

Exploring

2002

October 2001

Introduction

1

System Basics

2

Introduction to Interactive Working

3

System Handling

4

Working with Models

5

Introductory Example

6

Creation of Objects

7

Updating Objects

8

Deletion of Objects

9

Manipulating Objects

10

Transformation Techniques

11

Dimensioning and Symbols

12

How to contact us:

In Germany:

EUKLID Software GmbH
Vor dem Lauch 19
D-70567 Stuttgart
Tel: +49 / 711 / 7 22 84-0 · Fax: +49 / 711 / 7 22 84- 293
Internet: <http://www.euklid-software.de>

In Switzerland:

EUKLID Software GmbH
St. Gallerstrasse 151
CH-8645 Jona, SG
Tel: +41 / 55 / 21270-43 · Fax: +41 / 55 / 21270-44

You have questions, recomments to...

... products

... or other subjects of the technical
information system?

For further information please call our offices.

© 2001 EUKLID Software GmbH
All rights reserved.

AcrobatReader™ is a registered trademark of Adobe Inc.
I-DEAS™ is a registered trademark of SDRC Inc.
PARASOLID™ is a registered trademark of EDS Corp.
SolidWorks® is a registered trademark of SolidWorks Corp.

Standardization

13

Visualization Techniques

14

Construction Aids

15

Bill of Material Generation

16

Configuring the System

17

Plotting

18

Mixed-up 3D-2D design

19

Data transfer

20

Data Safety

21

Appendix

22

Glossary

23

Contents

1	Introduction	1-1
2	System Basics	2-1
2.1	System Philosophy	2-1
2.2	The Data Structure	2-5
2.2.1	Effect Objects	2-13
2.3	The User Interface	2-18
2.3.1	Data Input Philosophy	2-18
2.3.2	Input Errors	2-22
2.3.3	Action Digression	2-22
2.3.4	Model Inquiry and Modification	2-23
2.4	Special Features	2-24
2.4.1	Mutual Association between Geometry and Dimensions	2-24
2.4.2	Hierarchical Layer/Structure Concept	2-24
2.4.3	Standardization	2-25
2.4.4	Programming Language AQL	2-27
2.4.5	Free Configurability	2-27
2.5	Complex Applications	2-28
2.5.1	Multiple Elevation Design	2-28
2.5.2	Design Variants	2-30
2.5.3	Detail Design	2-31
2.5.4	Kinematic Simulation, Assembly Checks and Snag Tests	2-32
2.5.5	Functional Contexts	2-33
3	Introduction to Interactive Working	3-1
3.1	Hardware and Operating System	3-1
3.2	Starting <i>EUKLID Design</i>	3-2
3.2.1	Start options	3-2
3.3	Basic Elements	3-7
3.3.1	The Screen Areas	3-7
3.3.2	The Mouse	3-23

3.3.3	The Cursor	3-24
3.3.4	Popup Menus	3-26
3.3.5	Use of Scrollbars	3-29
3.3.6	The Keyboard	3-29
3.4	Help in EUKLID Design	3-36
3.4.1	The Online Help	3-36
3.4.2	The Online Documentation	3-37
3.4.3	The Tooltips	3-38
3.5	Finishing a Session	3-41
3.6	Program Interruption	3-42
3.7	Abort the Running AQL Program	3-42
4	System Handling	4-1
4.1	Input Facility	4-1
4.2	Parameters and Properties	4-5
4.2.1	Types of Properties and Parameters	4-7
4.3	Other Parameter Types	4-10
4.3.1	Parameter Lists	4-10
4.3.2	Alternative Parameter	4-14
4.4	Object Names	4-16
4.4.1	Assigning of Names	4-17
4.4.2	Display and Selection of Names	4-18
4.4.3	Changing of Names	4-20
4.4.4	Deletion of Names	4-20
4.5	Identification of Objects	4-21
4.5.1	Selection Mechanisms	4-21
4.5.2	Selection Modes	4-31
4.6	Object Groups	4-35
4.6.1	Creation of Groups	4-36
5	Working with Models	5-1
5.1	Application Concept	5-1
5.1.1	Using EUKLID Design Tools	5-2
5.1.2	Structuring the Product Development Process	5-6
5.1.3	Structuring the Product	5-13
5.2	Layering	5-18
5.2.1	The Layer Menu	5-19
5.2.2	Creating Layers	5-20
5.2.3	Setting the Active Layer and Model	5-21

5.2.4	Changing the Layer Status	5-22
5.2.5	Moving Objects to Another Layer	5-24
5.2.6	Moving Layers	5-26
5.2.7	External Layers	5-27
5.2.8	Converting Layers to User Object	5-34
5.2.9	Adding a User-defined Object Definition to a Layer	5-35
5.2.10	Deleting Layers	5-36
5.2.11	Show corresponding Layer of an object	5-38
5.3	File Operations	5-39
5.3.1	Creation of New Models	5-44
5.3.2	Opening Models	5-45
5.3.3	Reopening Models	5-49
5.3.4	Inserting Models	5-50
5.3.5	Open models directly	5-51
5.3.6	Closing Models	5-52
5.3.7	Saving models	5-53
5.3.8	First time saving a model	5-53
5.3.9	Saving a model again	5-54
5.3.10	Saving Models in new files	5-56
6	Introductory Example	6-1
6.1	The Design Example	6-3
6.2	Creating the Design	6-4
6.3	Design of Further Spoke Variants	6-11
6.4	Editing Parameters	6-17
6.5	Enlarge Window	6-22
6.6	Saving the Example Model	6-25
7	Creation of Objects	7-1
7.1	Explicit Construction	7-1
7.1.1	Selection of Menu	7-3
7.1.2	Selection of Action Group	7-3
7.1.3	Selection of Action	7-3
7.1.4	Input of Properties and Parameters	7-4
7.2	Implicit Construction	7-5
7.3	Object Types	7-6
7.3.1	Basic Objects	7-6
7.3.2	Geometry Objects	7-14
7.3.3	Contours	7-15

7.3.4	Planes	7-17
7.3.5	Text	7-18
7.3.6	Measures	7-19
7.3.7	Symbols	7-20
7.3.8	Copy Vectors	7-22
7.3.9	Coordinate Systems	7-23
7.3.10	Variables	7-27
7.3.11	Groups	7-29
7.3.12	Tables	7-30
7.3.13	Table Index	7-30
7.3.14	User Defined Objects	7-31
8	Updating Objects	8-1
8.1	Examination of Design Data Structures	8-2
8.1.1	Descending through the Data Structure	8-3
8.1.2	Ascending through the Data Structure	8-5
8.2	Editing an Object	8-6
8.2.1	Editing Properties and Parameters	8-6
8.2.2	Editing Actions	8-14
8.3	Editing Objects in Current Mode	8-28
8.3.1	Quick Edit of the Object Closest to the Cursor Position	8-28
8.3.2	Quick Edit of the Object Last Created	8-29
8.3.3	Discontinuing Quick Edit Mode	8-29
8.4	Editing via Iteration (Specific Value Entry)	8-30
8.5	Adjusting a Sketched Profile (Adjust Action)	8-34
8.6	Undoing the Last Edit	8-35
9	Deletion of Objects	9-1
9.1	Deletion of Objects in Delete Mode	9-3
9.1.1	Deletion of Single Objects in Delete Mode	9-3
9.1.2	Deletion of Object Sets	9-4
9.2	Deletion of Objects in Edit Mode	9-7
9.2.1	Deletion of Objects in Edit Mode	9-7
9.2.2	Action-specific Deletion of Objects in Edit Mode	9-9
9.3	Undoing Delete Actions	9-10
9.3.1	Undoing the Latest Object-specific Delete Action in Delete Mode	9-11
9.3.2	Undoing the Latest Delete Action in Edit Mode	9-11
9.3.3	Recovering Currently Object-specifically Deleted Object Sets in Edit Mode	9-12

9.3.4	Undoing a Previous Delete Action	9-13
10	Manipulating Objects	10-1
10.1	Trimming	10-2
10.1.1	Trimming of a Line up to a Line	10-2
10.1.2	Trimming of two Lines up to an Intersection Point	10-2
10.2	Chamfer	10-3
10.2.1	Chamfer between two Lines	10-3
10.2.2	Chamfer between two Shortened Newly Created Lines	10-3
10.3	Fillet	10-4
10.3.1	Fillet between two Lines	10-4
10.3.2	Fillet between two Shortened Newly Created Lines	10-4
11	Transformation Techniques	11-1
11.1	Copying Objects	11-2
11.1.1	Copying Objects by Defining a Copy Vector	11-4
11.1.2	Copying Objects by Defining a Mirroring Axis	11-6
11.2	Duplicating Objects	11-9
12	Dimensioning and Symbols	12-1
12.1	Dimensioning	12-2
12.1.1	Dynamical Positioning of Measures (Crosshair Guidance)	12-3
12.1.2	Creating Distance Dimensions	12-6
12.1.3	Radius Dimensions	12-10
12.1.4	Diameter Dimensions	12-11
12.1.5	Angle Dimensions, Opening Angle and Length Dimensions of Circular Arcs	12-12
12.1.6	Chain and Referenced Dimensions	12-14
12.1.7	Dimensioning Parameters and Properties	12-17
12.1.8	Manipulation of Witness Lines	12-22
12.1.9	Adjusting a Sketched Profile	12-24
12.2	Coordinate Tables	12-25
12.3	Symbols	12-27
12.3.1	General Symbols	12-28
12.4	Symbol Balloons	12-31
12.4.1	Welding Symbols According to DIN 1912	12-33

13	Standardization	13-1
13.1	User-defined Object Types and User-defined Actions	13-2
13.1.1	The Creation of User-defined Objects in Models	13-7
13.1.2	Changing User-defined Objects in Models	13-8
13.1.3	Using User-defined Action in Models	13-9
13.1.4	Conception of Object Types	13-10
13.1.5	Defining Object Types	13-11
13.1.6	Saving User-defined Object Types	13-20
13.1.7	Appending Object Type Icons in the User Interface	13-20
13.1.8	Saving the Configuration	13-20
13.1.9	Loading User-defined Object Types	13-21
13.1.10	Adopting Object Type Definitions for User-Defined Objects	13-22
13.1.11	Editing Object Type Definitions	13-23
13.1.12	Access to Objects of User-defined Objects	13-24
13.1.13	Converting User-defined Objects into Layers	13-24
13.1.14	Conception of Actions	13-25
13.1.15	Defining Actions	13-26
13.1.16	Saving User-defined Actions	13-43
13.1.17	Appending User-defined Action Icons to the User Interface	13-43
13.1.18	Saving the User-defined Action Configuration	13-43
13.1.19	Loading User-defined Actions	13-44
13.1.20	Editing Action Definitions	13-45
13.1.21	Creating User Icons	13-46
13.1.22	Saving User-defined Object Types and User-defined Actions	13-54
13.1.23	Appending Object Type Icons and Action Icons to the User Interface	13-55
13.1.24	Saving the Object Type and Action Configuration	13-64
13.1.25	Nesting User-defined Object Types and User-defined Actions	13-67
13.1.26	Definition Example for User-defined Object Types and User-defined Actions	13-73
13.1.27	Notes on the Data Structure	13-82
13.2	Working with Variables	13-91
13.2.1	Simplification of Construction Modifications	13-95
13.2.2	Formula Processing	13-97
13.2.3	Calculation for Planes and Closed Contours	13-114
13.3	Tables	13-117
13.3.1	Table Design	13-122
13.3.2	Creating Tables	13-124
13.3.3	Example for an Internal Table	13-134
13.3.4	Creating an Index	13-136

13.3.5	Data Extraction from Tables	13-139
13.3.6	Nesting of Tables	13-142
13.3.7	Application of Tables to Standard Components	13-146
14	Visualization Techniques	14-1
14.1	Windows	14-1
14.1.1	The Desktop	14-1
14.1.2	Types of Windows	14-2
14.1.3	Basic Elements of Model Windows	14-2
14.1.4	Open a Model Window	14-3
14.1.5	Change size and position	14-4
14.1.6	Activate Models and Model Windows	14-4
14.1.7	Arrange Model windows	14-5
14.1.8	Close Model Windows	14-5
14.1.9	The View of a Window	14-6
14.1.10	View using the function keys F5 - F9	14-9
14.2	Grid	14-9
14.3	Line Style	14-10
14.4	Colors of Objects	14-12
14.5	Suppression of Hidden Lines	14-14
14.6	Visualization of Layers (LAVIS)	14-18
14.6.1	Symbolic Visualization of Models	14-18
14.6.2	Creating and Changing a Structure Model	14-20
15	Construction Aids	15-1
15.1	Sketcher	15-2
15.1.1	Introduction	15-2
15.1.2	Application Features	15-3
15.1.3	Sketching an Elementary Object	15-6
15.2	Designing with the Sketcher	15-13
15.2.1	Constructing a Closed Sequence of Line Segments	15-14
15.2.2	Constructing the Bore Hole	15-18
15.2.3	Constructing the Side View	15-19
15.2.4	Shading a Cross Section	15-23
15.3	Sketching of Contours and Planes	15-25
15.3.1	Sketching a Contour – contour_sketch	15-27
15.3.2	Notes on the Application of the Sketching Action	15-29
15.3.3	Properties of the Sketching Action	15-29
15.3.4	Sketching a Symmetric Profile – automirror	15-31

15.4	Automatic Creation of Contours and Planes	15-32
15.4.1	General Notes on the Creation of Contours and Planes	15-33
15.4.2	Procedure	15-38
15.5	Macros	15-42
15.5.1	Indefinite Lines	15-42
15.5.2	Rectangle (with rounded corners)	15-42
15.5.3	Oblong Hole	15-43
15.5.4	Polygon	15-43
15.5.5	Groove	15-43
15.5.6	Shaft End	15-44
15.5.7	Reticule of a Circle	15-44
15.6	Drawing Frames	15-45
16	Bill of Material Generation	16-1
16.1	Storage of Bill of Material Data in the Model	16-3
16.2	Bill of Material Output	16-7
16.2.1	Extracting Bills of Material to Models	16-7
16.2.2	Extracting a Bill of Material to a File	16-13
16.3	Creating a Bill of Material	16-16
16.3.1	Creation of Bill of Material Balloons	16-17
16.3.2	Creation of a Bill of Material Balloon as a Copy of an Existing one	16-19
16.3.3	Extracting the Bill of Material to the Model	16-20
16.3.4	Extracting a Bill of Material to a File	16-21
16.4	Bill of Material Configuration	16-22
16.4.1	Bill of Material Data Configuration	16-22
16.4.2	Configuration of Bills of Material in Models	16-24
16.4.3	Configuration of Bills of Material in Files	16-30
17	Configuring the System	17-1
17.1	The System Configuration	17-2
17.2	The User-oriented Configuration	17-5
17.3	The Model-oriented Configuration	17-5
17.4	Loading the Configuration	17-6
17.5	Saving the Current Configuration	17-7
17.5.1	Saving System	17-7
17.5.2	User Specific Saving	17-8
17.6	Configuring the Interprocess Communication (IPC)	17-9
17.7	Configuring the User Interface	17-11
17.7.1	Switching the Cursor Icon On and Off	17-11

17.7.2	Switching the Exec Cursor On and Off	17-11
17.7.3	Setting the Dialog Language	17-12
17.7.4	Setting the Colors	17-12
17.7.5	Define Default viewport type	17-12
17.7.6	Define Function keys	17-13
17.7.7	Configuration of the Main Menu	17-14
17.8	Configuration of the file system	17-25
17.8.1	File search rule (location table)	17-25
17.8.2	The Standard Library "+"	17-28
18	Plotting	18-1
18.1	Setting Plot Properties	18-2
18.2	Handling Texts	18-3
18.3	Handling Line Thickness	18-4
18.4	Handling Line Patterns	18-6
18.5	Handling Colors	18-7
18.6	Settings for HP-GL	18-8
18.7	Plotting in Batch Mode	18-9
18.8	Plot Output of Variable Extracts	18-10
18.9	Plot Output of Fixed Extracts	18-13
19	Mixed-up 3D-2D design	19-1
19.1	SolidJoin	19-1
19.1.1	Procedure	19-2
19.1.2	Additional functionality	19-3
19.2	I-DEAS link	19-5
19.2.1	Configuration of data transfer from I-DEAS Drafting Setup	19-6
19.2.2	EUKLID Design as a 2D component of I-DEAS Master Series	19-14
19.2.3	Explicit data transfer from I-DEAS to EUKLID Design	19-23
19.2.4	Explicit data transfer to I-DEAS Master Modeler	19-26
20	Data transfer	20-1
20.1	Converting Draft Data	20-2
20.1.1	General Element Characteristics	20-3
20.1.2	Elements transferred	20-5
20.1.3	Configuring the converter	20-9
20.1.4	Mapping tables	20-9
20.2	Data Transfer via IGES, DWG and DXF	20-15
20.2.1	Data transfer via IGES	20-16

20.2.2	Data transfer via DXF (V12)	20-18
20.2.3	Data conversion	20-19
20.2.4	Parameter files	20-21
20.2.5	Limits of data conversion	20-48
20.2.6	Subdirectory and files in the directory “design_iges”	20-51
20.2.7	Error messages	20-52
20.3	Interface to External Applications	20-53
21	Data Safety	21-1
21.1	Procedure in Case of Error	21-1
21.1.1	User Errors	21-2
21.1.2	Program Errors	21-5
21.1.3	System Errors	21-6
21.1.4	Error Reports	21-7
21.2	Logging Sessions	21-8
22	Appendix	22-1
22.1	System Data Conventions	22-1
22.1.1	File Names	22-1
22.1.2	Real Numbers	22-1
22.1.3	Lengths and Angles	22-2
22.1.4	Character Strings	22-2
23	Glossary	23-1
	Index	Index-1

1 Introduction

The principal difference between **EUKLID Design** and earlier CAD systems lies in its database structure. It is possible to define and process not only the objects themselves but also their relation to each other via the action. **EUKLID Design** standards may thus be observed and adhered to very simply.

It is possible also to inquire, display, modify and delete design relations, e.g. *perpendicular to* or *parallel to* at any time. The design relations are as easy to change at a later time as the geometry, allowing geometric, design logic and design detail changes to be carried out in the same manner.

Operation of **EUKLID Design** is supported by a highly sophisticated interactive user interface with self-explanatory icons. Many actions also embody where necessary comprehensive online documentation, which may be invoked at any time during interactive working.

Structure of the Manual Series

This user documentation consists of the following volumes:

Exploring

This manual contains an overview of the functionality and the principles of **EUKLID Design** and an introductory example to learn the handle of the basic functions.

Open Architecture (AQL)

This manual contains the description of the programming language which can be used to analyze, generate and modify the contents of **EUKLID Design** models.

Introduction

Target Group and Purpose of the Manual

This user documentation is intended for experienced design staff. No prior knowledge of CAD systems is necessary.

Training courses are also available on various aspects of **EUKLID Design**. More detailed information may be obtained from the *CAD/CAM strässle GmbH* Sales Office.

Summary of Contents

Exploring **EUKLID Design** has the following structure:

- *System Basics*: contains a detailed description of the operating methods of **EUKLID Design**
- *User interface*
- *Operation Basics*: describes how to use **EUKLID Design**
- *Modelling*
- *Introductory Example*: will help to familiarize you with the basic functions of the CAD system.
- The next chapters contain a more detailed description of the functions of **EUKLID Design**.
- *Data safety*: describes the data safety measures included in **EUKLID Design** and provides help if faults occur.
- *Appendix*: contains information about the display of data by the system and font tables.

Typographical and Notational Conventions

The following conventions are used throughout the Users Guide as a means of highlighting important information:

- Inputs and outputs are printed in `Courier` font.
- Keys are enclosed in pointed brackets, e.g. <Tab> key.
- Examples are highlighted by the *Example* heading.



Time-saving tips or exceptions are separated from ordinary text by the mark left from this text.

Descriptions of special cases such as disturbances, error sources etc. are highlighted by the mark left from this text.

2 System Basics

2.1 System Philosophy

EUKLID Design is a design system with capabilities above and beyond those of conventional drafting systems. Like other systems, it works with geometrical objects and actions. Besides geometrical objects, this system makes other design elements available as objects, e.g. basic objects such as lengths, angles, strings or variables. In addition, **EUKLID Design** provides the means to create your own user-defined objects and actions.

The objects can be incorporated into the construction in various ways, according to the requirements to be met and the design objects already available. As in all conventional CAD systems, a line may be defined in the following ways:

- Line between two points
- Horizontal line through a point
- Line of given length from one point at a given angle (polar coordinates)
- Line parallel to a given line at a defined distance
- Tangent from a point to a circle

In contrast to normal CAD systems, the action by which an object is created is stored separately and is an integral part of the data structure.

Relations known as design relations are created as a result of these actions. These relations, such as *perpendicular to* or *parallel to*, are accessible at any point in the design process. Observance of design rules is simple and straightforward. The dependencies can be retrieved, evaluated, updated and deleted at all times.

It is just as easy to update the constructional relations at a later date as to amend the geometry, the result being that **EUKLID Design** will carry out all geometrical and design amendments with the same facility and efficiency.

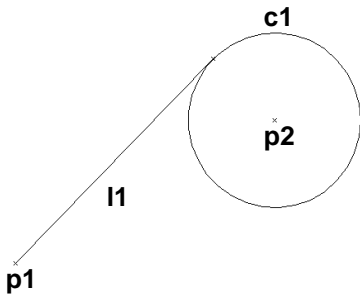
System Basics



As with conventional CAD systems, an object can also be created absolutely (without a create action) by users who program in AQL (see *Open Architecture (AQL)* documentation).

Example 1

A line is to be defined as a tangent from a point $p1$ to a circle $c1$.

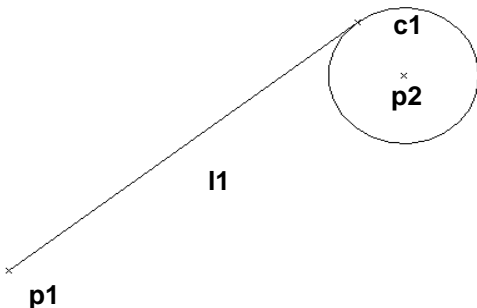


The action *tangent* causes the line to be recorded in the data structure as a tangent with the parameters $p1$ and $c1$



The circle (not the point of contact) is the parameter of the tangent.

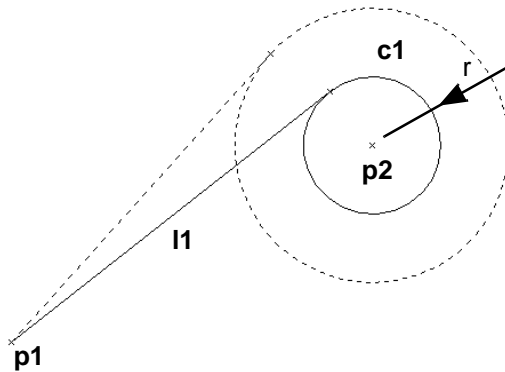
If the position or size of the circle is changed, the tangent as such is not affected. It moves with the circle and remains as before defined as a tangent from a point ($p1$) to a circle ($c1$). This means that all that has changed is one of the two parameters of the tangent; $l1$ remains a tangent.



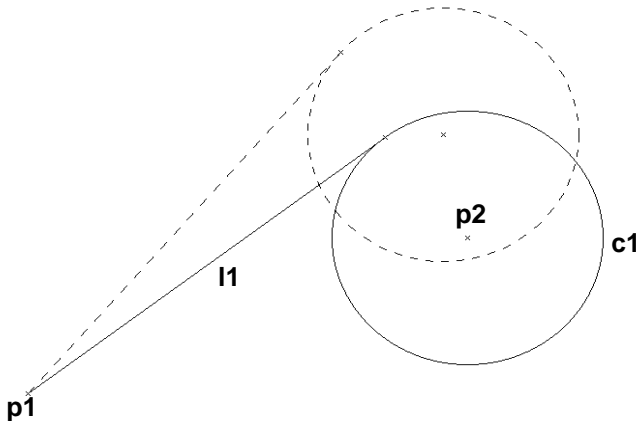
Example 2

The following example illustrates the effects of parameter modification.

When the radius of the circle is changed, the geometry of the circle changes. The center $p2$ and the root of the tangent $p1$ remain unchanged. Only the tangent itself needs to be recalculated.

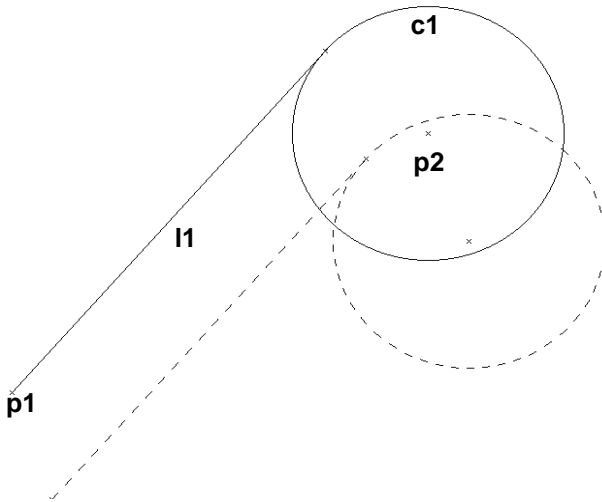


If the center of the circle – point $p2$ – is moved, the radius and therefore the size of the circle remain unchanged. Only the position relative to point $p1$ changes. The tangent $l1$ is now defined by $p1$ and the changed position of $c1$, and is recalculated on this basis.

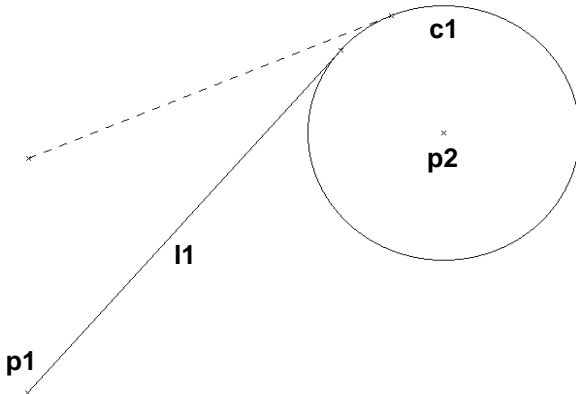


System Basics

If, for example, the position of point $p1$ is amended, while leaving all other values unchanged, this has the effect of displacing bodily the whole construction. Only $p1$ is actually defined relative to the coordinate system; all other objects, however, are governed exclusively by $p1$, thus their position relative to $p1$ remains unchanged and they are displaced with $p1$.



If, however, point $p2$ is defined relative not to point $p1$ but also directly to the coordinate system (this constitutes a change in the constructional relation), $p1$ can be displaced alone without the circle. Only the tangent $l1$ changes.



2.2 The Data Structure

The object-oriented data structure of **EUKLID Design** guarantees a high degree of flexibility. **EUKLID Design** models consist of a group of objects and actions. The relations between objects or between objects and operations may be redefined or dissolved as desired. This makes **EUKLID Design** suitable for those applications which are difficult or even impossible to support with conventional CAD systems.

The procedure for creating a design object is programmed as follows:

- Record object in data structure
- Store object create action separately
- Execute action and thus create relation between the object and its parameters

The data structure consists of the following elements:

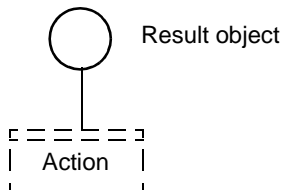
An **object** is an **EUKLID Design** design element. It always has an associated type (e.g. *line*) and can be identified within the data structure through its type and attribute.



Object with properties

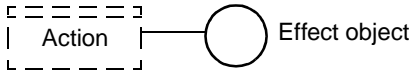


A **result object** is an object created as the result of an action.

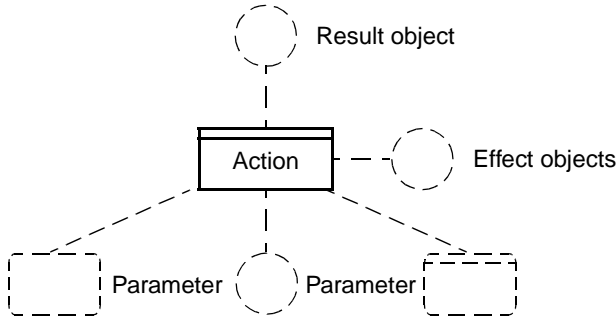


System Basics

An **effect object** is an object created by an action in addition to the result object, e.g. start and end points of a parallel line.



An **action** is an **EUKLID Design** function. It is (mostly) stored in the data structure and recalculated if a parameter is changed.

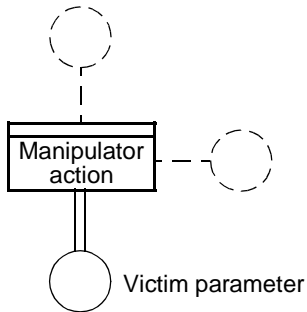


Executed actions are normally stored in the data structure. If a parameter of an action changes, the action is recalculated (evaluated). This may result in a recalculation of the complete model to maintain the consistency of the relations that were formed by these actions. Actions of this type are used to create geometrical objects such as lines.

A **drop action** is an action that is not stored in the data structure. It is executed once only at the request of the user. An example of this is 'plot', which is not initiated automatically every time an object changes.

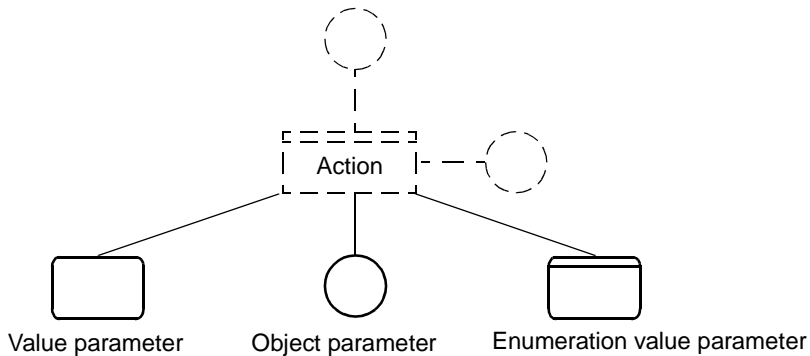


A **manipulator action** changes parameters (victim parameters).

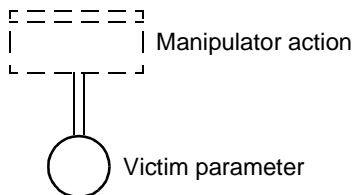


Example: The action *trim two lines at a common intersection* (*trim_2lines*) changes the parameters line 1 and line 2.

Parameters define how actions differ, e.g. geometry and placing of objects.

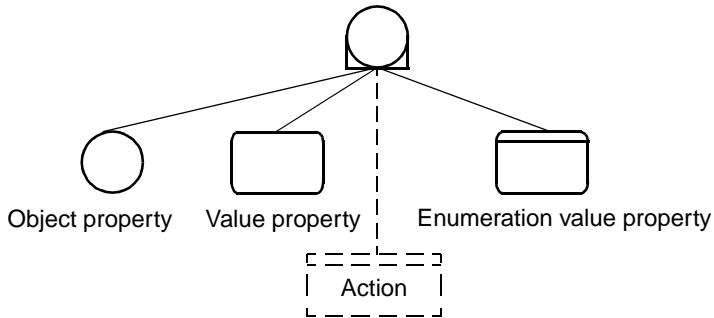


A **victim parameter** is a parameter changed by a manipulator action.



System Basics

Properties describe the attributes of the object that are allocated independently of its creation, e.g. z-value and line pattern. They are therefore associated with the object, not the action.



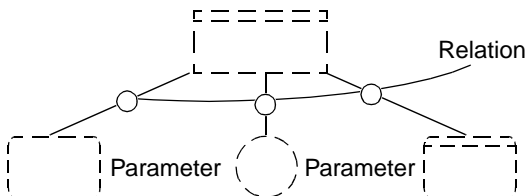
A **value** is an entry by which values are represented. Every object has a value for data representation purposes (object value). It can act as a parameter for actions and relations or can be associated with an object as a property.



An **enumeration value** is a value in a specification (e.g. circle with/without axes).

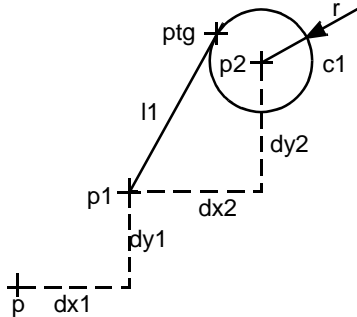


A **relation** arises as a result of the logical operation between the action and your parameters or your created objects.



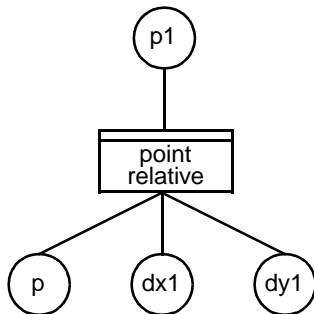
Example 1

The line from example 1 in chapter „*System Philosophy*“ on page 2-1 is defined as the tangent from a point $p1$ to a circle $c1$.



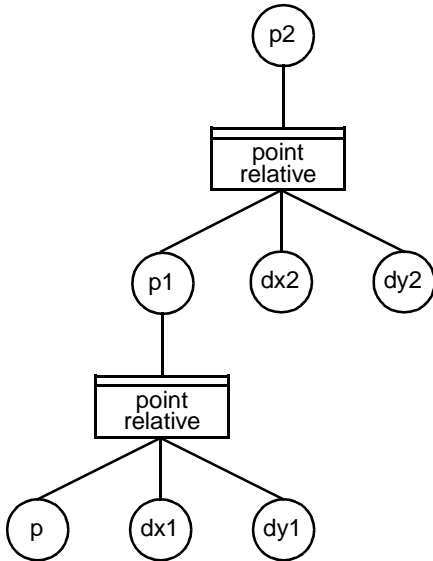
$p1$ is defined as a point relative to a point. The parameters of point $p1$ are:

- The origin of the coordinate system p as reference point
- The coordinates proper $dx1$ and $dy1$

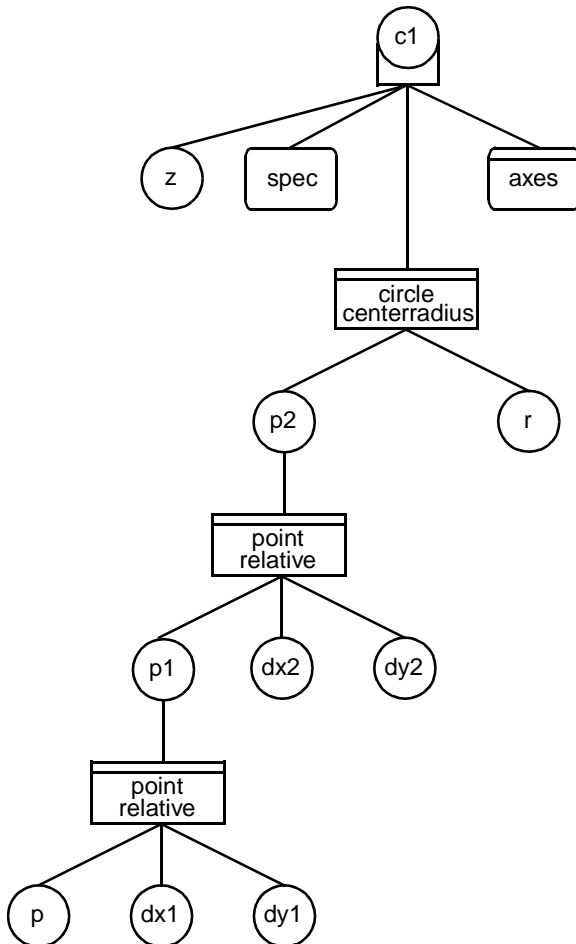


System Basics

The point $p2$ is constructed in a similar manner, but refers not to the origin of the coordinate system as reference point, but rather to the previously defined point $p1$. The resulting data structure is shown in the figure below.

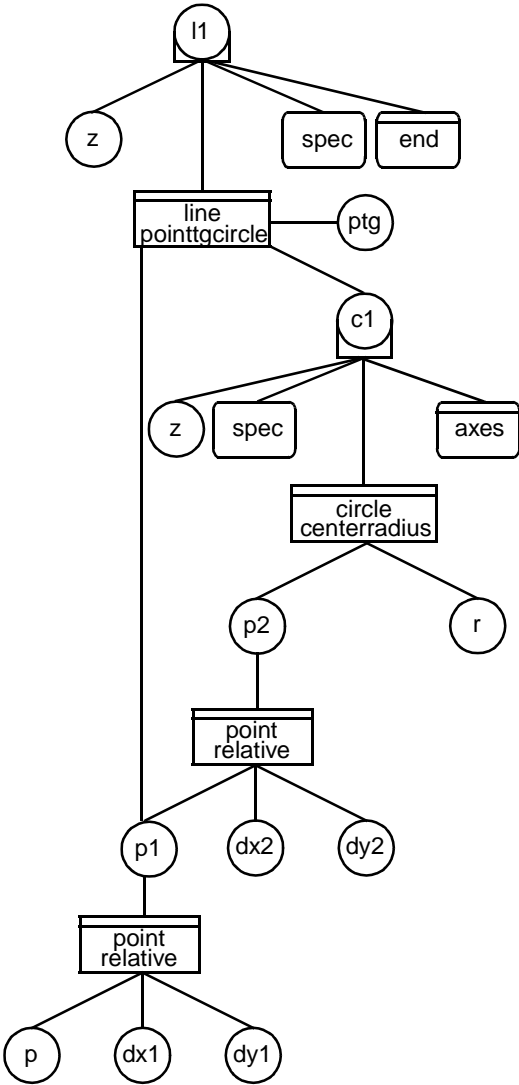


The subsequent development of the data structure on creating the circle is illustrated in the figure below. The circle $c1$ has point $p2$ at its center (first parameter) and the sketched radius r .



System Basics

The tangent *l1* is fully defined by the two objects point *p1* and circle *c1*. The internal definition of the tangent now only requires creation of the two relations to *p1* and *c1*.

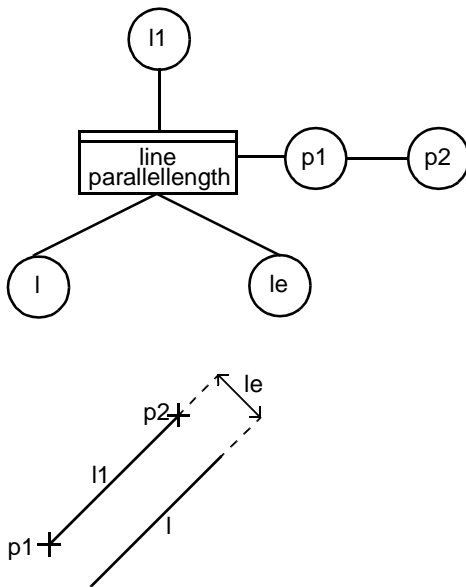


2.2.1 Effect Objects

EUKLID Design optimizes the size of the data structure and improves information quality by creating effect objects at the same time as result objects.

