0x6b600000,0x00000000)},
  {VT_exponent,22,0x3a91ce,DOUBLEWITHTWODWORDINTREE(0x6b700000,0x00000000)},
  {VT_exponent,22,0x3a91cf,DOUBLEWITHTWODWORDINTREE(0x6b800000,0x00000000)},
  {VT_exponent,22,0x3a91d0,DOUBLEWITHTWODWORDINTREE(0x6b900000,0x00000000)},
  {VT_exponent,22,0x3a91d1,DOUBLEWITHTWODWORDINTREE(0x6ba00000,0x00000000)},
  {VT_exponent,22,0x3a91d2,DOUBLEWITHTWODWORDINTREE(0x6bb00000,0x00000000)},
  {VT_exponent,22,0x3a91d3,DOUBLEWITHTWODWORDINTREE(0x6bc00000,0x00000000)},
  {VT_exponent,22,0x3a91d4,DOUBLEWITHTWODWORDINTREE(0x6bd00000,0x00000000)},
  {VT_exponent,22,0x3a91d5,DOUBLEWITHTWODWORDINTREE(0x6be00000,0x00000000)},
  {VT_exponent,22,0x3a91d6,DOUBLEWITHTWODWORDINTREE(0x6bf00000,0x00000000)},
  {VT_exponent,22,0x3a91d7,DOUBLEWITHTWODWORDINTREE(0x6c000000,0x00000000)},
  {VT_exponent,22,0x3a91d8,DOUBLEWITHTWODWORDINTREE(0x6c100000,0x00000000)},
  {VT_exponent,22,0x3a91d9,DOUBLEWITHTWODWORDINTREE(0x6c200000,0x00000000)},
  {VT_exponent,22,0x3a91da,DOUBLEWITHTWODWORDINTREE(0x6c300000,0x00000000)},
  {VT_exponent,22,0x3a91db,DOUBLEWITHTWODWORDINTREE(0x6c400000,0x00000000)},
  {VT_exponent,22,0x3a91dc,DOUBLEWITHTWODWORDINTREE(0x6c500000,0x00000000)},
  {VT_exponent,22,0x3a91dd,DOUBLEWITHTWODWORDINTREE(0x6c600000,0x00000000)},
  {VT_exponent,22,0x3a91de,DOUBLEWITHTWODWORDINTREE(0x6c700000,0x00000000)},
  {VT_exponent,22,0x3a91df,DOUBLEWITHTWODWORDINTREE(0x6c800000,0x00000000)},
  {VT_exponent,22,0x3a91e0,DOUBLEWITHTWODWORDINTREE(0x6c900000,0x00000000)},
  {VT_exponent,22,0x3a91e1,DOUBLEWITHTWODWORDINTREE(0x6ca00000,0x00000000)},
  {VT_exponent,22,0x3a91e2,DOUBLEWITHTWODWORDINTREE(0x6cb00000,0x00000000)},
  {VT_exponent,22,0x3a91e3,DOUBLEWITHTWODWORDINTREE(0x6cc00000,0x00000000)},
  {VT_exponent,22,0x3a91e4,DOUBLEWITHTWODWORDINTREE(0x6cd00000,0x00000000)},
  {VT_exponent,22,0x3a91e5,DOUBLEWITHTWODWORDINTREE(0x6ce00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a91e6,DOUBLEWITHTWODWORDINTREE(0x6cf00000,0x00000000)},
  {VT_exponent,22,0x3a91e7,DOUBLEWITHTWODWORDINTREE(0x6d000000,0x00000000)},
  {VT_exponent,22,0x3a91e8,DOUBLEWITHTWODWORDINTREE(0x6d100000,0x00000000)},
  {VT_exponent,22,0x3a91e9,DOUBLEWITHTWODWORDINTREE(0x6d200000,0x00000000)},
  {VT_exponent,22,0x3a91ea,DOUBLEWITHTWODWORDINTREE(0x6d300000,0x00000000)},
  {VT_exponent,22,0x3a91eb,DOUBLEWITHTWODWORDINTREE(0x6d400000,0x00000000)},
  {VT_exponent,22,0x3a91ec,DOUBLEWITHTWODWORDINTREE(0x6d500000,0x00000000)},
  {VT_exponent,22,0x3a91ed,DOUBLEWITHTWODWORDINTREE(0x6d600000,0x00000000)},
  {VT_exponent,22,0x3a91ee,DOUBLEWITHTWODWORDINTREE(0x6d700000,0x00000000)},
  {VT_exponent,22,0x3a91ef,DOUBLEWITHTWODWORDINTREE(0x6d800000,0x00000000)},
  {VT_exponent,22,0x3a91f0,DOUBLEWITHTWODWORDINTREE(0x6d900000,0x00000000)},
  {VT_exponent,22,0x3a91f1,DOUBLEWITHTWODWORDINTREE(0x6da00000,0x00000000)},
  {VT_exponent,22,0x3a91f2,DOUBLEWITHTWODWORDINTREE(0x6db00000,0x00000000)},
  {VT_exponent,22,0x3a91f3,DOUBLEWITHTWODWORDINTREE(0x6dc00000,0x00000000)},
  {VT_exponent,22,0x3a91f4,DOUBLEWITHTWODWORDINTREE(0x6dd00000,0x00000000)},
  {VT_exponent,22,0x3a91f5,DOUBLEWITHTWODWORDINTREE(0x6de00000,0x00000000)},
  {VT_exponent,22,0x3a91f6,DOUBLEWITHTWODWORDINTREE(0x6df00000,0x00000000)},
  {VT_exponent,22,0x3a91f7,DOUBLEWITHTWODWORDINTREE(0x6e000000,0x00000000)},
  {VT_exponent,22,0x3a91f8,DOUBLEWITHTWODWORDINTREE(0x6e100000,0x00000000)},
  {VT_exponent,22,0x3a91f9,DOUBLEWITHTWODWORDINTREE(0x6e200000,0x00000000)},
  {VT_exponent,22,0x3a91fa,DOUBLEWITHTWODWORDINTREE(0x6e300000,0x00000000)},
  {VT_exponent,22,0x3a91fb,DOUBLEWITHTWODWORDINTREE(0x6e400000,0x00000000)},
  {VT_exponent,22,0x3a91fc,DOUBLEWITHTWODWORDINTREE(0x6e500000,0x00000000)},
  {VT_exponent,22,0x3a91fd,DOUBLEWITHTWODWORDINTREE(0x6e600000,0x00000000)},
  {VT_exponent,22,0x3a91fe,DOUBLEWITHTWODWORDINTREE(0x6e700000,0x00000000)},
  {VT_exponent,22,0x3a91ff,DOUBLEWITHTWODWORDINTREE(0x6e800000,0x00000000)},
  {VT_exponent,22,0x3a9200,DOUBLEWITHTWODWORDINTREE(0x6e900000,0x00000000)},
  {VT_exponent,22,0x3a9201,DOUBLEWITHTWODWORDINTREE(0x6ea00000,0x00000000)},
  {VT_exponent,22,0x3a9202,DOUBLEWITHTWODWORDINTREE(0x6eb00000,0x00000000)},
  {VT_exponent,22,0x3a9203,DOUBLEWITHTWODWORDINTREE(0x6ec00000,0x00000000)},
  {VT_exponent,22,0x3a9204,DOUBLEWITHTWODWORDINTREE(0x6ed00000,0x00000000)},
  {VT_exponent,22,0x3a9205,DOUBLEWITHTWODWORDINTREE(0x6ee00000,0x00000000)},
  {VT_exponent,22,0x3a9206,DOUBLEWITHTWODWORDINTREE(0x6ef00000,0x00000000)},
  {VT_exponent,22,0x3a9207,DOUBLEWITHTWODWORDINTREE(0x6f000000,0x00000000)},
  {VT_exponent,22,0x3a9208,DOUBLEWITHTWODWORDINTREE(0x6f100000,0x00000000)},
  {VT_exponent,22,0x3a9209,DOUBLEWITHTWODWORDINTREE(0x6f200000,0x00000000)},
  {VT_exponent,22,0x3a920a,DOUBLEWITHTWODWORDINTREE(0x6f300000,0x00000000)},
  {VT_exponent,22,0x3a920b,DOUBLEWITHTWODWORDINTREE(0x6f400000,0x00000000)},
  {VT_exponent,22,0x3a920c,DOUBLEWITHTWODWORDINTREE(0x6f500000,0x00000000)},
  {VT_exponent,22,0x3a920d,DOUBLEWITHTWODWORDINTREE(0x6f600000,0x00000000)},
  {VT_exponent,22,0x3a920e,DOUBLEWITHTWODWORDINTREE(0x6f700000,0x00000000)},
  {VT_exponent,22,0x3a920f,DOUBLEWITHTWODWORDINTREE(0x6f800000,0x00000000)},
  {VT_exponent,22,0x3a9210,DOUBLEWITHTWODWORDINTREE(0x6f900000,0x00000000)},
  {VT_exponent,22,0x3a9211,DOUBLEWITHTWODWORDINTREE(0x6fa00000,0x00000000)},
  {VT_exponent,22,0x3a9212,DOUBLEWITHTWODWORDINTREE(0x6fb00000,0x00000000)},
  {VT_exponent,22,0x3a9213,DOUBLEWITHTWODWORDINTREE(0x6fc00000,0x00000000)},
  {VT_exponent,22,0x3a9214,DOUBLEWITHTWODWORDINTREE(0x6fd00000,0x00000000)},
  {VT_exponent,22,0x3a9215,DOUBLEWITHTWODWORDINTREE(0x6fe00000,0x00000000)},
  {VT_exponent,22,0x3a9216,DOUBLEWITHTWODWORDINTREE(0x6ff00000,0x00000000)},
  {VT_exponent,22,0x3a9217,DOUBLEWITHTWODWORDINTREE(0x70000000,0x00000000)},
  {VT_exponent,22,0x3a9218,DOUBLEWITHTWODWORDINTREE(0x70100000,0x00000000)},
  {VT_exponent,22,0x3a9219,DOUBLEWITHTWODWORDINTREE(0x70200000,0x00000000)},
  {VT_exponent,22,0x3a921a,DOUBLEWITHTWODWORDINTREE(0x70300000,0x00000000)},
  {VT_exponent,22,0x3a921b,DOUBLEWITHTWODWORDINTREE(0x70400000,0x00000000)},
  {VT_exponent,22,0x3a921c,DOUBLEWITHTWODWORDINTREE(0x70500000,0x00000000)},
  {VT_exponent,22,0x3a921d,DOUBLEWITHTWODWORDINTREE(0x70600000,0x00000000)},
  {VT_exponent,22,0x3a921e,DOUBLEWITHTWODWORDINTREE(0x70700000,0x00000000)},
  {VT_exponent,22,0x3a921f,DOUBLEWITHTWODWORDINTREE(0x70800000,0x00000000)},
  {VT_exponent,22,0x3a9220,DOUBLEWITHTWODWORDINTREE(0x70900000,0x00000000)},
  {VT_exponent,22,0x3a9221,DOUBLEWITHTWODWORDINTREE(0x70a00000,0x00000000)},
  {VT_exponent,22,0x3a9222,DOUBLEWITHTWODWORDINTREE(0x70b00000,0x00000000)},
```

```
  {VT_exponent,22,0x3a9223,DOUBLEWITHTWODWORDINTREE(0x70c00000,0x00000000)},
  {VT_exponent,22,0x3a9224,DOUBLEWITHTWODWORDINTREE(0x70d00000,0x00000000)},
  {VT_exponent,22,0x3a9225,DOUBLEWITHTWODWORDINTREE(0x70e00000,0x00000000)},
  {VT_exponent,22,0x3a9226,DOUBLEWITHTWODWORDINTREE(0x70f00000,0x00000000)},
  {VT_exponent,22,0x3a9227,DOUBLEWITHTWODWORDINTREE(0x71000000,0x00000000)},
  {VT_exponent,22,0x3a9228,DOUBLEWITHTWODWORDINTREE(0x71100000,0x00000000)},
  {VT_exponent,22,0x3a9229,DOUBLEWITHTWODWORDINTREE(0x71200000,0x00000000)},
  {VT_exponent,22,0x3a922a,DOUBLEWITHTWODWORDINTREE(0x71300000,0x00000000)},
  {VT_exponent,22,0x3a922b,DOUBLEWITHTWODWORDINTREE(0x71400000,0x00000000)},
  {VT_exponent,22,0x3a922c,DOUBLEWITHTWODWORDINTREE(0x71500000,0x00000000)},
  {VT_exponent,22,0x3a922d,DOUBLEWITHTWODWORDINTREE(0x71600000,0x00000000)},
  {VT_exponent,22,0x3a922e,DOUBLEWITHTWODWORDINTREE(0x71700000,0x00000000)},
  {VT_exponent,22,0x3a922f,DOUBLEWITHTWODWORDINTREE(0x71800000,0x00000000)},
  {VT_exponent,22,0x3a9230,DOUBLEWITHTWODWORDINTREE(0x71900000,0x00000000)},
  {VT_exponent,22,0x3a9231,DOUBLEWITHTWODWORDINTREE(0x71a00000,0x00000000)},
  {VT_exponent,22,0x3a9232,DOUBLEWITHTWODWORDINTREE(0x71b00000,0x00000000)},
  {VT_exponent,22,0x3a9233,DOUBLEWITHTWODWORDINTREE(0x71c00000,0x00000000)},
  {VT_exponent,22,0x3a9234,DOUBLEWITHTWODWORDINTREE(0x71d00000,0x00000000)},
  {VT_exponent,22,0x3a9235,DOUBLEWITHTWODWORDINTREE(0x71e00000,0x00000000)},
  {VT_exponent,22,0x3a9236,DOUBLEWITHTWODWORDINTREE(0x71f00000,0x00000000)},
  {VT_exponent,22,0x3a9237,DOUBLEWITHTWODWORDINTREE(0x72000000,0x00000000)},
  {VT_exponent,22,0x3a9238,DOUBLEWITHTWODWORDINTREE(0x72100000,0x00000000)},
  {VT_exponent,22,0x3a9239,DOUBLEWITHTWODWORDINTREE(0x72200000,0x00000000)},
  {VT_exponent,22,0x3a923a,DOUBLEWITHTWODWORDINTREE(0x72300000,0x00000000)},
  {VT_exponent,22,0x3a923b,DOUBLEWITHTWODWORDINTREE(0x72400000,0x00000000)},
  {VT_exponent,22,0x3a923c,DOUBLEWITHTWODWORDINTREE(0x72500000,0x00000000)},
  {VT_exponent,22,0x3a923d,DOUBLEWITHTWODWORDINTREE(0x72600000,0x00000000)},
  {VT_exponent,22,0x3a923e,DOUBLEWITHTWODWORDINTREE(0x72700000,0x00000000)},
  {VT_exponent,22,0x3a923f,DOUBLEWITHTWODWORDINTREE(0x72800000,0x00000000)},
  {VT_exponent,22,0x3a9240,DOUBLEWITHTWODWORDINTREE(0x72900000,0x00000000)},
  {VT_exponent,22,0x3a9241,DOUBLEWITHTWODWORDINTREE(0x72a00000,0x00000000)},
  {VT_exponent,22,0x3a9242,DOUBLEWITHTWODWORDINTREE(0x72b00000,0x00000000)},
  {VT_exponent,22,0x3a9243,DOUBLEWITHTWODWORDINTREE(0x72c00000,0x00000000)},
  {VT_exponent,22,0x3a9244,DOUBLEWITHTWODWORDINTREE(0x72d00000,0x00000000)},
  {VT_exponent,22,0x3a9245,DOUBLEWITHTWODWORDINTREE(0x72e00000,0x00000000)},
  {VT_exponent,22,0x3a9246,DOUBLEWITHTWODWORDINTREE(0x72f00000,0x00000000)},
  {VT_exponent,22,0x3a9247,DOUBLEWITHTWODWORDINTREE(0x73000000,0x00000000)},
  {VT_exponent,22,0x3a9248,DOUBLEWITHTWODWORDINTREE(0x73100000,0x00000000)},
  {VT_exponent,22,0x3a9249,DOUBLEWITHTWODWORDINTREE(0x73200000,0x00000000)},
  {VT_exponent,22,0x3a924a,DOUBLEWITHTWODWORDINTREE(0x73300000,0x00000000)},
  {VT_exponent,22,0x3a924b,DOUBLEWITHTWODWORDINTREE(0x73400000,0x00000000)},
  {VT_exponent,22,0x3a924c,DOUBLEWITHTWODWORDINTREE(0x73500000,0x00000000)},
  {VT_exponent,22,0x3a924d,DOUBLEWITHTWODWORDINTREE(0x73600000,0x00000000)},
  {VT_exponent,22,0x3a924e,DOUBLEWITHTWODWORDINTREE(0x73700000,0x00000000)},
  {VT_exponent,22,0x3a924f,DOUBLEWITHTWODWORDINTREE(0x73800000,0x00000000)},
  {VT_exponent,22,0x3a9250,DOUBLEWITHTWODWORDINTREE(0x73900000,0x00000000)},
  {VT_exponent,22,0x3a9251,DOUBLEWITHTWODWORDINTREE(0x73a00000,0x00000000)},
  {VT_exponent,22,0x3a9252,DOUBLEWITHTWODWORDINTREE(0x73b00000,0x00000000)},
  {VT_exponent,22,0x3a9253,DOUBLEWITHTWODWORDINTREE(0x73c00000,0x00000000)},
  {VT_exponent,22,0x3a9254,DOUBLEWITHTWODWORDINTREE(0x73d00000,0x00000000)},
  {VT_exponent,22,0x3a9255,DOUBLEWITHTWODWORDINTREE(0x73e00000,0x00000000)},
  {VT_exponent,22,0x3a9256,DOUBLEWITHTWODWORDINTREE(0x73f00000,0x00000000)},
  {VT_exponent,22,0x3a9257,DOUBLEWITHTWODWORDINTREE(0x74000000,0x00000000)},
  {VT_exponent,22,0x3a9258,DOUBLEWITHTWODWORDINTREE(0x74100000,0x00000000)},
  {VT_exponent,22,0x3a9259,DOUBLEWITHTWODWORDINTREE(0x74200000,0x00000000)},
  {VT_exponent,22,0x3a925a,DOUBLEWITHTWODWORDINTREE(0x74300000,0x00000000)},
  {VT_exponent,22,0x3a925b,DOUBLEWITHTWODWORDINTREE(0x74400000,0x00000000)},
  {VT_exponent,22,0x3a925c,DOUBLEWITHTWODWORDINTREE(0x74500000,0x00000000)},
  {VT_exponent,22,0x3a925d,DOUBLEWITHTWODWORDINTREE(0x74600000,0x00000000)},
  {VT_exponent,22,0x3a925e,DOUBLEWITHTWODWORDINTREE(0x74700000,0x00000000)},
  {VT_exponent,22,0x3a925f,DOUBLEWITHTWODWORDINTREE(0x74800000,0x00000000)},
```