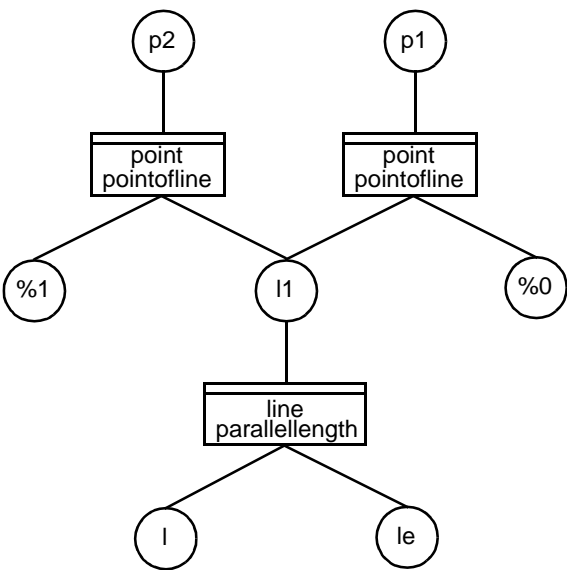
Example

When a parallel line *l1* is created, the start and end points of the line are created as effect objects.



System Basics

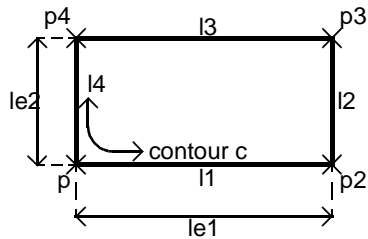
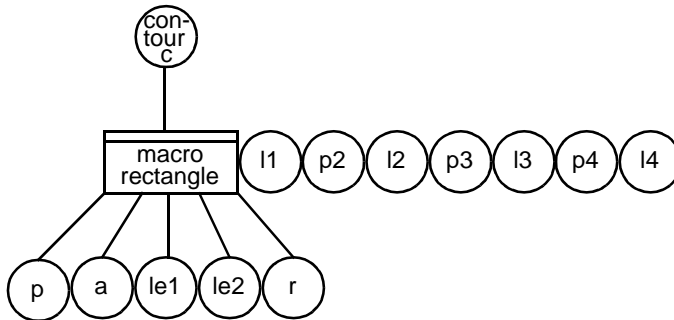
The explicit creation by the user of start and end points as points on a line would also require the length to be entered. The result would be a much larger data structure.



A further advantage of effect objects is direct access to the creating action without having to enter the data structure.

Example

When a rectangle is created, the corner points and lines are created as effect objects.

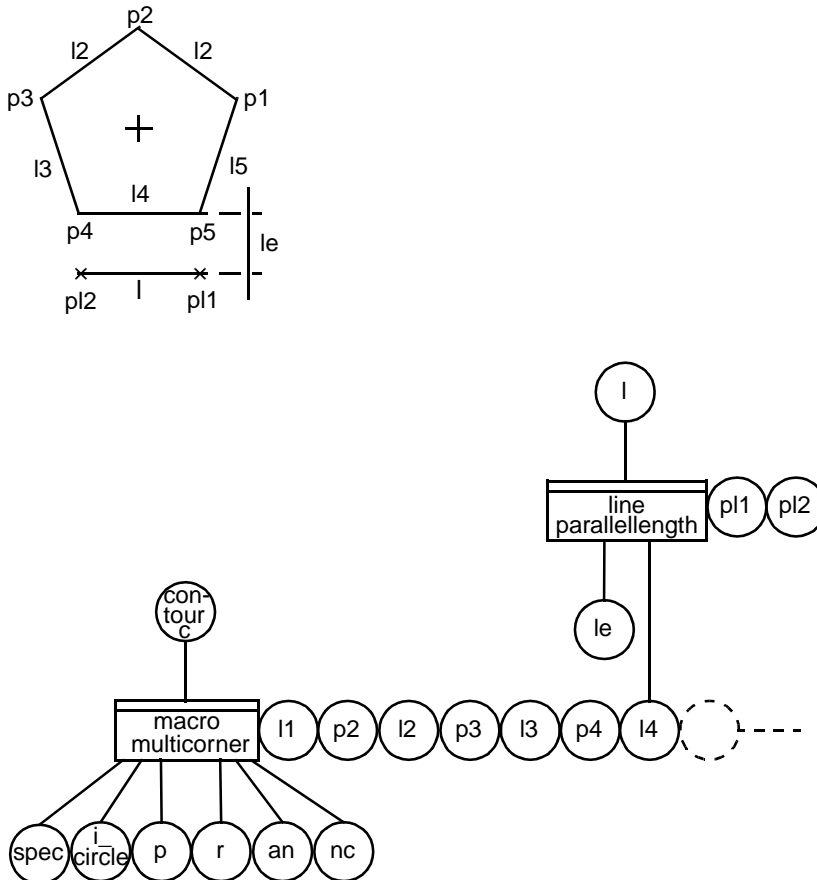


System Basics

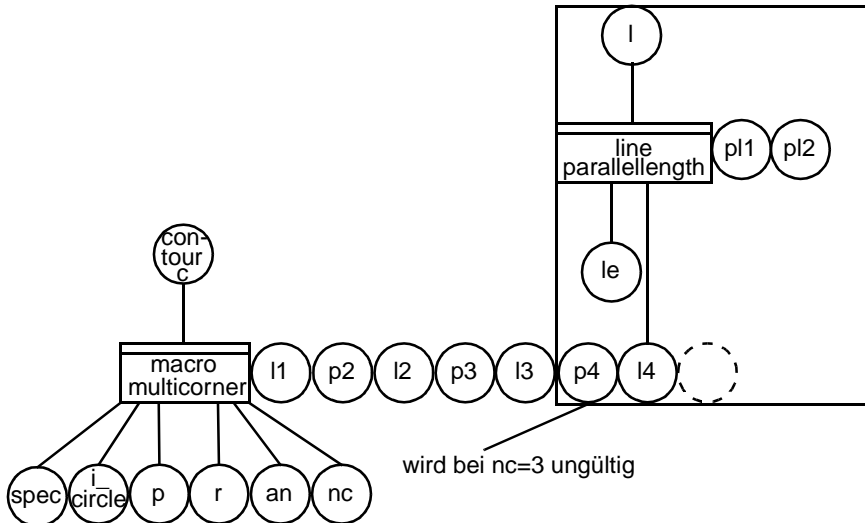
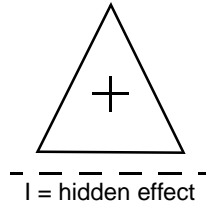
Effect objects, the number of which are reduced by a parameter associated with the action, become inactive, as do all their related elements (hidden effect), but remain in the data structure.

Example

When creating a polygon, the parameter *nc* defines the number of corners and thus the number of effect objects (missing corner points and lines).



If the parameter *nc* is subsequently edited such that the number of corners and thus the number of effect objects is reduced, the surplus effect objects and their related elements become inactive until the original design becomes valid again as the result of further editing.



System Basics

2.3 The User Interface

The system philosophy of ***EUKLID Design*** is supported by a sophisticated user interface. Hierarchically structured screen menus with modern icon selection facilities provide the user at a glance with information regarding the status of the system and the design. Considerable emphasis was also placed during the design of the user interface on a high degree of user-friendliness coupled with acceptable response times.

For each action, the user can call up easy-to-use, hierarchically structured online documentation at the touch of a key. A manual can be dispensed with during the session.

2.3.1 Data Input Philosophy

EUKLID Design differentiates between four main operating modes:

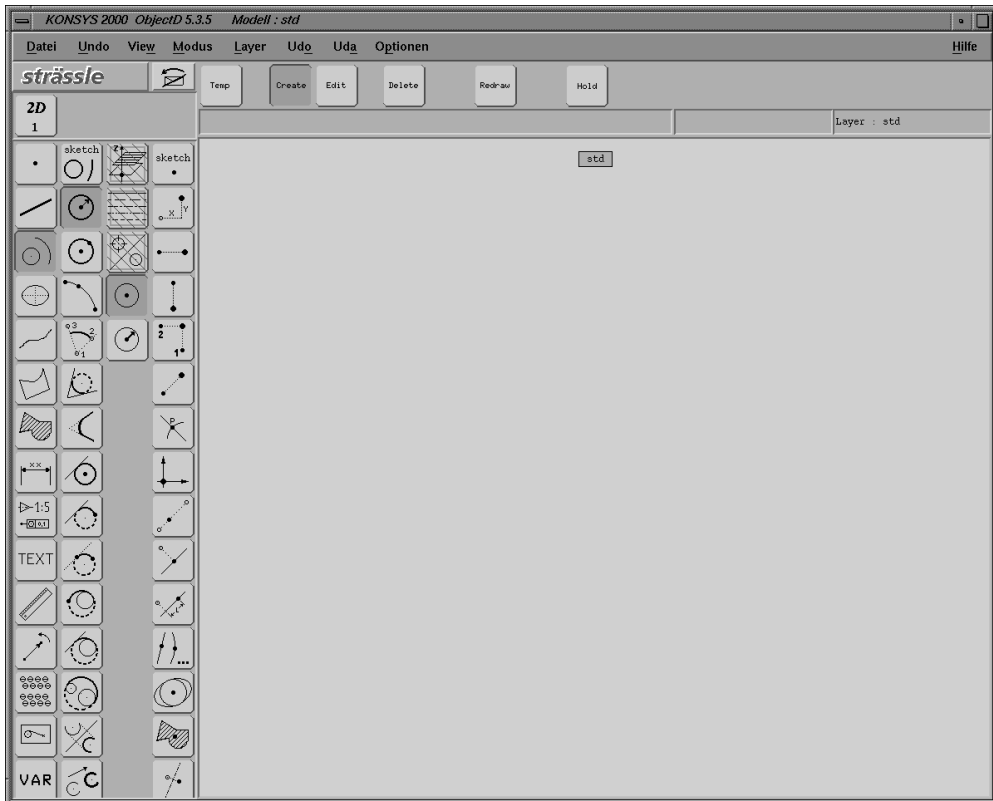
- Create
- Edit/browse
- Delete
- Temp

Create mode is used to create objects and relations using actions.

Edit mode is used to update action parameters and to redefine the existing design relations between geometric objects. This mode also provides information regarding the structure of the design.

Delete mode is used to remove selected objects from the design.

Temp mode covers a number of different groups of actions. It interrupts all other operating modes. After leaving *Temp* mode, work can be resumed at the point where the previously selected mode was interrupted.



The main input method under **EUKLID Design** is the hierarchically structured menu columns optionally on the left or right side of the screen.

In *Create* mode, the first column contains the action groups of the active menus and in *Edit* and *Delete* modes the types of objects on which the operations are to be performed. The first column is always visible.

The second column lists the actions. In the case of *circle*, for example, all the possible actions with which a circle can be created (or was created) are displayed:

- Circle with center point and radius
- Circle with center point and point on circumference
- Circle or arc through three points

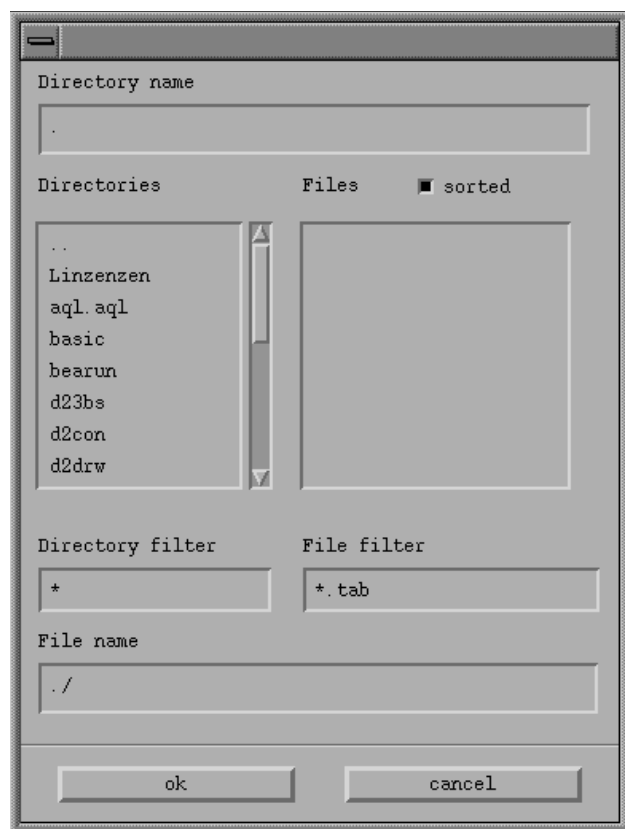
System Basics

The third column is used to enter the parameters of the selected action (second column) for the chosen object. The parameters are requested by the system in the correct sequence. A highlighting facility emphasizes the next parameter to be input.

Parameters may be assigned data in a number of ways, according to requirements and the state of progress of the design:

- it is possible to refer interactively to previously designed objects; these can be selected directly from within the drawing area.
- Precise positions, lengths, angles, text etc. can be input in alphanumeric form in the text field using the keyboard, or can be extracted from the model or read from tables.
- A third input option is the sketching facility, incorporating rubberbanding within the drawing area.
- The icons in the fourth column can be used for implicit design, i.e. subsequent creation. For example, parameter values may be taken from existing objects and coupled with arithmetical operations, such as the radius of a circle
 $K_1 := \text{length of line } L_1 / 2.$

In the case of complex actions, the system displays additional, movable dialog boxes, allowing data input specific to the command group concerned via scrollbars. Examples of functions supported in this way are layering, tables etc.



System Basics

2.3.2 Input Errors

Because of the object-oriented user interface of **EUKLID Design** the system simply will not accept the input of incorrect or incomplete parameter sequences. The user can only select those icons which are actually visible and valid.

The user has also the option of discontinuing at any point an action already invoked, simply by selecting a different icon. **EUKLID Design** will then react immediately – whatever the current system status – and will continue operation with the new action.

With this facility, for example, incorrect values may be corrected before concluding parameter input. Another example is interrupting object creation to change the action while still using the same object. It is also possible to change objects (e.g. from circle to line) or status, e.g. from *Create* to *Edit* mode, at any time.

2.3.3 Action Digression

EUKLID Design offers the option of digressing at any point from a current action to a different one, afterwards resuming the original action from the point at which it was interrupted.

Principal applications of this digression facility are twofold: firstly, to enlarge or reduce (i.e. zoom) or displace (pan) sections of a larger design, for example to facilitate element identification, and secondly, to define implicitly, with the aid of column four of the menu, parameters (including design elements) that are required for the current action but which do not yet exist.

2.3.4 Model Inquiry and Modification

It is possible, by using *Edit* mode, to display the entire design history of a model at any point on the screen, which users define themselves by selecting an object. The selected object is then highlighted on the screen.

The original input menu with its four columns serves here as an information system. Highlighting is used to display both the action creating the object and those of its parameters specified by the user.

If a parameter is itself an object, the action of this parameter may be displayed by descending through the data structure.

It is possible in this way to evaluate model inquiries quickly and efficiently. The original input menu now serves as the output medium containing precisely that information with which the user is familiar to permit the design to be read and assessed at a glance.

Furthermore, by simply touching the object when in *Edit* or *Delete* mode, it is possible to see from the symbol that appears on the cursor how the objects of a certain type were created.

It is possible to use looking at and assessing previously defined data interdependencies and geometrical characteristics for modifying models. The menus of the *Edit* mode are therefore more than an output medium; they may also be used as an update input menu in the same way as with normal inputs under *Create* mode.

The updating of dimensions or relations is nothing new from the computing point of view. It is, however, carried out at just one previously defined point within the design and may give rise to any number of automatic recalculations, depending on which and how many objects are governed by the one being updated.

2.4 Special Features

EUKLID Design possesses in addition a few other notable features, namely:

- The mutual association between geometry and dimensions
- The hierarchical layer/structure concept
- Standardization of objects and actions
- The AQL programming language
- Freely configurable

2.4.1 Mutual Association between Geometry and Dimensions

Since **EUKLID Design** is object-oriented and permits the definition of any relation between objects, the dimensioning capability of **EUKLID Design** will, as long as the topology remains the same, adapt itself automatically to any changes in the geometry.

Another option offered by **EUKLID Design** is to derive dimension variants from iterating. In other words, the designer is able to sketch the component first and afterwards apply the definitive dimensions. In this case, the system assists the user by highlighting all the base objects which have influence on the start object. The user may then select a base object which should be changed. The data structure is iterated to achieve the specified value of the start object.

This function is available for lengths, angles, proportions and real numbers. This means, for instance, that the moment of inertia of a surface can be set to a predefined value instead of a particular value.

2.4.2 Hierarchical Layer/Structure Concept

A simple and straightforward method of structuring the design is by layer hierarchy. It is possible, for example, to assign components, assemblies, detail enlargements or just the dimensions to separate layers; these may then be set to active, selectable or invisible. These statuses may be different for each view-object.

The various layers may be so defined that parts of the design which belong together logically are assigned to the same layer and can be accessed en bloc without affecting

the rest of the design. When a design is fully organized on this system, a hierarchical tree structure of layers results. This structure can also be shown separately and used for identification purposes.

The hierarchical layer facility makes the deletion of components or details a simple and elegant procedure. If a layer is deleted, all layers subordinate to it are implicitly also deleted.

It is also possible to “bequeath” to layers particular properties from user-defined objects. It is thus possible, for example, to define a layer as a “part” and to allocate all the relevant predefined properties of that “part” to this layer (e.g. price, stock location, part number).

Layers can also be used to allow several designers to work on a model at the same time. The chief designer can thus divide his project into sub-projects that can be worked on by the individual detail designers, yet still control the necessary relations through the sub-projects.

2.4.3 Standardization

EUKLID Design provides the following features for the standardization that is becoming an increasingly important factor in design:

2.4.3.1 User-definable Objects (UDO)

User-definable objects are used to define frequently used standards, such as standard parts and factory standards, in the system.

This allows users to define their own objects, such as screws, pins etc. These can have freely definable properties (e.g. diameter, length) and are recognized in exactly the same way as system-defined objects (e.g. lines).

Instances of these objects, with any properties and in any quantities, can then be introduced into the model.

System Basics

Such objects can also inherit the properties of other already defined objects both statically, when they are defined, and dynamically in each individual instance.

Any number of display formats can be defined for each object; a different format can then be selected each time the object is viewed. This means, for example, that a screw could perhaps only be shown as a cross within a viewport.

2.4.3.2 User-definable Actions (UDA)

User-definable actions enable ***EUKLID Design*** to be extended to include frequently-required functions. This provides, on the one hand, the facilities already available in the ***EUKLID Design*** dialog and on the other, through AQL, the full scope of a modern programming language. Many actions in ***EUKLID Design*** are already implemented as UDAs by the manufacturer, which serves to underline the power of this tool.

UDA parameters can be specified as required and their features defined. If, for example, the user has no create type for line, this can, assuming it has an auxiliary construction, be linked into the system. No programming knowledge is necessary. If no possible auxiliary construction is available or if the desired function is of a different nature (e.g. special zoom), this can be achieved by integrating an AQL module.

External programs or output from such programs can also be integrated easily into ***EUKLID Design***.

User-defined actions can be linked into the user interface in such a way that it is not possible to distinguish them from those defined by the system. This also applies to the availability in AQL.

2.4.3.3 User-definable Data Tables (UDT)

Tables are used to store data (numbers, character strings) which can then be accessed from the model. These tables can also be retrieved from databases. Access takes place in a similar way to that used in databases. For example, the rule defining how a screw length is calculated can be stored in the model, and the most suitable screw can then be retrieved from the table.

2.4.4 Programming Language AQL

The **EUKLID Design** programming language AQL provides programmer access to the system. AQL may be used to inquire a **EUKLID Design** model and as a command language. Possible applications are:

- Extending the functionality in AQL of integrated UDAs
- Any specific user extracts
- Complex model generation using program logic
- Interface to calculation routines
- Production planning and control facilities
- Interface to external programs

Further information may be obtained from Volume *Open Architecture (AQL)*.

2.4.5 Free Configurability

All system properties, such as menus, menu hierarchy and standard values, can be defined by the user and are stored as follows according to type:

- at the system configuration level (e.g. standard menu for drawing frame)
- at the individual user level (e.g. colors, language)
- at the individual model level (e.g. views, inch/mm setting)
- at the session level (e.g. cursor properties)

All user definable features, such as objects and actions, can be integrated into the user interface without “break”.

It is thus possible to present the end user with the most comfortable, efficient and ergonomically effective user interface for each application.

2.5 Complex Applications

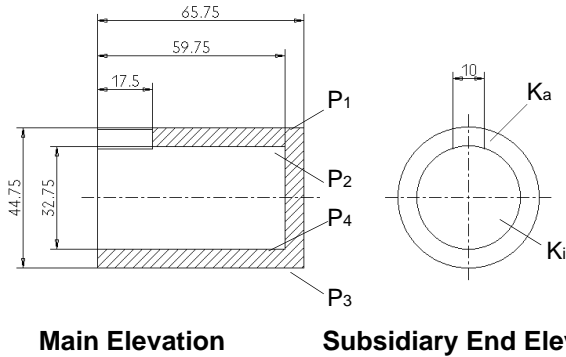
The scope of **EUKLID Design** is many-faceted and in no way restricted to 2D-CAD work. Thanks to its completely new data structure philosophy, the range of applications supported by **EUKLID Design**, working as a design system truly worthy of the name, go far beyond what can be expected of today's conventional 2D-CAD systems. The spectrum includes even facilities and methodologies which up to now were the exclusive domain of 3D-systems.

Below is a brief summary of particularly important application areas, exemplifying the technical solutions supported by **EUKLID Design**, plus details of the auxiliary facilities:

- Multiple elevation design
- Design variants
- Detail design
- Kinematic simulation, assembly checks and snag tests
- Functional contexts
- Constructional calculations
- Standardization of designs

2.5.1 Multiple Elevation Design

The design relations in the data structure of **EUKLID Design** make possible the generation of multiple elevations on the computer. The wide range of actions allows variants to be derived from a main elevation and any number of associated subsidiary elevations or sectional views without making changes to the dependent views and sections. It is sufficient to modify just the main elevation.



In the previous example, the end elevation is defined in relation to the main elevation. The following procedure was adopted:

- The diameter of the circle C_i in the end elevation is governed by the distance between the points P_2 and P_4 in the main elevation.
- The outside diameter C_o in the end elevation is identical to the length of the line between points P_1 and P_3 in the main elevation.

If any dimensions are changed in the main elevation, the objects of the subsidiary end elevation are automatically amended accordingly. Data consistency is thus guaranteed in every case.



To incorporate the groove in the sleeve correctly, the reverse procedure is required. The display on the right becomes the main elevation and the one on the left the dependant one. The relations are therefore not fixed.

System Basics

2.5.2 Design Variants

The relational data structure permits the creation of any design variant, provided it does not cause any complex topology changes.

Knowledge of the constructional relations means that any geometrical or arithmetical values, plus texts, may be changed without affecting the design logic.

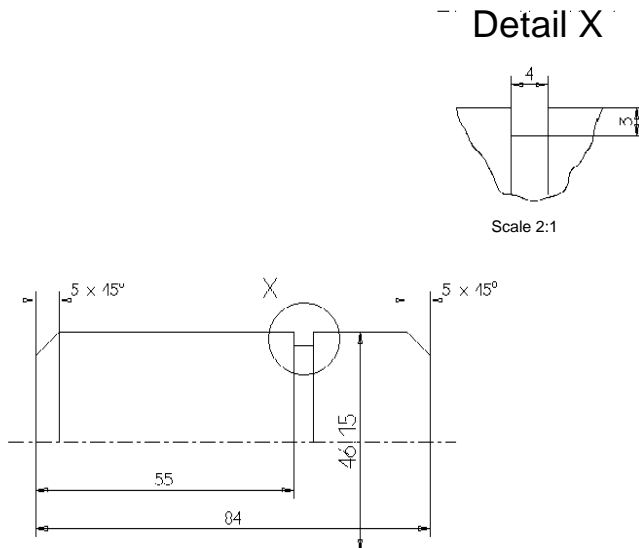
Design variants are thus only one aspect of relational design.

The **tables function** is an important tool in the construction of design variants. Similar to a data base, it permits the storage and retrieval of parameter values which are frequently used in designs.

2.5.3 Detail Design

The constructional relations of **EUKLID Design** facilitate the display of constructional details without extra effort. A local coordinate system, to a different scale if desired, is used to extract and display enlargements of any parts of the design, for example to apply more detailed or special dimensions.

2



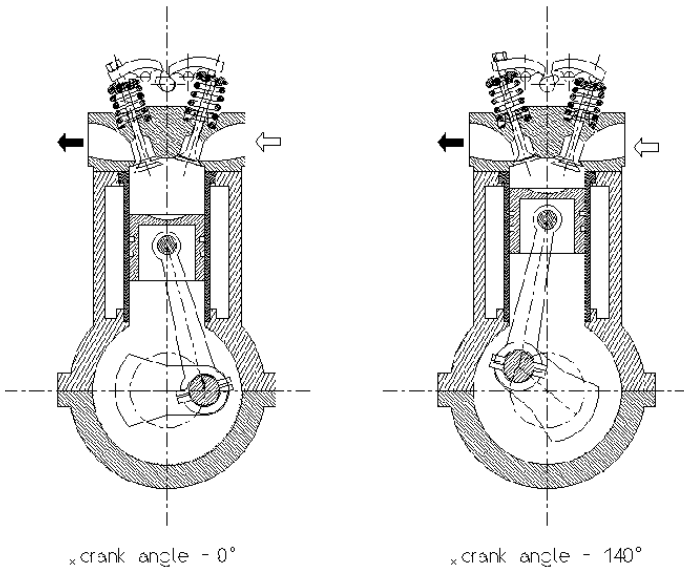
The detail design facility is useful in combination with the **layer/view facility**, so that only relevant details are displayed.

The details thus derived are also automatically updated whenever the main design is changed.

System Basics

2.5.4 Kinematic Simulation, Assembly Checks and Snag Tests

The capabilities of **EUKLID Design** make kinematic simulation and optical snag tests, which are usually typified by the interdependence of complex geometries, an extremely simple and elegant procedure to carry out. With the **variables facility** it is usually sufficient to change a few parameters in order to alter, simulate or assess the movements or discrete specific positions of a selected component. The example given below, of an internal combustion engine, illustrates the principle best.



As the crankshaft is rotated, the positions of the connecting rod and piston change accordingly. The rotation of the camshaft affects in its turn, via the cams, the movement of the rocker arms, causing the valve springs to flex and the valves to open and close.

It is sufficient to input one parameter, in this case the angle of the crankshaft, to simulate and observe the internal movements of the entire engine, thanks to the interdependencies of the data structure. Where more complex simulations are desired, the AQL programming language can be used.

2.5.5 Functional Contexts

Every component, as a rule, has a certain relationship to its environment by means of which it fulfils its function. For this reason, when a component is incorporated into the design, the other components around and near it are inevitably affected. For example, a screw requires in one component a threaded hole and in another a clearance hole, in order to achieve its purpose of "forming a mechanical bond". This type of relation is known as the functional context of the design.

EUKLID Design defines unambiguous relations between components and assemblies, guaranteeing the consistency of the design at all times. Functional relationships may be defined as a **UDA/UDO**, composed of the UDAs/UDOs for the various components and subassemblies. The design relations defined in the data structures cause any modification to a given component to be reflected automatically in the design of the components related to it.

3 Introduction to Interactive Working

3.1 Hardware and Operating System

The files created with ***EUKLID Design*** are hardware platform independent. It is thus possible to work with ***EUKLID Design*** in a heterogeneous network. The approved hardware platforms and operating system versions are listed in the release notes.

You can only use your version of ***EUKLID Design*** on the hardware specified platforms, e.g. you can run an SGI version only on an SGI workstation.

In a heterogeneous network it is essential for path and file names to be defined in accordance with the conventions of all the installed operating systems.

No problems are caused as a result of the different conventions for path separators ("/" and "\"). Please do **not use special characters in path and file names**, such as \$, %, blanks and umlauts! The first character should be a letter.

Shared files must be available in directories which are accessible to all the installed hardware platforms. Sufficient access rights are to be guaranteed. The network should be shaped transparently by corresponding mounts.

Introduction to Interactive Working

3.2 Starting *EUKLID Design*

Depending on the hardware platform you have different possibilities to start ***EUKLID Design***: on Windows NT for example via the start menu of the task bar, on the UNIX platforms for example via the start script *design* in a shell.

3.2.1 Start options

In the following the start options are explained for the UNIX version. You may add them also to the corresponding shortcut on Windows NT.

design <Version> [Options] [<Model name> ...]

Call parameter	Meaning
design	If you start without a model name, a model with the name <i>std</i> is processed. No model file is read or created, unless you had stored a model with the configuration. In this case the model of the configuration will be used (see „ Configuring the System “ on page 17-1).
<Version>	The version you want to start, e.g. <i>v570</i> .
Options	Strings <u>preceded by</u> <code>"-</code>
<Model name> ..	Strings <u>not preceded by</u> <code>"-</code> . You can specify one or several models, i.e. any string not preceded by <code>"-</code> , which doesn't belong to any start option, is interpreted as model. The last model which was read will be active.



EUKLID Design assigns the suffix *.mod* to all models on principle. Therefore the suffix may be omitted when a model is specified.



The length of the file name is limited by the operating system. You must take this into consideration when you exchange data between different operating systems.

The following options can be used on all platforms:

-access <AQL program>

This option calls the specified AQL program before and after every store or load operation, and whenever a new model is started (please refer to the enclosed example in the installation directory: `/#/aql.aql/file_aql.aql`).

The specified AQL program is searched in the following directories (in this order):

- working directory
- \$HOME/design_config
- `/#/aql.aql`

-batch

EUKLID Design runs in batch mode. The user interface is not started. The model is not stored when the program is terminated.



This option can be combined with the option `-start` or `-stop`.

-cleanup

This option is used to remove those objects that cannot be evaluated (due to an error) when the model is loaded, e.g. invalid objects.

-config_dir <Path>

This option specifies the directory of the system configuration. By default the directory `/#/design_config` is used.

Example:

<i>Start option</i>	<i>specified directory</i>
<code>-config_dir #/general_config</code>	<code>h:\general_config\design_config</code>

-config_dir_user <Pfad>

This option specifies the directory of the user configuration. By default the directory `~/design_config` is used.

Example:

<i>Start option</i>	<i>specified directory</i>
<code>-config_dir_user ~/work</code>	<code>h:\usr\meier\work\design_config</code>

Introduction to Interactive Working

-display <Server>:<Screen>

This option specifies that the user interface should run on another workstation (or X-terminal). In normal cases, the <Screen> option should be zero. TCP/IP must be running for both workstations and the target workstation must have enabled this service ("*xhost +*" command).

-geometry <width>x<height>+<xoff>+<yoff>

This option specifies the size and position of the *EUKLID Design* window in pixels.

xoff specifies the offset of the left edge of the window to the left side of the display.

-xoff specifies the offset of the right edge of the window to the right edge of the display.

yoff specifies the offset of the top of the window to the top of the display. *-yoff* specifies the offset of the bottom of the window to the bottom of the display.

Example: design -geometry 600x600-10+350
 (The minus offsets may not be supported on all X-servers.)

-help or -h

Explaining start options.

-iconic

Start as icon.

-language <Name>

This option determines the language of the user interface. Please select a language name in the user interface (see chapter Configuring the System „[Setting the Dialog Language](#)“ on page 17-12) and enter this option.

-protocol <File>

If this option is specified, a session protocol file will be generated. This protocol file has the form of an AQL program.

-read_only <Model name>

The model will be loaded for *read only* access. You can still load external sublayers of that model for read/write access.

-scheme <Name>

This option determines the colors of the user interface. Please select the name of a color scheme in the user interface (see chapter Configuring the System „[Setting the Colors](#)“ on page 17-12) and enter this option.

-start <AQL program>

This option will call an AQL program just after all model files have been loaded and before the program becomes active.

-stop <AQL program>

This option will call an AQL program just before the session is terminated.

-sync

The option -sync will switch the X-window system to synchronous mode. The program will operate much slower but the display will be updated immediately after each modification. This option should only be used for debugging purposes.

-verbose

Additional information, e.g. on loading UDO and UDA, are output in the start-window.

-writeback

This option sets the batch mode active (without user interface), starts **EUKLID Design**, loads and rewrites the specified models and terminates the program.



This start option can be used to convert a model to a new format (version converting if necessary).

Introduction to Interactive Working



Window size

When ***EUKLID Design*** is started it uses the whole area of the screen or comes up with the size and position of the last session.



Start with model

You can specify a model name at the program start. If this is the name of an existing model, it is displayed in the drawing area. If no model with this name exists, the message "*File <model_name> is no Bea model, new model is initialized*" is displayed in the start-window and a new, empty model with the specified name is generated.



Reduced windows

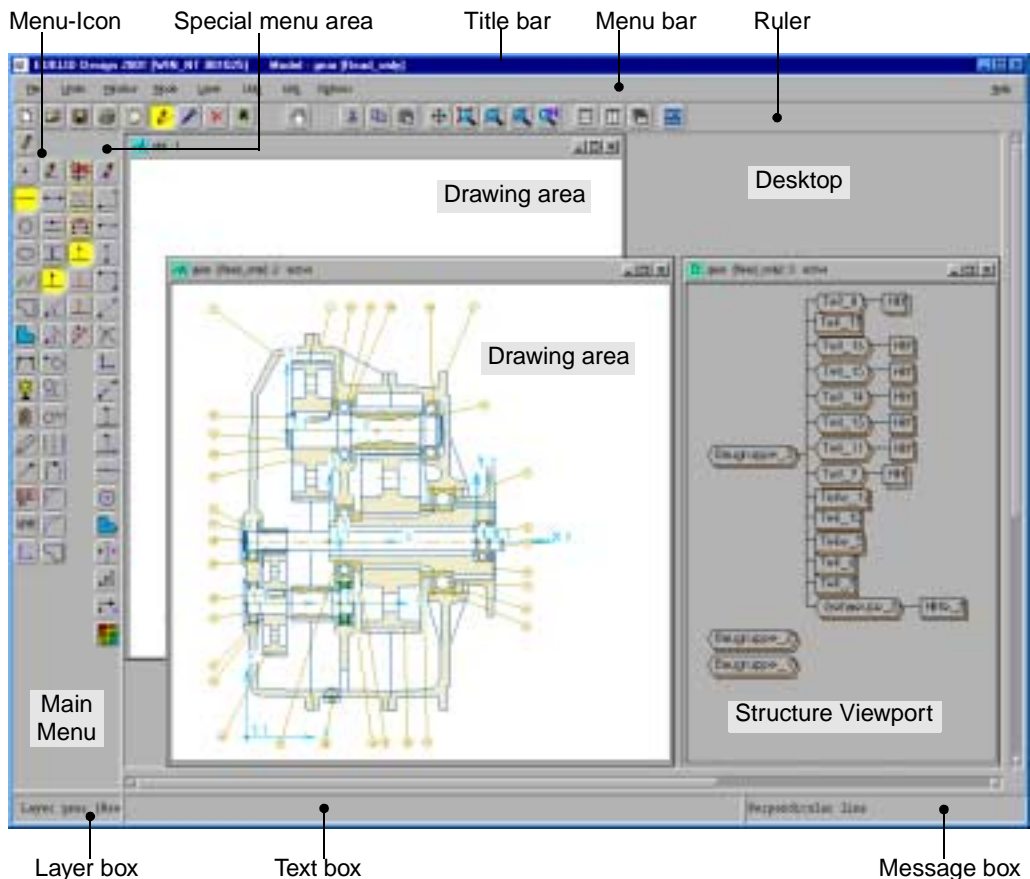
Please note that full operability can no longer be guaranteed if windows are greatly reduced in size (for example icons may overlap or popup menus can no longer be accessed). In such instances you need only increase the size of the window again.

3.3 Basic Elements

This chapter describes usage and behavior of the user interface.

3.3.1 The Screen Areas

The user interface consists of various screen areas. On the time of delivery, the default setting is as follows:



Introduction to Interactive Working

3.3.1.1 The Desktop

In the desktop, you can open up to 10 different model viewports and position them freely. You can determine which model should be displayed in which viewport.

Iconized viewports will be positioned in the upper left corner of the desktop. For further information see chapter „[Windows](#)“ on page 14-1.




3.3.1.2 The Drawing Area

In the drawing area of the model viewport, the model is displayed, positions are specified and drawing elements are identified as parameters.

You can open a popup menu via the right mouse button. Depending on the system status, the menu includes different menu commands (see section „[Popup Menus](#)“ on page 3-26).

3.3.1.3 The Main Menu (vertical menu bar)

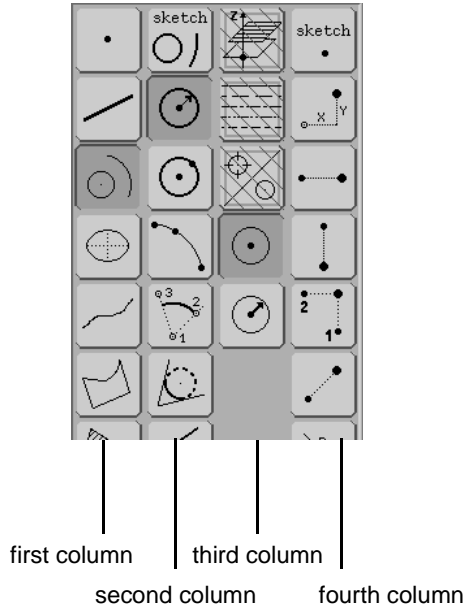
The main menu consists of four menu columns. The content and meaning of the main menu depends upon the current mode of the program:

<i>Icon</i>	<i>Mode</i>	<i>Meaning</i>
	Create	Create objects
	Edit	Inform about objects Change objects
	Delete	Delete objects

The <↑> and <↓> keys allow you to scroll through the icons, provided all icons don't fit on the screen.

Introduction to Interactive Working

The menu columns contain objects, action groups, available actions and their parameters.



Create-Mode

Column	Type	Meaning
First column	Action group	What do you want to create?
Second column	Actions	How do you want to create it?
Third column	Parameter	Specifications needed
Fourth column	Value	Selection of predetermined values
	Action (recursion)	Subsequent creation of parameter objects

Introduction to Interactive Working

Edit-Mode

Column	Type	Meaning
First column	Object type	Which object do you want to change?
Second column	Action	Info: How was the object generated?
		Change: Do you want to redefine an object? (Popup Menu)
Third column	Parameter	Info: About values and text input
		Change: - Values and text input - Relations to objects
Fourth column	Action	Info: Creation type of the parameter object
		Change: Do you want to change the relation to a new, non existing object?

Delete-Mode

Column	Type	Meaning
First column	Object type	Which object do you want to delete?

Menus

In the Create-Mode, the actions of **EUKLID Design** are combined to action-groups and menus. You can switch to any menu you like.

You can switch to another menu in this way:

- Please click on the menu-icon with the left mouse-button. Each mouse-click activates another menu. After the last menu, the program returns to the first menu. Clicking with the middle mouse-button, you can go backwards through the menus.
- Please press the right mouse-button over the menu-icon. In the subsequent popup menu, you select with the entry *Go to* the menu, in which you want to change.

You may define your own menus and action groups and integrate them into the user interface. Please refer for further information to the section „[Configuration of the Main Menu](#)“ on page 17-14.

Introduction to Interactive Working

Action Groups

Actions are combined to meaningful units, the action-groups. For example you find in the action-group *Lines* (at the time of delivery) all actions that generate an object *Line*.

The action-groups of the active menu are shown in the first column.

The actions of the active action-group are shown in the second column.

You can adapt and modify the action-groups when required. For further information please refer to the section „[Action Group Menus](#)“ on page 17-15.

Object Groups

During the selection-phase of the Edit- and Delete-Mode, the first menu-column shows the available object-types. In **EUKLID Design**, the object-types were combined in two object-groups. You may generate also object-groups of your own.



Object-groups are described in more detail in the section „[Object Group Menus](#)“ on page 17-22.

3.3.1.4 The Ruler



<i>Icon</i>	<i>Meaning</i>
	New model
	Open model
	Save model
	Print
	Temp-Mode
	Create-Mode Create objects
	Edit-Mode Show / Change objects
	Delete-Mode Delete objects
	Redraw
	STOP-Icon Stop the running action
	Hold

Introduction to Interactive Working

Icon	Meaning (Forts.)
	Cut objects
	Copy objects
	Paste objects
	Pan
	Zoom all
	Zoom in
	Zoom out
	Zoom undo
	Arrange viewports vertical
	Arrange viewports horizontal
	Arrange viewports overlapping
	Informations about EUKLID Design and CCS Informationssysteme

You can switch on and off the ruler via the Options-Menu, *Window layout* → *Ruler*.

3.3.1.5 The Menu Extension Area

This area is used to output information on the creation type of an object in the Edit-Mode, show icons with arrows to go through the data structure etc.

3.3.1.6 The Menu-Icon

You can change from one menu to another by clicking with the left mouse button. Another menu is activated with each mouse-click. After the last menu, the program returns to the first menu. Clicking with the middle mouse-button, you can go backwards through the menus.

3.3.1.7 The Text Box

The text box is used to output data (prompt texts) and to input text and numbers.

3.3.1.8 The Layer Box (Output Area for the active Layer)

In this box, the name of the active layer is displayed. If the cursor is located over this box you can display the *Layer Status* dialog box via clicking the right mouse button.

3.3.1.9 The Message Box

In this box, the program shows logged values for parameter sketching functions, e.g. for sketching the radius of a circle.

Introduction to Interactive Working

3.3.1.10 The Title Bar

The windows title bar includes the following informations:

Product name, version number, Computer platform, creation date and the name of the active model.

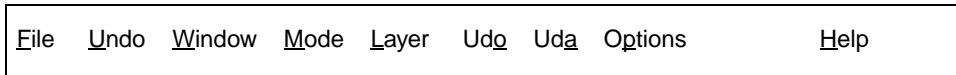
Example: EUKLID Design 2001 [WIN_NT 000924] Model: std

3.3.1.11 The Menu Bar

This section describes the pull-down menus and their actions. The menus of the menu bar as well as some actions can simply be selected by key combinations (shortcuts).

The menus of the menu bars are called by pressing the <ALT> key and the letter underlined in the menu; e.g. press <ALT>+F to call the File menu.

If such a shortcut is available, the key combination is shown on the right side of the menu entry.



The File-Menu

The *File*-Menu contains the actions that are needed to create, load and save models.

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
New model ... Open model ... Reopen model ... Add model ... Close model ... Close all ...	Ctrl+n Ctrl+o	„File Operations“ on page 5-39
Print		see Online-Help
Admin Waste-basket	Shift+F10 Shift+F11	see Online-Help
Save model ... Save model as ... Save configuration	Ctrl+s	„File Operations“ on page 5-39 „Saving the Current Configuration“ on page 17-7
Update location table	Shift+F12	„File search rule (location table)“ on page 17-25
Set standard library		„The Standard Library “+”“ on page 17-28
Recently used files ...		„Open models directly“ on page 5-51
<... filename ...>		List with recently used model files
Exit ...	Ctrl+x	„Finishing a Session“ on page 3-41

The Undo-Menu

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
<u>U</u> ndo	Ctrl+u	„Undoing Delete Actions“ on page 9-10
Edit <u>l</u> ast object	Ctrl+l	Edit-Mode with last created object

Introduction to Interactive Working

The Window-Menu

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
<u>R</u> edraw	Ctrl+r	Updating the drawing area
<u>R</u> efresh	Ctrl+f	Refresh the user interface
New window Horizontal Vertical Standard Overlapping		„Windows“ on page 14-1
<... model name ...>		List with actually opened models. The active model is marked.

The Mode-Menu

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
<u>C</u> reate	Ctrl+c	Create objects
<u>E</u> dit	Ctrl+e	Show / Change objects
<u>D</u> elete	Ctrl+d	Delete objects
<u>T</u> emp	Ctrl+t	see below
<u>H</u> old	Ctrl+h	
<u>B</u> rowse	Ctrl+b	
<u>Q</u> uick edit	Ctrl+q	

Temp

The Temp-Mode interrupts the current action to activate an action from the Temp-Menu. These actions do not create any data structure objects (so called "Drop" actions), like *zoom*, *pan*,

You can disable the Temp-Mode via clicking it a second time. The program returns to the interrupted action.

Hold

As long as the Hold-Mode is activated, a re-evaluation of actions and dependent objects is prevented. You can edit several object-parameters simultaneously for example.

By another clicking this Menu-entry or the Temp-Icon, the Hold-Mode is deactivated and a possible evaluation is started.



The Hold-Mode can be activated only during the Edit-Mode.

Browse

The Browse-Mode is used to traverse the data structure during selection (e.g. at the setting of parameters or properties of UDAs or UDOs). It is activated automatically.

You can leave the Browse-Mode in this way:

- Via clicking the *Browse-Icon*,
- selecting the *Browse-Option* in the Mode-Menu,
- <CTRL>b, or
- pressing <Esc>.

Quick edit

The Quick edit mode is described in the section „*Quick Edit of the Object Closest to the Cursor Position*“ on page 8-28.

Introduction to Interactive Working

The Layer-Menu

This menu is used to create and change layers.

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
<u>S</u> tatus	Shift+Ctrl+s	
Save external Load external		
Convert to UDO		
Is a		

For a detailed description of this menu, please refer to section „[The Layer Menu](#)“ on [page 5-19](#).

The Udo-Menu

This menu is used to define user defined object types (UDO).

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
Load Unload		
Init definition Edit definition Abort definition		
Close definition Save definition Save definition as		
Dissolve Convert to Layer		
Is a		

The Uda-Menu

This menu is used to define user defined actions (UDA).

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
Load Unload		
Init definition Edit definition Abort definition		
Close definition Save definition Save definition as		

For a detailed description of these menus, please refer to chapter „[Standardization](#)“ on [page 13-1](#).

Introduction to Interactive Working

The Options-Menu

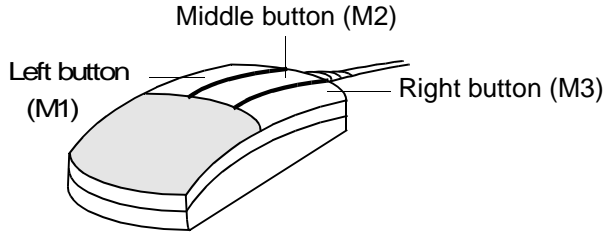
<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
Window Layout ... Icon Properties ... Cursor Icon Execcursor Keypad		„Configuring the User Interface“ on page 17-11 „The Keypad Tool“ on page 3-30
Language Color scheme Default Viewport type Function keys		„Configuring the User Interface“ on page 17-11
Tooltips ... Configure Help / Documentation		see Online-Help

The Help-Menu

<i>Menu entry</i>	<i>Shortcut</i>	<i>Description in section...</i>
Documentation On context		„The Online Documentation“ on page 3-37 „The Online Help“ on page 3-36

3.3.2 The Mouse

A three-button mouse is normally used.



The mouse buttons have the following functions:

<i>Mouse button</i>	<i>Meaning in drawing area</i>	<i>Meaning in Main Menu</i>
M1	Select objects	Select icons
<Shift> + M1	Expand the selection set	Expand the selection set (Edit- and Delete-Mode)
<Ctrl> + M1	Expand or reduce the selection set	Expand or reduce the selection set
M2	<ul style="list-style-type: none">• Switch the selector-Parameter (Edit-Mode)• Create arc while sketching contour or plane• see <Ctrl> + M1	<ul style="list-style-type: none">• Input text into text fields• Menu-Icon: Change the menus backwards
M3	Popup menus	<ul style="list-style-type: none">• Popup menus• Show or hide the keypad tool over active text box








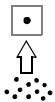
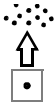

An **entry** is made by pressing the left mouse button.

EUKLID Design commands are composed and executed using sequences of these identified icons, popup menu fields and design elements.






Introduction to Interactive Working

3.3.3 The Cursor

The shape of the cursor changes in accordance with the area in which it is situated or depending on the operating state:

<i>Cursor shape</i>	<i>Meaning</i>
	Select an icon or field in the dialog box; Identify an object in the drawing area.
	Select a function in the popup menu.
	Drawing area: No input possible. Press the right mouse button in order to open a popup menu.
	Multiple selection made
	Object identification in UDO
	Selection in UDO
	Indirect Selection mode
	Selection of objects and their sons
	Selection of objects and their ancestors
	Free sketching position in the sketcher

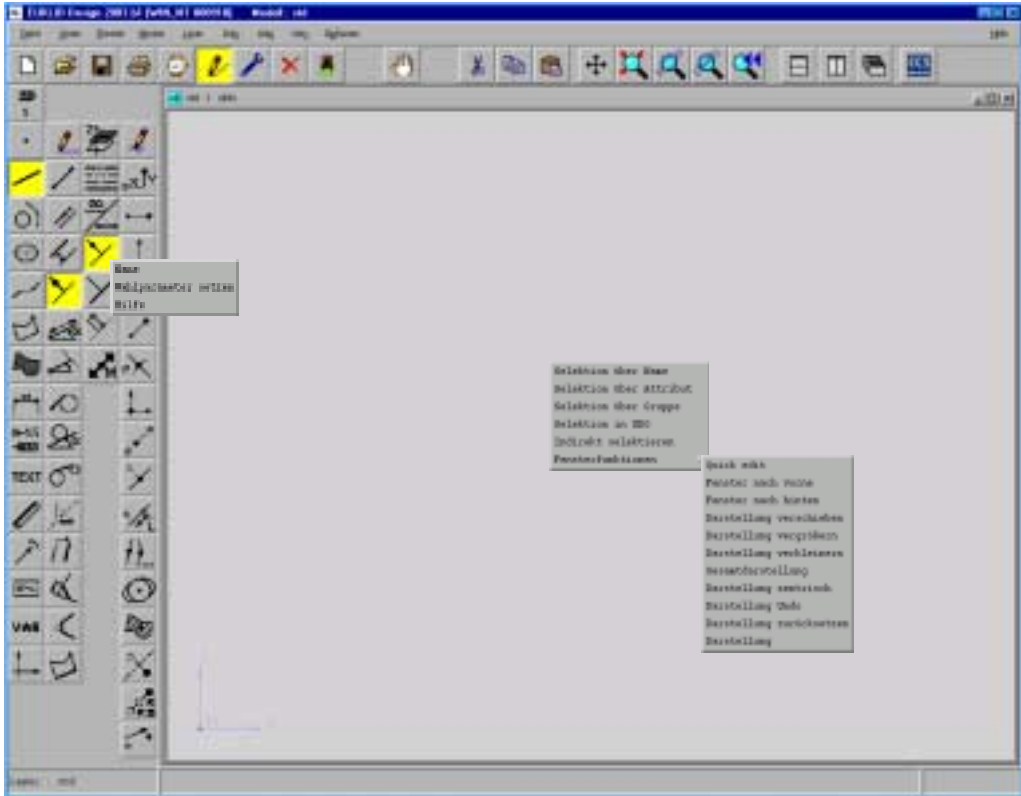
Introduction to Interactive Working

<i>Cursor shape</i>	<i>Meaning</i>
	Identify object in the sketcher
	Position on object in the sketcher
	Specification of auxiliary positions in the drawing area (crosshairs); e.g. window identification.
	The program is busy; no input possible.
	Switch the Selector-Parameter (Edit-Mode) You can have shown the different possibilities per keystroke.

Introduction to Interactive Working

3.3.4 Popup Menus

Some functions are reachable via popup menus. Press the right mouse button in the drawing area or over an icon in the main menu.



Popup menu in the drawing area

<i>Menu entry</i>	<i>Meaning</i>
Keep action	See Online-Help: Sketch actions
Sketch edit	
Sketch reset	
Quick edit	The object (of any type) nearest to the cursor is switched to the Edit-Mode.
Front	See Online-Help: Actions of the action group <i>Window</i>
Back	
Pan	See Online-Help: Actions of the action group <i>View</i>
Zoom in	
Zoom out	
Zoom all	
Zoom center	
Zoom undo	
Zoom reset	
View	See „ <i>The View of a Window</i> “ on page 14-6.

Popup menu in the drawing area

<i>Menu entry</i>	<i>Meaning</i>
Select by name	See „ <i>Selection by Name</i> “ on page 4-23.
Select by attribute	See „ <i>Selection by Attribute</i> “ on page 4-25.
Select by group	See „ <i>Selection by group</i> “ on page 4-24.
Select in UDO	See „ <i>Selection in User Defined Objects</i> “ on page 4-26.
Select indirect	See „ <i>Indirect Selection</i> “ on page 4-28.
Viewing	See „ <i>Windows</i> “ on page 14-1.

Introduction to Interactive Working

Popup menu on an action's icon

<i>Menu entry</i>	<i>Meaning</i>
Redefine	See „ Redefining Objects “ on page 8-15.
Name	See „ Object Names “ on page 4-16.
Layer	See „ Show corresponding Layer of an object “ on page 5-38.
Delete /	See „ Deletion of Objects in Edit Mode “ on page 9-7.
Undo delete	See „ Undoing Delete Actions “ on page 9-10.
Show sons	Shows all visible objects depending on the selected object ("sons")
Show ancestors	Sows all visible objects on which the selected object depends ("fathers")
Remove action	See „ Action-specific Deletion of Objects in Edit Mode “ on page 9-9.
Attributes	Define user attributes
Tooltips	See „ The Tooltips “ on page 3-38.
Help	See „ The Online Help “ on page 3-36.

3.3.5 Use of Scrollbars

Some dialog boxes include scrollbars used to display text requiring more space than available in a dialog box.

Via scrollbars at the right and bottom edges of lists in dialog boxes you can move (scroll) the content of the window horizontally or vertically. They consist of

- An upper and lower (or left and right) positioning arrow
- The scrolling box.

Selecting the upper or lower **positioning arrow** moves the list upwards or downwards, respectively, by one window's field or line. Selecting the left or right positioning arrow moves the list to the left or right by a predefined distance.

The **scrolling box** shows the size and position of the visible part of the list in proportion to the entire list. The position, and thus the part of the list being displayed, may be changed by selecting any point along the scrollbar.

3.3.6 The Keyboard

The keyboard is used to enter text in dialog boxes or in the text input box. The keyboard layout depends on the country where the program is installed.

All details in this section apply to inputs in the active **EUKLID Design** window unless explicitly stated otherwise.

3.3.6.1 Alphanumeric Input

Alphanumeric characters can be input via the keyboard and dialog boxes.

Keyboard inputs which are valid parameters are placed into the text box. The <RETURN> key is used to conclude an alphanumeric character string.

If the text box contains data proposed by the program, the <RETURN> key merely confirms this.

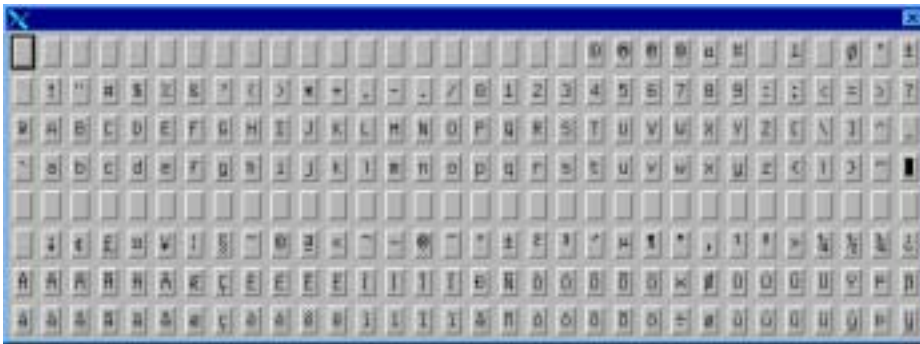
Introduction to Interactive Working

The Keypad Tool

Via the dialog box *Keypad* you can input characters, that you cannot reach with the keyboard

You can open the dialog box in the menu *Options, Keypad* or, if the cursor is located in the active text box, by pressing the right mouse button. You can close the dialog box in the same way.

The dialog box shows all available characters of the active font.



The octal input method

To input special characters, you may also use the octal input method.

`\xyz`

xyz stands for the octal ASCII-code of the special character.

Example:

If you input `'\101'` into the text box you get an 'A' in the drawing area.

Predefined function of some keys

Some keys have a predefined function. The location of these keys may be different depending on the target computer.

<i>Key</i>	<i>Meaning</i>
<ESC>	Cancel the current action or subaction. Close a dialog box Deselect an object (Edit-Mode).
<RETURN>	Confirm or terminate a text input
<TAB>	Toggle between alternatives
<SHIFT> + <TAB>	Reversing alternatives

Editing of Text Inputs

After entering an arbitrary character in the text box, you can use the following keys for editing in the text box (according to Motif rules):

<i>Key</i>	<i>Meaning</i>
<DELETE>	Deletes the character at the right of the cursor
<BACKSPACE>	Deletes the character at the left of the cursor
←	Moves cursor one position to the left
→	Moves cursor one position to the right
<SHIFT> ←	Selects the text from right to left
<SHIFT> →	Selects the text from left to right

You can insert text by the center mouse button in the text box.

You can delete the text selected by the <BACKSPACE> key.

Introduction to Interactive Working

<i>Key</i>	<i>Meaning</i>
<CTRL> ←	Moves the cursor one word to the left
<CTRL> →	Moves the cursor one word to the right

Input of numbers and calculations in the text box

Numbers input via the keyboard, which are valid as parameters, are placed into the text box. Either numbers or calculations may be input. The calculation is carried out (like a pocket calculator) and the result used as parameter value.

Valid characters for input of numbers and calculations

1 ... 0 Numerals, e.g.: 1479

. Decimal point, e.g.: 1479.87

, Decimal comma, e.g.: 1479,97



The decimal comma is not allowed in tables or in inputs made in AQL.

The valid operators are identical with those which may be used for "variables from formulas". These are listed in the table [„Operators“ on page 13-98](#) in the chapter Standardization.

Calculations may only include constants as arguments. No variables may be used.

The <RETURN> key is used to conclude the character string input.

Input of Special Characters by Keyboard

Alternatively, special characters can be input via the **keyboard**. Examples of the necessary key combinations are given in the table below.

<i>Input</i>	<i>Appearance</i>
@/0	∅
@#	□
@=	≈
@+ -	±
@ _	⊥
@ss	β
@c	ç
@m	μ
@%	°
@ @	@
@<0,1,...,9>	Special characters in ASCII code See the X-Windows table
@<xy>	See table „ <i>Input of Umlauts by Keyboard</i> “ on page 3-34

Introduction to Interactive Working

Input of Umlauts by Keyboard

<i>y x</i>	<i>A</i>	<i>a</i>	<i>E</i>	<i>e</i>	<i>I</i>	<i>i</i>	<i>O</i>	<i>o</i>	<i>U</i>	<i>u</i>
`	À	à	È	è	Ì	ì	Ò	ò	Ù	ù
'	Á	á	É	é	Í	í	Ó	ó	Ú	ú
^	Â	â	Ê	ê	Î	î	Ô	ô	Û	û
"	Ä	ä	Ë	ë	Ï	ï	Ö	ö	Ü	ü
~	Ã	ã					Õ	õ		

All other characters combinations preceded by the "@" character are interpreted literally.

3.3.6.2 Special Functions on Keys

Scrolling

The table below shows which **key combinations** can be used to scroll the window in different directions.

<i>Key</i>	<i>Result</i>
<CTRL> <↑>	Scroll window upwards
<CTRL> <↓>	Scroll window downwards
<CTRL> <→>	Scroll window to the right
<CTRL> <←>	Scroll window to the left

Scrolling the Menu Area

Key	Meaning
<↑>	The icon bar on which the cursor is positioned is moved upwards
<↓>	The icon bar on which the cursor is positioned is moved downwards

Function Keys

You can define function keys (shortcuts) by the command *Define function keys* in the *Options* menu (see section „*Define Function keys*“ on page 17-13).

Calling menus and menu commands

You may also call menus and menu commands by the respective key combinations.

- You can call menus by the <ALT> key and the letter underlined in the menu designation.

Example: You can call the menu Eile by <ALT>+<F>.

Subsequently, the menus stays open.

- Now you can call the menu command desired by entering the letter underlined in the menu command.

Example: You can call the action *Open model...* by entering the letter <o>.

- The most important menu commands may also be called directly (without opening the respective menu). The corresponding key combinations are shown behind the commands.

Example: You can call the action *Open model...* by <CTRL>+<o>.

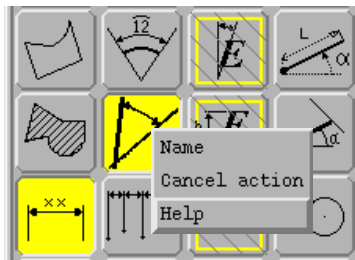
Introduction to Interactive Working

3.4 Help in *EUKLID Design*

3.4.1 The Online Help

EUKLID Design provides an Online Help facility in HTML format which you can view with a browser like Netscape or Internet Explorer.

Please press the right mouse button and choose *Help* in the popup menu over an icon of the main menus 2nd or 3rd column:



Your installed web browser displays the corresponding page with informations. Inside the online help windows the hyperlinks are displayed blue. Clicking on a hyperlink you can open its target. Please refer to the functionality of the popup menu of the right mouse button within your browser if your cursor is located over a hyperlink. There you can, for example, open a new window and open its target in that window.

In addition you can get this information via the *Help*-Menu in the menu bar (see next section).

Configuring the Online Help

Via the menu *Options, Configure Help/Documentation* you can define the browser for viewing the Online Help. For further information please press the Help button of the dialog box.

3.4.2 The Online Documentation

Via the *Help*-Menu you can step into the Online Documentation and also into the Online Help.

Documentation

This action starts the main menu of the **EUKLID Design** online documentation where you can follow the hyperlinks into a specific manual. They correspond to the printed manuals.

The online documentation is generated in PDF format. You can view it, for example, with *Adobe Acrobat Reader*. Hyperlinks are displayed blue. Index and table of contents contain hyperlinks too.

If you click on a hyperlink you will jump to its target. You may open a specific chapter or section using the lefthanded bookmarks.

On context

This action opens the online help for the active action (see previous section).

Configuring the Online Documentation

Via the menu *Options, Configure Help/Documentation* you can define the browser for viewing the Online Documentation. For further information please press the Help button of the dialog box.

Introduction to Interactive Working

3.4.3 The Tooltips

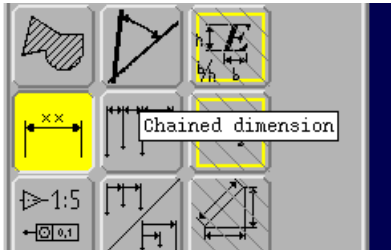
The Tooltips offer you additional information shown in a small window at the cursor-position. You have to position the mouse over the required element.

You can get Tooltips

- for icons of the user interface,
- for objects in the drawing area.

Tooltips for icons of the user interface

For each icon you may get a Help-Text which is displayed simultaneously in the text box:



You may activate the Tooltips:

- Stay over the requested icon without moving the mouse.
- The Help-Text changes dynamically when you move the cursor to another icon.

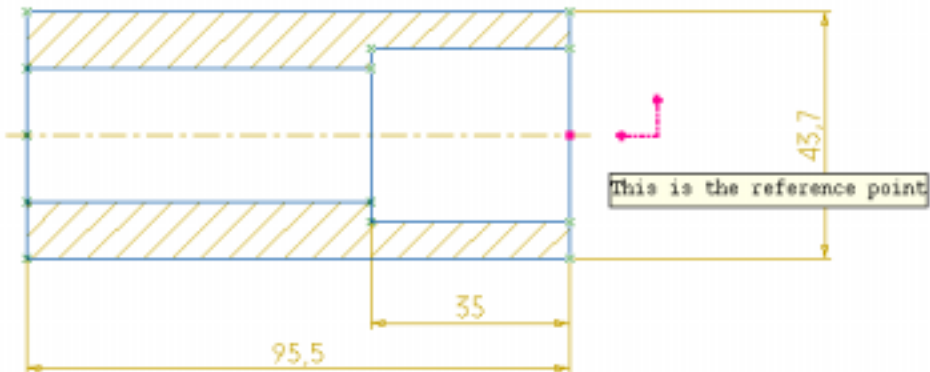
You may deactivate the Tooltips:

- Leave the menu, e.g. the Main menu, with the cursor.
- Wait a time without moving the mouse.

Tooltips for objects in the drawing area

In the Edit- or Delete-Mode you may get a Help-Text for each object in the drawing area. It displayed simultaneously in the text box (max. two lines).

You see the Tooltips of the closest object before any selection This may help you with the selection of the right object if several objects have the same distance from the cursor for example.



You may activate the Tooltips:

- Stay near the requested object without moving the mouse.
- The Help-Text changes dynamically when you move the cursor to another object.

You may deactivate the Tooltips:

- Leave the window or drawing area with the cursor.
- Wait a time without moving the mouse.
- Select an object.

Introduction to Interactive Working

You may change the text of Tooltips:

- Select the required object.
- Choose *Tooltips* in the popup menu of the icon of the corresponding action.
- Enter the text into the dialog box. You can delete or overwrite the text which is offered as default.
Please note: A 'line feed' is displayed.

Your text is stored with the object and is component of the model.

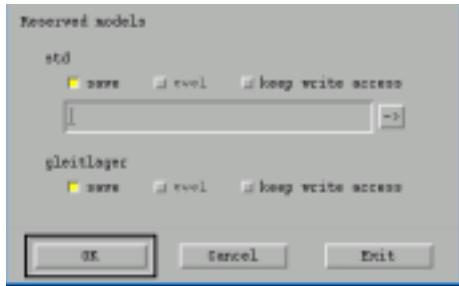
Switch the Tooltips on or off

Via the menu *Options, Tooltips* you can switch on or off the Tooltips. You can also define the start and hold delay. For further information please refer to the Help of the dialog box.

3.5 Finishing a Session

You can finish the program with *Exit ...* in the *File*-Menu or by the key combination <CTRL>+<x>.

The following dialog box is opened subsequently:



Reserved models are listed. The following options are available:

Button / Option	Meaning
save	You define whether the specified model shall be saved at the program end or not. By default, the model is saved.
keep write access	You may keep the access right for writing of the model. Default: Release the model
OK	The dialog box is closed, settings are executed.
Cancel	The <i>Exit</i> command is canceled.
Exit	The program is quit, no settings are executed.
	Enter a filename if you never saved the model yet. For informations on the subsequent dialog box please refer to section „ <i>Dialog box for file names</i> “ on page 5-40

Introduction to Interactive Working

3.6 Program Interruption

EUKLID Design may be terminated by pressing the key <DELETE> (kill signal) in the active start window.

Please note that the current model is not saved. The protocol file up to the point of interruption is retained, however.

3.7 Abort the Running AQL Program

<DELETE> key (kill signal) or <Ctrl> c

in the active start window: the running AQL program will be aborted.

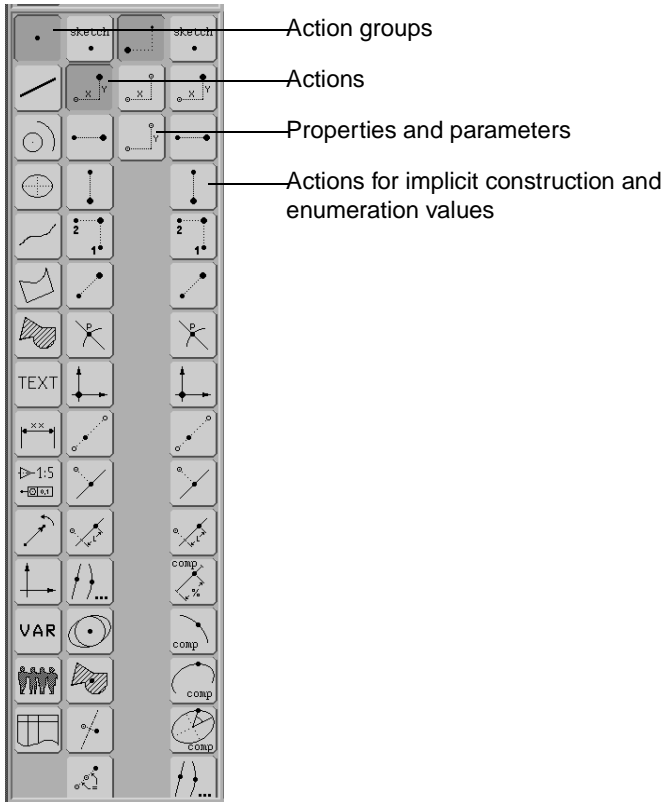
4 System Handling

4.1 Input Facility

A major characteristic of ***EUKLID Design*** operation is the menu-driven input facility. All activating inputs consist of a sequenced identification of menu fields or constructional elements;

The dynamic menu fields contain icons providing a self-explanatory display of the program functions appropriate to the status selected. You select the icon corresponding to the function required with the mouse. The menus are structured as follows:

System Handling



In *Create* mode, the icons of the action groups remain permanently visible in the first column of the menu area until another menu is selected. After a change of menu the topmost action group is always activated. All the corresponding actions are displayed in the second menu column. The third column contains the properties and parameters of the active action, the topmost icon again being activated. The fourth column contains the actions for the implicit construction of an object parameter, or values if there is only a finite number of input options for a parameter (e.g. a line of finite or infinite length).

The program repeats a given action until you explicitly select a different one. It is possible to discontinue any current action at any time by clicking on another action or

pressing the <ESC> button. If you click on another action group or action, the menu columns that follow will be arranged accordingly.

In the *Edit* and *Delete* mode, the first menu column displays a selected object group. In **EUKLID Design**, the object types are grouped together in an object group; in addition, you can create your own object groups.

If you have selected an object in *Edit* mode, **EUKLID Design** displays the action with which the object was created in the second menu column and the associated properties and parameters in the third menu column.

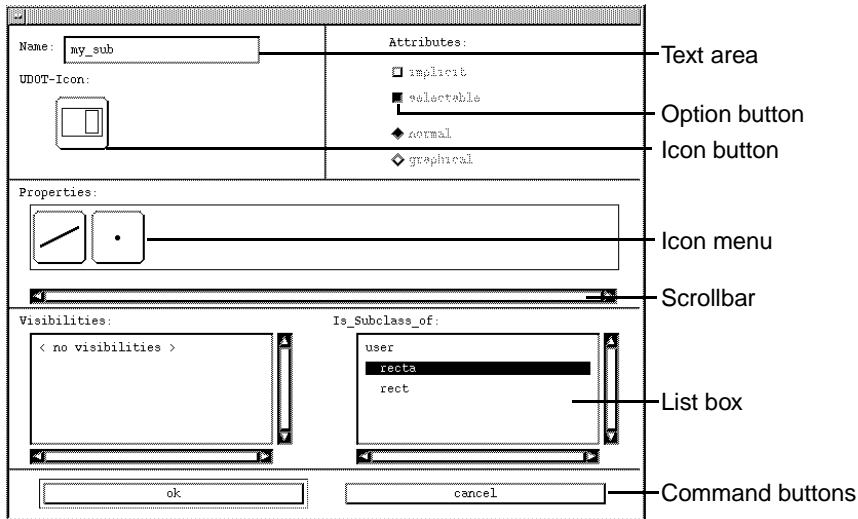
In *Delete* mode you can delete objects directly, e.g. by selecting them in the drawing area.

Temp mode actions have only an interrupting effect, allowing, for example, details or a selection of windows to be enlarged during the creation of an object.

When **EUKLID Design** requires an object parameter, you have the choice to create this object recursively (implicit construction) or to select an object.

Inputs in the text field are only necessary when parameters have to be furnished with specific values. In the case of more complex inputs, the system displays dialog boxes to allow more comprehensive user prompting.

System Handling



A *text field* displays information and allows text to be entered from the keyboard.

Option buttons contain options that can be activated by clicking on them.

An *icon button* displays an icon and allows a popup menu to be called up via the right mouse button.

An *icon menu* enables a selection to be made by clicking on the desired icon.

The *scrollbar* enables you to move through a list when there are more choices than can be displayed in the box.

A *list box* contains a list of choices that you can click on. Only one item in the list can be selected.

Command buttons contain commands that you initiate by clicking the appropriate button.

4.2 Parameters and Properties

EUKLID Design distinguishes between parameters and properties:

Action or objects can be controlled through parameters and properties. If a property or parameter changes, the action is invoked again and any dependent objects are recalculated.

The **third column** of the main menu first lists all properties and then all parameters necessary for the current object type (or the action) and the specified action.

The system expects parameters to be entered from top to bottom. If all the parameters required for an object have not yet been entered, one already entered can be corrected by clicking on it and entering a new value.

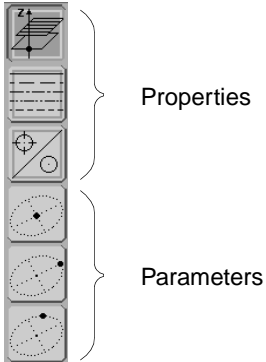
Properties

Properties describe those features of objects that are independent of how they are created, e.g. z-value and line pattern. An action that does not create any objects will therefore not require any properties.

The advantage of properties is that they do not have to be reentered when an action is redefined. Furthermore, in the case of actions that generate several effect objects, properties can be changed individually for each object.

System Handling

You can recognize properties on the user interface from the way they are displayed – with a different colored frame – as shown in the following illustration:



Parameters

Parameters define how actions differ, e.g. geometry and object placement.

4.2.1 Types of Properties and Parameters

EUKLID Design uses several **types of parameters and properties** to satisfy the various requirements and provide the required degree of user friendliness. To achieve greater clarity, each type of parameter and property on the user interface is represented by a different graphic:



Mandatory parameter/mandatory property



Optional parameter/optional property

4.2.1.1 Mandatory Parameters and Properties

Mandatory properties and mandatory parameters have no predefined value.

The user must specify these for the desired (create) action. Since these are mandatory entries, the system prompts for them in sequence from top to bottom without further action by the user.

If all the mandatory parameters and properties required for an action have not yet been entered, one already entered can be corrected by clicking on it and reentering. The program prompts for parameters and properties until all the necessary data has been entered, even if the input sequence has been changed.



Mandatory parameters may also be included in parameter lists and alternative parameter structures.

Parameters and properties can be objects, inputs to dialog boxes, values and enumeration values, each of which expects and enables different user actions depending on their type.

When, for example, a line is created between two points, a start point, among other things, will be expected. This must either be identified by the user or be able to be created by an implicit construction.

System Handling

The external appearance of a compulsory parameter or property on the user interface is represented by an **icon**. Any other parameters or properties subsequently required are highlighted.

Example



4.2.1.2 Optional Parameters and Properties

As a contribution towards user friendliness, optional parameters and optional properties have predefined values.

As we are now dealing with **optional entries**, the program does not automatically prompt for the input of these parameters and properties. It simply makes the optional parameter or property available to the user in the third menu column.

If an optional parameter or property is to be entered, the program is notified by clicking on the icon. Only then can and must the parameter be entered. If all the mandatory parameters and properties for an object have not yet been entered, one already entered can be corrected by clicking on it and reentering.

An example of this is the dash pattern for lines, curves, etc., which is interpreted as a predefined optional parameter and therefore does not have to be reentered every time by the user.



- Every parameter can have a default value assigned to it by the user. This value is saved when the configuration is saved.
- Some optional parameter and property values can be predefined by a group action (e.g. the z-value).
- Optional parameters may also be included in parameter list structures (see section „*Other Parameter Types*“ on page 4-10).

Icons for optional parameters and properties are displayed with a hatching pattern of top left to bottom right.

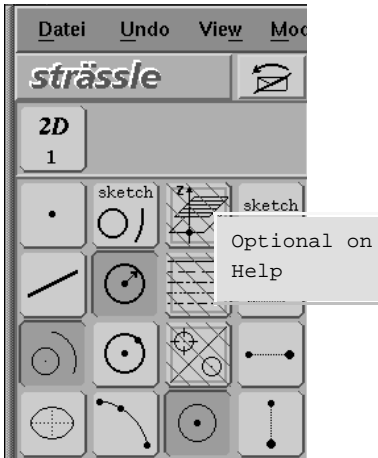
Example



Defining a Parameter as an Optional Parameter

You can define a parameter as an optional parameter as follows:

- ➡ Click the icon of the desired optional parameter or property with the right mouse button and choose *set optional parameter* in the popup menu.
- ➡ Then enter the parameter value (object, value or enumeration value).



Cancel Optional Parameter Definition

- ➡ Click the icon of the desired parameter or property with the right mouse button and choose *reset optional parameter* from the popup menu.

Properties of UDOs and parameters of UDAs can also be set in the definition environment if required. In other words, the UDO/UDA defines the default value for a parameter or property. This only applies to simple objects and values, e.g. lengths (when entered from a text area or a dialog box).

4.3 Other Parameter Types

In addition to the types mentioned above, the following parameter types also exist:



Parameter list



Alternative parameter

4.3.1 Parameter Lists

A parameter list is a collation of any number of objects into a list. The resulting object is created from this list, together with other parameters and properties. For example, a spline is created from a list of vertices, a surface from a list of adjacent objects.

An advantage of lists is that they can subsequently be changed: in edit mode, the list can be extended or shortened as desired, or the individual elements can be replaced by others.

In the case of a spline, for example, additional vertices can be added to or removed from the list at any time, or individual vertices replaced. This causes a change to the spline.

Parameter icons of the *parameter list* type are situated between upper and lower arrow icons.

Several parameter icons of different types (a sequence) can be situated between the arrow icons.

Alternative parameters (also in a sequence) are used where a choice of various object types (type class) is permissible.

The list is completed

- by clicking on the lower arrow icon
- by choosing *confirm* from the popup menu in the drawing area (right mouse button), provided you are in the multi-selection mode, i.e. you have selected more than one object with the mouse in the rectangle.

Create and Edit Procedures

Create

As with other parameters, list parameters are defined by

- identifying an object (clicking on it in the drawing area)
- using one of the selection mechanisms provided by the popup menu that can be called up in the drawing area with the right mouse button (see section „[Selection Mechanisms](#)“ on page 4-21)
- selecting a group (holding down the left mouse button and dragging the selection rectangle)
- entering the parameter value in the text area
- implicit creation via the fourth menu column.

A design element (e.g. a vertex for a spline) is highlighted in the drawing area after it has been entered in the parameter list. The list is closed by clicking on the lower arrow icon. The creation of the result object can then be concluded or the input of a further parameter may be required.