```
  {VT_exponent,22,0x3a9260,DOUBLEWITHTWODWORDINTREE(0x74900000,0x00000000)},
  {VT_exponent,22,0x3a9261,DOUBLEWITHTWODWORDINTREE(0x74a00000,0x00000000)},
  {VT_exponent,22,0x3a9262,DOUBLEWITHTWODWORDINTREE(0x74b00000,0x00000000)},
  {VT_exponent,22,0x3a9263,DOUBLEWITHTWODWORDINTREE(0x74c00000,0x00000000)},
  {VT_exponent,22,0x3a9264,DOUBLEWITHTWODWORDINTREE(0x74d00000,0x00000000)},
  {VT_exponent,22,0x3a9265,DOUBLEWITHTWODWORDINTREE(0x74e00000,0x00000000)},
  {VT_exponent,22,0x3a9266,DOUBLEWITHTWODWORDINTREE(0x74f00000,0x00000000)},
  {VT_exponent,22,0x3a9267,DOUBLEWITHTWODWORDINTREE(0x75000000,0x00000000)},
  {VT_exponent,22,0x3a9268,DOUBLEWITHTWODWORDINTREE(0x75100000,0x00000000)},
  {VT_exponent,22,0x3a9269,DOUBLEWITHTWODWORDINTREE(0x75200000,0x00000000)},
  {VT_exponent,22,0x3a926a,DOUBLEWITHTWODWORDINTREE(0x75300000,0x00000000)},
  {VT_exponent,22,0x3a926b,DOUBLEWITHTWODWORDINTREE(0x75400000,0x00000000)},
  {VT_exponent,22,0x3a926c,DOUBLEWITHTWODWORDINTREE(0x75500000,0x00000000)},
  {VT_exponent,22,0x3a926d,DOUBLEWITHTWODWORDINTREE(0x75600000,0x00000000)},
  {VT_exponent,22,0x3a926e,DOUBLEWITHTWODWORDINTREE(0x75700000,0x00000000)},
  {VT_exponent,22,0x3a926f,DOUBLEWITHTWODWORDINTREE(0x75800000,0x00000000)},
  {VT_exponent,22,0x3a9270,DOUBLEWITHTWODWORDINTREE(0x75900000,0x00000000)},
  {VT_exponent,22,0x3a9271,DOUBLEWITHTWODWORDINTREE(0x75a00000,0x00000000)},
  {VT_exponent,22,0x3a9272,DOUBLEWITHTWODWORDINTREE(0x75b00000,0x00000000)},
  {VT_exponent,22,0x3a9273,DOUBLEWITHTWODWORDINTREE(0x75c00000,0x00000000)},
  {VT_exponent,22,0x3a9274,DOUBLEWITHTWODWORDINTREE(0x75d00000,0x00000000)},
  {VT_exponent,22,0x3a9275,DOUBLEWITHTWODWORDINTREE(0x75e00000,0x00000000)},
  {VT_exponent,22,0x3a9276,DOUBLEWITHTWODWORDINTREE(0x75f00000,0x00000000)},
  {VT_exponent,22,0x3a9277,DOUBLEWITHTWODWORDINTREE(0x76000000,0x00000000)},
  {VT_exponent,22,0x3a9278,DOUBLEWITHTWODWORDINTREE(0x76100000,0x00000000)},
  {VT_exponent,22,0x3a9279,DOUBLEWITHTWODWORDINTREE(0x76200000,0x00000000)},
  {VT_exponent,22,0x3a927a,DOUBLEWITHTWODWORDINTREE(0x76300000,0x00000000)},
  {VT_exponent,22,0x3a927b,DOUBLEWITHTWODWORDINTREE(0x76400000,0x00000000)},
  {VT_exponent,22,0x3a927c,DOUBLEWITHTWODWORDINTREE(0x76500000,0x00000000)},
  {VT_exponent,22,0x3a927d,DOUBLEWITHTWODWORDINTREE(0x76600000,0x00000000)},
  {VT_exponent,22,0x3a927e,DOUBLEWITHTWODWORDINTREE(0x76700000,0x00000000)},
  {VT_exponent,22,0x3a927f,DOUBLEWITHTWODWORDINTREE(0x76800000,0x00000000)},
  {VT_exponent,22,0x3a9280,DOUBLEWITHTWODWORDINTREE(0x76900000,0x00000000)},
  {VT_exponent,22,0x3a9281,DOUBLEWITHTWODWORDINTREE(0x76a00000,0x00000000)},
  {VT_exponent,22,0x3a9282,DOUBLEWITHTWODWORDINTREE(0x76b00000,0x00000000)},
  {VT_exponent,22,0x3a9283,DOUBLEWITHTWODWORDINTREE(0x76c00000,0x00000000)},
  {VT_exponent,22,0x3a9284,DOUBLEWITHTWODWORDINTREE(0x76d00000,0x00000000)},
  {VT_exponent,22,0x3a9285,DOUBLEWITHTWODWORDINTREE(0x76e00000,0x00000000)},
  {VT_exponent,22,0x3a9286,DOUBLEWITHTWODWORDINTREE(0x76f00000,0x00000000)},
  {VT_exponent,22,0x3a9287,DOUBLEWITHTWODWORDINTREE(0x77000000,0x00000000)},
  {VT_exponent,22,0x3a9288,DOUBLEWITHTWODWORDINTREE(0x77100000,0x00000000)},
  {VT_exponent,22,0x3a9289,DOUBLEWITHTWODWORDINTREE(0x77200000,0x00000000)},
  {VT_exponent,22,0x3a928a,DOUBLEWITHTWODWORDINTREE(0x77300000,0x00000000)},
  {VT_exponent,22,0x3a928b,DOUBLEWITHTWODWORDINTREE(0x77400000,0x00000000)},
  {VT_exponent,22,0x3a928c,DOUBLEWITHTWODWORDINTREE(0x77500000,0x00000000)},
  {VT_exponent,22,0x3a928d,DOUBLEWITHTWODWORDINTREE(0x77600000,0x00000000)},
  {VT_exponent,22,0x3a928e,DOUBLEWITHTWODWORDINTREE(0x77700000,0x00000000)},
  {VT_exponent,22,0x3a928f,DOUBLEWITHTWODWORDINTREE(0x77800000,0x00000000)},
  {VT_exponent,22,0x3a9290,DOUBLEWITHTWODWORDINTREE(0x77900000,0x00000000)},
  {VT_exponent,22,0x3a9291,DOUBLEWITHTWODWORDINTREE(0x77a00000,0x00000000)},
  {VT_exponent,22,0x3a9292,DOUBLEWITHTWODWORDINTREE(0x77b00000,0x00000000)},
  {VT_exponent,22,0x3a9293,DOUBLEWITHTWODWORDINTREE(0x77c00000,0x00000000)},
  {VT_exponent,22,0x3a9294,DOUBLEWITHTWODWORDINTREE(0x77d00000,0x00000000)},
  {VT_exponent,22,0x3a9295,DOUBLEWITHTWODWORDINTREE(0x77e00000,0x00000000)},
  {VT_exponent,22,0x3a9296,DOUBLEWITHTWODWORDINTREE(0x77f00000,0x00000000)},
  {VT_exponent,22,0x3a9297,DOUBLEWITHTWODWORDINTREE(0x78000000,0x00000000)},
  {VT_exponent,22,0x3a9298,DOUBLEWITHTWODWORDINTREE(0x78100000,0x00000000)},
  {VT_exponent,22,0x3a9299,DOUBLEWITHTWODWORDINTREE(0x78200000,0x00000000)},
  {VT_exponent,22,0x3a929a,DOUBLEWITHTWODWORDINTREE(0x78300000,0x00000000)},
  {VT_exponent,22,0x3a929b,DOUBLEWITHTWODWORDINTREE(0x78400000,0x00000000)},
  {VT_exponent,22,0x3a929c,DOUBLEWITHTWODWORDINTREE(0x78500000,0x00000000)},
```