Edit

List parameters can be displayed in their order of entry by clicking on the lower arrow icon and then be modified if necessary. If the parameters are design elements, these are highlighted in the drawing area, the first one after the first click on the arrow icon, the second one after the second click, etc. The cycle returns to the first element after the last one in the list.



By clicking on the upper arrow icon, you simply access the parameter list and the first element in the list is selected.

Each highlighted element can be edited by clicking on the corresponding object type icon between the upper and lower arrow icons.

System Handling

Elements of the parameter list can not only be selected with the lower arrow icon (see above), but also with the icons in the fourth menu column.

☞ Click on the lower arrow icon in the third menu column.

The following icons, among others, will be listed in the fourth menu column:



selects the first element in the list



selects the next element



selects the previous element



selects the last element in the list.

Objects can be edited as follows:

- Select the desired element from the list as described above and click on the object type icon within the parameter list (third menu column). The selected object can then be
 - further queried and edited by clicking on the arrow icon above the fourth menu column (access the data structure)
 - redefined with the same or another action
 - replaced by another object of the same type (e.g. identification of the new object in the drawing area)



- By clicking on the redefine icon in the fourth menu column

This enables the list for the selected action to be redefined. The action and all other settings remain unchanged, the previous list is deleted and started over.



Redefining an action using the second column creates a new action. All parameters must be reentered. In contrast, the action remains unchanged when redefined using the fourth column. Only the parameter list is redefined.

Extending and Shortening of Parameter Lists in Edit Mode

Parameter lists can be extended or shortened.

After clicking on the lower arrow icon in the third menu column, the following icons, among others, appear in the fourth menu column, allowing the list to be extended or shortened:



Clicking on this icon allows another object to be inserted before the last selected element in the list.



Clicking on this icon allows another object of the same type to be inserted after the last selected element of the list.



Clicking on this icon allows the last selected element to be removed from the list.

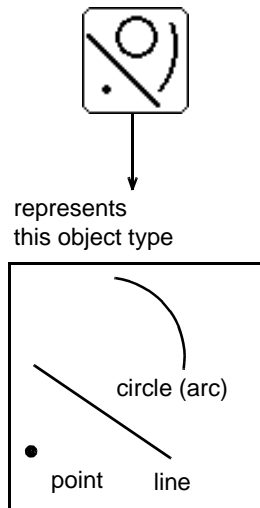
Typical applications are the creation of contour, spline and text from partial strings.

System Handling

4.3.2 Alternative Parameter

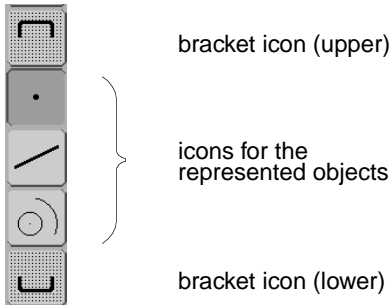
This type of parameter enables **a selection from a group of possible object types** to be made. The action type column can be considerably shortened with this parameter type so that, for example, the creation of an intersection between two design elements only requires one type of action, not one each for the various combinations “line-line” and “line-circle” etc. The choice of the objects involved is made using the corresponding alternative icons, which in this case facilitates all combinations of the objects displayed.

The **icon** of an alternative parameter is visually distinguished from other parameter icons by colored brackets, as shown in the following illustration.



Clicking on the alternative icon causes the represented object icon to appear in its place, enclosed by the upper and lower bracket icons (expanded status).

Opened (expanded) alternative parameter:



The alternative parameter is closed again by clicking on the lower bracket icon.

A particular object type can be chosen by clicking on one of the icons between the brackets

- to ensure that when entering a position in the drawing area, only one object of the specified type is selected (and not any one of those represented by the alternative icons). This is necessary, for example, if the length of a contour and not an individual contour element (that may lie over the contour) is to be evaluated.
- to create the particular parameter required for the input by implicit construction. All appropriate actions for this are provided in the fourth menu column, as with other parameters.



A logical linking of alternative parameters and list parameters is possible and occurs, for example, when contours are created. In this case, objects of the *line* or *circle (arc)* types (= alternative parameters) are arranged together to form a contour (= list).

System Handling

Behavior when Creating and Editing

- Create
When requesting an alternative parameter and then specifying a position in the drawing area, an object is identified whose type corresponds to one of the represented objects and is closest to the click position.
- Edit
In the edit mode, an alternative parameter is expanded by clicking on its icon and the design element itself is highlighted in the drawing area.

The parameter object can be edited in the usual way or can be queried or redefined using the fourth menu column.

4.4 Object Names

A name is dependent on the particular object to which it is assigned. This means that a name will continue to appear in the list of names for as long as the object concerned remains part of the data structure. A deleted object may still be addressed via its name as long as the session is in progress. Deleted objects which have names are also saved, even if they are not related to any other objects.

All objects, except those that already have a *name* parameter (variable, table, index) may be assigned names. In these exceptional cases the name is assigned as a parameter and can only be changed with the aid of the parameter mechanisms.

4.4.1 Assigning of Names

You can assign a name to an object as follows:

- in *Create* or *Edit* mode via text area
- individually (see Online Help, *Object Name*) and for selected sets (see Online Help, *Names and Assigning Names in Rectangle*) using symbols.

In *Create* mode, you can define a name in the following way before creating the object:

1. Choose *name* in the popup menu of the desired action icon.
2. Enter the name in the text area and press the <RETURN> key.
3. Furnish the requested parameters of the action.

You can also define the name of a parameter object of an action. To do so,

1. Choose *name* in the popup menu of the concerned parameter icon.
2. Enter the object name in the text area.
3. Select the parameter in drawing area or create the parameter object via the fourth menu column.

In *Edit* mode, you can assign an object name as follows:

1. Click on the object group icon in the first menu column.
2. Select the appropriate object in the drawing area. The creation action with its properties and parameters and manipulation actions, if any, are displayed in the menu area.
3. Choose *name* in the popup menu of the creation action icon.
4. Input the desired name in the *text area*.

If you have selected *Edit* mode from the *Edit last object* item in the *Undo* menu or via *quick edit* from the popup menu in the drawing area, actions 1 and 2 are not needed.

System Handling

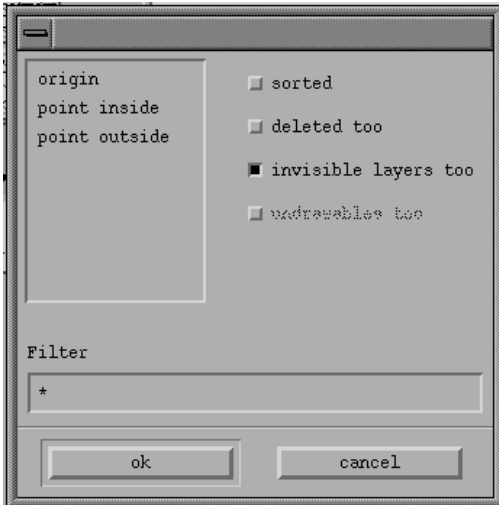
4.4.2 Display and Selection of Names

You can check before actually selecting by displaying the relationship between name and object. There are three possible ways:

- via the name selection dialog box
- in *Edit* mode via the text area
- via symbols singly (see Online Help, *Object Name*) and for selected sets (see Online Help, *Create Names and Symbols*)

Display and Selection via Name Selection dialog box

1. Choose *select_by_name* in the drawing area popup menu. The name selection dialog box is opened with all objects of the desired type, that have a name.



You can list all names sorted, including deleted and invisible objects by pressing the concerned buttons and enter a pattern in the *filter* field (usefull if lists are very long).

2. Select one or more objects from the list by clicking on their names in the list box. The object and/or objects named are highlighted within the drawing area (provided that it lies within the visible drawing area).
3. If the object displayed is the right one, click on *OK* to conclude the identification by name. If the object displayed is not the right one, select *cancel* or another name.

Example:

A line is to be drawn through two points, one of which lies within the visible drawing area and the other outside it. The two points have unique names; *point inside* is visible, and *point outside* is invisible.

1. Click on the line icon in the first menu column of the main menu in *Create* mode.
2. Select the action *line through two points*.
3. Select the first (i.e. visible) point in the drawing area as first parameter.
4. Choose *select_by_name* in the drawing area popup menu. The name selection dialog box is opened.
5. Click on the name of the line *point outside* in the dialog box.
6. Click on *OK*. The action is concluded and the line will be displayed on the screen.

Display via Text Area in Edit Mode

1. Click on the object group icon in the first menu column.
2. Select the object in the drawing area. The creation action with its properties and parameters and manipulation actions are displayed in the second menu column.
3. Choose *name* in the popup menu of the action icon.
The object name is displayed in the text area if the object was assigned a name.

System Handling

4.4.3 Changing of Names

You can change an object name in *Edit* mode as follows:

1. Click on the object group icon in the first menu column.
2. Select the appropriate object in the drawing area. The creation action with its properties and parameters and manipulation actions, if any, are displayed in the menu area.
3. Choose *name* in the popup menu of the creation action icon.
4. Input the desired name in the Text area.

If you have selected *Edit* mode from the *Edit last object* item in the *Undo* menu or via *quick edit* from the popup menu in the drawing area, actions 1 and 2 are not needed.

4.4.4 Deletion of Names

The deletion of names is done in *Edit* mode as follows:

1. Click on the object group icon in the first menu column.
2. Select the appropriate object in the drawing area. The creation action with its properties and parameters and manipulation actions, if any, are displayed in the menu area.
3. Choose *name* in the popup menu of the creation action icon.
4. Remove the name in the text area and press the <RETURN> key.



Blanks are significant characters too.



Names represented by symbols can be deleted via the action *Deletion of Names and Name Symbols in Rectangle* (see Online Help).

4.5 Identification of Objects

In *EUKLID Design* objects are identified by different selection mechanisms.

4.5.1 Selection Mechanisms

In *EUKLID Design*, an object can be identified by different selection mechanisms because not all objects are selectable by their geometry, for example length or angle.

You can directly use the normal selection mechanisms with the left mouse button:

- Selection by distance
- Selection in rectangle

You can call other selection mechanisms via the drawing area popup menu:

- Selection by name
- Selection by attribut
- Selection in UDO
- Indirect Selection

In *Create* mode, you can use all selection mechanisms.

In *Edit* mode, you can only use the selection in UDO and indirect selection if you want to replace an object parameter with another object.

In *Delete* mode, you cannot use the selection in UDO and indirect selection.

In *Browse* mode, you cannot use indirect selection because it works parallel.

System Handling

4.5.1.1 Selection by Distance

This is the default selection method for a visible object. The cursor has the following shape in the viewport:



Click near to the desired object in the drawing area. The object nearest the cursor position and of the desired type is then identified and, if necessary, highlighted.



If several objects share the same position on the screen, they are highlighted. The system switches to the multi-selection mode. The cursor shape changes as follows:



The selection set can be prepared in this mode.

If the *cursor icon* option in the *options* menu is enabled, the icon of the action that created the nearest object is displayed. This object is identified by clicking the left mouse button.

4.5.1.2 Selection in Rectangle

In this mode all objects of the selected type of which the parts are inside a selection rectangle will be selected.

1. Press the left mouse button and move the mouse. The anchor point of the rectangle will be at this position.
2. Move the cursor. The cursor will change to a rectangle.



The current cursor position will be diagonally opposite the anchor point of the rectangle.

3. Release the mouse button at the desired position. The object within the rectangle will be identified and, if necessary, highlighted.

If the *cursor icon* option in the *options* menu is enabled, the selected object set is highlighted.



If you selected several objects, although only one is permitted, you will have to repeat the selection, or a specific object must be selected from the set in question (individual selection mode).

4.5.1.3 Selection by Name

You can identify objects which are otherwise difficult or nearly impossible to identify by name, which is an attribute predefined by the system. This is the case if, for example:

- An object cannot be identified directly, e.g. length, angle, string, table etc.
- The object to be identified overlies or is overlain by another object of the same type.
- The object to be identified cannot be uniquely identified (or not at all) among the total set of objects in the design, due to high complexity and/or lack of detail clarity.
- An object is located outside the currently visible drawing area.

A prerequisite is that the object must have a name.

Another useful application is for the naming of a comparison object in order to create a calculated index.

System Handling

4.5.1.4 Selection by group

This allows you to compose a selection set from a large number of objects. This set can consist of

- Individual elements of group
- Individual objects and their dependent objects (sons)
- Individual objects and their precedent objects (ancestors)

The selection set can be further edited using the established selection mechanisms.

Selecting individual elements of groups

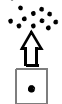
Use the right mouse button to select Selection by group from the popup menu, then, again using the right mouse button, Select group object contents. The name selection dialog box that follows allows you to select the groups, whereby only their individual elements are selected, but not the group objects themselves.



This functionality allows you to completely delete all individual elements of groups, for example.

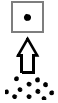
Selecting objects and their dependent objects (sons)

Use the right mouse button to select Selection by group from the popup menu, then, again using the right mouse button, Select dependent objects. Now you may select objects and their dependents (sons) using the established selection mechanisms (direct, in rectangle, right mouse button). Thereby the cursor shape appears as follows:



Selecting objects and their precedent objects (ancestors)

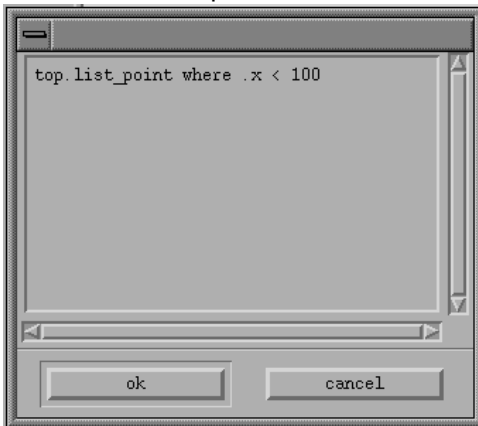
Use the right mouse button to select Selection by group from the popup menu, then, again using the right mouse button, Select precedent objects. Now you may select objects and their direct fathers using the established selection mechanisms (direct, in rectangle, right mouse button). Thereby the cursor shape appears as follows:



4.5.1.5 Selection by Attribute

With this selection mechanism, you can select an object using an AQL expression.

1. Choose *select by attribute* in the drawing area popup menu. A text dialog box is called up.
2. Define an AQL expression.



3. Click on *OK* to accept the selection, click on *cancel* to cancel it or select a new object. All objects covered by this expression will be selected if they are of the expected type.

System Handling

4.5.1.6 Selection in User Defined Objects

With this selection mechanism, you can select objects within a user-defined object using the standard selection mechanism.



In *Edit* and *Delete* mode, this selection mechanism is excluded because user defined objects are closed objects that can only be used as an entity.

1. Choose *select in user* in the drawing area popup menu. The four menu columns are cleared and the cursor changes to the U-form.



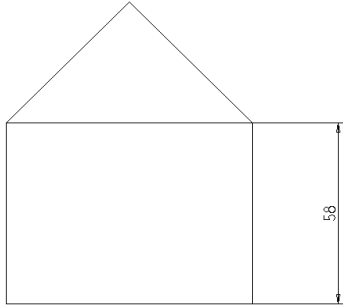
2. Select the desired user defined object using the standard selection mechanism. The user defined object will be highlighted and the cursor form changes to a rhombus.



3. Select the desired object within this user defined object using the standard selection mechanism.

Example

The height of an UDO *house* is to be dimensioned.



- ☞ Select the dimension icon in the first menu column in *Create* mode.
- ☞ Select the distance measurement *measure_p1c1p1c2* in the second menu column.
- ☞ Choose *Selection in user* in the drawing area popup menu.
- ☞ Identify the UDO in the drawing area.
- ☞ Identify the first point to be measured in the UDO.
- ☞ Choose *Selection in user* in the drawing area popup menu.
- ☞ Identify the UDO in the drawing area.
- ☞ Identify the second point to be measured in the UDO.
- ☞ Enter the dimension position.

System Handling

4.5.1.7 Indirect Selection

With this selection mechanism, you can select an object indirectly in *create* mode, i.e. you can select a dependent object related to the object to be selected, descend to the corresponding object in the data structure and select it. The action by which it was created can be displayed using the arrow icon in the special menu area.

Indirect selection is particularly suitable for the selection of basic objects such as length and angle that cannot be selected directly.

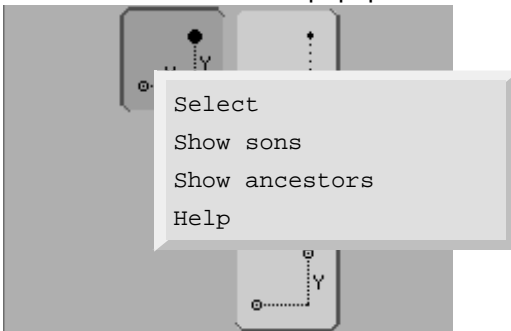
Proceed as follows:

1. Choose *Select indirect* in the drawing area popup menu. The popup disappears, the four columns are cleared and the form of the cursor changes to a vertical arrow.



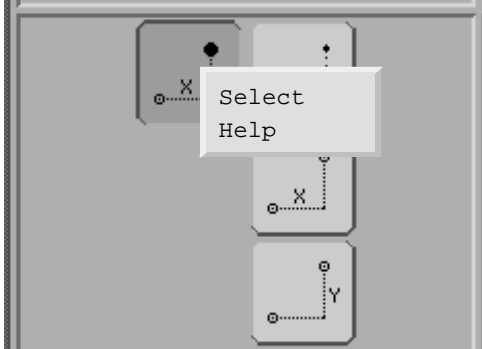
2. Select an object dependent on the object to be selected. The creation and manipulation actions of the dependent object are shown in the second column. In the third column the parameters of the creation action are shown.
3. Descend to the desired object.
4. Call the second column popup menu and/or the third column popup menu above the object parameter to be selected, depending on whether the object is to be selected via its creation action or directly via the object parameter.

In the second column the popup menu has the following entries:



<i>Select</i>	The result object of the creation action is selected.
<i>Show sons</i>	All dependent objects of the selected object will be highlighted.
<i>Show ancestors</i>	All objects the object depends on will be highlighted.
<i>Help</i>	The help function is called.

In the third column the popup menu has two entries:



<i>Select</i>	If you choose select the highlighted parameter will be selected (even if it is no object parameter).
<i>Help</i>	The help function is called.

5. Choose *select* in the popup menu of the concerning menu column.

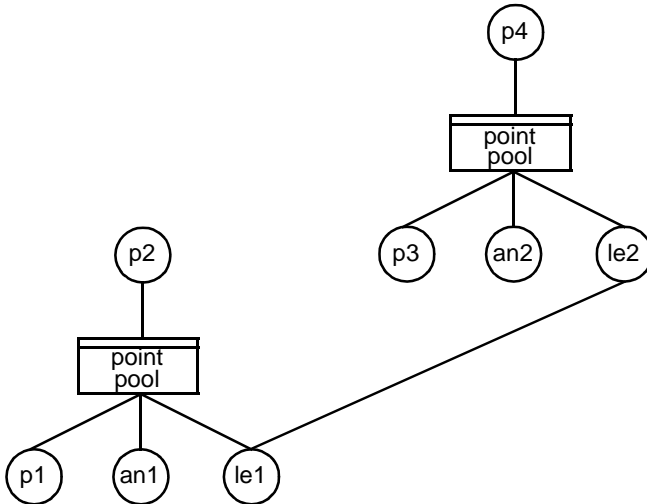


If you select an object of a wrong type or a value parameter you will get a warning. Then select via popup menu of the concerning menu column until the right type is selected.

Example

To create a point with polar coordinates *point_pool*, a length should be selected indirectly to provide the *distance* parameter. The point p2, with its parameters p1, an1, le1, already exists in the data structure as a point with polar coordinates.

System Handling



1. Change to *Create* mode.
2. Click on the action group icon *point*.
3. Click on the action icon *point_pool*.
4. Furnish the parameters *reference point P3* and *angle opposite the horizontal an2*.
5. Choose *Select indirect* in the drawing area popup menu. The popup menu is closed, the four menu columns are displayed empty and the cursor shape changes to a vertical arrow.
6. Select point p2. The creation and manipulation actions of the point are shown in the second menu column. In the third menu column the parameters of the point are shown.
7. Click on the parameter *distance (le)* in the third menu column. The parameter is highlighted in the drawing area.
8. Choose *Select* in the third column popup menu above the parameter icon *distance le1*. The parameter *distance le1* will be selected, i.e. the value of the length *le1* will be used for the parameter *distance le2* of the action *point_pool*.

4.5.2 Selection Modes

EUKLID Design provides two selection modes:

- Group mode
- Single mode

4.5.2.1 Group Mode

Group mode is active when several objects that, for example, overlap one another or are equidistant from each other, have been selected. In this mode the cursor has the following shape:



All selected objects are highlighted. You can now prepare the selection set:

- The left mouse button is used to select other objects instead of the currently selected set of objects.
- The middle mouse button is used to add objects to the selection set if they are not yet part of it, or they will be removed from the set if they are already selected.
- The right mouse button is used to call the drawing area popup menu. This popup offers additional functionality to manipulate the selection set.

Accept

With this menu command, you can confirm the selected objects. If only one object is allowed but if several objects are selected, the system will show the error message *Please refine the selection until 1 object is left.* and the system will return to group mode.

Deselect all

With this menu command, you can remove all selected objects from the selection set. You must then start the selection again. The <ESC> key has the same effect.

Select by name

With this menu command, you can extend or restrict the set of selected objects using the *object name selection* dialog box. You can modify the set by clicking on the names of the objects you want to add or remove from the set. Those named objects that to the set will be highlighted.

System Handling

Select by attribute

With this menu command, you can redefine the selection by entering an AQL expression. You can access the current selection through the designator *program.current_selection* in this expression.

Single mode

With this menu command, you can select single mode. You can also switch to single mode by pressing the <TAB> key. Once in single mode, the <TAB> key can be used to switch to the next element of the selection.

Delete all

With this menu command, you can delete the selected set of objects.

Move to layer

With this menu command, you can move the selected set of objects to another layer. The layer selection dialog box will be opened. Click on the name of the desired layer.

Viewing

With this menu command, you can call the viewport manipulation submenu.

4.5.2.2 Single Mode

In this selection mode only one object is selected. You can also examine each object of the selected set in detail.

When single mode is selected, the first object of the selected set is highlighted. The four menu columns are cleared and in the second column the creation action and, if applicable, the manipulation actions of the object are shown. In the third column the parameters of the creation action are displayed. If the object has a name it is displayed in the text window at the bottom. The cursor shape changes to the following cursor shape.



With the right mouse button, you can call the drawing area popup menu. This popup offers additional functionality of the selection set.

Accept

With this menu command, you can select the object you are just examining. All other objects will be removed from the selection set.

Deselect

With this menu command, you can remove the object you are just examining from the selection set and the next object will be presented. If the removed object is the last selected object, you have to start the selection again.

Group mode

With this menu command, you can return to Group mode.

Next

With this menu command, you can activate the next object of the selection set. So it is easier to find the right object you want to select. This is important for cases when different objects are lying one behind the other or if it is important to examine the relations of an object. If the actual object is the last object, the first object will be displayed.

You can switch to the next object by pressing the <TAB> key.

System Handling

Previous

With this menu command, you can activate the previous object of the selection set. This makes it easier to find the right object you want to select. This is important for cases when different objects are lying one behind the other or if it is important to examine the relations of an object. If the actual object is the first object, the last object will be displayed.

You can switch to the previous object by pressing <SHIFT>+<TAB>.

Show sons

With this menu command, you can highlight all visible objects that are dependent on the current object.

Show ancestors

With this menu command, you can highlight all visible objects on which the current object depends.

Show environment

This option is not available for all object types.

With this menu command, you can visualize the context in which an object is used.

Name

With this menu command you can give an object a name to make future selections easier. Enter the name of the object in the text area.

Delete

With this menu command you can delete the selected object.

Move to layer

With this menu command, you can move the selected object to another layer. The *Layer selection* dialog box will appear.

Viewing

With this menu command, you can call the viewport manipulation popup menu.



If only one object is allowed but several objects are selected, you must repeat the selection or select a specific object from the set in question (single mode).

4.6 Object Groups

An object group is a grouping of logically related objects, and may be regarded as a saved selected object set.

As an object can belong to several groups, the structures created are not strictly hierarchical, but should be regarded as network structures.

An object group may include both ordinary objects and existing object groups. A name is always assigned to the new object group.

Object groups may be selected using the normal selection mechanism, augmented or reduced by other object groups or individual objects and highlighted on the screen.

With the object groups facility e.g. related geometries (parts, views ...) may be assigned a name, varied as to contents and displayed on the screen.

The object group is a separate object type in ***EUKLID Design***.



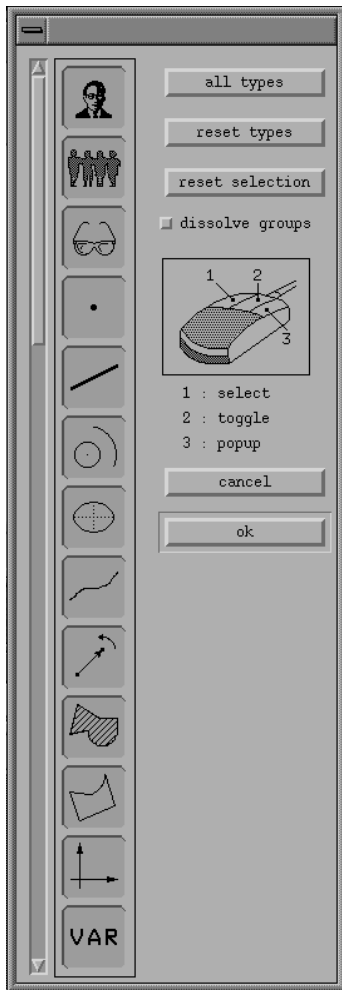
In contrast to layers, an object can belong to several groups.

System Handling

4.6.1 Creation of Groups



Groups are created using the *group create* action. ***EUKLID Design*** opens the selection dialog box for entering the objects that are to belong to the group (selection set).



The listbox contains the types of objects that can be selected. The default is all object types. If only one object type is to be selected, the default can be changed by clicking on the appropriate type icon.

All object types can be selected using the *all types* button.

System Handling

All object types in the listbox can be deselected using the *reset types* button.

The selection set in the drawing area can be deselected using the *reset selection* button.

Objects in already existing groups can be selected using the *dissolve groups* button. In this case the objects from the groups are selected and not the groups themselves.

Objects in the drawing area can be selected by pressing the left mouse button and dragging to form a selection set.

With the middle mouse button the objects selected will be added if they are not yet part of the selection and/or the objects which are already in the selection, will be removed from the selection set.

The *cancel* button terminates the selection without entering a selection set.

The selection set is confirmed by pressing the *OK* button.

The *OK+name* button that is also present in the selection dialog box for some actions saves the selection set for further use in the form of a *group* object.

If you want to add a group to the selection set, proceed as follows:

1. Click on the object type icon *group* in the listbox.
2. Choose *select by name* in the drawing area popup menu.
The name dialog box is displayed.
3. Select the name of the desired group in the listbox. The relevant group is highlighted in the drawing area.
4. Click on *OK* in the dialog box. The group is added to the selection set.

After assembling the selection set, **EUKLID Design** requests the input of a group name. Enter the desired name in the text area.

4.6.1.1 Displaying and Changing the Objects of a Group

Created groups may be displayed and updated in *Edit* mode. This is done using the usual selection mechanisms.

4.6.1.2 Deletion of Groups

As with all objects, deletion of a group is carried out in *Delete* mode. When deleting a group, only the group object is actually deleted and not the members of the group.



It is possible to delete individual objects of a group.

5 Working with Models

5.1 Application Concept

EUKLID Design is not an electronic drawing board, but goes beyond this in allowing a product that it has been used to develop to be viewed in various ways. The drawing is only one important alternative among a number of possible representations. The structuring of a product into assemblies and parts, or the parts lists themselves, are also often of interest. Consequently, these are also views that are supported by the program as standard. The open data structure of **EUKLID Design** also allows user-specific views to be created.

The aim of this chapter is to show how you can gain the greatest possible benefit in this respect using **EUKLID Design**. This includes the structuring of both the development process and the product that is to be designed. This chapter therefore concentrates less on describing new functions and more on illustrating typical user problems using familiar program functionality. This will be clarified by the use of a comprehensive example.

Working with Models

5.1.1 Using *EUKLID Design* Tools

This section will show how the individual *EUKLID Design* tools can be used.

5.1.1.1 Using Layers

Layers are used to arrange the the model according to various structural criteria. The structure format is strictly hierarchical, i.e. an object that is located in a layer cannot be in any other layer at the same time. This type of structuring is therefore suitable for modules and parts. On the other hand, as layers can contain other layers, it is possible to construct structures of any depth. The degree of detail of the structure is determined by the user.

Internal Layers

These contain the objects that are central to the product. Internal layers are made available immediately available the model is loaded. Internal layers should be used for data that must be available to the entire product team and that is central to the design.

External Layers

External layers should be used if the model is to be broken down according to organizational criteria. This has the following benefits:

- It allows distributed concurrent processing of a model by several designers, each responsible for their own area.
- Regarding the data volume it enables lean models which contain only the required information. These models are easy to use despite the existence of relations.

Relations between layers

In principle, all relations between layers are permitted, irrespective of whether they are external or not. To keep the models lean, the relations should always be formed outwards from the parent layer (internal layer). If this is not possible, the data to be used should be concentrated in the parent layer. This leads to clean models in the sense of top-down structuring of the product model. If relations based on external layers are created, part of the data structure of this layer must be duplicated. This leads to slower system performance.

5.1.1.2 Using User Defined Objects

User defined objects (UDOs) allow new object types to be introduced into the system. As with system defined objects, e.g. *line*, user defined objects can also be identified and evaluated under a type name, e.g. *screw*. This allows properties to be inherited from existing objects. For instance, a *screw* is also a *part* and can therefore inherit the predefined properties of *part*, without these having to be defined again in *screw*. Such a UDO is both a *screw* and a *part*.

Properties

These are features that describe the object itself in more detail. For a screw, for example, these are the nominal diameter and the length, but not the placement point or the angle of insertion. Once such a UDO is created, any number of actions (UDAs) can be defined to place them. The properties that characterize the object do not then have to be redefined every time, since these are attached to the object itself, and are thus assigned on creation independently of the action.

5.1.1.3 Using User Defined Actions

User defined actions (UDAs) allow specific application requirements to be implemented quickly without having to supply them as programs with the system. The user interface then corresponds exactly to the programmed actions. Usually, user defined objects (UDOs) are also created by user defined actions. Typically, therefore, UDOs and UDAs are used together.

Action types

- Actions that are performed by the user are normally stored in the data structure. If one of the action's parameters changes, the action will be recalculated (evaluated). This may possibly initiate a recalculation of the entire model to maintain the consistency of the relationrelations formed by these actions. Actions of this type are used, for example, to create geometric objects such as lines, etc.
- Another action type is only performed once on request and is not stored in the data structure (*drop* action). Actions of this type are, for example, plot, zoom, repeat, etc.

Parameters

Working with Models

Parameters are the variables that control how the action proceeds. Typical parameters for the creation of a *screw* UDO are, for example, the placement point and the angle of insertion. An alternative UDA for the placement of the *screw* could, for example, specify the center line as the parameter that determines the position of the screw.

5.1.1.4 Using the *Is a* Relation

The *is a* relation enables aspects that were not known when a UDO or layer was defined to be introduced retrospectively. For instance, a *screw* can be declared later to be a *tension screw*. The special feature of a tension screw is that it must be tightened to a specific torque to fulfil its function. A property of *tension screw* would therefore be the *torque* variable. A *screw* dealt with in this way would therefore be a *screw* and a *part* (see above) and a *tension screw*.

This possibility is also of special interest for post-design production planning. The *Is a* relation can be used to include manufacturing aspects (e.g. NC parameters) in the model without the designer having to take this into account at the outset.

5.1.1.5 Using the Layer Structogram

The Layer structogram is used to display the model structure at a time when no graphical representation of a part has been defined. It shows the product structure and allows actions that are to be used for product modelling to be identified as user interfaces.

5.1.1.6 Using Groups

In contrast to layers, groups allow models to be structured like a network, in other words objects situated in one group can also be part of another group. Some actions, such as copying, etc., permit groups as parameters, and so enable the copied object to be modified dynamically.

5.1.1.7 Using Tables

Tables are intended to be used in the assignment of standard parameters to parts and manufacturing forms. The evaluated data is located in a line of a table or distributed over several tables. In this way, a parameter set as used in the specification of a part is determined. The parameters are therefore assigned specific values (which can no longer be selected at will) and are allocated to a uniquely classifiable part or manufacturing form.

Working with Models

5.1.2 Structuring the Product Development Process

5.1.2.1 Specifying the Product

Every development process starts with the specification of the product to be developed. This specification is usually in a form that has to be converted into technically feasible facts. Initial calculations are often necessary for this purpose.

Example

Specification in words:

“Design a lifting platform for passenger cars”

Specification converted into facts that can be evaluated technically:

- Two-column lifting platform
- Max. load capacity 35000N
- Lifting stroke 2m
- Lifting time for 2m max. 10 sec.
-

Applying this to the *EUKLID Design Model*

- ☞ Open a new model and create there a layer *spec*.
- ☞ Make this layer active.
- ☞ Define the appropriate variables.
 - Load capacity = 35000
 - Stroke = 2000
 - t = 20

5.1.2.2 Design Proposal (Determining the Principles)

At this stage, the technical specification needs to be further refined and solutions to the design problems found. It is often necessary to examine several alternatives. This phase produces less documentation and concentrates more on incorporating the know-how of the designer.

Example

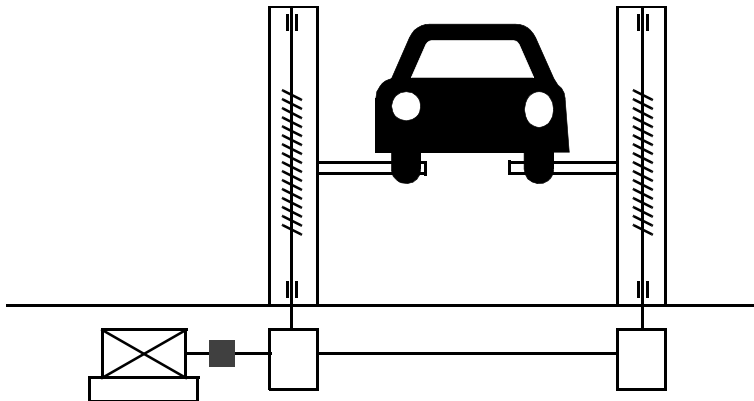
The following are selected from a number of options:

- Type of lifting platform (number of columns)
 - Type of drive (screw jacks / hydraulic, ...)
- etc.

Applying this to the *EUKLID Design Model*

A decision was made in favour of the following arrangement:

- Two-column non-recessed platform
 - Electro-mechanical drive with central motor, distribution gearboxes and screw jacks
 - 4 swing-out lifting arms to support the vehicle
- ☞ Create a *draft* layer and keep your ideas in this in the form of line sketches, initially without dimensions. Pay particular attention to the functional features of the design, for example the ability to simulate the lifting function.



Working with Models



In the case of very complex designs, it is recommended that you put down in writing how and why a particular solution was chosen so that the reasoning behind decisions can be appreciated. This can be done in ***EUKLID Design*** with a text editor and the ASCII file created is then stored in the project with Data Management functionality.

5.1.2.3 Calculations

Once you have decided how the design is to be carried out, the rough calculations can be performed.

Example

The gear ratio, the motor power, the screw jack design (pitch, number of starts, diameter) etc. are calculated.

The required motor power will be calculated as representative of all calculations. In simplified form (without taking account of inertia forces), this is derived from the load, the lifting speed and the efficiency according to the following formula:

$$P = F * \text{stroke} / t * 1 / \eta$$

Applying this to the ***EUKLID Design Model***

To keep the design as flexible as possible, it makes sense to store these calculations with the model and to base the further design on them.

☞ Make the *spec* layer active.

☞ Create the appropriate variables, e.g. from formulae.

To keep the model consistent, the data for load, stroke, time, etc. is read from the existing variables.

$$\text{Motor power } (F * h / t * 1 / \eta) = 7000W$$



In the case of complex calculations that cannot be covered by a simple formula, it is helpful to link some actions (UDAs) into AQL programs that produce result variables (as effect or result objects). The results of external calculation programs can be imported into the model either through AQL (file

linking etc.) or Data Management. In the latter case, a consistent relation between calculation and model does not exist.

5.1.2.4 Project Planning / Task Distribution

Example

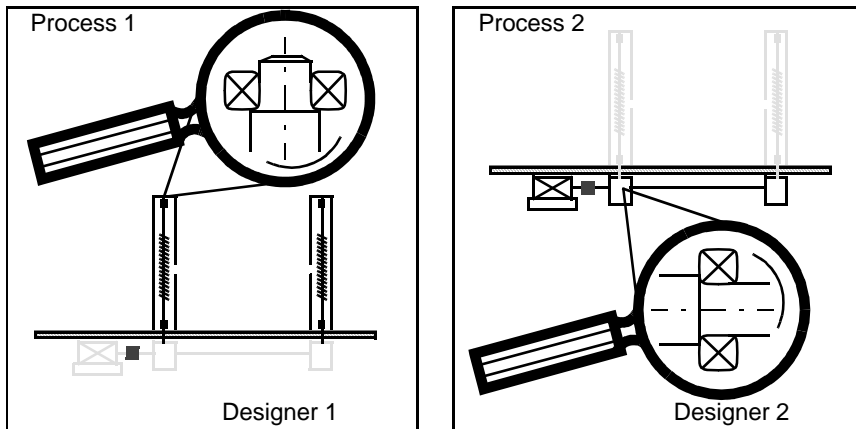
To enable the project to be processed quickly, the lifting platform is to be developed by two designers. The work will be divided into the following assemblies

- Drive assembly with motor, gearboxes, main shaft and screw jack coupling
- Column assembly with screw jack, column, swivelling arm, support and controller

The design leadership is taken by the designer of the column assembly (designer 1), to whom designer 2 supplies the drive.

Applying this to the *EUKLID Design Model*

To enable the drive assembly to be designed separately from the columns, only a few fitting dimensions and the specification data for the complete design need to be provided. On the other hand, the designer of the column is not interested in, for example, details of the gearbox bearings. A model structure that takes this into account works with external layers. This is shown in the following illustrations:



Working with Models

The lead designer (designer 1) works on the model without special limitations. Designer 2 loads the model into a dedicated process in read mode. The external layers to be processed by designer 2 are loaded explicitly later and the required layers made active. Designer 2 then continues the allocated design tasks. After completion, the external (processed) layers are resaved. Designer 2 can incorporate the modifications made by designer 1 by opening the model again. Designer 1 can see the work done by designer 2 by loading the appropriate external layers. You can prevent accidental editing conflicts by blocking the appropriate layers in each case.

5.1.2.5 Design Phase (Subprojects)

Once the specification has been firmed up and it is clear who will work on which assembly, design details such as bearings, guides, fixings etc. and the choice of purchased parts must then be clarified. This is the task of the designer of the respective assembly, who will refer to the specification and calculations as necessary.

Example

The designer of the drive assembly is responsible for the arrangement of the motor. The required drive power is known from the calculation, and this must now be related to torque and rpm. The motor is then selected from a standard catalog.

Applying this to the *EUKLID Design Model*

The motor is defined as a user-defined object (UDO). Properties of this UDO are the motor type, the rpm and the power. The required properties are selected from a table. This *motor* UDO is positioned by a UDA that has the placement point and the angle of installation as parameters. After supplying the properties/parameters from the corresponding variables and the geometry of the model, a consistent logical relation now exists between motor and specification. If the specification is changed (e.g. faster lifting), the motor selection is changed (greater power). The system issues a message if this power class is not available within the motor type.

5.1.2.6 Detailing

The detailing of the piece parts as finished, standard workshop drawings can, in the case of major projects, be performed as a sub-project within the assembly sub-project.

The detail drawing does not have to be created from scratch; all that needs to be done is to copy the parts of the design drawing that are relevant for the part in question. The necessary details are then added to the copy. This method ensures that the relation of the piece parts to the design drawing is maintained. If the design is modified, the piece parts are automatically modified accordingly at the same time.

Applying this to the *EUKLID Design Model*

- ☞ Make the piece part layer active.
- ☞ Create a coordinate system using an appropriate scale for the piece part drawing.
- ☞ Create the copy vector between the design drawing and required detail drawing.
- ☞ Copy the objects from the design drawing into the detail drawing. The selection set is then copied into the new coordinate system and enlarged or reduced accordingly.
- ☞ Complete the piece part drawing (dimensions, symbols, text, drawing frame etc.).

5.1.2.7 Integrating Subprojects into the Complete Project Model

The design should be complete once all subprojects have been integrated. In practice, the individual process steps influence each other and the whole therefore bears more resemblance to a process of iteration rather than one that is purely sequential. *EUKLID Design* supports this in a particular way.

In many manufacturing facilities, assembly drawings are needed in addition to design drawings. For the sake of consistency, these are derived from the piece part detail drawings by copying useable geometry groups from the piece part drawing into the assembly. The classic design process is complete once fitting dimensions, assembly notes, etc., have been added to the assembly drawings.

5.1.2.8 Derived Documents (Parts List, Ordering Lists, ...)

As the model contains all the necessary data, it is a simple matter to generate derived documents, such as parts lists, etc., from it. Using suitable procedures, *EUKLID Design* stores not only geometric data in the model but also specification, logical and organizational data. For example, the data necessary for parts lists lies in the parts or in an associated parts list banner. The calculation of the cost of the designed product can also be automated in this way.

Working with Models

Applying this to the *EUKLID Design* model

Select the action to create a parts list and select all the parts for the list. A parts list consistent with the model will be created. If a part is subsequently modified, the parts list will be modified at the same time.

5.1.2.9 Production Planning

In addition to parts information for the parts lists, production data can also be extracted from the model, e.g. the control geometry for an NC program. For this purpose, it is often necessary to have or prepare additional data that does not come from the classical design process. ***EUKLID Design*** provides the facility to enter such information (e.g. for manufacturing technologies) into the model retrospectively. The *is a* relation together with UDOs provide this facility. The various aspects of the model can be defined in the UDOs. Options here might be:

- Job sheet for the part
- Manufacturing information at the machine tool
(e.g. spindle speed, tool, cooling, ...)
- Purchasing information (e.g. supplier, alternative supplier, store, etc.)

Example

A milled contour is to be shown as such for a part. In addition, the necessary manufacturing data is to be stored in the model.

Applying this to the *EUKLID Design* Model

- ☞ Add the required contour to a new sublayer of the layer in which it was logically located.
- ☞ If not already available, create a UDO with the aspects that are of interest to manufacturing, e.g. for the user defined object type (UDOT) *milled contour* tool, these might be spindle speed, lubrication, etc..
- ☞ Pass these properties on to the contour by linking the UDO to the layer containing the contour with an *is a* relation.
- ☞ Enter the properties of the UDO in relation to the design. The advantage of this procedure is that if the model is modified, the manufacturing data is automatically updated accordingly. For example, the reduction of a radius of the contour may make it necessary to choose another tool. According to how the UDO is designed,

this can be performed automatically or attention can be drawn to it by an error message.

☞ One UDA can now be used to derive all milled contours. This UDA contains all *milled contours* as parameters and can, for example, produce an NC program. The *milled contour* aspect can now also be used as an identification and selection criterion in the model.

5.1.3 Structuring the Product

The *layer* object and the UDO are used for structuring purposes in **EUKLID Design**. These two objects are derived from each other and are distinguished only by their user interfaces, which are adapted for the respective purposes. The usual grouping by parts and assemblies is only one example of the structuring possibilities. For another user group, structuring according to different criteria may be of interest, e.g. fluid transmission in pipework etc. These requirements can also be taken into account, since the significance of an object type (UDOT) is determined dynamically by the user and is not predefined by the program. This grouping does not have to be specified in advance as top-down but can, if the structure is hierarchical, be entered later.

5.1.3.1 Parts and Assemblies Structure

Top-down / Bottom-up Structuring

The classical product structuring procedure is based on grouping into assemblies and piece parts. In such cases, the degree of detailing increases as design work proceeds. The main assemblies the product should have are decided first. This is a classic top-down procedure that makes sense in the phase when the product specification is relatively imprecise. This procedure can inhibit creativity during detail design. Parts, assemblies and methods of manufacture are in a state of flux and need constant revision. It therefore makes more sense to give the designer freedom down to line level and then transfer the resulting geometry into the product structure (bottom-up).

Introducing the “Part” Information

It must be possible in bottom-up structuring to also be able to introduce the *part* information retrospectively. On the other hand, standard parts carry this information with them. Other

Working with Models

classes of information such as *purchased part* or *electrical part* are also relevant. These can be inserted in existing structures either right at the beginning or retrospectively.

Example

A freely defined geometry is to be declared a part (bottom-up).

Applying this to the *EUKLID Design Model*

- ☞ Add the geometry of the part into a new sublayer of the layer to whose assembly the part belongs. The new layer will be given the name of the part.
- ☞ Use the *is a* relation for this layer. For *part*, use whichever UDOT is appropriate for your requirements (e.g. *purchased part*, etc.) or create a suitable new UDOT and enter the appropriate properties, taking into account relations to the rest of the design.
- ☞ The appropriate geometry can now also be identified and selected in the model according to the *part* aspect. This is valid in both the geometry and structure view-ports.

5.1.3.2 Standardization

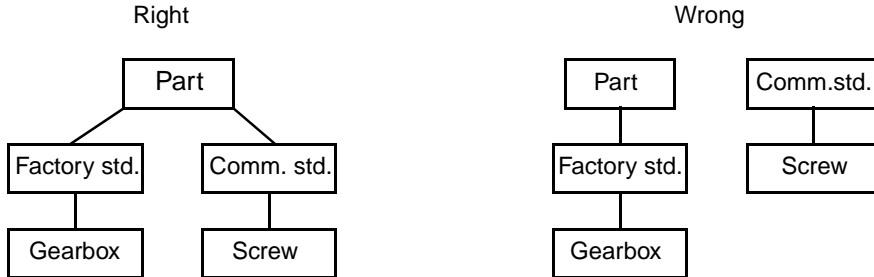
Commercial and works standard parts

As with other parts, commercial and works standard parts are represented as UDOs and UDAs. These are placed in libraries for further use and assigned the appropriate parameters. It is normally advisable to save one part in one UDOT. While a variable expression is used to specify a commercial standard part, a table should be used for factory standard parts. It is thus possible to create a whole family of parts using an index pointer (which may be a parameter).

Applying this to the *EUKLID Design model*

As we are dealing with “parts”, the UDOT *part* or *standard part* can be used right from the start when deriving the UDOT definition for the standard part. The “parts” class will therefore already be inherited by the standard part when it is defined. For subsequent classification purposes, it could be useful to derive new classes of *part* (e.g. *standard part*, *purchased part*) and use these in the derivation procedure. This class concept should be considered carefully before creating a standard parts library, as it has enormous consequences in terms of usability. Particu-

lar attention should be given to ensuring that there is not a surfeit of classes with the same meaning that are not derived from a common basic class.



Example

A screw is introduced into the model as an instance of a UDOT/UDA. The screw is then known as the object *screw* and can be identified. As with other object types (e.g. line), it can be allocated to a (user) menu.

Standard Manufacturing Forms

The same applies to standard manufacturing forms as to standard parts. In contrast to parts, the parameters will be more manufacturing oriented and less choice related. A part usually consists of many manufacturing forms, the constructive relationships of which determine the part. As availability down to the single object (point, line, etc.) is of greater importance, the representation is related more to UDAs.

Libraries

Standardized parts and manufacturing forms are best managed in libraries. When creating libraries, the version control of these standards must be taken into account. It must be ensured that models that contain obsolete forms can be reused and brought up to date.

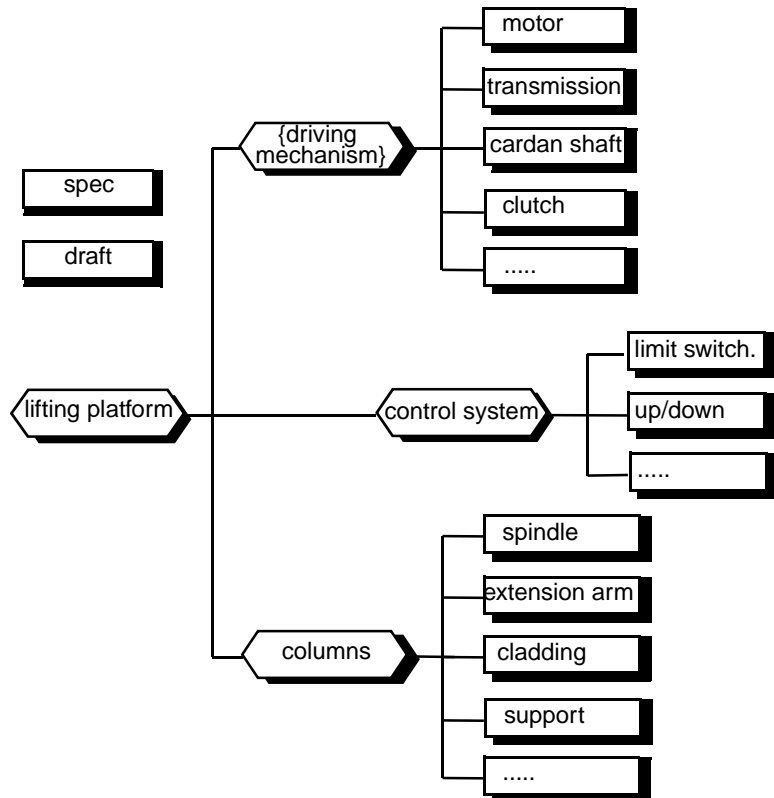
Working with Models

5.1.3.3 The Layer Structogram

EUKLID Design contains a tool to visualize your product model. You can use this to show parts symbolically without having to define their geometry in advance. These graphics can also be plotted and used as an abstract means of structuring in the concept phase. Parts, assemblies etc., can be identified in the finished design by either their symbolic or geometric representation. A clear relationship exists between these forms of representation. The Layer structogram is thus just another “view” of the product model. The structuring remains the responsibility of the user and is not determined by the system.

Example

The following diagram shows the structure for the lifting platform:



5.2 Layering

The **EUKLID Design** layering facility offers a large number of unlimited options for the logical structuring of drawings. It is possible, for example, to set up complex hierarchies of assembly components related by way of their geometry and function. Component-oriented design is fully supported by comprehensive multicolor layering (see section „[Application Concept](#)“ on page 5-1) as is the simultaneous editing of a model by several users. Furthermore, as far as displays are concerned, it is also possible to control whether objects in layers can be seen or selected.

When you create a new model, the system generates a first layer (topmost layer of the hierarchy) automatically. This layer is the active one when the model is first loaded and carries the name of the model.

The number of layers available is unlimited; each layer is identified by a unique, freely definable name. Newly created objects are allocated to the layer that is currently active. Each object is allocated to a single layer; a layer may comprise as many objects (including other layers) as required.



A comfortable tool to visualize layers is the Layer structogram, see chapter „[Visualization of Layers](#)“ on page 14-18.

5.2.1 The Layer Menu

All **EUKLID Design** layering actions are controlled by the *layer* menu. The layering facility is invoked by clicking on *Layer* in the menu bar.

Layer		Help
Create	Shift+Ctrl+e	
Delete	Shift+Ctrl+l	
Set active	Shift+Ctrl+a	
Status	Shift+Ctrl+s	
Move objects	Shift+Ctrl+v	
Move sublayer	Shift+Ctrl+u	
Save external		
Load external		
Convert to user		
Is a		

- Create:

Deleting layers
- Delete:

Deleting layers
- Set active:

Setting the active layer and model
- Status:

Manipulating the layer state
- Move objects:

Moving objects to other layers
- Move sublayer:

Moving layers
- Save external:

Saving external layers
- Load external:

Loading external layers
- Convert to user:

Converting layers to user defined objects
- Is a:

Defines types of layer

Working with Models

5.2.2 Creating Layers

You may create and name any number of layers to allow the structuring of a design.

☞ Choose *Create* in the *Layer* menu.

☞ Input the name of the layer in the text area. The dialog box for setting layers active is opened.



☞ Input whether the new layer must be activated.

If you select *set active* and confirm with *ok*, the new layer becomes the active layer and the previously active level is selectable. The *selectable* status allows existing objects in the layer concerned to be identified and changed.

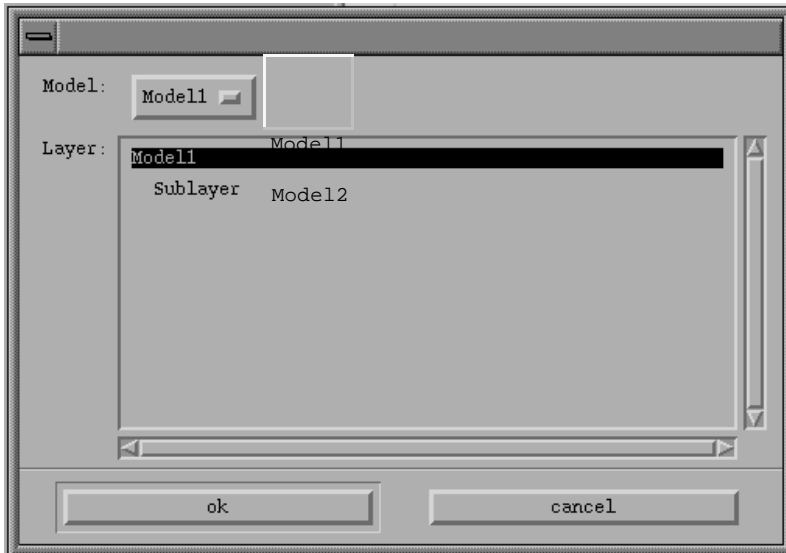
Otherwise the new layer is selectable and can be seen. The *visible* status allows existing objects to be displayed.

The new layer is a sublayer of the previously active layer and lies just one level beneath it in the hierarchy. In the list box of the layer dialog box, its name is therefore shifted 2 places to the right compared to the previously active layer. The position in the hierarchy of a new layer can be determined by making a particular layer active before creating the new one.

5.2.3 Setting the Active Layer and Model

The current working layers are active layers. All newly created objects are assigned to these layers. Objects in these layers are selectable and visible. Only one layer is active at a time. If a hitherto inactive layer is set to active, the previously active layer is set to selectable.

☞ Choose *Set active* in the *Layer* menu. The layer dialog box will appear. The active model will be the default model.



To change the active model,

- ☞ select the desired model in the *Model* field via popup menu. The sublayers of the new model will be listed in the *Layer* list box. The top layer of the model will become active.
- ☞ If required, select another layer from the list.
- ☞ Click *OK*. The selected model will become active. The name of the active model is displayed in the *Layer* field.

To change the active layer of the active model,

- ☞ Select the requested layer in the *Layer* list box. The name of the selected layer is highlighted in the *Layer* status dialog box.

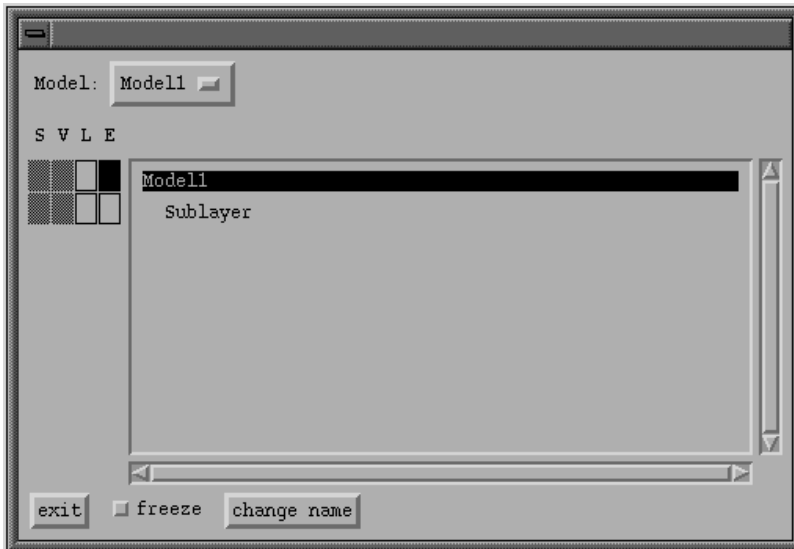
Working with Models

☞ Click *OK*. The selected layer will become active. The name of the active layer is displayed in the *Layer* field.

5.2.4 Changing the Layer Status

The status of layers may be changed via the *layer status* box.

☞ ► Choose *Status* in the *Layer* menu. The *Layer status* dialog box will appear:



This dialog box allows you to set the following :

S (selectable)

Objects in selectable layers can be selected, deleted and have their parameters modified. These states are view specific and have to be set in the *Viewport* specification dialog box (see section [„The View of a Window“ on page 14-6](#) in chapter Visualization Techniques). It is only shown here.

V (visible)

Objects in visible layers can only be seen. This state is view specific and has to be set in the *Viewport* specification dialog box (see section „[The View of a Window](#)“ on page 14-6 in chapter Visualization Techniques). It is only shown here.

L (locked)

Objects in locked layers cannot be modified or deleted.

☞ Click the *L* button. The button is highlighted.

E (external)

External layers are saved in dedicated model files (see section „[External Layers](#)“ on page 5-27).

☞ Click the *E* button. The button is highlighted.

Freeze

The updating of models when the status of a layer is changed is suspended so that the time-consuming, especially in the case of complex models, reformatting of the graphic is omitted.

☞ Click the *freeze* button. The button is highlighted.

Change name

The names of layers can be modified.

☞ Select the layer to be modified.

☞ Click *Change name*.

☞ Enter the new name in the text field.

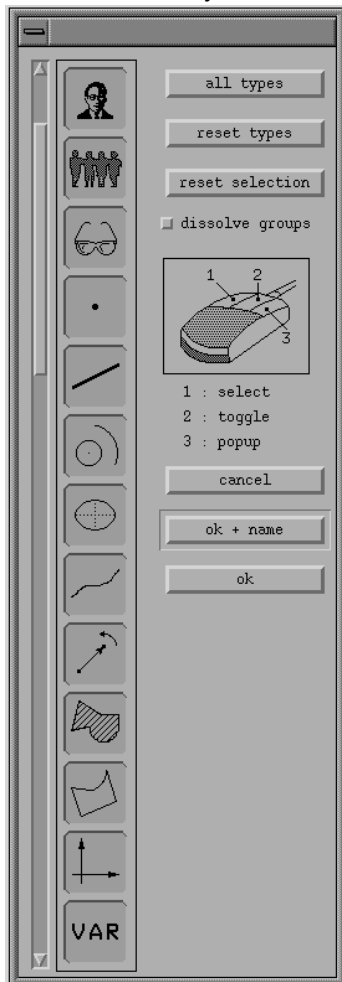
☞ Click *Exit* when you have changed all the states you want.

Working with Models

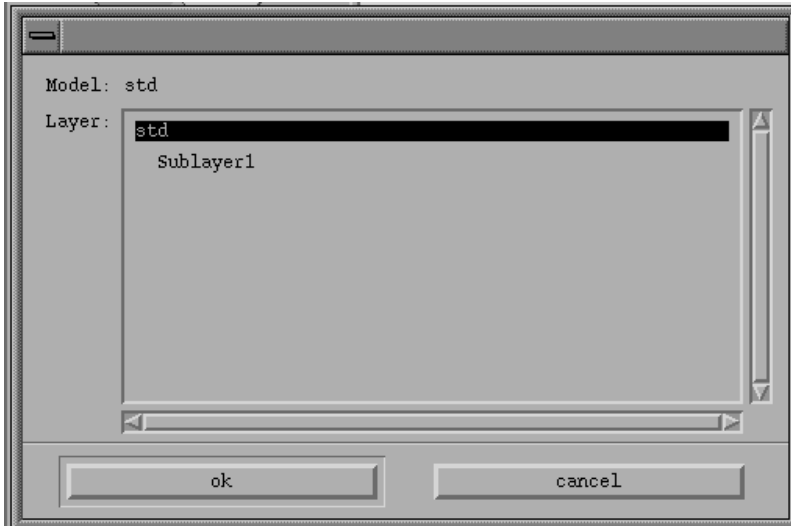
5.2.5 Moving Objects to Another Layer

With this action you can incorporate a bottom up structure.

☞ Choose *Move objects* in the *Layer* menu. The group dialog box will appear.



- ☞ Select a group of objects of a desired type.
- ☞ Click *OK* or *OK+name*. The layer box will now appear.



- ☞ Select the desired layer from the list box.
- ☞ Click *KK*. The selected objects will be moved to the target layer.

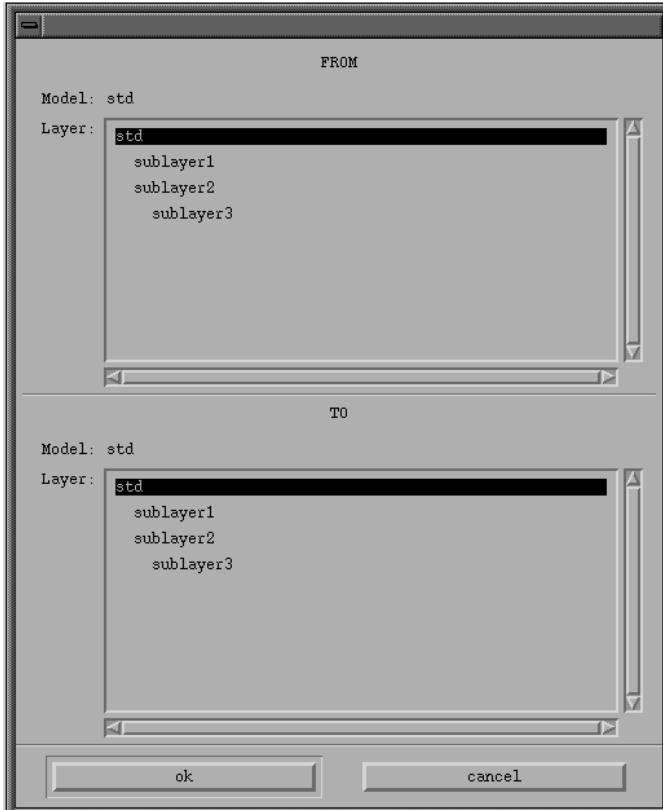
In *Edit* mode, you can use the *Layer* command of the popup menu of the second menu column to move the object created by the action to another layer. The layer box is displayed.

Working with Models

5.2.6 Moving Layers

A layer may be placed below any other layer in the hierarchy. This facilitates the handling of assemblies, components and auxiliary dimension layers which may contain dimensions, for example.

☞ Choose *Move sublayer* in the *Layer* menu. The dialog box for moving sublayers will appear.



- ☞ Select the layer to be moved in the upper list box.
- ☞ Select the target layer from the lower list box.
- ☞ Click *OK*. The layer will be moved below the target layer.

5.2.7 External Layers

The hierarchical structure of layers can be spread over several model files. External layers offer a way to split a project into smaller subprojects that are stored on separate model files.

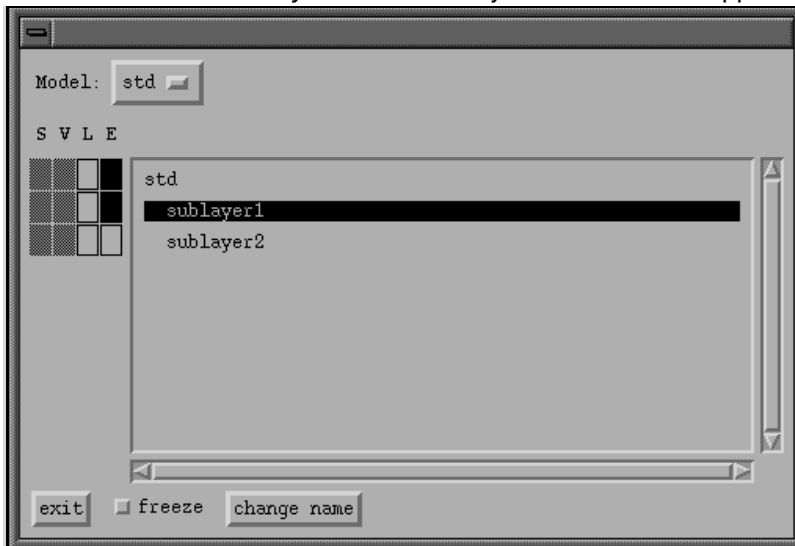
- ☞ This reduces the size of the model on which the system has to operate. When designing a part, other parts do not have to be loaded into memory (performance).
- ☞ This allows more than one designer to work on the design simultaneously.

One gets most benefits of external layers in well structured projects (see chapter [„Application Concept“ on page 5-1](#)).

5.2.7.1 Declaring External Layers

A layer will be declared external in the following way:

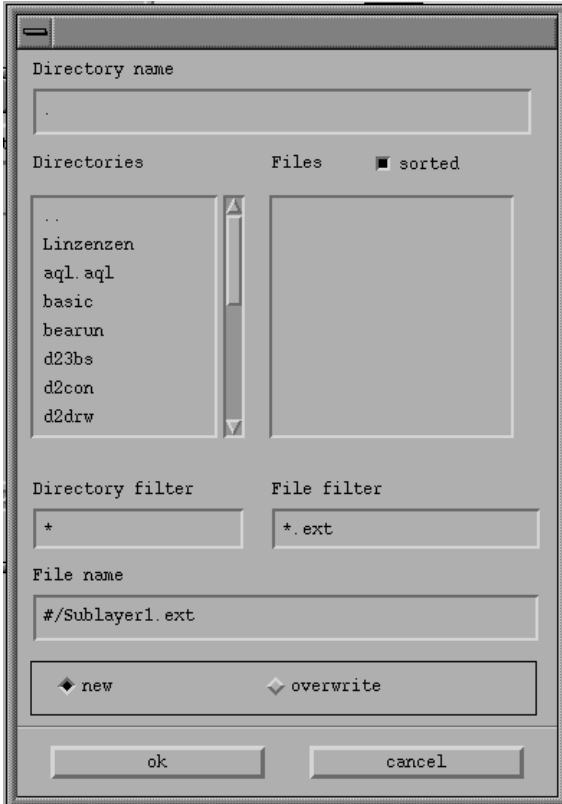
- ☞ Choose *Status* in the *Layer* menu. The layer status box will appear.



- ☞ Click the *E* button to the left of the layer concerned. The button is highlighted.

Working with Models

The file dialog box will appear.



The default file name is the name of the layer and the extension `.ext`.

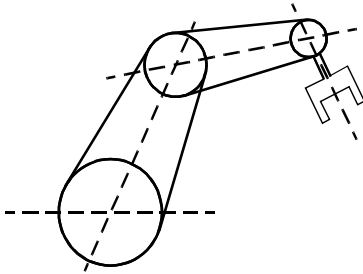
- ☞ Correct the model file name if desired.
- ☞ Click **OK**.

The father layer of a set of external layers contains the shared data between those sub-layers. All actions that operate across file boundaries will be stored in the file of the enclosing layer. Whenever an external layer references objects belonging to a father layer, the external layer file will contain a copy of the objects it references. The father layer file will contain records to link these copies to the original objects.



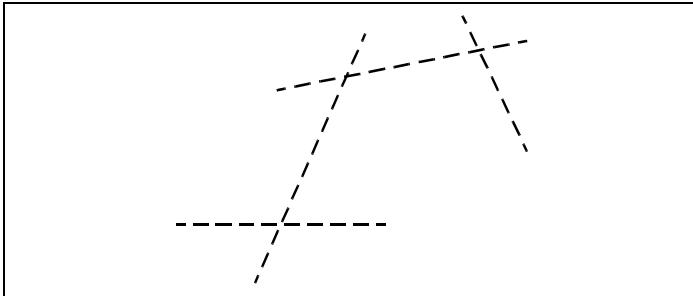
We recommend that users store shared objects in the father layer.

Example



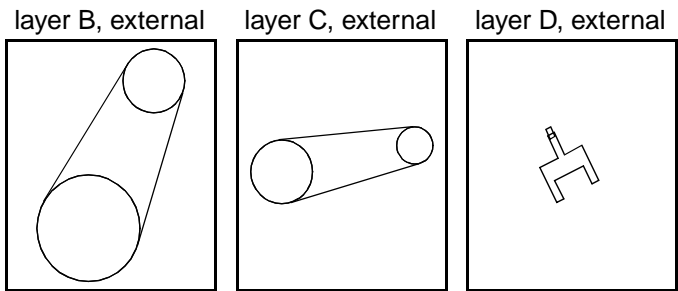
A robot has been designed based on the construction geometry framework stored in layer A.

layer A, external

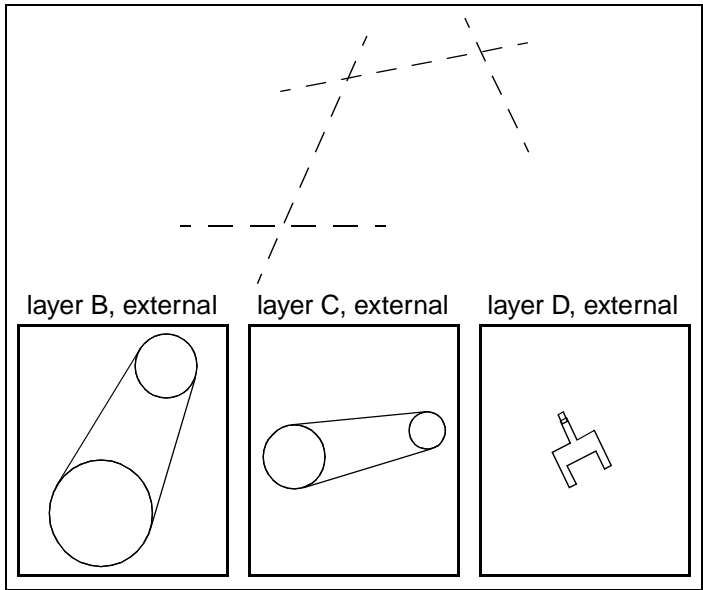


The various parts of the robot are described in detail in layers B, C and D. When designing a part, the other parts do not need to be loaded into memory.

Working with Models



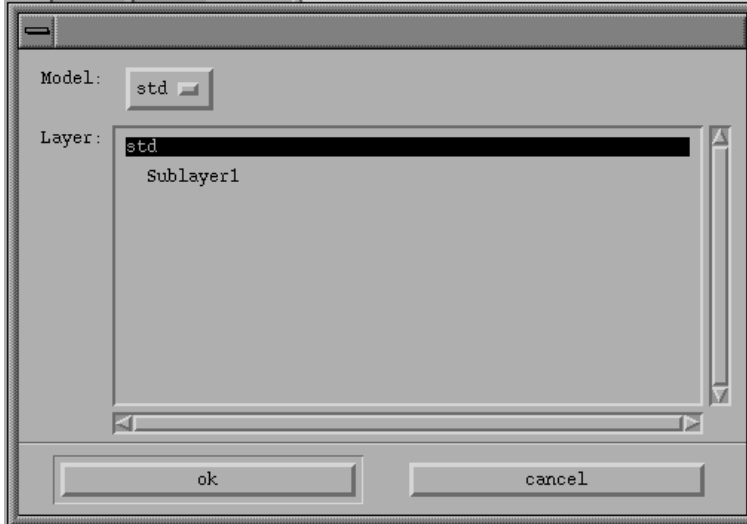
layer A, external



5.2.7.2 Saving External Layers

An external layer will be loaded in the following way:

☞ Choose *Save external layer* in the *Layer* menu. The layer dialog box will appear:



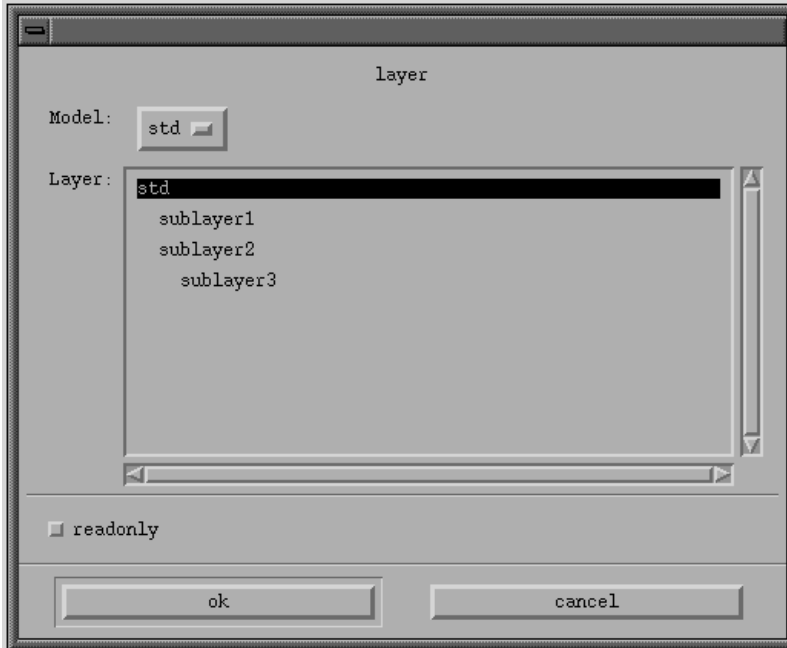
- ☞ Select the external layer in the list box.
- ☞ Click *OK*. The file dialog box will appear.
The default file name is the name of the layer and the extension *.ext*.
- ☞ Correct the model file name if necessary.
- ☞ Click *OK*. The requested layer is saved.

Working with Models

5.2.7.3 Loading External Layers

An external layer will be loaded in the following way:

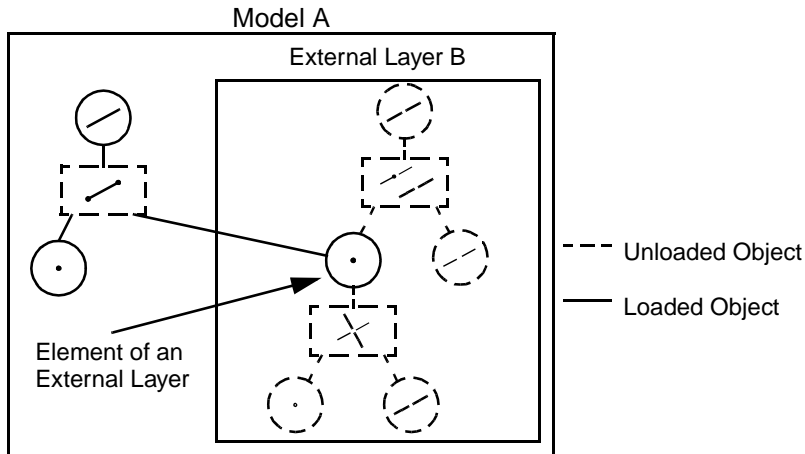
☞ Choose *Load external* in the *Layer* menu. The layer box will appear:



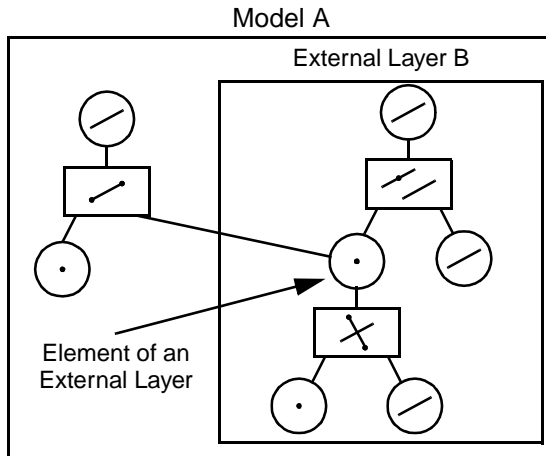
- ☞ Select the external layer in the list box.
- ☞ Click *readonly* if you only want to read the external layer.
- ☞ Click *OK*. The requested layer will be loaded.

Even if an external model is not loaded, it will be there and will contain only those objects that are referenced by the father layer:

External Layer not Loaded



External layer Loaded



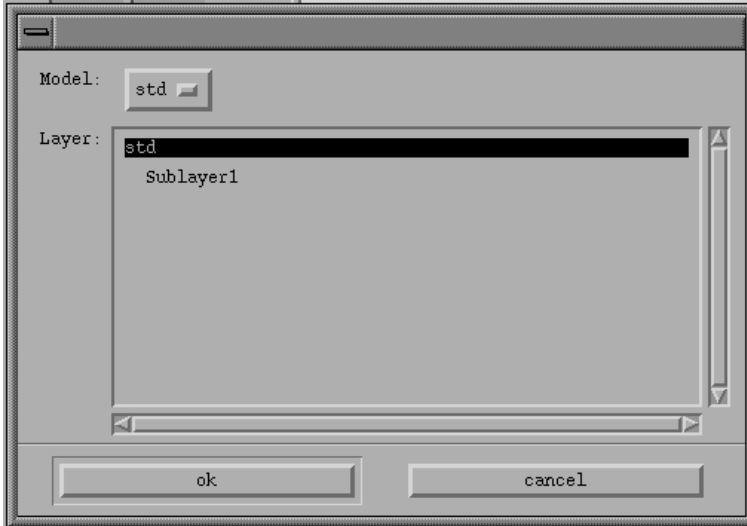
The system will nevertheless allow references between objects stored on external layers. If a lot of such references occur, the size of the external layer model file may be big.

Working with Models

5.2.8 Converting Layers to User Object

A layer can be converted to a UDO instance.

☞ Choose *Convert to user* in the *Layer* menu. The layer box will appear:



☞ Select the layer to be converted in the list box.

☞ Click *OK*. The layer is closed, it is not visible anymore in the layer menu.



To convert a user object back to a layer, use the *Convert to layer* command from the *UDO* menu.