```
  {VT_exponent,22,0x3a929d,DOUBLEWITHTWODWORDINTREE(0x78600000,0x00000000)},
  {VT_exponent,22,0x3a929e,DOUBLEWITHTWODWORDINTREE(0x78700000,0x00000000)},
  {VT_exponent,22,0x3a929f,DOUBLEWITHTWODWORDINTREE(0x78800000,0x00000000)},
  {VT_exponent,22,0x3a92a0,DOUBLEWITHTWODWORDINTREE(0x78900000,0x00000000)},
  {VT_exponent,22,0x3a92a1,DOUBLEWITHTWODWORDINTREE(0x78a00000,0x00000000)},
  {VT_exponent,22,0x3a92a2,DOUBLEWITHTWODWORDINTREE(0x78b00000,0x00000000)},
  {VT_exponent,22,0x3a92a3,DOUBLEWITHTWODWORDINTREE(0x78c00000,0x00000000)},
  {VT_exponent,22,0x3a92a4,DOUBLEWITHTWODWORDINTREE(0x78d00000,0x00000000)},
  {VT_exponent,22,0x3a92a5,DOUBLEWITHTWODWORDINTREE(0x78e00000,0x00000000)},
  {VT_exponent,22,0x3a92a6,DOUBLEWITHTWODWORDINTREE(0x78f00000,0x00000000)},
  {VT_exponent,22,0x3a92a7,DOUBLEWITHTWODWORDINTREE(0x79000000,0x00000000)},
  {VT_exponent,22,0x3a92a8,DOUBLEWITHTWODWORDINTREE(0x79100000,0x00000000)},
  {VT_exponent,22,0x3a92a9,DOUBLEWITHTWODWORDINTREE(0x79200000,0x00000000)},
  {VT_exponent,22,0x3a92aa,DOUBLEWITHTWODWORDINTREE(0x79300000,0x00000000)},
  {VT_exponent,22,0x3a92ab,DOUBLEWITHTWODWORDINTREE(0x79400000,0x00000000)},
  {VT_exponent,22,0x3a92ac,DOUBLEWITHTWODWORDINTREE(0x79500000,0x00000000)},
  {VT_exponent,22,0x3a92ad,DOUBLEWITHTWODWORDINTREE(0x79600000,0x00000000)},
  {VT_exponent,22,0x3a92ae,DOUBLEWITHTWODWORDINTREE(0x79700000,0x00000000)},
  {VT_exponent,22,0x3a92af,DOUBLEWITHTWODWORDINTREE(0x79800000,0x00000000)},
  {VT_exponent,22,0x3a92b0,DOUBLEWITHTWODWORDINTREE(0x79900000,0x00000000)},
  {VT_exponent,22,0x3a92b1,DOUBLEWITHTWODWORDINTREE(0x79a00000,0x00000000)},
  {VT_exponent,22,0x3a92b2,DOUBLEWITHTWODWORDINTREE(0x79b00000,0x00000000)},
  {VT_exponent,22,0x3a92b3,DOUBLEWITHTWODWORDINTREE(0x79c00000,0x00000000)},
  {VT_exponent,22,0x3a92b4,DOUBLEWITHTWODWORDINTREE(0x79d00000,0x00000000)},
  {VT_exponent,22,0x3a92b5,DOUBLEWITHTWODWORDINTREE(0x79e00000,0x00000000)},
  {VT_exponent,22,0x3a92b6,DOUBLEWITHTWODWORDINTREE(0x79f00000,0x00000000)},
  {VT_exponent,22,0x3a92b7,DOUBLEWITHTWODWORDINTREE(0x7a000000,0x00000000)},
  {VT_exponent,22,0x3a92b8,DOUBLEWITHTWODWORDINTREE(0x7a100000,0x00000000)},
  {VT_exponent,22,0x3a92b9,DOUBLEWITHTWODWORDINTREE(0x7a200000,0x00000000)},
  {VT_exponent,22,0x3a92ba,DOUBLEWITHTWODWORDINTREE(0x7a300000,0x00000000)},
  {VT_exponent,22,0x3a92bb,DOUBLEWITHTWODWORDINTREE(0x7a400000,0x00000000)},
  {VT_exponent,22,0x3a92bc,DOUBLEWITHTWODWORDINTREE(0x7a500000,0x00000000)},
  {VT_exponent,22,0x3a92bd,DOUBLEWITHTWODWORDINTREE(0x7a600000,0x00000000)},
  {VT_exponent,22,0x3a92be,DOUBLEWITHTWODWORDINTREE(0x7a700000,0x00000000)},
  {VT_exponent,22,0x3a92bf,DOUBLEWITHTWODWORDINTREE(0x7a800000,0x00000000)},
  {VT_exponent,22,0x3a92c0,DOUBLEWITHTWODWORDINTREE(0x7a900000,0x00000000)},
  {VT_exponent,22,0x3a92c1,DOUBLEWITHTWODWORDINTREE(0x7aa00000,0x00000000)},
  {VT_exponent,22,0x3a92c2,DOUBLEWITHTWODWORDINTREE(0x7ab00000,0x00000000)},
  {VT_exponent,22,0x3a92c3,DOUBLEWITHTWODWORDINTREE(0x7ac00000,0x00000000)},
  {VT_exponent,22,0x3a92c4,DOUBLEWITHTWODWORDINTREE(0x7ad00000,0x00000000)},
  {VT_exponent,22,0x3a92c5,DOUBLEWITHTWODWORDINTREE(0x7ae00000,0x00000000)},
  {VT_exponent,22,0x3a92c6,DOUBLEWITHTWODWORDINTREE(0x7af00000,0x00000000)},
  {VT_exponent,22,0x3a92c7,DOUBLEWITHTWODWORDINTREE(0x7b000000,0x00000000)},
  {VT_exponent,22,0x3a92c8,DOUBLEWITHTWODWORDINTREE(0x7b100000,0x00000000)},
  {VT_exponent,22,0x3a92c9,DOUBLEWITHTWODWORDINTREE(0x7b200000,0x00000000)},
  {VT_exponent,22,0x3a92ca,DOUBLEWITHTWODWORDINTREE(0x7b300000,0x00000000)},
  {VT_exponent,22,0x3a92cb,DOUBLEWITHTWODWORDINTREE(0x7b400000,0x00000000)},
  {VT_exponent,22,0x3a92cc,DOUBLEWITHTWODWORDINTREE(0x7b500000,0x00000000)},
  {VT_exponent,22,0x3a92cd,DOUBLEWITHTWODWORDINTREE(0x7b600000,0x00000000)},
  {VT_exponent,22,0x3a92ce,DOUBLEWITHTWODWORDINTREE(0x7b700000,0x00000000)},
  {VT_exponent,22,0x3a92cf,DOUBLEWITHTWODWORDINTREE(0x7b800000,0x00000000)},
  {VT_exponent,22,0x3a92d0,DOUBLEWITHTWODWORDINTREE(0x7b900000,0x00000000)},
  {VT_exponent,22,0x3a92d1,DOUBLEWITHTWODWORDINTREE(0x7ba00000,0x00000000)},
  {VT_exponent,22,0x3a92d2,DOUBLEWITHTWODWORDINTREE(0x7bb00000,0x00000000)},
  {VT_exponent,22,0x3a92d3,DOUBLEWITHTWODWORDINTREE(0x7bc00000,0x00000000)},
  {VT_exponent,22,0x3a92d4,DOUBLEWITHTWODWORDINTREE(0x7bd00000,0x00000000)},
  {VT_exponent,22,0x3a92d5,DOUBLEWITHTWODWORDINTREE(0x7be00000,0x00000000)},
  {VT_exponent,22,0x3a92d6,DOUBLEWITHTWODWORDINTREE(0x7bf00000,0x00000000)},
  {VT_exponent,22,0x3a92d7,DOUBLEWITHTWODWORDINTREE(0x7c000000,0x00000000)},
  {VT_exponent,22,0x3a92d8,DOUBLEWITHTWODWORDINTREE(0x7c100000,0x00000000)},
  {VT_exponent,22,0x3a92d9,DOUBLEWITHTWODWORDINTREE(0x7c200000,0x00000000)},
```