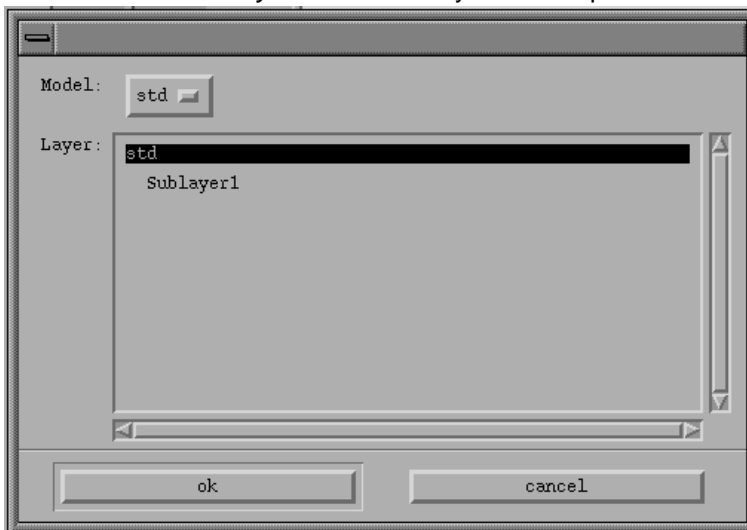
5.2.9 Adding a User-defined Object Definition to a Layer

The use of a user-defined object type definition (UDOT) is especially useful to add manufacturing and logistical information to the layer.

The object information will be copied from the UDOT to the layer, the action information will not be duplicated. The properties of the type will be queried as additional parameters.

☞ Load the UDOT.

☞ Choose *Is a* in the *Layer* menu. The layer box is opened.



☞ Select the layer that must be extended with the UDO content.

Working with Models

☞ Click **OK**. The UDOT dialog box will now appear:



- ☞ Select the name of the requested UDOT.
- ☞ Click **OK**.
- ☞ Enter the properties of the UDO. The object data is copied from the UDOT to the layer.

The layer now has the same properties as the UDOT. It can be addressed both as a UDO of a particular type or as a layer.

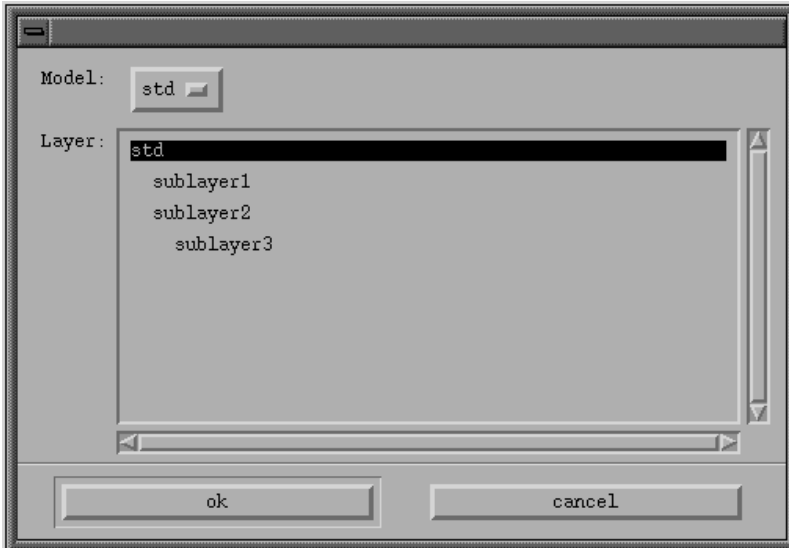
5.2.10 Deleting Layers

It is possible to delete entire components, assemblies, dimensions etc. en bloc, i.e. not at object level, if the layer hierarchy is properly structured.



The main layer cannot be deleted, being the basis of the model concerned.

☞ Choose *Delete* in the *Layer* menu. The system displays the layer box.

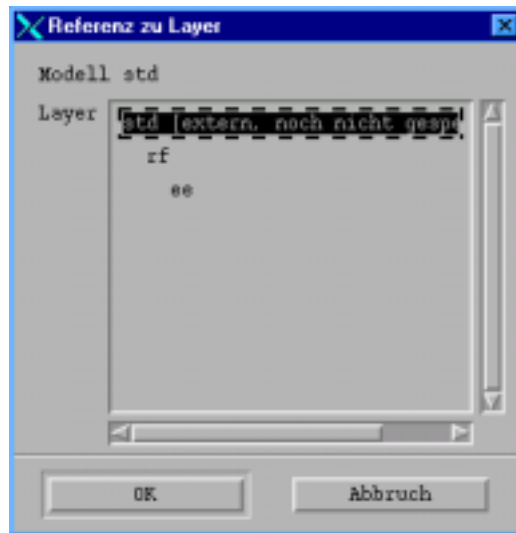


☞ Select the layer in the list box you want to delete.

☞ Click *OK*.

Working with Models

5.2.11 Show corresponding Layer of an object



5.3 File Operations

Each **EUKLID Design** model (also tables, UDOs, UDAs external layers) contains a unique identification number (ID). It will be located and loaded via the file search rule (see „[File search rule \(location table\)](#)“ on page 17-25).

Please **do not** copy model files via the operating system! Otherwise multiple files contain the same identification number. Please use the save actions of **EUKLID Design**.



The contents of the active model can be saved in the user configuration file. Every new model will get the contents of the configuration file (see „[User Specific Saving](#)“ on page 17-8).

Saved Informations

If you save a model in a file the following informations are added:

- Model Data
Your design containing relations to other files: tables, UDOs, UDAs, external layers.
- Change Comments, File History
With each saving (model, configuration, UDx, external layers) you may add a descriptive comment.

Your comment is automatically completed with:

- User (login name)
- Computer name
- Product version, hardware platform, creation date
- Date, time
- Last pathname of the model.

At the moment of opening and closing files you may have a look at the file history as a list of comments in a separate list box.

- File Preview
When saving, the current screen contents are 'photographed' and embedded into the model file as a quick preview graphics. You can view it quickly later on **without**

Working with Models

opening the file and get a quick overview in the file dialog box by stepping through the listed file items.



If you want to have a benefit of the preview graphics, please think about which typical sector you want to see later as a preview and adjust your drawing area to it before you save.

- **One or more Backups**

From the second saving into the same file always the last state is kept as internal backup version. In addition, from the third saving you may choose to keep the last backup as additional one. So you can keep several backup versions. They are saved also in the model file (in compressed form).

At model load you can select specific backup versions:

- You may view the preview graphics of a specific backup version. In addition you get also the file history up to this version.
- If you want to load a specific backup version it will be extracted automatically into a separate model file (with the same ID) with an automatically generated file name. This file will be opened.

At model save you may remove internal backup versions. A separate dialog box shows the list of backups. Double clicking an entry you may view the preview graphics.

Dialog box for file names

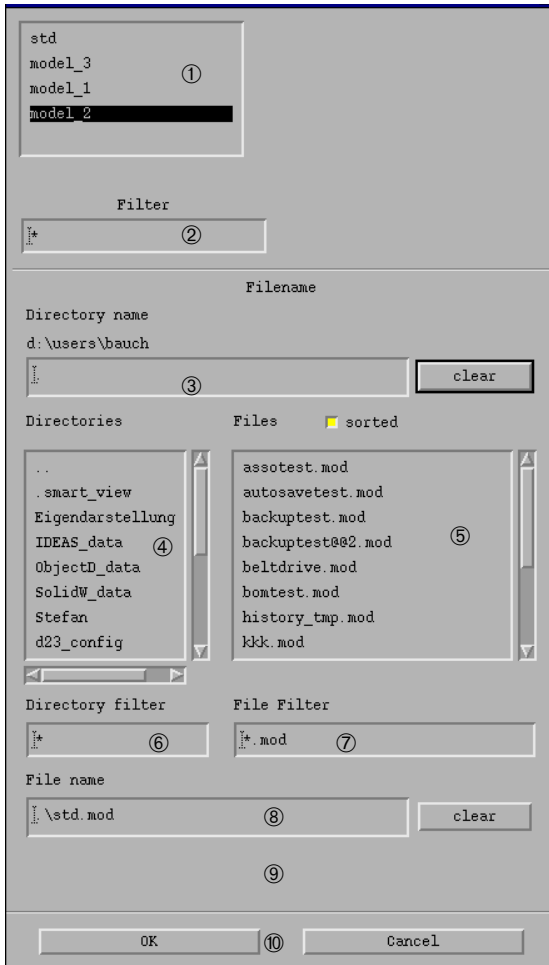
For all file inputs and outputs, the file dialog box is opened to enter the file name.

You can move the cursor between the input areas with the mouse or the <TAB> key.



Maybe the list boxes show only parts of the entries in the case of a long list or only the beginning of the name in the case of long file names. Using the scrollbar, you can view the next part of the list or the end of the file name.

The following illustration shows an example of the fields in the dialog box which are arranged according to the appropriate action:



- ① You can select the desired model here (model list box).
(Only in association with model operations).
- ② The model filter can be entered here. Conclude with the <RETURN> key.

Working with Models

- ③ The desired directory can be entered here, e.g.
 - # Installation directory
 - ~ The users home directory
 - .. Parent directory of the current directory
 - . Current directory (working directory)
 - + Library directory specified in the configurationWhen selecting via the *Directories* list box, the directory name will copied into this box.
- ④ Select the required directory from this list. It will be copied into the *Directory name* list box. The contents of the list depend on the search expression used in the directory filter ⑤.
- ⑤ Select the required file from this list. It will be copied into the *File name* list box. The contents of the list depend on the search expression used in the file filter ⑥. The list can be sorted in alphabetical order using the *sorted* button.
- ⑥ The search expression for directories can be entered here. Conclude with the <RETURN> key.
- ⑦ The search expression for files can be entered here.

Examples of filters:

<i>*.mod</i>	all model files of the specified directory (default argument for model operations)
<i>*</i>	all files in the specified directory
<i>../*</i>	all files of the parent directory
<i>parts/t*.mod</i>	all model files beginning with <i>t</i> of the directory <i>parts</i>
<i>~</i>	all model file of the home directory
<i>#</i>	all model files of the installation directory
<i>+</i>	all model files of the library directory specified in the configuration file

- ⑧ The desired file name can be entered here. Blanks are not allowed. Please pay attention to the conventions of your operating system (number of characters, accents, etc.). When selecting a file from the *Files* list box, the file name selected will be copied into this box.

EUKLID Design uses default file name extensions for file types if no extension is specified. When reading or writing other file types, **EUKLID Design** automatically uses the following file name extensions:

Extensions related to models:

<i>.ext</i>	External layers
<i>.mod</i>	Model files
<i>.s2s</i>	EUKLID model files
<i>.tab</i>	Tables of SIGGRAPH-Design
<i>.top</i>	Model files of SIGGRAPH-Design
<i>.uda</i>	User defined actions
<i>.udo</i>	User defined objects
<i>.udt</i>	Tables of EUKLID Design
<i>.udv</i>	User defined values
<i>.x_t</i>	Parasolid model files
<i>.x_b</i>	Parasolid model files in binary format

Extensions of other created or used files:

<i>.aql</i>	AQL program files
<i>.bom</i>	Bill of materials files (standard configuration)
<i>.cgm</i>	CGM files
<i>.cps</i>	PostScript files (colored)
<i>.dwg</i>	DWG files
<i>..dxf</i>	DXF files
<i>.hpg</i>	HPGL files
<i>.iga</i>	IGES parameter files
<i>.igs</i>	IGES files
<i>.pff</i>	Picture File files
<i>.plt</i>	Plot files
<i>.ps</i>	PostScript files (black and white)

Extensions of configuration files:

<i>.agp</i>	Action groups configuration files
<i>.cfg</i>	Configuration files
<i>.lta</i>	File Locator Table
<i>.mnu</i>	Menu configuration files
<i>.ogp</i>	Object group configuration files

Working with Models

⑨ Options can be chosen here (depending on the action):

<i>new</i>	Creates a new file when saving. If a file with this name already exists, you have to specify in a popup menu whether you want to overwrite the existing file or specify a different file name in the <i>File name</i> box.
<i>overwrite</i>	Overwrites an existing file when saving, e.g. when saving intermediate statuses of the existing model.
<i>Read only</i>	Opens a file in read mode: The model cannot be saved.
<i>Keep write access</i>	Reserve file in write mode further on. Protection against overwriting by another user.

⑩ You can start (*OK*) or cancel (*Cancel*) the operation.

5.3.1 Creation of New Models

Each model gets a name which is used to build the corresponding file name.

☞ Choose *New model* in the *File* menu.

☞ Enter the required file name in the text box via keyboard. A viewport with this model name will be opened and set active.

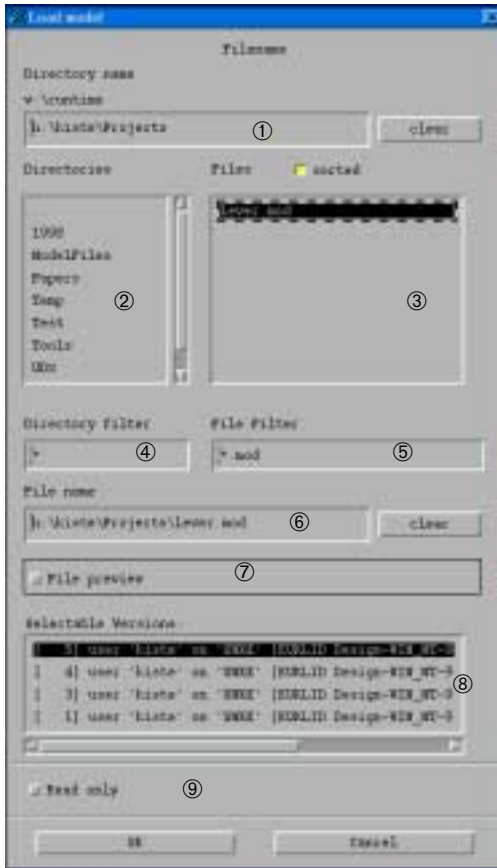
The model name will be shown in the dialog boxes of models and layers

.You can open any number of models, but only 10 viewports (see chapter Visualization Techniques, section „*Windows*“ on page 14-1).

5.3.2 Opening Models

EUKLID Design models may be read and displayed via file name.

☞ Choose *Open model* in the *File* menu. The following file dialog box is displayed:



5

Working with Models

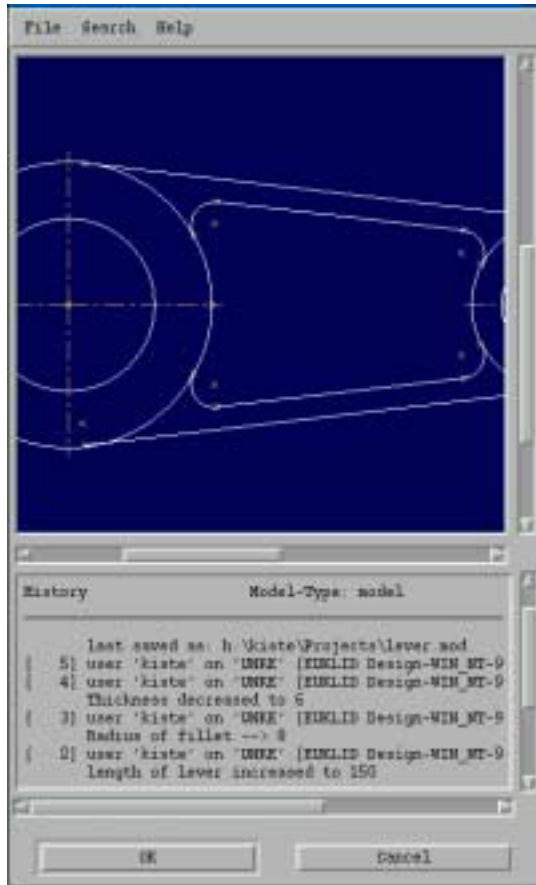
The dialog box defaults are as follows:

- ① Name of the current working directory
- ② Names of subdirectories
- ③ Names of the model files
- ④ Filter for directories
- ⑤ Filter for model files (*.mod*)
- ⑥ Name of the current model (empty provided no model is loaded)
- ⑦ Don't show the preview graphics
- ⑧ List of available backup versions (empty provided no model is selected)
- ⑦ Open in write mode

Open a model

Proceed as follows when opening:

- ☞ Select the name of the desired directory from the *Directories* list box if the file is not in the working directory. The name will be copied into the *Directory name* box.
- ☞ Enter the file name by clicking in the list *Files* or via keyboard into the box *File name*.
- ☞ After selecting, the models internal backup versions are displayed in the *Selectable Versions* box. Please select a specific backup version in the list box if you want to see its preview graphics or file history. Press *File preview* and the following dialog box will be opened:



The upper part of the dialog box shows the view of your model you defined before the saving. You may pan the display by pressing and holding the left mouse button and moving the mouse or with the scrollbars.

Below you see the file history as a list of change comments. This list box has also scrollbars for panning.

Using the menus at the top of the dialog box you can

- write the file history into a file,
- search within the history.

Working with Models



If you click on a backup version in the box *Selectable Versions* ® in the dialog box *Open model* (see [page 5-45](#)) the selected version is displayed simultaneously in the preview box. You need not close the preview box.

If you want to load a specific backup version an independent file (with the same ID) with the same name, added with "@@n" (n = version number) will be extracted. This file will be opened.

- ☞ Click on *Read only* if you want to open the specified file in read mode.
- ☞ Click on *OK*. The model will be opened.

5.3.3 Reopening Models

If a model that has already been opened with write access by another user is opened in read mode, it can be opened and displayed in write mode after it has been closed by the other user. The changes made by the other user are then included. In addition, this read mode function can be used to incorporate changes that another user has made in write mode. This is important for concurrent access to models.

☞ Choose *Reopen model* in the *File* menu. The following dialog box is opened:



- ☞ Select the name of the model to be opened if it is not already highlighted.
- ☞ Click on *Read only* if the specified file is to be opened in read mode.
- ☞ Press *OK*. The model is reopened.



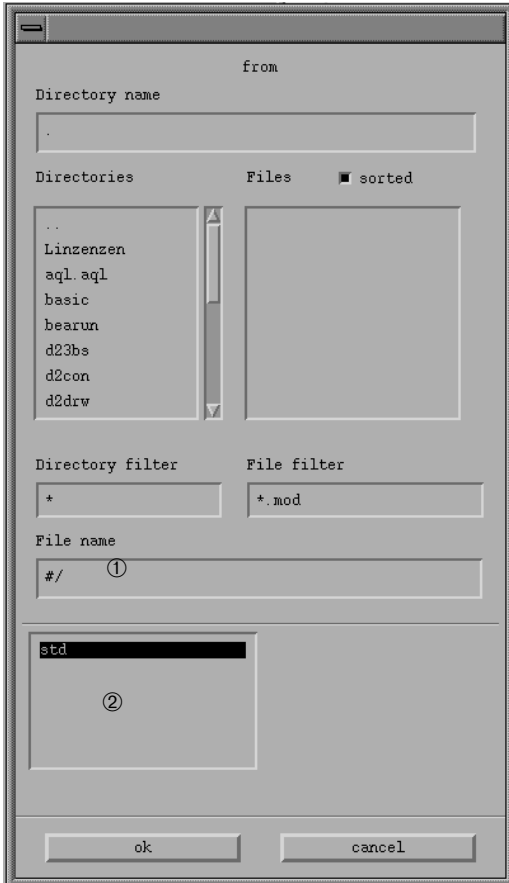
Ensure that only one user at a time has write access. The last write operation always acquires the access right.

Working with Models

5.3.4 Inserting Models

The contents of another model can be inserted into a model

☞ Choose *Add model* in the *File* menu. The following dialog box is opened:



- ☞ ① Enter the model name whose content you wish to insert by clicking in the *Files* box or via the keyboard in the *File name* box.
- ☞ ② Choose the name of the model to be expanded from the model box if it is not the active model.
- ☞ Click on *OK*. The contents of the source model will be inserted into the selected model.

5.3.5 Open models directly

--> Menü-Eintrag: ...Recently used files

--> Index !!

Working with Models

5.3.6 Closing Models

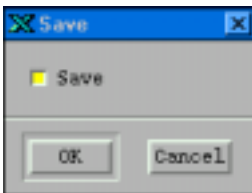
You can close opened models with or without saving.

To do this, proceed as follows:

☞ Choose *Close model* in the *File* menu. The model box is opened:



- ☞ Choose the name of the model to be closed if it is not already selected.
- ☞ Click on *Keep write access* if you want to keep the write access further on.
- ☞ Click on *OK*. An appropriate dialog box is opened depending on the save mode:
 - For models in write mode, the following dialog box is opened with model saving as default:



- ☞ Switch *Save* off if the model is not to be saved.
- ☞ Click on *OK*. The model will be closed with or without saving as specified.
 - For models in read mode, **EUKLID Design** asks via a dialog box whether the model is to be closed taking possible changes into account.
- ☞ Click on *Discard changes & Close* or *Cancel action*. The model will be closed as specified or the action will be cancelled.

5.3.7 Saving models

You may save the content of the active model in a file.

Proceed as follows:

☞ Choose *Save model* in the *File* menu. The model dialog box will be opened.



☞ Click on the name of the model to be saved in the list box if you do not want to save the active model.

☞ Click on *OK*.

Dependent on a possible existing file you must or can enter some additional information.

5.3.8 First time saving a model

The file dialog box will be opened (see [page 5-45](#)).

☞ Switch off *Keep write access* (default) if you want to release the write access.

☞ Enter the desired file name.

☞ Enter a descriptive change comment.

☞ Click on *OK*. The model will be saved in this file.

Working with Models

5.3.9 Saving a model again

If the file already exists you can save a new version.

The dialog box *Save model* will be opened.



- ☞ ① Switch off *Keep write access* (default) if you want to release the write access.
- ☞ Switch on *Keep last...version* if you want to keep the last backup version as additional backup. Otherwise it will be overwritten with each saving.
- ☞ ② Clicking on *More info* you can display the preview graphics and the file history of the last backup version.
- ☞ ③ After clicking on *Remove Backups* you can choose in a list box one or more backup versions which you do not want to keep any longer. Via double click on an

entry of the list box you may display the preview graphics and the file history of the selected backup version.



If you want to select multiple entries in the list box please hold the <Shift> or <Ctrl> key while clicking with the left mouse button (Windows NT behavior).

☞ ④ Enter a change comment.

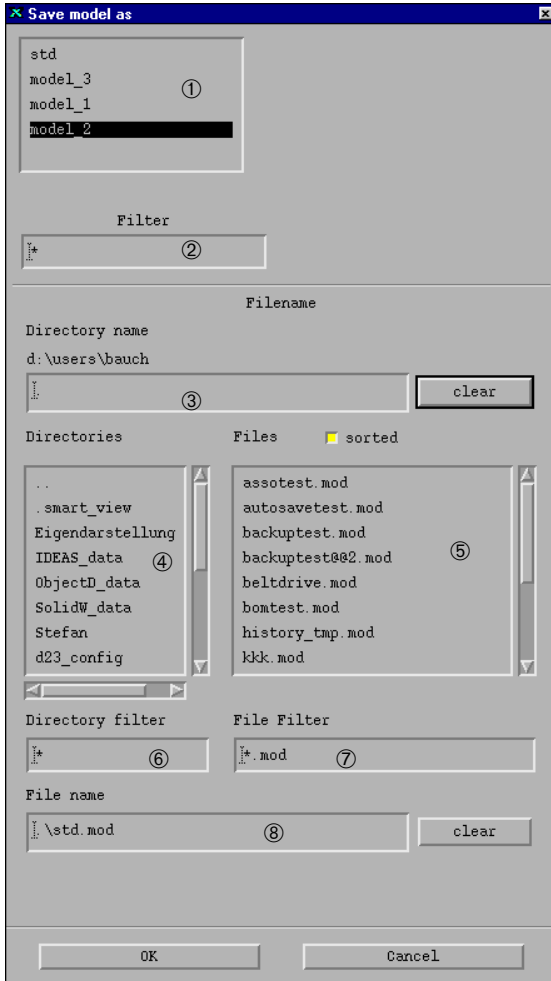
☞ Click on *OK*. The model will be saved in this file.

Working with Models

5.3.10 Saving Models in new files

You can save a model in any new file at any time.

☞ Choose *Save model as* in the *File* menu. The following file box is opened:



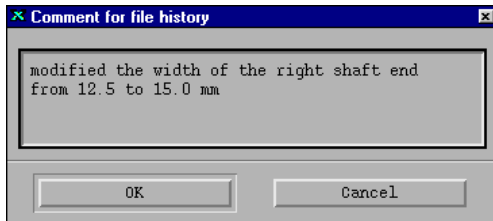
The dialog box defaults are as follows:

- ① Model choose box, the active model is highlighted
- ② Current working directory
- ③ List of subdirectories
- ④ List of the model files in the current directory
- ⑤ Filter expression for directories
- ⑥ Filter for model files (*.mod*)
- ⑦ Selected file name

Proceed as follows when saving:

- ☞ Select the requested model whether you don't want to save the active one.
- ☞ Select the requested directory from the *Directories* list box whether you don't want to save the file in the working directory. The name will be copied into the *Directory name* box.
- ☞ Enter the file name by clicking in the *Files* list box or via keyboard on the *File name* box.
- ☞ Click on *OK*.

The following dialog box will be opened to enter a change comment:



- ☞ Enter a comment to describe your last changes.
- ☞ Click on *OK*. The model will be saved.



Each model gets a unique identification. If you want to create a similar design you have to save the model with *Save model as*. A new model will be created with a new identification. Therefore you should not use *Save model as* for saving interim copies.

Working with Models

6 Introductory Example

In this chapter, a simple example – the design of a handwheel – is used to show how the system works. The design can be repeated step by step, allowing both beginners and experienced users to become quickly familiar with the operation of the system and how easy it is to use.



This description is based on the standard configuration of the system at the time of despatch. Any adaptations made by the client to actions within action groups are not taken into account.

During the design of the handwheel you will learn how to

- create and delete objects,
- use the fourth menu column for implicit design,
- invoke important actions and functions, e.g. the Sketcher and
- work with variables and formulae.

In the „*Editing Parameters*“ on page 6-17 section, you will practice how to

- change and edit parameters so you can produce design variants very easily.

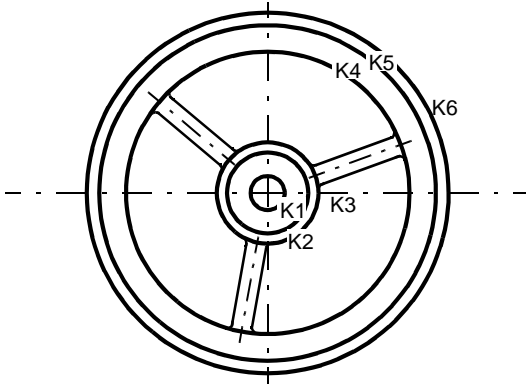
Introductory Example

A number of examples are supplied; these can be found in the directory **#/examples**.

The already designed handwheel that you will find in the file “rad2.mod” has three spokes, each 120 degrees apart.

The number of spokes should be capable of being changed so that variants of the handwheel can be created very rapidly. You may define as many spokes as you wish. The angular spacing between each spoke is determined automatically using a formula.

In the section *„Designing with the Sketcher“ on page 15-13* (see chapter Construction Aids), you will produce two views (front and side views) of a sample assembly.



Preparation

To prevent possible loss of data, make a backup copy of the original working example.

6.1 The Design Example

The handwheel is designed in the following steps:

- creation of the base circles C1 to C6.
- construction of a spoke by
drawing the spoke center line,
drawing lines parallel to the spoke center line,
adding fillets between the parallel lines and the circles C3 and C4,
deleting the parallel straight lines and redrawing exactly,
copying/mirroring the half spoke about the center line,
adjusting the center line.
- construction of further spokes (design variant) using a formula and a logical operation with the action “copy with rotary vector”.

Beginners will find it helpful to read through the section about the „*Sketcher*“ on page [15-2](#) first.

Introductory Example

6.2 Creating the Design

Creation of the base circles C1 to C6

1. Click the *circle_sketch* icon in the “circles” action group.



2. The system automatically selects the action *sketch a point* (in this case the center point of the circle). Click in the middle of the drawing area with the left mouse button and the system will draw the center point of the circle at this point.



3. Determine the radius of the circle by moving the mouse out from the center point and click the left mouse button. The first circle C1 is then drawn.



4. Construct the concentric circles C2 to C6 by always clicking first on the center point of C1 with the mouse and then defining the radii of C2 to C6 as described in 3 above.

Drawing the spoke center line

The spoke center line is created as an infinite line (line through a point with an angle and a length) and later redefined as a “line through 2 points”. The advantage of this is that the finished handwheel can be rotated by changing the angle.

1. Click on the *line_pointalength* icon in the “lines” action group.



2. Click on the *dotted mode* parameter and set the definition *dotted* in the dialog box.



3. Click the *line type* parameter and the *infinite* icon in column 4.



4. Click on the center point of the concentric circles in the drawing area as the starting point for the infinite line.



A temporary line is now drawn that can be positioned dynamically (though the center point) by moving the mouse.

5. Define the angle of the infinite line by moving the mouse into the drawing area and then clicking the left mouse button. Enter, for example, “20” degrees in the text field and press <ENTER> or select one of the fixed angles in column 4, e.g. 45°.



You have now created an infinite construction line.

Introductory Example

Drawing a line parallel to the spoke center line

The line parallel to the spoke center line (as part of the spoke) is again drawn using the Sketcher.

1. Click on the *line_sketch* icon.



2. Click in the drawing area on the infinite construction line already drawn. The symbol showing the possible action of the activated Sketcher – the parallel displacement of the straight line – and a temporary line now appear next to the cursor.
3. Now move the temporary line parallel to the spoke center line by the desired amount, towards the top left for instance, and click the left mouse button.
4. The parallel line is now drawn as an infinite line.

Adding fillet between the parallel lines and the circles C3 and C4

Proceed as follows to draw the connection between the parallel straight lines and the circles:

1. Click on the *circle_linecircle* icon in the “circles” action group.



All actions on the circle object create a view with a cross as standard. You can suppress this by clicking on the specific property and selecting the icon **without** a cross in column 4.

2. Select the parallel straight line by clicking on it with the mouse in the drawing area.



3. Then click on the circle C3 **above** (or to the left of) the parallel straight line. This position determines the position of the arc formed (internal or external fillet).



If, on the other hand, you click on the circle below (or to the right of) the parallel straight line, the arc will be drawn below the circle.

4. Now move the mouse until the radius of the arc is about right and click the left mouse button. Alternatively, enter an exact value in the text field (e.g. the value “3.5”) and press <ENTER>.



The system has now created an arc between the line and the circle and has also drawn two points, the tangential points.

Now construct the fillet between the parallel lines and the circle C4 in a similar manner.

Introductory Example

Deleting the parallel straight lines and redrawing

In the following step, the overlength parallel lines are first deleted and then redrawn as straight lines between the tangential points.

1. Change to delete mode by clicking on the *Delete* icon in the icon bar.
2. Click on the *line* icon in column 1. The system is now ready to delete individual lines. You must identify these by clicking on them in the drawing area.
3. Click on the parallel straight lines to delete them. If you accidentally delete the wrong line, you can reverse the action with UNDO.
4. Change to create mode by clicking on the *Create* icon in the icon bar and click on the “lines” action group in column 1. The Sketcher is thus active as the first action in the list.



5. Click on the first tangential point of the arc as the first point of the straight line.



6. Click on the second tangential point of the arc.



The construction of the precise connection between the arcs – instead of the parallel construction line – is now complete.

Copying/mirroring the half spoke about the center line

In the following procedure you will create the second half of the spoke using the *copy* function.

1. Click on the *vectors* action group.



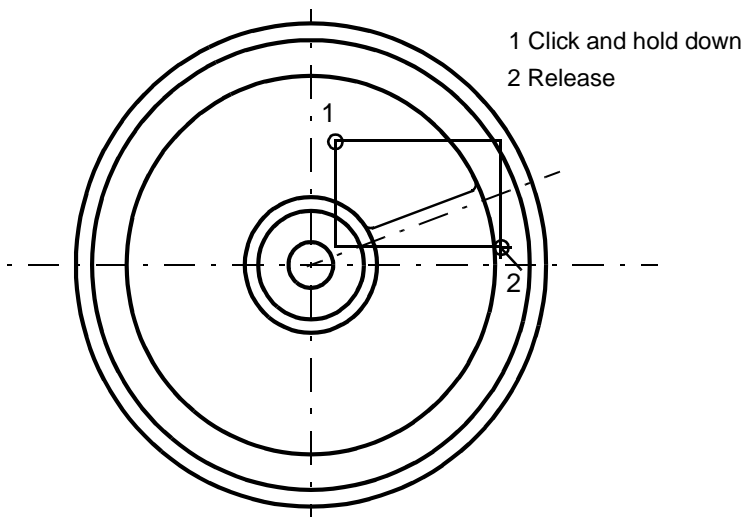
2. Click on the *group_copymirror* icon to activate the mirror function.



3. In the drawing area, select the center axis as the axis around which the spoke is to be mirrored.



4. The *selected set* icon in the menu field is highlighted. You can then select what is to be mirrored by dragging the mouse with the left button held down. The half spoke must be totally enclosed within the rectangle. The selected set is then highlighted.



Introductory Example



This procedure also mirrors the associated points. If you want to prevent this, the pop-up menu displayed when you click the right mouse button provides various options.

5. Terminate the procedure by clicking on the lower *arrow* icon, pressing <ENTER> or clicking on *confirm* in the pop-up menu displayed when you click the right mouse button. The upper half of the spoke is now mirrored about the center line.



Adjusting the center line

The arbitrary center line does not correspond exactly to our requirements and will be redrawn using the Sketcher in the following step.

1. Click on the *line* icon. This makes the Sketcher active again as the first action in the list.
2. Click on the *line_end* icon in column 3 and on the *centerline* icon in column 4.



3. Select the intersection between C3 and the arbitrary center line in the drawing area. As the first point of the new center line, the system displays the intersection between the circle and the straight line objects. You will recognize this by the appearance of the *intersection* symbol next to the cursor.
4. Move the mouse until the *line_pointpoint* symbol appears next to the cross hairs.

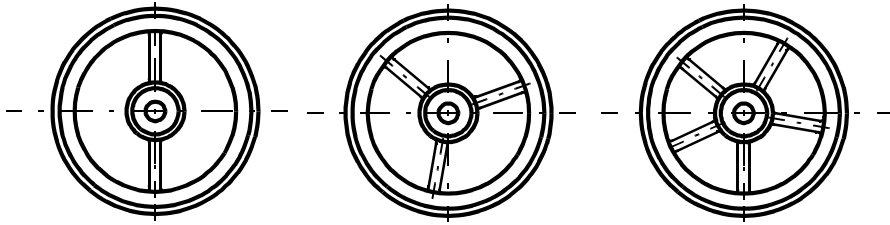


Press the right mouse key and click on *stop action* in the pop-up menu. The “line_pointpoint” action is retained. You can now insert a point at the second intersection between straight line and circle.

5. In the drawing area, select the the intersection between C4 and the arbitrary center line. The new, exact center line is now drawn in and you may delete the old one.
6. Change to delete mode by clicking on the *delete* icon in the icon bar.
7. Click on the *lines* icon and delete the infinite line in the drawing area by clicking on it.

6.3 Design of Further Spoke Variants

The advantage of **EUKLID Design** consists in the functionality that allows variants of a design to be created rapidly. According to the requirements, the handwheel can be designed, for instance, with two, three or five spokes.



Also, every individual variant has design properties, e.g. spoke thickness or diameter of drilling, that must each be modified in respect of each variant. The following explains the design of the spokes taking this functionality into account.

The design of additional spokes is performed in two stages:

- Definition of the necessary variables.
The following rules apply for the handwheel:
The number of spokes to be copied is “ $n-1$ ”, where n is the total number of spokes;
the angle of the copy vector is 360 degrees divided by the total number of spokes n .
- Copying the first spoke using variables and formulae.

Introductory Example

Defining the variables

First, the number of spokes is defined:

1. Click on the *variable* action group in create mode.



2. Click on the “real number” action.



3. If necessary, enter a value for the *text height* parameter by clicking on the icon and then entering a value.



4. Enter “3” from the keyboard as the first value and press <ENTER>.



5. Define the position in the drawing area where the variable is to appear.



If no suitable point exists, you can create this implicitly using column 4 as follows:
Click on the *point_sketch* icon in the fourth menu bar. Click on the desired position in the drawing area with the left mouse button. The point will be drawn in at this position and you can continue with the definition of the formula.

6. Enter the name of the variable *n* from the keyboard and press <ENTER>. You have now defined the first variable *number of spokes*.



In the second step, enter the number of copies; this has a relationship to the total number (total number of spokes minus 1).

7. Click on the *formulavvariable* action, enter the formula “n-1” from the keyboard and press <ENTER>.



A square icon with a rounded border containing the formula $E=mc^2$.

The system now expects an input for the unknown “n”.

8. Click on the icon for variables in the now opened *alternative* parameter and then select the variable in the drawing area by clicking on it with the mouse.



A square icon with a rounded border containing the text **VAR**.

9. Define the position at which the second variable is to appear in the drawing area.



A square icon with a rounded border containing the text **VAR** and a small black dot (cursor) to its left.

If no suitable position exists, you can create it implicitly using column 4.

10. Enter the variable name *copies* from the keyboard and press <ENTER>. You have now defined *number of copies* as the second variable.



A square icon with a rounded border containing the text **VAR=**.

In the third and last step, you will create the angle “alpha” of the copy vector, this being calculated according to the formula “alpha = 360/n”. The icons to be clicked are identical to those in the second step just described.

11. The *formulavvariable* action is still active. Enter the formula “360/n” from the keyboard and press <ENTER>.



A square icon with a rounded border containing the formula $E=mc^2$.

The system now expects an input for the unknown “n”.

12. Click on the icon for variables in the now opened *alternative* parameter and then select the variable in the drawing area by clicking on it with the mouse.



A square icon with a rounded border containing the text **VAR**.

Introductory Example

13. Define the position at which the third variable is to appear in the drawing area.



If no suitable point exists, you can create it implicitly using column 4.

14. Enter the variable name *alpha* from the keyboard and press <ENTER>. You have now defined the third variable *angle of copy vector*. The variable will be displayed at the specified position. Its value is 120.



Copying the original spoke

All the variables are now predefined and can be used in the construction of further spokes.

1. Activate the vectors action group.



2. Click on the *group_copy* icon.



3. Define the number of copies by clicking the “number” parameter and then click on the *variable* icon in column 4. Then select the *copies* variable in the drawing area by clicking the required number with the mouse.



4. The copied spokes must be drawn displaced by the copy angle. This is again done implicitly: first click on the *vector_rotator* icon in column 4 and then select the center point of the circle in the drawing area as the point of rotation.



5. The corresponding copy angle is again constructed using column 4 by clicking on the *variable as angle* icon and then selecting the copy angle “alpha” in the drawing area.



6. The *selected set* icon in the menu field for the amount selected is now highlighted and you can select the number to be copied (= spokes) in the drawing area by dragging the mouse with the left button held down. The spoke must be completely enclosed within the rectangle. The selection is now highlighted.