```
  {VT_exponent,22,0x3a92da,DOUBLEWITHTWODWORDINTREE(0x7c300000,0x00000000)},
  {VT_exponent,22,0x3a92db,DOUBLEWITHTWODWORDINTREE(0x7c400000,0x00000000)},
  {VT_exponent,22,0x3a92dc,DOUBLEWITHTWODWORDINTREE(0x7c500000,0x00000000)},
  {VT_exponent,22,0x3a92dd,DOUBLEWITHTWODWORDINTREE(0x7c600000,0x00000000)},
  {VT_exponent,22,0x3a92de,DOUBLEWITHTWODWORDINTREE(0x7c700000,0x00000000)},
  {VT_exponent,22,0x3a92df,DOUBLEWITHTWODWORDINTREE(0x7c800000,0x00000000)},
  {VT_exponent,22,0x3a92e0,DOUBLEWITHTWODWORDINTREE(0x7c900000,0x00000000)},
  {VT_exponent,22,0x3a92e1,DOUBLEWITHTWODWORDINTREE(0x7ca00000,0x00000000)},
  {VT_exponent,22,0x3a92e2,DOUBLEWITHTWODWORDINTREE(0x7cb00000,0x00000000)},
  {VT_exponent,22,0x3a92e3,DOUBLEWITHTWODWORDINTREE(0x7cc00000,0x00000000)},
  {VT_exponent,22,0x3a92e4,DOUBLEWITHTWODWORDINTREE(0x7cd00000,0x00000000)},
  {VT_exponent,22,0x3a92e5,DOUBLEWITHTWODWORDINTREE(0x7ce00000,0x00000000)},
  {VT_exponent,22,0x3a92e6,DOUBLEWITHTWODWORDINTREE(0x7cf00000,0x00000000)},
  {VT_exponent,22,0x3a92e7,DOUBLEWITHTWODWORDINTREE(0x7d000000,0x00000000)},
  {VT_exponent,22,0x3a92e8,DOUBLEWITHTWODWORDINTREE(0x7d100000,0x00000000)},
  {VT_exponent,22,0x3a92e9,DOUBLEWITHTWODWORDINTREE(0x7d200000,0x00000000)},
  {VT_exponent,22,0x3a92ea,DOUBLEWITHTWODWORDINTREE(0x7d300000,0x00000000)},
  {VT_exponent,22,0x3a92eb,DOUBLEWITHTWODWORDINTREE(0x7d400000,0x00000000)},
  {VT_exponent,22,0x3a92ec,DOUBLEWITHTWODWORDINTREE(0x7d500000,0x00000000)},
  {VT_exponent,22,0x3a92ed,DOUBLEWITHTWODWORDINTREE(0x7d600000,0x00000000)},
  {VT_exponent,22,0x3a92ee,DOUBLEWITHTWODWORDINTREE(0x7d700000,0x00000000)},
  {VT_exponent,22,0x3a92ef,DOUBLEWITHTWODWORDINTREE(0x7d800000,0x00000000)},
  {VT_exponent,22,0x3a92f0,DOUBLEWITHTWODWORDINTREE(0x7d900000,0x00000000)},
  {VT_exponent,22,0x3a92f1,DOUBLEWITHTWODWORDINTREE(0x7da00000,0x00000000)},
  {VT_exponent,22,0x3a92f2,DOUBLEWITHTWODWORDINTREE(0x7db00000,0x00000000)},
  {VT_exponent,22,0x3a92f3,DOUBLEWITHTWODWORDINTREE(0x7dc00000,0x00000000)},
  {VT_exponent,22,0x3a92f4,DOUBLEWITHTWODWORDINTREE(0x7dd00000,0x00000000)},
  {VT_exponent,22,0x3a92f5,DOUBLEWITHTWODWORDINTREE(0x7de00000,0x00000000)},
  {VT_exponent,22,0x3a92f6,DOUBLEWITHTWODWORDINTREE(0x7df00000,0x00000000)},
  {VT_exponent,22,0x3a92f7,DOUBLEWITHTWODWORDINTREE(0x7e000000,0x00000000)},
  {VT_exponent,22,0x3a92f8,DOUBLEWITHTWODWORDINTREE(0x7e100000,0x00000000)},
  {VT_exponent,22,0x3a92f9,DOUBLEWITHTWODWORDINTREE(0x7e200000,0x00000000)},
  {VT_exponent,22,0x3a92fa,DOUBLEWITHTWODWORDINTREE(0x7e300000,0x00000000)},
  {VT_exponent,22,0x3a92fb,DOUBLEWITHTWODWORDINTREE(0x7e400000,0x00000000)},
  {VT_exponent,22,0x3a92fc,DOUBLEWITHTWODWORDINTREE(0x7e500000,0x00000000)},
  {VT_exponent,22,0x3a92fd,DOUBLEWITHTWODWORDINTREE(0x7e600000,0x00000000)},
  {VT_exponent,22,0x3a92fe,DOUBLEWITHTWODWORDINTREE(0x7e700000,0x00000000)},
  {VT_exponent,22,0x3a92ff,DOUBLEWITHTWODWORDINTREE(0x7e800000,0x00000000)},
  {VT_exponent,22,0x3a95a0,DOUBLEWITHTWODWORDINTREE(0x7e900000,0x00000000)},
  {VT_exponent,22,0x3a95a1,DOUBLEWITHTWODWORDINTREE(0x7ea00000,0x00000000)},
  {VT_exponent,22,0x3a95a2,DOUBLEWITHTWODWORDINTREE(0x7eb00000,0x00000000)},
  {VT_exponent,22,0x3a95a3,DOUBLEWITHTWODWORDINTREE(0x7ec00000,0x00000000)},
  {VT_exponent,22,0x3a95a4,DOUBLEWITHTWODWORDINTREE(0x7ed00000,0x00000000)},
  {VT_exponent,22,0x3a95a5,DOUBLEWITHTWODWORDINTREE(0x7ee00000,0x00000000)},
  {VT_exponent,22,0x3a95a6,DOUBLEWITHTWODWORDINTREE(0x7ef00000,0x00000000)},
  {VT_exponent,22,0x3a95a7,DOUBLEWITHTWODWORDINTREE(0x7f000000,0x00000000)},
  {VT_exponent,22,0x3a95a8,DOUBLEWITHTWODWORDINTREE(0x7f100000,0x00000000)},
  {VT_exponent,22,0x3a95a9,DOUBLEWITHTWODWORDINTREE(0x7f200000,0x00000000)},
  {VT_exponent,22,0x3a95aa,DOUBLEWITHTWODWORDINTREE(0x7f300000,0x00000000)},
  {VT_exponent,22,0x3a95ab,DOUBLEWITHTWODWORDINTREE(0x7f400000,0x00000000)},
  {VT_exponent,22,0x3a95ac,DOUBLEWITHTWODWORDINTREE(0x7f500000,0x00000000)},
  {VT_exponent,22,0x3a95ad,DOUBLEWITHTWODWORDINTREE(0x7f600000,0x00000000)},
  {VT_exponent,22,0x3a95ae,DOUBLEWITHTWODWORDINTREE(0x7f700000,0x00000000)},
  {VT_exponent,22,0x3a95af,DOUBLEWITHTWODWORDINTREE(0x7f800000,0x00000000)},
  {VT_exponent,22,0x3b8fe0,DOUBLEWITHTWODWORDINTREE(0x7f900000,0x00000000)},
  {VT_exponent,22,0x3b8fe1,DOUBLEWITHTWODWORDINTREE(0x7fa00000,0x00000000)},
  {VT_exponent,22,0x3b8fe2,DOUBLEWITHTWODWORDINTREE(0x7fb00000,0x00000000)},
  {VT_exponent,22,0x3b8fe3,DOUBLEWITHTWODWORDINTREE(0x7fc00000,0x00000000)},
  {VT_exponent,21,0x68e90,DOUBLEWITHTWODWORDINTREE(0x7fd00000,0x00000000)},
  {VT_exponent,21,0x68e91,DOUBLEWITHTWODWORDINTREE(0x7fe00000,0x00000000)},
  {VT_exponent,21,0x68e98,DOUBLEWITHTWODWORDINTREE(0x7ff80000,0x00000000)},
```

```
}; // End of acofdoe array
```

11.18 Procedure for WriteDouble

WriteDouble uses the array defined in **Data definition for double storage** to find common floating point values for CAD data.

```
/* Internal definitions for floating point data */
union ieee754_double
{
   double d;
/* This is the IEEE 754 double-precision format. */
   struct
   {
#    if defined(PRC_BIG_ENDIAN)
     unsigned int negative:1;
     unsigned int exponent:11;
  /* Together these comprise the mantissa.  */
     unsigned int mantissa0:20;
     unsigned int mantissa1:32;
#    endif
#    if defined(PRC_LITTLE_ENDIAN)
  /* Together these comprise the mantissa.  */
     unsigned int mantissa1:32;
     unsigned int mantissa0:20;
     unsigned int exponent:11;
     unsigned int negative:1;
#    endif
   } ieee;
};

// Macros for LITTLE ENDIAN machines
#if defined(PRC_LITTLE_ENDIAN)
#   define DOUBLEWITHTWODWORD(upper,lower)  lower,upper
#   define UPPERPOWER       (1)
#   define LOWERPOWER       (!UPPERPOWER)

#   define NEXTBYTE(pbd)                    ((pbd)--)
#   define PREVIOUSBYTE(pbd)                ((pbd)++)
#   define MOREBYTE(pbd,pbend)              ((pbd)>=(pbend))
#   define OFFSETBYTE(pbd,offset)           ((pbd)-=offset)
#   define BEFOREBYTE(pbd)                  ((pbd)+1)
#   define DIFFPOINTERS(p1,p2)              ((unsigned)((p2)-(p1)))
#   define SEARCHBYTE(pbstart,b,nb) (unsigned char *)memchr((pbstart),(b),(nb))
#   define BYTEAT(pb,i)                     *((pb)+(i))
// Macros for BIG ENDIAN machines
#elif defined(PRC_BIG_ENDIAN)
#   define DOUBLEWITHTWODWORD(upper,lower)  upper,lower
#   define UPPERPOWER       (0)
#   define LOWERPOWER       (!UPPERPOWER)

#   define NEXTBYTE(pbd)                    ((pbd)++)
#   define PREVIOUSBYTE(pbd)                ((pbd)--)
#   define MOREBYTE(pbd,pbend)              ((pbd)<=(pbend))
#   define OFFSETBYTE(pbd,offset)           ((pbd)+=offset)
#   define BEFOREBYTE(pbd)                  ((pbd)-1)
#   define DIFFPOINTERS(p1,p2)              ((p1)-(p2))
#   define SEARCHBYTE(pbstart,b,nb) (unsigned char *)memrchr((pbstart),(b),(nb))
#   define BYTEAT(pb,i)                     *((pb)-(i))
```

```
#else
#   error "Big/Little endian to be defined"
#endif

// Common macros and types
#define MAXLENGTHFORCOMPRESSEDTYPE      ((22+1+1+4+6*(1+8))+7)/8

#define NEGATIVE(d)     (((union ieee754_double *)&(d))->ieee.negative)
#define EXPONENT(d)     (((union ieee754_double *)&(d))->ieee.exponent)
#define MANTISSA0(d)    (((union ieee754_double *)&(d))->ieee.mantissa0)
#define MANTISSA1(d)    (((union ieee754_double *)&(d))->ieee.mantissa1)

typedef unsigned char PRCbyte;
typedef unsigned short PRCword;
typedef unsigned PRCdword;

static PRCdword
        stadwZero[2]={DOUBLEWITHTWODWORD(0x00000000,0x00000000)},
        stadwNegativeZero[2]={DOUBLEWITHTWODWORD(0x80000000,0x00000000)};

/* End of Internal definitions */


// Internally used functions

/* Internal functions */

static int stCOFDOECompare(const void* pcofdoe1,const void* pcofdoe2)
{
   return(EXPONENT(((const sCodageOfFrequentDoubleOrExponent *)pcofdoe1)-
>u2uod.Value)-
          EXPONENT(((const sCodageOfFrequentDoubleOrExponent *)pcofdoe2)-
>u2uod.Value));
}

#if defined(PRC_BIG_ENDIAN)
static void *memrchr(const void *buf,int c,size_t count)
{
   unsigned char
      *pcBuffer=(unsigned char *)buf,
      *pcBufferEnd=pcBuffer-count;

   for(;pcBuffer>pcBufferEnd;pcBuffer--)
      if(*pcBuffer==c)
         return(pcBuffer);

   return(NULL);
}
#endif

/* End of internal functions */

# define STAT_V
# define STAT_DOUBLE
# define add_bits AddBits

void WriteDouble( double value )
{
   union ieee754_double *pid=(union ieee754_double *)&value;
   int
```