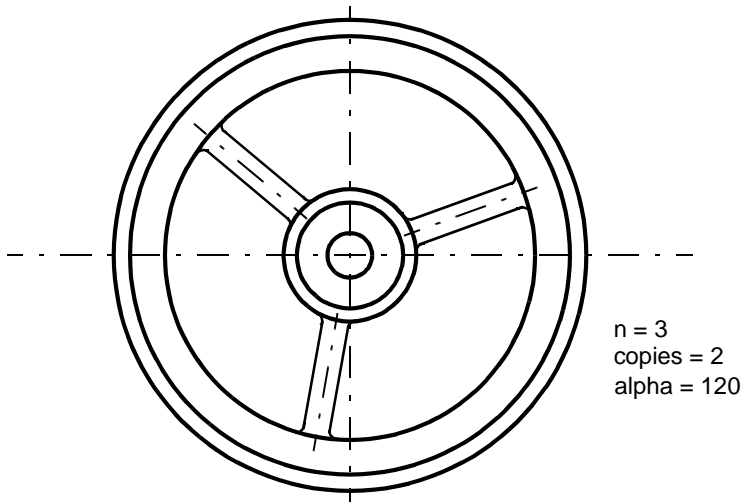
This procedure also mirrors the associated points. If you want to prevent this, the pop-up menu displayed when you click the right mouse button provides various options.

Introductory Example

7. Terminate the procedure by clicking on the lower *arrow* icon, pressing <ENTER> or clicking on *confirm* in the pop-up menu displayed when you click the right mouse button. The spoke is now copied twice about the center of the circle with the copy vector “alpha” 120 degrees.



The result should appear in the drawing area as shown below:



6.4 Editing Parameters

The power of **EUKLID Design** is especially useful when it comes to the specific modification of designs. You will be able to produce several variants with very little effort and without having to create a complex new design.

The following examples are shown:

- changing the radius of circle C3
- changing the radius of the fillet
- varying the number of spokes.

Changing the radius of circle C3

First, click on the *edit* icon in the icon bar to switch the system into the edit mode. Then perform the following steps:

1. Click on the *circle* icon.



2. Select C3, the circle to be edited, by clicking it with the mouse in the drawing area. Columns 2 and 3 are displayed in the menu area: the actions for the circle (circle with center point and radius) and their parameters.



3. Click on the *radius* icon in column 3, as this is to be changed.

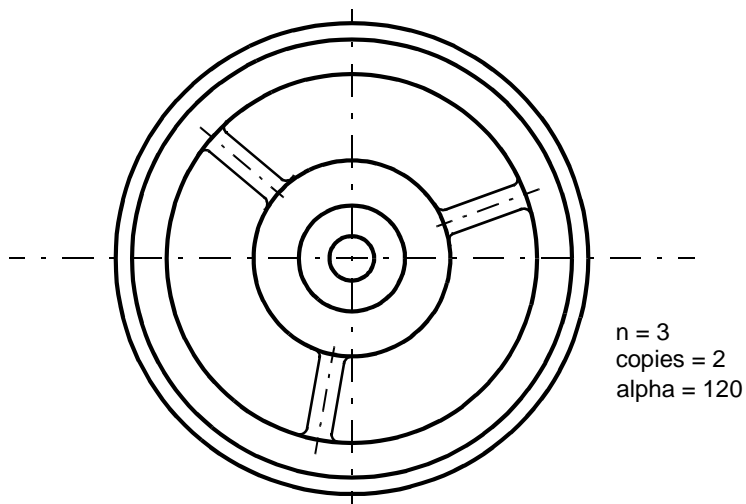


4. You may now define a new radius in two ways:
either from the keyboard, enter the new value in the text field and press <ENTER>
or change it dynamically in the drawing area. Use the mouse to move the cursor close to the circle C3, press the left mouse button and drag the mouse with the button held down. Release the left mouse button to fix the circle at the selected radius.



If you have accidentally selected too small or too large a radius, the message “circle with this radius cannot be created” appears. Correct your input and define a new radius as described above.

Introductory Example



The length of the spokes depends on the radius of C3: the circle is recalculated and displayed again after the new radius value is entered. The length of the spokes is then automatically adjusted to fit the new circle.

Changing the radius of the fillet

Switch the system into edit mode and activate the *circle* icon.



1. Select the fillet to be edited, i.e. the fillet or circle drawn between the straight line of the original spoke and C4 (not the mirrored fillet) by clicking on it with the mouse in the drawing area. Columns 2 and 3 are displayed in the menu area: the actions for the circle and its parameters.



2. Click on the *radius* icon in column 3.

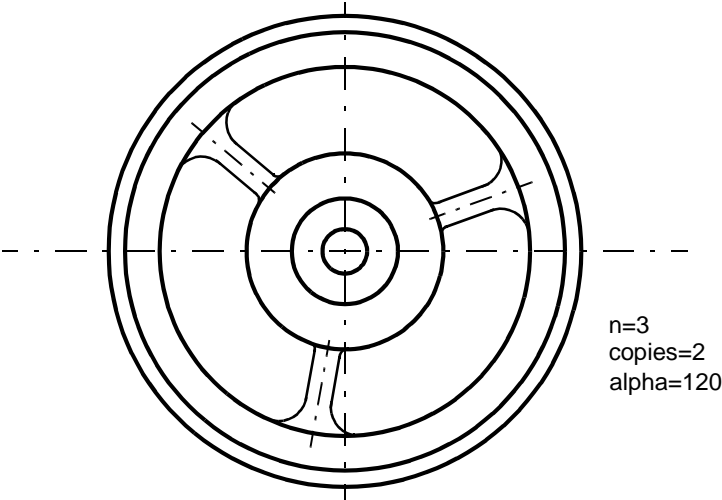


3. You may now define a new radius in two ways:
either from the keyboard, enter the new value in the text field and press <ENTER>
or change it dynamically in the drawing area. Use the mouse to move the cursor close to the circle C3, press the left mouse button and drag the mouse with the button held down. Release the left mouse button to fix the circle at the selected radius.



If you have accidentally selected too small or too large a radius, the message “circle with this radius cannot be created” appears. Correct your input and define a new fillet as described above.

Introductory Example



Varying the number of spokes

To vary the number of spokes, proceed as follows:

First, click on the *edit* icon in the icon bar to switch the system into edit mode. Then perform the following steps:

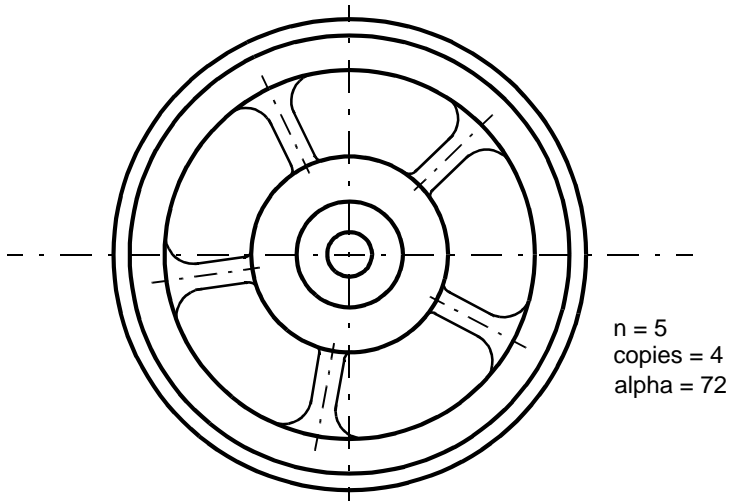
1. Click on the *variable* icon.



2. Select the variable to be edited in the drawing area (click on $n = 3$). Columns 2 and 3 are displayed again in the menu area with the associated action and parameters.
3. Click on the *real_number* icon.



4. Enter the new value (e.g. "5") in the text field from the keyboard and press <ENTER>. The system calculates the new view and you can check immediately whether the design is still satisfactory with these parameters.



Introductory Example

6.5 Enlarge Window

Sometimes it is necessary to enlarge a portion of a drawing if, for example, elements in the drawing area need to be selected or if the way the model is displayed in the drawing area is not sufficient for close detailed design. There are two ways of enlarging the window (zoom):

- from the popup menu displayed when the right mouse button is clicked: *enlarge window*
- using *enlarge window* on the second menu page.

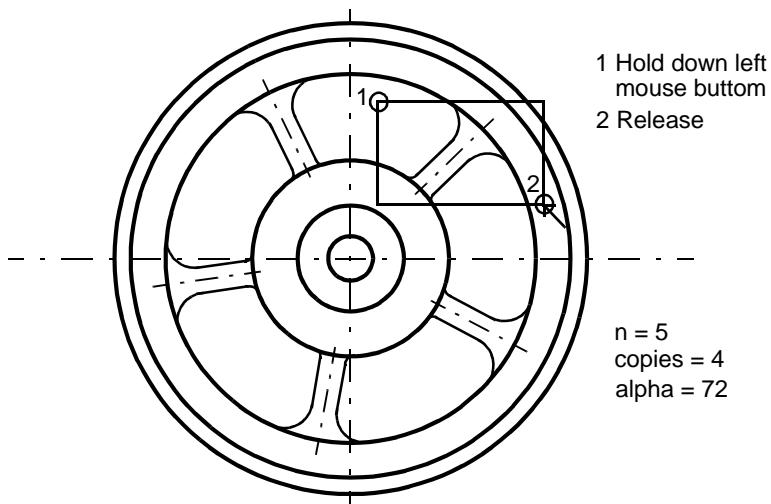
Proceed as follows:

1. With the cursor in the drawing area, press and hold the right mouse button. Select *enlarge window* from the pop-up menu and release the mouse button. If the system is in another operating mode, you will first have to access the pulldown menu from the “viewport functions” field.

Alternatively, activate the second menu page and click on the icons *viewing* and *view_zoom_in*.

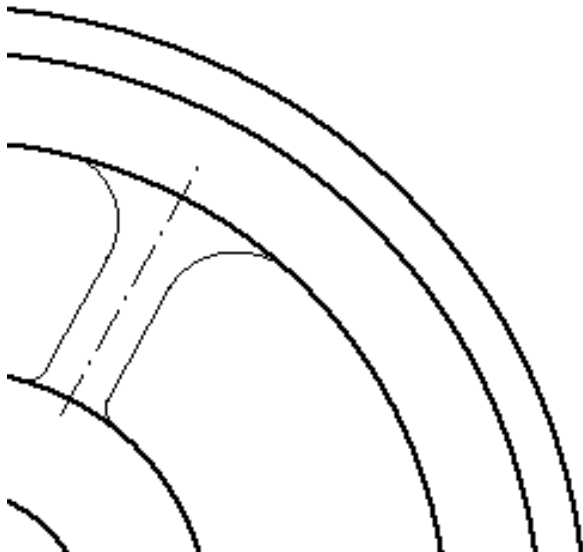


2. The portion of the window in the drawing area to be enlarged can be defined by dragging the cursor with the left mouse button pressed to form a rectangle around it, as when making a selection during the copying procedure.



Introductory Example

The system now displays an enlarged view of the selected portion in the drawing area:



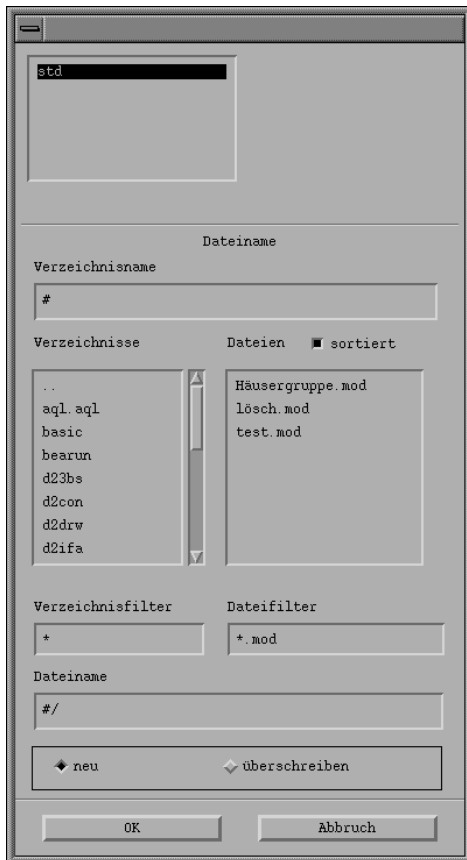
This action can be repeated as often as required, or reversed with *reduce window*.

6.6 Saving the Example Model

You should save your work at regular intervals to prevent possible loss of data. Save the models to your data media by entering a path name.

- To save, click on *file* in the menu bar and then on *save model as*

The following dialog box will be displayed:



Introductory Example

Enter

- the “directory” and
- the “file name”

by clicking on the corresponding input field and entering the path or file name respectively from the keyboard. Then specify whether you wish to

- save the model as a new file by clicking on *new* or
- overwrite the old file by clicking on *overwrite*.

If this model was saved earlier under this name, you can save it again by clicking on *save model ...* or using the key combination <CTRL>+<s>.

7 Creation of Objects

Objects are created in *Create* mode using the menu area. **EUKLID Design** recognizes two different design techniques:

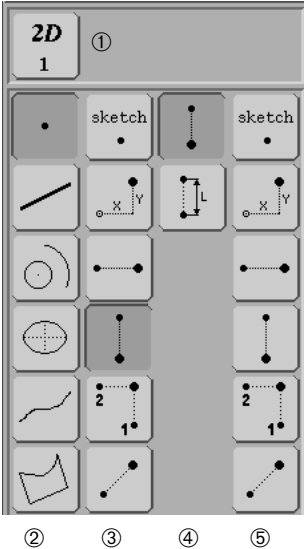
explicit construction	creation of objects by actions the parameter objects of which exist
implicit construction	creation of parameter objects while furnishing parameters to actions

7.1 Explicit Construction

In *Create* mode, you can generate objects in stages, as follows:

- ☞ Select the menu in the special menu area ①.
- ☞ Select the action group in the first menu column ②.
- ☞ Select the action in the second menu column ③.
- ☞ Furnish the properties and parameters in the third menu column ④ with objects in the drawing area, values and/or enumeration values in the menu column ⑤.

Creation of Objects



7.1.1 Selection of Menu

You choose the menu by selecting the menu icon in the menu extension area. The default selection is the first 2D menu. In the first menu column, the corresponding action groups, in the second menu column the actions of the default first action group and in the third menu column the properties and parameters of the default first action are displayed.

The menu selected remains effective until you select a different one. If you switch to *Edit* or *Delete* mode the 2D menu is displayed. A switch to *Temp* mode will only cause a digression.

7.1.2 Selection of Action Group

You choose the action group by selecting the appropriate icon in the first menu column. The default selection is the first action group in the list. The icon selected is highlighted and the corresponding actions are displayed in the second menu column.

The action group selected remains effective until you select a different one. If you switch to *Edit* or *Delete* mode the selection is maintained in form of an object group. A switch to *Temp* mode will only cause a digression.

7.1.3 Selection of Action

You define the action by selecting the appropriate icon in the second menu column. The default selection is the first action in the list. The icon selected is highlighted and the corresponding properties and parameters are displayed in the third column.

The action selected remains effective until a different action group or action is selected.

Creation of Objects

7.1.4 Input of Properties and Parameters

Once the action has been selected, the properties and parameters, which are either mandatory or optional, are displayed in the third menu column. Mandatory properties and parameters are requested by **EUKLID Design** in the third menu column from top to bottom. The parameter to be furnished is indicated by highlighting the corresponding icon.

In case of the explicit construction, the following rules apply to the input:

Object

The actions for implicit construction are displayed in the fourth menu column.

☞ Identify an existing object via the selection mechanism within the drawing area.

Value

The fourth menu column remains empty.

☞ Input the desired value either via the keyboard in the text area or opened dialog box, or as a sketch point via the mouse in the drawing area. In special cases, you can enter successive parameters into a dialog box.

Enumeration Value

The possible inputs are displayed in the fourth column.

☞ Input the desired enumeration value either via mouse in the fourth menu column by selecting the icon of the desired enumeration value or via the keyboard in the text area or in an opened dialog box.

After having furnished all mandatory parameters with values, the action selected is executed and the object is created.

7.2 Implicit Construction

You can recursively create parameter objects, which do not exist at the time of the result object's creation, via the fourth menu column during the creation action of the result object. The depth of recursivity is unrestricted.

If an parameter object is required, the actions for the respective object type are displayed in the fourth menu column – as for the explicit construction in the second menu column. In addition, the actions of the basic object types (length, angle and string) are displayed.

If you select an action icon in the fourth menu column **EUKLID Design** will digress from the creation action in hand to create the parameter object. The first menu column shows the original action group. The second, third and fourth menu columns are changed in accordance with the current action. Any further implicit creation of a parameter object for the requested parameter object of the original action increments the "interrupt level" by one. The action last selected is thus the first to be processed, then the last but one and so on, until the level of the original action is reached.

It is thus theoretically possible to start a design from a bill of material entry and to conclude with the construction of the very first point. The limits to this facility are set not by the system, but by the ability to comprehend the actions.

Example

A text whose reference point is not yet defined, is to be created.

- ☞ Select the icon of the action group *text* in the first menu column.
- ☞ Select the icon of the action *text_absolute* in the second menu column.
- ☞ Furnish the parameter *string*.
- ☞ Select the icon of the desired action in the fourth menu column for the implicit construction of the *reference point P* parameter. The second, third and fourth menu columns are changed in accordance with the current action.
- ☞ Furnish the required parameters. The text is created.

Parameters which in addition to a object also require selection of a position with the mouse to define their position (e.g. circle for tangent position) require subsequent specification of the position on the implicitly created object.

Creation of Objects

7.3 Object Types

In addition to classic CAD elements, **EUKLID Design** provides further design elements as objects, such as basic objects like length or variables. Moreover, **EUKLID Design** offers the facility to define own objects. A full list of object types, with their actions and parameters, may be found in Volume *Functions*.

7.3.1 Basic Objects

A basic object is required to store certain parameters of explicit objects in the data structure and to allow, e.g. to specify the source of the data. Basic objects therefore only become visible when contained as parameters in another object (in the third menu column with the actions in fourth menu column). Basic objects are only displayed, for example, when dependencies need to be made visible during explicit dimension input, and then only temporarily.

Basic objects may therefore only be created by means of the fourth menu column. The following basic objects are available:

- angle
- column_name – table column name
- coord – world coordinate system
- length
- nilprim – virtual object
- number – real number
- posball – balloon position
- posmeas – measure position
- posttext – tolerance
- prop – distance ratio
- rawval – roughness
- string
- view – View
- wcoord – coordinate system
- weldfork

7.3.1.1 angle

The object type *angle* is used wherever angles in degrees occur as parameters of other objects. The value is shown as a double precision real point number (see Appendix).

Example

Parameter *r* of the action *variable_ofangle*.

7.3.1.2 column_name – Table Column Name

The object type *column_name* is used as parameter of the action *tab_sub*.

Example

Parameter *col* of the action *tab_sub*

7.3.1.3 coord – Coordinates

The object type *coord* is used wherever world coordinates are used as parameters of other objects.

7.3.1.4 length

The object type *length* is used wherever lengths in millimeters or inches occur as parameters of other objects. The value is shown as a double precision real point number (see Appendix).

Example

Parameters *dx* and *dy* of the action *point_relative*.

Creation of Objects

7.3.1.5 nilprim – Virtual Object

The object type *nilprim* is used as parameter of the action *tab_instance_row*.

Example

Parameter *refer_nil* of the action *tab_instance_row*

7.3.1.6 number – Real Number

The object type *number* is used wherever real numbers are used as parameters of other objects.

Example

Parameter *r* of the action *angle_scalprod*

7.3.1.7 posball – Balloon Position

The object type *posball* is used as parameter of the actions *symbol_balloon* and *symbol_balcopy*.

Example

Parameter *posballoon* of the action *synbol_balloon*

7.3.1.8 posmeas – Measure Position

The object type *posmeas* is used as parameter to position the measure of the action group *measure*.

Example

Parameter *pos* of the action *measure_relative*

7.3.1.9 posttext – Tolerance

The object type *posttext* is used as parameter of the action group *measure*.

Example

Parameter *post* of the action *measure_relative*

7.3.1.10 prop – Distance Ratio

The object type *prop* is used wherever distance ratios are used as parameters of other objects.

Example

Parameter *p* of the action *point_pointbetween*

7.3.1.11 rawval – Roughness

The object type *rawval* is used as parameters of the action *symbol_raw*.

Example

Parameter *a* of the action *symbol_raw*

Creation of Objects

7.3.1.12 string

The object type *string* is used wherever character strings are used as parameters of other objects.

Example

Parameter *st* of the action *text_absolute*

Strings can be implicitly constructed:

- From a keyboard input
- From the objects *length*, *angle* and *text* (not block text)
- By concatenating several strings or parts of strings
- From a string in a table or in a bill of material balloon

7.3.1.13 view

The object type *string* is used wherever views are used as parameters of i.e. of view-ports.

Example

Parameter *view* of the action *display_open*

7.3.1.14 wcoord – Coordinate System

The object type *wcoord* is used wherever coordinates are used as parameters of other objects.

Example

Parameter *txpos* of the action *symbol_comment*

7.3.1.15 weldfork – Weld Symbol Configuration

The object type *weldfork* is used as parameters of the action *symbol_weld*.

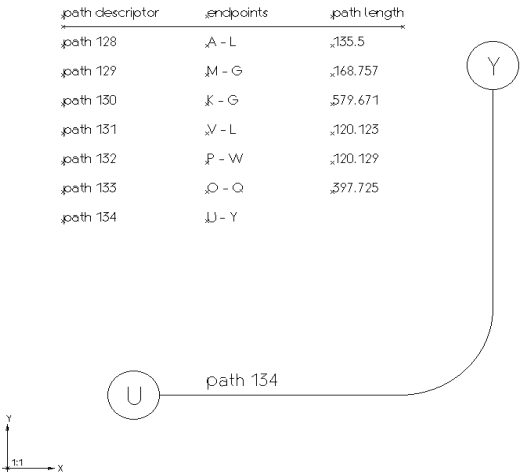
Example

Parameter *fork* of the action *symbol_weld*

Example

The following example illustrates how basic objects may be created implicitly.

The path between U and Y is to be added to an existing list of paths. Paths are represented in the model as contours. The task at hand is to insert a path as text in a specific column and line. The text is to be read from a length (string from length), which in turn derives its value from the length of a contour (length from contour length). This is a dependent construction, in which path changes automatically cause the list to be updated.



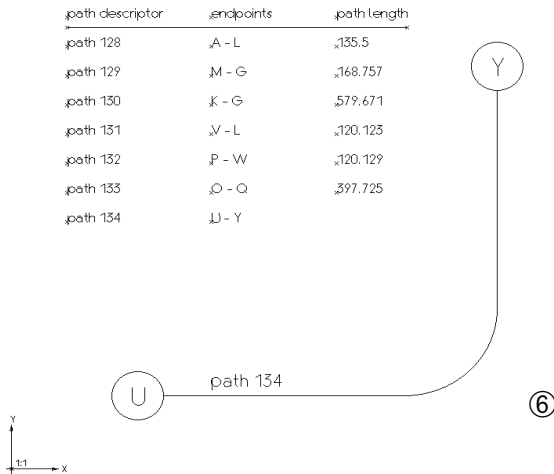
1. Choose the action group *text* in the first menu column in *Create* mode.
The action *horizontal_text* and the parameter *String* are set by the system in the standard configuration.
2. Choose the action *string_oflength* for parameter *string* in the fourth menu column.
The system interrupts from the action *text_absolute* in favor of the action *string_oflength*. The system then calls for the parameter *length*.

Creation of Objects

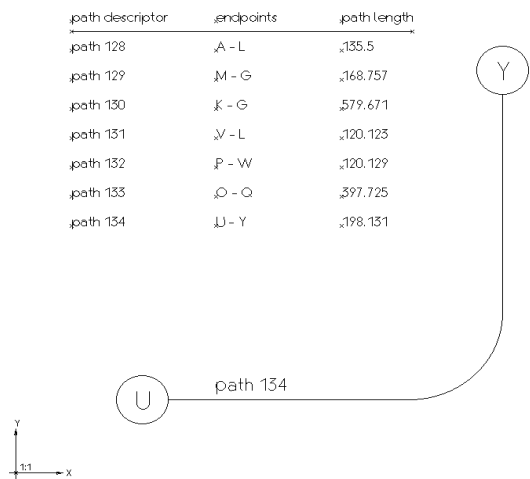
3. Choose the action *length_offline* in the fourth menu column.

The system interrupts from the action *text_absolute* in favor of the action *string_offlength*. The system then calls for the outstanding parameters.

4. Press the icon of the alternative parameter in the third menu column.
5. Select the icon of the parameter *contour* in the third menu column.
6. Identify contour in drawing area.



The system returns to the original action *text_absolute* and calls for outstanding parameters. All parameters for the original action *text_absolute* are now available and the text may now be created.



Creation of Objects

7.3.2 Geometry Objects

This heading covers all the classical CAD elements which are used in the creation of drawings and which are not specific to a particular sector of application:

Geometry objects are:



point



line



circle and/or arc



ellipse and elliptical curve



B spline

Each geometric object is displayed by a dedicated routine according to its shape, and may be identified within the drawing area or by name.

All geometry objects, with the exception of the point, possess direction, or orientation. The direction of the object is predefined (e.g. clockwise), but some may be influenced by the user. In the case of a line between two points, for example, the direction is dependent on the point sequence when the parameters are input. The calculations of the object direction are carried out by the system.

7.3.3 Contours

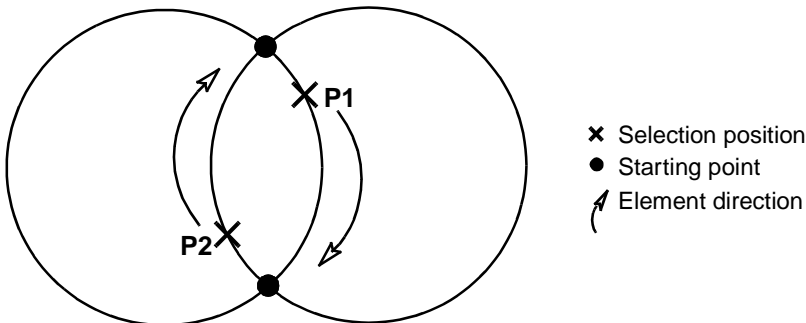


You should use the object type *contour* when the figures or parts of them are to be processed as a separate set. Once this set has been formed, you may perform various operations, such as rounding, on it. You do not need to carry out separately for each object.

Typical applications of contours are:

- Simple mensuration of the total length of a selection of individual objects (e.g. neutral axis of a bent sheet metal component not in the flat)
- Creation of complex roundings (e.g. cutting contour optimization in toolmaking)
- Complex crosshatchings (e.g. definition of sectioned planes by combining several individual contours)
- Definition of loci (e.g. link of a chain on part circumference of a sprocket and its tangents)
- Creation of parallel contours (e.g. routing contours in toolmaking)

You may use the objects *line*, *circle*, *arc* and *ellipse* to delineate a contour. You can define the contour elements individually or create contours automatically.



- × Selection position
- Starting point
- ↻ Element direction

Creation of Objects

Partial contours are automatically formed if the sequence of the selected objects or gaps between them does not allow the formation of a contiguous contour. As many partial contours as required are formed until the above conditions are no longer met.

The contour is displayed as a replica of the original objects from which it was created. The parallel contour is the exception. If the delineating elements are deleted, the contour remains visible, and vice versa. The contour is dependent on the delineating objects from which it is composed and adjusts itself automatically whenever one of these is changed. You may identify contours by the elements delineating them in the drawing area or by name.

7.3.4 Planes



Via the object type *plane* and the associated actions, you are able to

- crosshatch sections
- color selected areas
- mask out objects located underneath

You may use the objects *line*, *circle*, *arc* and *ellipse* to delineate an area. Apart from these external boundaries, islands inside areas can be used for delineation. These islands may also contain islands in their turn.

You can individually select or automatically create delineating elements.

The contour of an plane must be a closed contour. You must create planes delineated by elements which do not produce a closed contour using the action *plane_onconstruction* or you must close them using auxiliary elements which are later to be deleted.

Planes are dependent on the elements delineating them and adjust themselves automatically whenever one of these is changed.

The plane has two colors:

Object color hatching line color

Filling color Selecting a plane color only affects the filling color if the planes are filled

Crosshatching is displayed as parallel lines at a defined distance apart and at a defined angle. The crosshatching lines are not individual objects, but are the graphic representation of the object *crosshatching*.

You may identify crosshatchings by the elements delineating them in the drawing area or by name.

Creation of Objects

7.3.5 Text



The object type *Text* is used to incorporate text into a model.

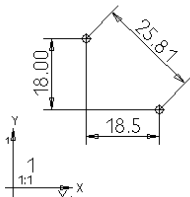
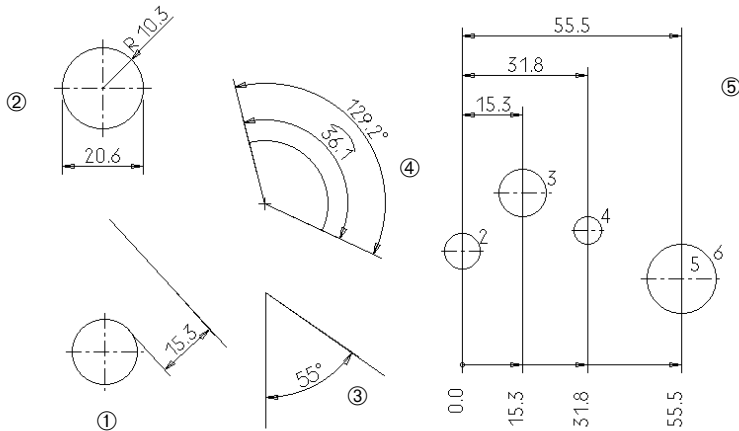
Text is displayed in characters of the previously defined character set. The text is output on the plotter in the same way as displayed on the screen.

Texts may be identified either by selection within the drawing area or directly by name.

7.3.6 Measures



Dimensioning under **EUKLID Design** conforms to DIN standard 406 and/or the corresponding DIN/ISO standards.



NR	X-Coord	Y-Coord	Ø
1	0.00	0.00	
2	117	109	9.1
3	132.25	123.75	12.09
4	148.75	114.25	6.96
5	172.500	102.000	
6	172.50	102.00	17.53

The following types of dimensions may be applied:

- ① Distance between objects of different types (e.g. point, line and circle)
- ② Radius and diameter
- ③ Angle

Creation of Objects

- ④ Length of arc (radian)
- ⑤ Reference point dimensions as
 - Cumulative dimensions
 - Chained dimensions
- ⑥ Coordinate tables

Dimensions are displayed with the help of dimension lines and, where necessary, auxiliary dimension lines. These may be identified either by selection within the drawing area or directly by name. Auxiliary dimension lines may be partly or wholly obscured by the object to which the dimensions have been applied.

7.3.7 Symbols

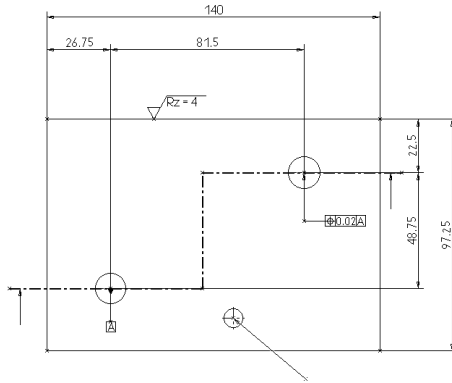


Symbols in classical technical drawing terminology are created by standard drawing techniques, usually with the aid of a stencil, and serve primarily to provide processing information and to enhance the readability of the drawing. Symbols are used to represent the following elements:

- Reference lines of various types
- Geometric tolerances
- Surface characteristics
- Taper characteristics (e.g. of cones)
- Section lines
- Bill of material balloons
- Welding characteristics according to DIN 1912

The individual geometrical elements of which a symbol is composed are not accessible as separate objects.

Each symbol is displayed by a dedicated routine according to its shape, and may be identified within the drawing area or by name.



drilled together with part 4



Creation of Objects

7.3.8 Copy Vectors



Copy vectors are the transformation rule by means of which you create a copy and a duplicate of a source geometry. There are corresponding actions for the various translation and rotation options.

Copy vectors are represented by arrows, which display symbolically the transformation characteristics of the vector concerned, and are identified in the drawing area by these arrows or by name.



Copy vectors between two coordinate systems cause also the transformation of the objects copied; the transformation also takes account of differences in the characteristics of the two coordinate systems (e.g. angle of orientation, scale etc). The dependence on the original thus remains unchanged. A useful application is the detailing of drawings.

7.3.9 Coordinate Systems



The coordinate system is the overall governing factor of all constructional elements. When starting **EUKLID Design** a coordinate system is automatically moved into the otherwise empty drawing area. You have the option of creating any number of coordinate systems and using them as a basis for the design.

A coordinate system is characterized by the following:

- geometric location within the drawing area
- scale or reference coordinate system
- differential scale along each axis (optional)
- angle of rotation



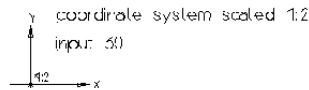
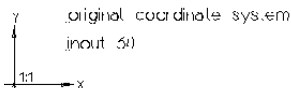
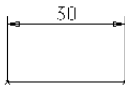
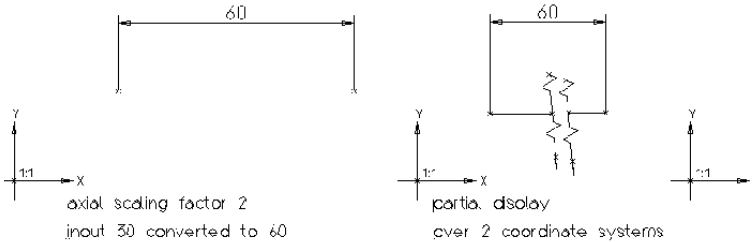
There are significant differences between axial scale and scaling factor. If an axial scale is defined, the data input is converted to this scale. If a scaling factor is used, the data is stored in the data structure as it was input. Only the display is scaled.

Useful applications of coordinate systems are:

- *Overall change of scale:* If you should discover during design work that a different scale would be more suitable, it is merely necessary to change the original coordinate system.
- *Mixed scales in one model:* Examples are: details in combination with copy vectors, and detailing of more than one component in a model to different scales.
- *Local coordinate calculation:* It is possible, for example, to apply a different coordinate system to each component, which will then be fully independent as to scale, location and orientation (these are known as component coordinates).
- *Partial display of large components:* Geometry data is stored in the data structure literally. It is thus possible to display a component in part without the need for manual correction when dimensioning or deriving NC data.
- *Distorted display:* The differential scaling of block diagrams for example.
- As default axes for area calculations

Creation of Objects

The coordinate system is represented graphically by two arrows aligned to the two axes. Coordinate systems may be identified within the drawing area or by name.



It is not possible to refer to the coordinate system directly. Therefore, whenever a coordinate system is created, a point of origin is also created. This point is actually the progenitor of all objects referring to the coordinate system concerned. If there is a point beneath that point, it is set to *deleted*.



Via the action called *point_origin*, you can reproduce the progenitor.

7.3.9.1 Working between Coordinate Systems

Generally speaking, constructional relations etc. are only formed within one coordinate system. In certain applications (such as subassembly construction), however, it may be necessary to establish references between a number of coordinate systems (such as dimensions between parts) or to reattach objects from one coordinate system to another (for example to create partial images). In such cases, note the following:

- All objects which relate to the point at the corresponding origin of co-ordinates via their relation chain are dependent on one co-ordinate system.
- The objects obtain their transformation from the coordinate system to which they are assigned (e.g. interpretation of angle or axis references, scale etc.). This has the result that when objects are transferred to a different coordinate system (e.g. separation, duplication, forming part of a model etc.) the graphical presentation changes if the transformation rules of the new coordinate system differ from those of the original system. This affects all objects which retain their mutual constructional relations.
- Constructional relations are formed between coordinate systems if the parameters of a particular object belong to different coordinate systems. In such a case it is not always clear to which coordinate system the newly created object belongs, nor how the parameters from other coordinate systems are to be interpreted (see notes on next page). References between coordinate systems should therefore only be created if they are carefully thought out, and should be limited to the minimum necessary.

7.3.9.2 Behavior of Objects in Relations between Coordinate Systems

- **Copy vectors** belong to both coordinate systems and may result in a delta transformation to the copied or duplicated set if there are differences between the original system (start point) and target system (end point). To maintain transparency, the copy vectors should if necessary lie directly between the origins of the coordinate systems.
- **Dimensions** are assigned the positional transformation from the coordinate system to which the object that is assigned as the first parameter (e.g. first point) belongs. The coordinate systems of the objects which are assigned to the subsequent parameters are also taken into account in determining the dimensions. The dimensions that are determined will only be useful, however, if all coordinate systems involved have the same scale.
Implausible combinations are normally rejected with an error message.

Creation of Objects



Other objects obtain the transformation from the object that was entered for the first parameter, and interpret the data of the subsequent parameters using the corresponding transformation. This leads to unexpected results in most cases, and should therefore be avoided.

7.3.9.3 Recommendations for Application

The following should be noted in practical applications:

- Always use **coordinate systems** purposefully and transparently; if work is shared between a number of people, it may well make sense to stipulate application conventions.
Only use coordinate systems to achieve a required graphical result if the specific effect of the coordinate system is actually necessary in that instance; for example, give preference to implement UDOs with a parametric angular position with the aid of constructional relations (such as *line with angle*) instead of with rotatable coordinate systems.
- Always set up **UDAs/UDOs** without coordinate systems if extensive environmental references are established in your application (for example in the case of form elements).
- Always form **dimension** relations between coordinate systems which ultimately derive their scaling from an identical reference coordinate system; differences in scale result in incorrect dimensions.
- **Functions which change the actions of objects** (such as separation, save part of a model etc.) should only be executed between coordinate systems with the same transformation rules, unless you deliberately intend to achieve specific transformations.
- If coordinate systems have different scales, limit relations between them to **copy vectors** only.

7.3.10 Variables

VAR

The object type *variable* provides functions outside the scope of CAD proper. Used in conjunction with user defined objects and actions, this results in a range of options for standardized calculations, the calculation of specification data etc, which may not only be derived from the CAD geometry but also be used to influence it (see main chapter „*Standardization*“ on page 13-1).

The principal applications of the object type *variable* are:

- *Storage of key design data in easily comprehensible form:* These may then be referred to during design work in such a way that a change to the data causes the design to be adjusted accordingly. If, for example, a gasoline engine is being designed, the key design data will include the power output, nominal rpm etc.
- *Calculations within the model,* including complex mensuration with nested variables and mathematical functions. This facility concerns other design data derived from the key design data, e.g. for the gasoline engine this would include the torque, swept volume, bore, stroke etc.
- *Reproduction of a table of figures within the drawing area,* with the option of translating the resulting figures to a geometric display: The use of calculated tables makes it easier to comprehend design contexts and thus to arrive at an optimum solution to the problem in hand.
- *Memory function for temporary "design figures":* Such facilities serve as a sort of designer's scratch pad and are allowed to be deleted after use.
- Display in numeric form of geometrical data which otherwise would be difficult to estimate.

It would be possible, for example, to relate the actual swept volume and the actual combustion volume as derived from the geometry. The result is the compression ratio, of which any changes in value may be monitored as the geometry is modified.