```
         i,
         fSaveAtEnd;
   PRCbyte
         *pb,
         *pbStart,
         *pbStop,
         *pbEnd,
         *pbResult,
         bSaveAtEnd;
   sCodageOfFrequentDoubleOrExponent
         cofdoe,
         *pcofdoe;

   cofdoe.u2uod.Value=value;
   pcofdoe = (sCodageOfFrequentDoubleOrExponent *)bsearch(
                  &cofdoe,
                  acofdoe,
                  sizeof(acofdoe)/sizeof(pcofdoe[0]),
                  sizeof(pcofdoe[0]),
                  stCOFDOECompare);

   while(pcofdoe>acofdoe &&
         EXPONENT(pcofdoe->u2uod.Value)==EXPONENT((pcofdoe-1)->u2uod.Value))
      pcofdoe--;

   while(pcofdoe->Type==VT_double)
   {
      if(fabs(value)==pcofdoe->u2uod.Value)
         break;
         pcofdoe++;
   }

   for(i=1<<(pcofdoe->NumberOfBits-1);i>=1;i>>=1)
      add_bits((pcofdoe->Bits&i)!=0,1 STAT_V STAT_DOUBLE);

   if
   (
     !memcmp(&value,stadwZero,sizeof(value)) ||
     !memcmp(&value,stadwNegativeZero,sizeof(value))
   )
      return;

   add_bits(pid->ieee.negative,1 STAT_V STAT_DOUBLE);

   if(pcofdoe->Type==VT_double)
      return;

   if(pid->ieee.mantissa0==0 && pid->ieee.mantissa1==0)
   {
      add_bits(0,1 STAT_V STAT_DOUBLE);
      return;
   }

   add_bits(1,1 STAT_V STAT_DOUBLE);

#  if defined(PRC_LITTLE_ENDIAN)
      pb=((PRCbyte *)&value)+6;
#  elif defined(PRC_BIG_ENDIAN)
      pb=((PRCbyte *)&value)+1;
#  endif
```

```
   add_bits((*pb)&0x0f,4 STAT_V STAT_DOUBLE);

   NEXTBYTE(pb);
   pbStart=pb;
#  if defined(PRC_LITTLE_ENDIAN)
      pbEnd=
      pbStop= ((PRCbyte *)&value);
#  elif defined(PRC_BIG_ENDIAN)
      pbEnd=
      pbStop= ((PRCbyte *)(&value+1))-1;
#  endif

   if((fSaveAtEnd=(*pbStop!=*BEFOREBYTE(pbStop)))!=0)
      bSaveAtEnd=*pbEnd;
   PREVIOUSBYTE(pbStop);

   while(*pbStop==*BEFOREBYTE(pbStop))
      PREVIOUSBYTE(pbStop);

   for(;MOREBYTE(pb,pbStop);NEXTBYTE(pb))
   {
      if(pb!=pbStart &&
          (pbResult=SEARCHBYTE(
              BEFOREBYTE(pb),*pb,DIFFPOINTERS(pb,pbStart)))!=NULL)
      {
         add_bits(0,1 STAT_V STAT_DOUBLE);
         add_bits(DIFFPOINTERS(pb,pbResult),3 STAT_V STAT_DOUBLE);
      }
      else
      {
         add_bits(1,1 STAT_V STAT_DOUBLE);
         add_bits(*pb,8 STAT_V STAT_DOUBLE);
      }
   }

   if(!MOREBYTE(BEFOREBYTE(pbEnd),pbStop))
   {
      if(fSaveAtEnd)
      {
         add_bits(0,1 STAT_V STAT_DOUBLE);
         add_bits(6,3 STAT_V STAT_DOUBLE);
         add_bits(bSaveAtEnd,8 STAT_V STAT_DOUBLE);
      }
      else
      {
                        add_bits(0,1 STAT_V STAT_DOUBLE);
                        add_bits(0,3 STAT_V STAT_DOUBLE);
                }
        }
        else
        {
           if((pbResult=SEARCHBYTE(
                 BEFOREBYTE(pb),*pb,DIFFPOINTERS(pb,pbStart)))!=NULL)
           {
              add_bits(0,1 STAT_V STAT_DOUBLE);
              add_bits(DIFFPOINTERS(pb,pbResult),3 STAT_V STAT_DOUBLE);
           }
           else
           {
              add_bits(1,1 STAT_V STAT_DOUBLE);
```

```
            add_bits(*pb,8 STAT_V STAT_DOUBLE);
         }
      }
   }
```

12 Tessellation Compression Support

12.1 General

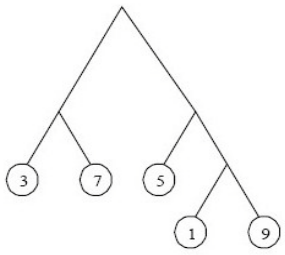
The following two sections describe numerical techniques used for compressed tessellation data (section 7.8).

12.2 Huffman Algorithm

Huffman coding is an algorithm used for lossless data compression (see Bibliography). The particular implementation of this algorithm used in the PRC format is described below.

The Hufffman algorithm used can support char (1 byte) or short (2 bytes) arrays as input, and compresses them using a number of bits which depends on maximum values that it contains. To compact data, a huffman tree is built in order to store the highest frequent elements with the smallest code. Each input value is located on a leaf.

Example : with the input, {1,3,5,7,7,9,5,3,3}, the following tree is build.



Value	Code	Code Length
3	00	2
7	01	2
5	10	2
1	110	3
9	111	3

This version of PRC imposes that code length is limited to 32.

The pseudo code below describes how the binary tree is stored in a bit field :

```
#include <string.h>

void HuffmanTreeCalculation(
   char* pcArray,
   unsigned uCharArraySize,
   unsigned uNumberOfBitsForValues,
   unsigned& uNumberOfLeaves,
   unsigned& uMaxCodeLength,
   unsigned*&puLeafValues,
   unsigned*&puLeafCodeLength,
   unsigned*&puLeafCodeValues);

static bool* AddBitInArray(
   bool* pbBitArray,
   bool bValue,
   unsigned& uMaxSize,
   unsigned& uSize)
{
   if (uSize == uMaxSize)
   {
      uMaxSize+=uMaxSize/10;
      bool* pbNew = new bool[uMaxSize];
      memcpy(pbNew,pbBitArray,uSize*sizeof(bool));
      delete [] pbBitArray;
      pbBitArray=pbNew;
   }
   pbBitArray[uSize]=bValue;
   uSize++;
   return pbBitArray;
}

// uNumberOfBitsForValues is an input of the Huffman algorithm.
// For instance, when writing Edge_status_array in XXX4.7.8.7,
// this number is equal to 2.
void Huffman(
   char* pcArray,
   unsigned uCharArraySize,
   unsigned uNumberOfBitsForValues,
   unsigned*& puHuffmanArray,
   unsigned& uHuffmanArraySize)
{
   unsigned uNumberOfLeaves,
      uMaxCodeLength,
      *puLeafValues,
      *puLeafCodeLength,
      *puLeafCodeValues;

   HuffmanTreeCalculation(
      pcArray,
      uCharArraySize,
      uNumberOfBitsForValues,
      uNumberOfLeaves,
      uMaxCodeLength,
```

```
    puLeafValues,
    puLeafCodeLength,
    puLeafCodeValues);

// here we have to allocate a dynamic array growing accordingly
unsigned uSize=0,uMaxSize=24*uCharArraySize;
bool *pbBitArray = new bool[uMaxSize];

// writing bit-by-bit the number of leaves
unsigned u,v;
for(u=0; u < uNumberOfBitsForValues + 1; u++)
   pbBitArray = AddBitInArray(
                    pbBitArray,
                    uNumberOfLeaves & (1<<u) ? true : false,
                    uMaxSize,
                    uSize);

// writing bit-by-bit the max code length on 8 bits
for(u=0; u < 8; u++)
   pbBitArray = AddBitInArray(
                    pbBitArray,
                    uMaxCodeLength &(1<<u) ? true : false,
                    uMaxSize,
                    uSize);

for(v = 0; v < uNumberOfLeaves; v++)
{
   // writing leaf value (same values as in pcArray, in different order)
   for(u=0; u < uNumberOfBitsForValues; u++)
      pbBitArray = AddBitInArray(
                       pbBitArray,
                       puLeafValues[v] &(1<<u) ? true : false,
                       uMaxSize,
                       uSize);

   // writing leaf code length
   for(u=0; u < 8; u++)
      pbBitArray = AddBitInArray(
                       pbBitArray,
                       puLeafCodeLength[v] &(1<<u) ? true : false,
                       uMaxSize,
                       uSize);

   // writing leaf code value
   for(u=0; u < puLeafCodeLength[v]; u++)
      pbBitArray = AddBitInArray(
                       pbBitArray,
                       puLeafCodeValues[v] &(1<<u) ? true : false,
                       uMaxSize,
                       uSize);
}

unsigned uBitsInUnsigned = sizeof(unsigned) * 8;
uHuffmanArraySize = uSize % uBitsInUnsigned ?
   (uSize / uBitsInUnsigned)+1 : uSize / uBitsInUnsigned;

puHuffmanArray = new unsigned[uHuffmanArraySize];

unsigned* puCurrent=puHuffmanArray;
unsigned uTot=0;
for (u=0;u<uHuffmanArraySize-1;u++,puCurrent++)
```

```
   {
      *puCurrent=0;
      for (v=0;v<uBitsInUnsigned;v++,uTot++)
         *puCurrent |= ( pbBitArray[uTot] ? 1 : 0 ) << v;
   }

   v=0;
   *puCurrent=0;
   for(u = uBitsInUnsigned*(uHuffmanArraySize-1);u < uSize; u++, uTot++, v++ )
      *puCurrent |= ( pbBitArray[uTot] ? 1 : 0 ) << v;
   delete [] pbBitArray;

}
```

Then the output unsigned integer array is used by **WriteCharacterArray** or **WriteShortArray** as described in section 11.

12.3 Basis pseudocode

This pseudo code describes basic calculation functions which have to be performed with IEEE standard (see Bibliography). All basic calculations below must be performed using floating-point processors working on 64 bits precision. All values, types and macros below belong to IEEE standard for single or double precision as well.

```
#include <float.h>
#include <math.h>

// For compilation on little-endian machines;
// please define PRC_BIG_ENDIAN otherwise
# define PRC_LITTLE_ENDIAN

class PrcPt
{
   double m_fx;
   double m_fy;
   double m_fz;
public :
   PrcPt() {}

   PrcPt(double fx, double fy, double fz) :
   m_fx(fx), m_fy(fy), m_fz(fz) {}

   PrcPt(const PrcPt& sPrcPt) :
   m_fx(sPrcPt.m_fx), m_fy(sPrcPt.m_fy), m_fz(sPrcPt.m_fz) {}
   void Set(double fx, double fy, double fz)
   { m_fx = fx; m_fy = fy; m_fz = fz; }

   double Dot(const PrcPt & sPt) const
   {
      return (m_fx*sPt.m_fx)+(m_fy*sPt.m_fy)+(m_fz*sPt.m_fz);
   }
   double LengthSquared()
   {
      return (m_fx*m_fx + m_fy*m_fy + m_fz*m_fz);
   }
```

```
   friend PrcPt operator + (const PrcPt& a, const PrcPt& b)
   {
      return PrcPt(a.m_fx+b.m_fx, a.m_fy+b.m_fy, a.m_fz+b.m_fz);
   }
   friend PrcPt operator - (const PrcPt& a)
   {
      return PrcPt(-a.m_fx, -a.m_fy, -a.m_fz);
   }
   friend PrcPt operator - (const PrcPt& a, const PrcPt& b)
   {
      return PrcPt(a.m_fx-b.m_fx, a.m_fy-b.m_fy, a.m_fz-b.m_fz);
   }

   friend PrcPt operator * (const PrcPt& a, const double d)
   {
      return PrcPt(a.m_fx * d, a.m_fy * d, a.m_fz * d);
   }

   friend PrcPt operator * (const double d, const PrcPt& a)
   {
      return PrcPt(a.m_fx*d, a.m_fy*d, a.m_fz*d);
   }
   friend PrcPt operator / (const PrcPt& a, const double d)
   {
      return PrcPt(a.m_fx/d, a.m_fy/d, a.m_fz/d);
   }
   friend PrcPt operator * (const PrcPt& a, const PrcPt& b)
   {
      return PrcPt( (a.m_fy*b.m_fz)-(a.m_fz*b.m_fy),
         (a.m_fz*b.m_fx)-(a.m_fx*b.m_fz),
         (a.m_fx*b.m_fy)-(a.m_fy*b.m_fx));
   }
   double Length();

   int Unitize();

   int MakeOrthoRep(PrcPt& sY, PrcPt& sZ);


   int AngleBetween(const PrcPt& sPt,double& dAngleRadians) const;
};

union ieee754_float
{
   float f;
   /* This is the IEEE 754 float-precision format. */
   struct
   {
#if defined(PRC_BIG_ENDIAN)
      unsigned int negative:1;
      unsigned int exponent:8;
      unsigned int mantissa:23;
#elif defined(PRC_LITTLE_ENDIAN)
      unsigned int mantissa:23;
      unsigned int exponent:8;
      unsigned int negative:1;
#else
#  error "Big/Little endian to be defined"
#endif
   } ieee;
};
```

```
double PrcPt::Length()
{
   ieee754_float fSomme;
   fSomme.f = (float) (m_fx*m_fx + m_fy*m_fy + m_fz*m_fz);
   double dSquared = (double) fSomme.f;
   if(fSomme.ieee.exponent > 127)
      fSomme.ieee.exponent = (fSomme.ieee.exponent - 127) /2 + 127;
   double dX_i;
   double dX_0 = (double) fSomme.f;
   unsigned int uCount = 0;
   while(uCount != 100) {
      dX_i = 0.5 * (dX_0 + dSquared / dX_0);
      if((double) dX_i == (double) dX_0) dX_0 = dX_i;
      uCount++;
   };
   if(uCount == 100) return (double) -1.0;
   return (double) dX_i;
}

int PrcPt::Unitize()
{
   double fLength = Length();
   if(fLength < FLT_EPSILON) return -1;
   m_fx = m_fx/fLength;
   m_fy = m_fy/fLength;
   m_fz = m_fz/fLength;
   return 0;
}

int PrcPt::AngleBetween(
   const PrcPt & sOther,double &dAngleRadians) const
{
   PrcPt sV1 = *this;
   PrcPt sV2 = sOther;
   double dV1Len = sV1.Length();
   double dV2Len = sV2.Length();
   if(dV1Len < FLT_EPSILON || dV2Len < FLT_EPSILON)return 1;
   sV1 = sV1 / dV1Len;
   sV2 = sV2 / dV2Len;
   double dDot = sV1.Dot(sV2);
   if (dDot > 1.0-1e-12)
      dDot = 1.0;
   else if (dDot < -1.0+1e-12)
      dDot = -1.0;
   dAngleRadians = acos(dDot);
   return 0;
}
int PrcPt::MakeOrthoRep(PrcPt& sY, PrcPt& sZ)
{
   PrcPt sX = *this;
   if(sX.Unitize())
      return -1;
   sY.Set(0,1.0,0);
   sZ = sX * sY;
   if(sZ.Unitize())
   {
      sY.Set(1.0,0,0);
      sZ = sX * sY;
      if(sZ.Unitize())
```

```
         return -1;
   }
   sY = sZ * sX;
   if(sY.Unitize())
      return -1;
   return 0;
}
```

Bibliography

D.A. Huffman, “A method for the construction of minimum-redundancy codes“, Proceedings of the I.R.E., septembre 1952, pp 1098-1102

Institute of Electrical and Electronics Engineers, *IEEE Standard for Binary Floating-Point Arithmetic*(IEEE 754-1985)

Internet RFCs 1950, *ZLIB Compressed Data Format Specification*, and 1951, *DEFLATE Compressed Data Format Specification*

ISO 10303, *Industrial automation systems and integration – Product data representation and exchange*