A variable is represented by its

- Value
- Locating point
- Name
- Text size (optional)

Creation of Objects

The variable is written left-justified from the locating point which you may define either by identifying in the drawing area or by implicit construction. The variable is displayed in the following form:

`name = value`

In *Edit* mode, the algebraic formula by which the variable resulted from calculations is displayed in brackets following the value itself.

Example

`Product = 386 (v1 * v2)`



You cannot identify a variable as object type *text*.

7.3.11 Groups



An object group is a grouping of logically related objects, and may be regarded as a "perpetually selected object set" (see main chapter *System Handling*, chapter „[Object Groups](#)“ on page 4-35).

An object group is defined by selecting in a rectangle and may include both ordinary objects and existing object groups. A name is always assigned to the new object group.

Object groups may be selected by name, augmented by other object groups or individual objects and highlighted on the screen.

With the object groups facility related geometries may be assigned a name, varied as to contents and displayed on the screen.

Each group is displayed by a dedicated routine according to its shape, and may be identified within the drawing area or by name.

Creation of Objects

7.3.12 Tables



The object type *Table* may be used to store frequently used parameter values for repeated application (chapter „[Tables“ on page 13-117](#)). The table may be used for complex standard design elements of the geometry and technical specification.

The data to be held in the table, may be of any type and content, and may be invoked in almost any function. The object type *table index* is used to extract the data.

Tables are identified by name.

7.3.13 Table Index



The object type *table index* is used to extract data from a table (section „[Creating an Index“ on page 13-136](#)) The index defines which row of the table is to be used.

By editing the index, it is possible to extract the corresponding data and carry out complex geometric variations.

Indexes are identified by name.

7.3.14 User Defined Objects



A user defined object (UDO) is a user defined set of objects (e.g. points, lines, variables) which may represent complex geometry, design logic and arithmetical calculations („[Working with Variables“ on page 13-91](#)). You can fully parametrize it (e.g. length, dimensions etc.).

You may incorporate an UDO in any model, applying in the required values to the properties just the same as with a normal object.

UDOs are displayed by a dedicated routine according to the shape of each contained object, and may be identified within the drawing area or by name.

Creation of Objects

8 Updating Objects

The following can be updated using the update facilities:

- Properties and parameters
- Actions
- Names (see section „[Changing of Names](#)“ on page 4-20)

Objects are updated in *Edit* mode using the menu area. The last object created can also be edited using the menu command *Edit last object* in the *Undo* menu, without having to switch to *edit* mode.



Sketched objects can also be updated via the *quick edit* menu command in the drawing area popup menu (see main chapter Construction Aids, chapter „[Sketcher](#)“ on page 15-2).

The following table gives an overview of the differences between these procedures.

	<i>Edit via Edit Icon</i>	<i>Quick Edit via Undo Menu</i>	<i>Quick Edit via Drawing Area Popup Menu</i>
Object	any object	last created object	sketched object
Selection	<ul style="list-style-type: none"> ☞ Switch to <i>Edit</i> mode ☞ Click object group icon in menu column 1 ☞ Select object in drawing area 	<ul style="list-style-type: none"> ☞ Choose the menu command <i>Edit last object</i> in <i>Undo</i> menu or Press key <CTRL>+<I> 	<ul style="list-style-type: none"> ☞ Position the cursor in the snap area of the object ☞ Choose the menu command <i>quick edit</i> in the drawing area popup menu

In *Edit* mode, you are able to obtain a precise view of the history of any drawing by descending through the data structure. From the design data, you can obtain a complete picture of how the component was designed.

Updating Objects

8.1 Examination of Design Data Structures

The ***EUKLID Design*** models contain the entire development history of the design. You may examine this for each object in *Edit* mode.

Each design consists of a hierarchy of objects related to one another. Each object is defined by one or more parameters corresponding to the underlying action. These parameters may be objects (either explicit or implicit), values or enumeration values.

The first stage of a descent through the data structures is to inquire about the interdependencies of objects and actions. It is thus possible to descend from one identified object to all other objects subordinate to it. In the same way, it is possible to return to the entry point, or to move back a number of steps to descend at another point. It is therefore possible to browse through the data structure for the purposes of diagnostics and information gathering.

8.1.1 Descending through the Data Structure

1. Switch to *Edit* mode. Only the first menu column containing the object type icons is displayed.
2. Click the desired object group icon in the first menu column.
3. Select the object to be examined in the drawing area. In the second menu column, the icons of the creating action and any executed manipulation actions are displayed. The third menu column shows the associated properties and parameters for this object.
4. Click on the property and/or parameter icon for which you want information.
 - If this is a value, the current value is displayed in the text field or the appropriate Geometric tolerances.
 - If this is an enumeration value, the icons of the possible values are output in the fourth menu column. The icon of the current value is highlighted.
 - If this is an object, an arrow icon is displayed in the special menu area above the third column and the icon of the creating action above the fourth menu column.
5. Click on the arrow icon. In the second menu column, the icons of the creating action and executed manipulation actions and in the third menu column the associated properties and parameters are displayed. The third menu column shows the associated properties and parameters for this object. In the special icon area above the second menu column, an arrow icon is displayed for ascending through the data structure.
6. Descend through the data structure by clicking alternately on parameter icon and arrow icon until you find the information you are looking for and/or you have found the action and/or the parameter to be changed.



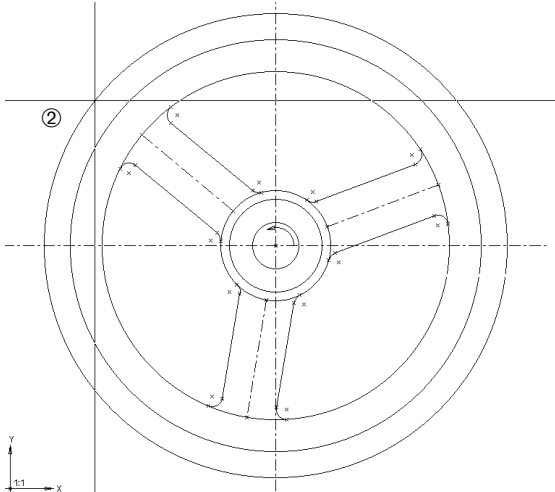
You can use the *show sons* and *show ancestors* menu command in the popup menu of the second menu column to visualize the dependent objects and/or the relations of an object.

The following description illustrates the update facilities of **EUKLID Design** with the example of the handwheel from the introductory example.

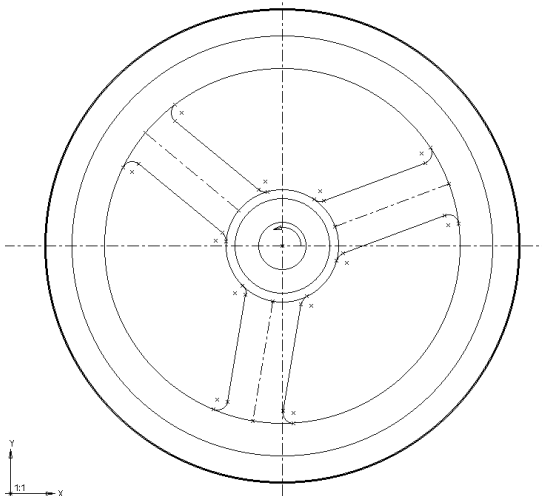
Updating Objects

Example 1: Inquiry of Object Action

1. Click the icon of the *circle* object group in the first menu column.
2. Select the outer circle in the drawing area.

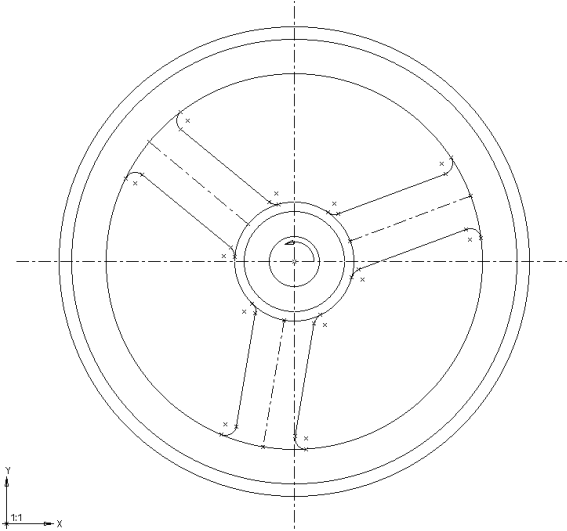


The circle is highlighted and the action *circle with center point and radius* is highlighted in the second menu column. The third menu column shows the icons *Z value*, *line type* and *center lines* as properties, center point and radius as mandatory parameters.



Example 2: Inquiry of Object Parameter

☞ Click the parameter icon *radius*. The keyboard icon is highlighted in the fourth menu column. The radius is then input via the keyboard. The current value of the radius is displayed in the text area.



8.1.2 Ascending through the Data Structure

8

After descending through the data structures, you ascend through the data structures as follows:

☞ Click the *arrow* icon above the second menu column until you reach the desired action and/or parameter.

Updating Objects

8.2 Editing an Object

You can edit the following categories of objects in *Edit* mode

- Properties and/or parameters
- Action

8.2.1 Editing Properties and Parameters

The properties of the object and the parameters of each action are visualized and edited using the third command column. ***EUKLID Design*** differentiates in this respect between

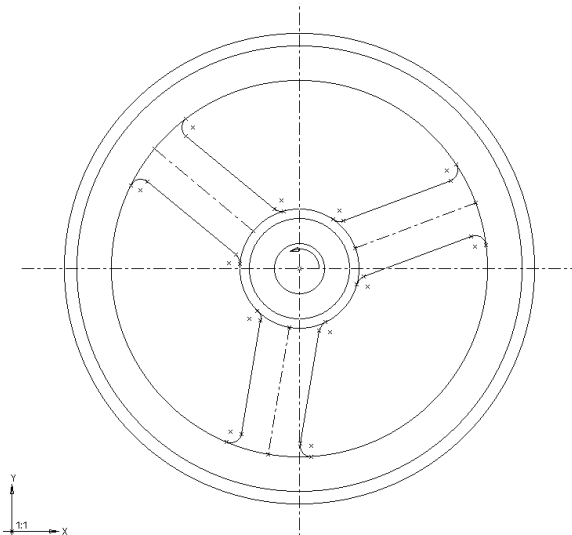
- Editing properties and parameters of a create action
- Editing properties and parameters of a manipulator action
- Editing properties and parameters in a complex design environment

8.2.1.1 Editing Properties and Parameters of a Create Action

To edit a property or parameter of the *create* action,

1. Click on the icon of the object group to which the object belongs in the first menu column in *Edit* mode. Default is the object type last clicked.
2. Select the object to be edited in the drawing area. The second menu column will now display the creation action and any manipulation actions of the object and the third menu column the properties and parameters of the action.
3. Click on the icon of the property and/or parameter to be changed in the third command column.
4. Reenter the property and/or parameter value (see main chapter *Creation of objects*, chapter *Explicit Construction*, section „*Input of Properties and Parameters*“ on page 7-4).

The radius of the outermost circle of the drawing *rad2*, whose parameter has already been selected in the 2. example *Inquiry of Object Parameter* in chapter „*Descending through the Data Structure*“ on page 8-3, section *Descending through the Data Structure*, may be defined anew either by input via the keyboard or simply by sketching.



Updating Objects

8.2.1.2 Editing Properties and Parameters of a Manipulator Action

To edit a parameter of a manipulator action

1. Click on the icon of the object group to which the object belongs in the first menu column in *Edit* mode. Default is the object type last clicked.
2. Select the object to be changed in the drawing area. The second menu column will now display the creation and manipulation actions of the object and the third menu column the properties and parameters of the action.
3. Click on the icon of the desired manipulator action in the third command column. The third menu column will now display the properties and parameters of the manipulator action.
4. Click on the icon of the property and/or parameter to be changed in the third command column.



The victim parameter cannot be edited. If you click on its icon, ***EUKLID Design*** outputs a message to this effect.

5. Reenter the property and/or parameter value (see chapter „*Input of Properties and Parameters*“ on page 7-4).

8.2.1.3 Editing Properties and Parameter in a Complex Design Environment

In the following cases, updating is a multi-stage process involving descending into the data structure and editing:

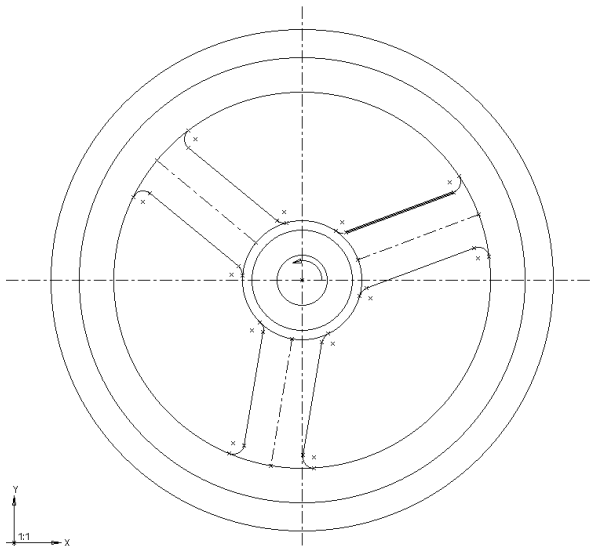
- You did not create the original design yourself and you do not know its internal construction and thus the interdependencies between the individual objects.
- For variant creation and changes to dimensions, you need to find the object whose parameters need to be changed to achieve the desired result.

The example *rad2* is used here to illustrate the update process.

Example

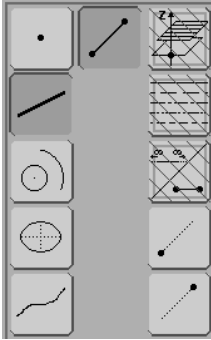
The task is to reduce the width of the spokes. Adopt the following procedure:

1. Select any line forming a spoke.
2. Establish, by means of the second and third menu column, whether the spoke selected is actually the original or one created by copying and mirroring.
3. Find the original spoke via a descent through the data structures. In this model, there is only one "genuine" spoke (with roundings). All others were derived from it by mirroring about the central axis and copying about the center point of the circle.

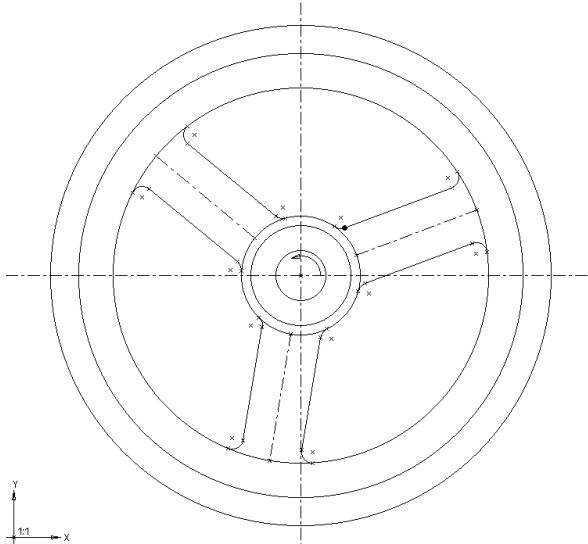


Updating Objects

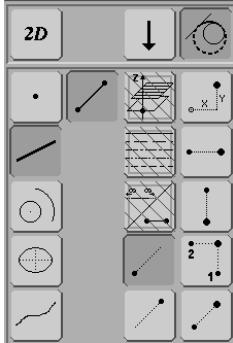
The following figure shows by which action the line was created. The line was defined as line between two points.



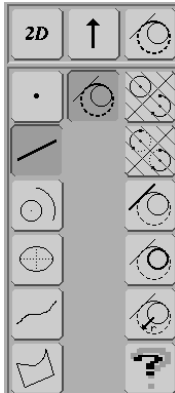
4. Click on the first point parameter icon. The starting point of the line in the spoke is highlighted in the drawing area.



In the menu area, the fourth menu column, from which the action of this point on an *arc tangential to a line and a circle* can be seen, is displayed.

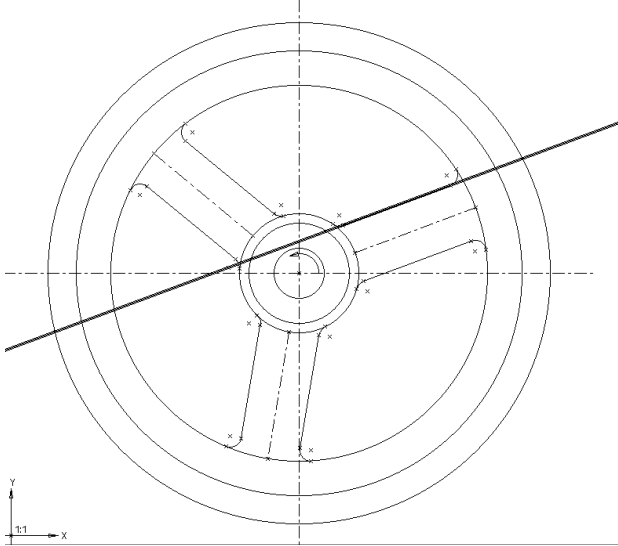


- Click on the arrow icon above the third menu column. The icon of the action that created the point (arc tangential to a line and a circle) is displayed in the second column. The *arc tangential to a line and a circle* icon is highlighted. The icons of the corresponding parameters are displayed in the third menu column, and the current action is displayed via the third column.

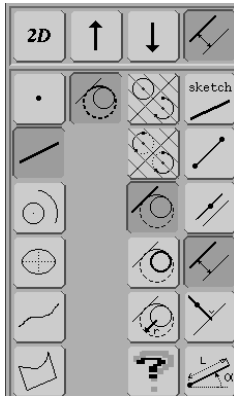


Updating Objects

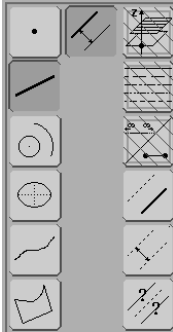
6. Click on parameter icon *line* in the third menu column. An infinite line, on which the "genuine" spoke lies, is displayed and highlighted.



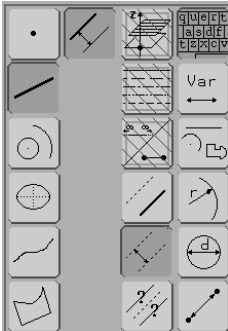
It may be seen from the fourth menu column that this *infinite line through 2 points* is an *line parallel to a line with distance*.



- Click on the arrow icon above the third menu column to descend further in the data structure. The system outputs the parameters of the parallel in the third menu column. The action *line parallel to a line with distance* of the infinite line to be examined is now highlighted in the second menu column.

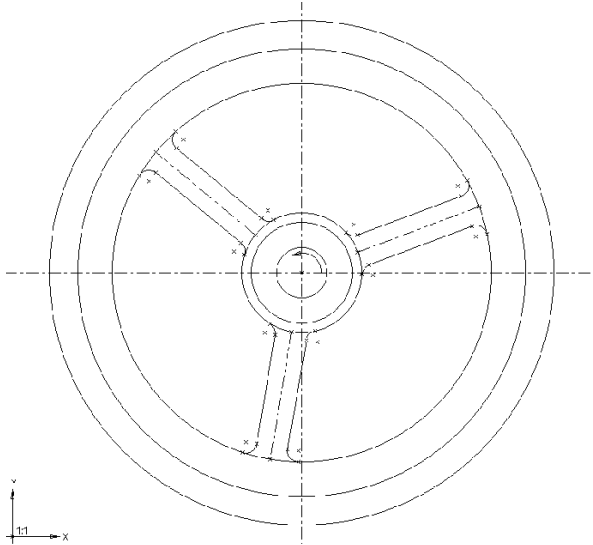


- Click on parameter icon *reference line* in the third menu column. **EUKLID Design** flags the center line of the spoke as a reference line. The infinite line, on which the "genuine" line of the spoke lies, is thus dependent on this center line. Consequently, only that distance needs to be changed to arrive at a variant of the wheel with thinner spokes.
- Click on parameter icon *distance from reference line* in the third menu column. The fourth menu column is displayed highlighting the keyboard icon. The current value is displayed in the text area.



Updating Objects

10. Change the current value via the keyboard. This causes all objects in the design which are dependent on this distance, to be recalculated and displayed anew. The result is a spoked wheel with thinner spokes.



Provided that the nature of the subsequent variant is comprehensible and predictable, you should make the parameters to be edited dependent on one central object (section „[Simplification of Construction Modifications](#)“ on page 13-95), guaranteeing direct access.

8.2.2 Editing Actions

You can change the actual interdependencies between objects via the actions (redefinition). **EUKLID Design** differentiates in this respect between

- Redefinition of objects
- Redefinition of object parameters
- Separation of objects
- Redefinition of object sets to simple actions

8.2.2.1 Redefining Objects

You may only redefine objects in active layers.

1. Choose *redefine* in the popup menu of the creation action icon. The second menu column will show all possible create actions for the selected object.
2. Click on the desired action icon.
3. Enter the required parameters as when creating objects.



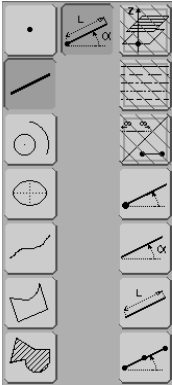
To cancel the redefinition, use the arrow icon above the second menu column.

Updating Objects

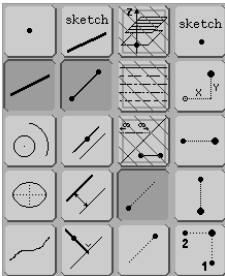
Example

The keyway of the *rad* model in the introductory example is created as a *line with length and angle* and shall be redefined as *line through two points*.

1. Click on the line icon in the first menu column in *Edit* mode.
2. Select the infinite line in the drawing area. The second menu column will highlight the creation action *line with length and angle* and the third menu column will display the properties and parameters of this action.



3. Choose *redefine* in the popup menu of the action icon. In the second menu column, the possible create actions for the selected line will be displayed.
4. Click on the action icon *line through two points* in the second menu column. The infinite line will be redefined to an *line through two points*.



5. Input the parameters for the new action via identifying the corresponding tangent points of the arcs in the drawing area.

The design of the exact connection between the arcs – instead of the parallel construction line – is now completed.

8.2.2.2 Redefining Object Parameters

In the case of complex parameter interdependencies you may carry out changes to the actions in the fourth menu column in the same way as the implicit creation of objects, in other words, you redefine the object parameters of a selected action in the fourth menu column.

You may input constants via the keyboard to furnish parameters of an object. It is usually more useful, however, to make parameters dependent on other factors, i.e. to make them variable. The scope of interdependencies defined in this way is considerable. The following options are available for deriving the length of a new line (these are just a few).:

- From a previously defined variable;
- From the distance between two existing points;
- From the radius of an existing circle;
- From an arithmetical expression (e.g. total, difference between two lengths etc.).

In the case of complex arithmetical expressions, arithmetical functions must be nested.

Example

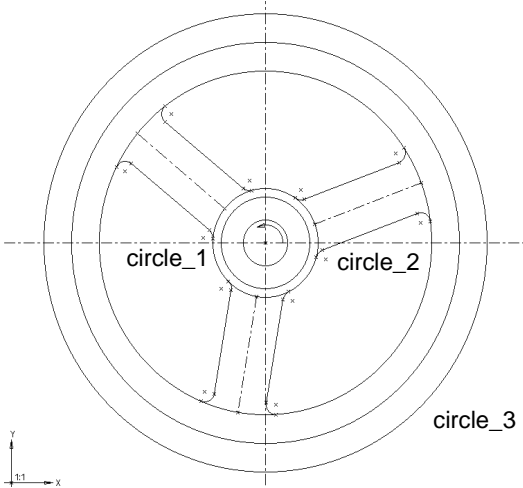
The radii of the three outer circles of the handwheel in the introductory example are independent of each other, making it necessary to update them all separately when creating a variant. By linking the three circle parameters it is possible to arrive at a variant by changing only one parameter. This may be achieved by introducing the following interdependencies into the design:

$$\begin{aligned}\text{radius_3} &= \text{radius_1} + \text{length} \\ \text{radius_2} &= \text{radius_1} + (\text{radius_3} - \text{radius_1})/2\end{aligned}$$

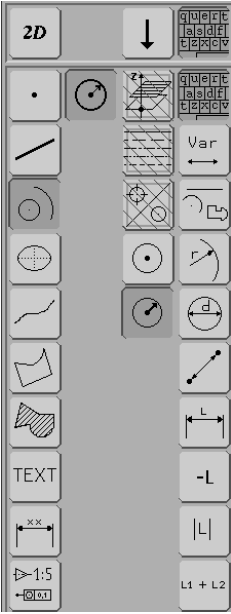
Therefore, each of the radius parameters of *circle_3* and *circle_2* must be redefined.

Updating Objects

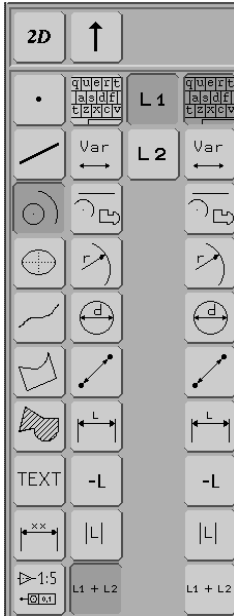
- 1. Select *circle_3*.



- 2. Click on the parameter icon *Radius*.



- Click on the arithmetic icon *sum of lengths* ($L_1 + L_2$) in the fourth menu column. The parameters of this new action are displayed and inquired in the third menu column.



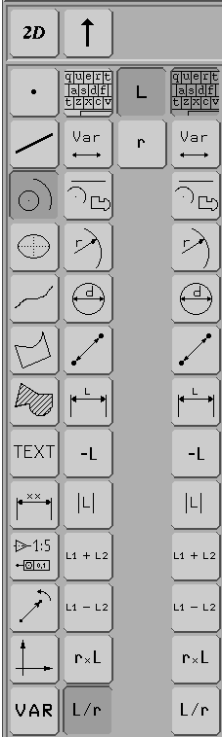
- Click on the radius icon in the fourth menu column to furnish parameter L_1 .
- Identify *circle_1* in the drawing area.
- Input the length L_2 via the keyboard. The relation to *radius_3* has now been established.

Changing the radius parameter of *circle_2*, the arithmetic icons have to be nested; the procedure is in principle the same.

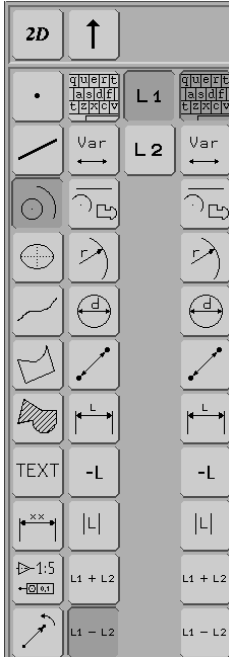
- Select *circle_2* in the drawing area.
- Click on the parameter icon *Radius*.
- Click on the arithmetic icon *sum of lengths* ($L_1 + L_2$) in the fourth menu column. The parameters of this new action are displayed and inquired in the third menu column.
- Click on the radius icon in the fourth menu column to furnish parameter L_1 .
- Identify *circle_1* in the drawing area.

Updating Objects

6. Click on the arithmetic icon *quotient of two values* (L/r) in the fourth menu column to furnish parameter $L2$. The parameters of this new action are displayed and inquired in the third menu column.



7. Click on the arithmetic icon *difference of two length* (L_1-L_2) in the fourth menu column to furnish parameter L . The parameters of this new action are displayed and inquired in the third menu column.



8. Click on the radius icon in the fourth menu column to furnish parameter $L1$.
9. Identify *circle_3* in the drawing area.
10. Click on the radius icon in the fourth menu column to furnish parameter $L2$.
11. Identify *circle_1* in the drawing area.
12. Input the value for r via the keyboard. The relation to *radius_2* has now been established.

Once these interdependencies have been defined, any change to `radius_1` will affect the entire wheel.

Updating Objects



- Working is considerably streamlined if parameter *radius_1* is stored in a variable.
- By descending further through the data structure, these actions for base objects would be accessible via the second menu column rather than the fourth menu column.

8.2.2.3 Redefinition of an Object Set to Simple Actions

By redefining objects, you can dissolve all relations between objects. All selected objects are reduced to 'simple' actions as far as possible (see section „[Action Types of Transformed Objects](#)“ on page 8-23). All relations within the selection are dissolved.

Calling the Action



The system allows objects to be redefined either individually (by clicking onto them inside the drawing area) or as a group selection. You may completely redefine as many objects of similar or dissimilar types as desired.

Example

Objects may be redefined as follows:

1. Click on *redefinition* icon.
2. Define the parameter *destination coordinate system*.
3. Select the objects to be redefined.
4. Click onto the lower arrow icon of the list parameter to confirm and carry out the redefinition action.

Action Types of Transformed Objects

The following alphabetic listing contains the action types used to redefine objects:

- **Angle** (created with absolute action, angle with same value as in original)
- **Circle**
- **Full circle** (circle with center point and radius)
Parameters:
Center point M with x- and y-distances to the coordinate system origin
Radius with same value as the corresponding radius of the original
- **Circle arc** (circular arc through three points)
Parameters:
First point with x- and y-distances to the coordinate system origin
Second and third points with x- and y-distances to the previous point
- **Coordinates** (absolute coordinates with same value as in original)
- **Coordinate system** (coordinate system with absolute coordinates)
Coordinate system with x/y scaling (*world_rota_scale*) remains unchanged.
- **Cropped image** (absolute cropped image as in original)
- **Measure** (same dimensions as in original by duplication of parameters *point, line, circle, length, angle, coordinates, text before nominal dimension, text after nominal dimension, string*)
- **Ellipse**
- **Full ellipse** (ellipse with midpoint, end point of primary axis and length of the secondary axis)
Parameters:
Midpoint with x- and y-distances to the coordinate system origin
End point of primary axis with x- and y-distances to midpoint M
Half length with same value as the corresponding length in original
- **Elliptical arc** (elliptical arc with end point of an axis, beginning point and end direction)
Parameters:
Midpoint with x- and y-distances to the coordinate system origin
End point of primary axis with x- and y-distances to midpoint
Beginning point of arc with x- and y-distances to midpoint
End direction point of arc with x- and y-distances to midpoint
- **Group** (same group as in original by duplication of parameters)
- **Index** (same index as in original by duplication of parameters)
- **Length** (created by absolute action, length with same values as in original)

Updating Objects

- **Line** (line through two points)
Parameters:
Beginning point with x- and y-distances to the coordinate system origin
End point with x- and y-distances to the beginning point
- **Number** (created with absolute action, number with same value as in original)
- **Plane/contour**
Plane comprised of design objects line, circle, ellipse (plane_on_construction)
contour comprised of design objects line, circle, ellipse (contour_on_construction)
- **Dimension position** (same dimension position as in original by duplication of parameters (*length, dimensions*) with same value)
- **Proportion** (same proportion as in original)
- **Point** (point with x- and y-distances to the origin of a coordinate system)
- **Roughness** (Same roughness as in original by duplication of parameters with same value)
- **Spline** (closed B-spline or open B-spline)
Open spline (normal, mirrored, copied) produces open spline, closed spline produces closed spline
- **String** (string with same value as in original)
- **Text** to specific dimension (same text as in original by duplication of parameters with same value)
- **Symbol** (same symbols as in original by duplication of parameters *point, line, circle, length, angle, coordinates, text before nominal dimension, text after nominal dimension, string*).
- **Symbol balloon** Copied symbol balloon changes to simple symbol balloon with same values as in original
- **Table** (same table as in original, not duplicated)
- **Text** (Text with angle, text block or mirrored text by duplication of parameters string, point, length, angle and line)
- **UDO** (same UDO as in original by duplication of parameters)
- **Variable** (same variables as in original by duplication of parameters *length, angle, point, string, number*)

8.2.2.4 Separating Objects

Separation lets you sever the relation between objects to their parameter objects, so that an independent portion of the model is created. I.e., undesired ancillary objects are reduced to simple actions and relations are maintained within the selection set. The same objects can be separated as can be duplicated.

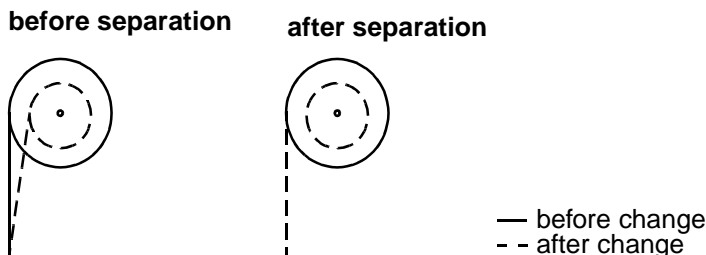
The objects whose parameter objects are separated are, in a manner similar to that of duplication, redefined as objects with 'simple' actions with respect to a coordinate system. The same actions are used as for duplication (see the section „[Duplicating Objects](#)“ on page 11-9).

There are three possibilities of separation available for governing the separation process.

- 'Separate selection from rest' (default): The objects retain their relation to those objects that are dependent upon them, so that following command execution, the selection is separated only to the outside and is linked to a coordinate system.
- 'Separate rest from selection': cuts all relations from outside the selection set into the selection set.
- 'Separate in both directions': cuts all relations (complete separation).

Example

The behavior of a separated object is demonstrated below using as an example a line of action *tangent to circle*.



Updating Objects

When the circle diameter is changed, the position of the line differs before and after separation:

Before separation, the line is positioned according to the change in circle diameter: it remains tangent to the circle.

After separation, the line is defined as a 'line between two points'. It retains its original position, even when the circle is changed, and so it is no longer tangent to the circle.

Separation is especially useful for storing portions of models, for simplifying complex designs or for severing table relations.

Calling the action

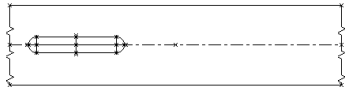


You can separate objects either individually or in groups by including several objects in a selection set.

Example

The fanwheel from our previous examples is now to be mounted on a bushing. This requires that the fanwheel dimensions must no longer be dependent upon the shaft diameter.

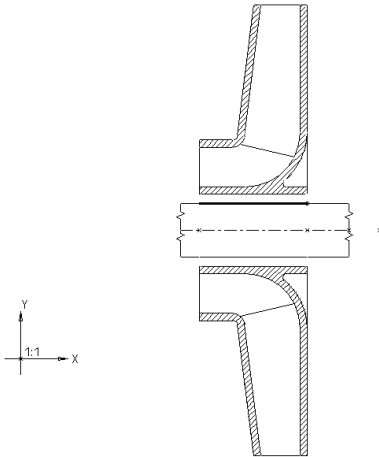
1. Click on the *Separate* icon.
2. Define the first parameter, *sever deleted relations* yes/no.
3. Define, as appropriate, the parameter *Type of separation* as 'Separate selection set from rest', 'Separate rest from selection set' or 'Separate in both directions'.



4. Define the parameter *Coordinate system*.

5. Select the objects to be separated inside the drawing area.
6. Click onto the lower *arrow* icon in the list parameter to confirm and carry out the separation action.

Result following separation, deletion of the feather key groove and changing of the shaft dimensions:



The relation between the objects is dissolved.



Separation can also be used to dissolve the dependency of a model upon a table

- by not including the table index in the selection for separation. The dependent object (*length*, *angle*, *string*) is then redefined to the same absolute value
- by using AQL for absolute redefinition of the dependent object (*length*, *angle*, *string*).

Updating Objects

8.3 Editing Objects in Current Mode

In the current mode, you are allowed to edit the following objects without switching to the *Edit* mode explicitly:

- the object closest to the cursor position
- the object last created

An action, which was interrupted by the *Quick Edit* mode, may be resumed from where it was left off after discontinuing this mode.

8.3.1 Quick Edit of the Object Closest to the Cursor Position

Edit the object closest to the cursor position in *Quick Edit* mode as follows:

1. Position the cursor on the object which is to edit.
2. Choose *2d viewing -> quick edit* and/or *quick edit* in the drawing area popup menu. **EUKLID Design** switches to *Edit* mode. The object closest to the cursor position is selected and highlighted. In the second menu column, the creation action and executed manipulator actions are displayed and in the third menu column, properties and parameters are displayed.

You are now able

- to edit properties and parameters
- to edit actions
- to descend through the data structures

8.3.2 Quick Edit of the Object Last Created

This action is useful for correcting the parameters of the creating action.

Edit the object last created in *Quick Edit mode* as follows:

- Choose *Edit last object* in the *Undo* menu or press the accelerator <CTRL>+ <I>. **EUKLID Design** switches to *Edit* mode. The object last created is selected and highlighted. In the second menu column, the creation action and executed manipulator actions are displayed and in the third menu column, properties and parameters are displayed.

You are now able

- to edit properties and parameters
- to edit actions
- to descend through the data structures

8.3.3 Discontinuing Quick Edit Mode

Discontinue the *Quick Edit* mode in one of the following ways:

☞ Change to another mode (*Create*, *Delete*).

or

☞ Press the <ESC> key.

Updating Objects

8.4 Editing via Iteration (Specific Value Entry)

The *iteration* action offers an alternative method of editing individual objects to the *Edit* mode previously presented.

Iteration manipulates the length or the value of objects, e.g. the length of lines, contours, the circumference of circles or values of variables. The system therefore holds a list of all basic objects influencing the start object. You can go through this list (graphical display) until you have reached the desired basic object. Then you can enter the specific value for the start object: the system computes (iterates) the basic objects data structure until the start object meets the required specific value.

User guidance is different from that of the Edit mode: you can directly find the basic objects, select one and receive system feedback, if the specific value could not be attained. In complex designs, the path through the *Edit* mode may sometimes be difficult to find.

You can access the following objects and use them to adapt the part being designed to specific design requirements.

- Dimensions
- Variables
- Lines
- Circles, Circular arcs
- Ellipses
- Contours
- Lengths
- Angles
- Proportions
- Real number

Basic objects are lengths, angles, proportions and real numbers. Only these basic objects appear in the list for iteration.

This lets you use the sketcher to create a rough model inside the drawing area that only approximates the exact coordinates of the final design. When the component is finished, iteration is used to specify the exact geometry and dimensional accuracy. Please read the section on [„Adjusting a Sketched Profile \(Adjust Action\)“ on page 8-34](#).

The *iteration* action can also be used to modify existing models. This is especially useful when

- it becomes difficult to ascertain the relations between individual objects in a complex model and therefore is difficult to know which parameters influence a particular geometry or
- the relation between a parameter and e.g. a dimension is mathematically complex, making it nearly impossible to give the dimension a specific value simply by changing a parameter value.

Procedure

Proceed as follows:

- Activate the *iteration* action.
- Open the alternative parameter and click on the object type to be edited.
- Select the object to be iterated inside the drawing area.
- Select the basic object to be affected by the change in design: by pressing the <TAB>-key you can proceed through the list of basic objects.
- Accept the selection of the basic object by pressing the right mouse button.
- Type the desired value of the selected parameter into the text entry box.

EUKLID Design then calculates and sets the influence parameter in that way, that the entered specific value matches the start object. The geometry is therefore changed at the expense of the selected basic object.

This procedure can be repeated as often as desired. If the specific value is not obtained, then it should be approximated in several appropriate steps.

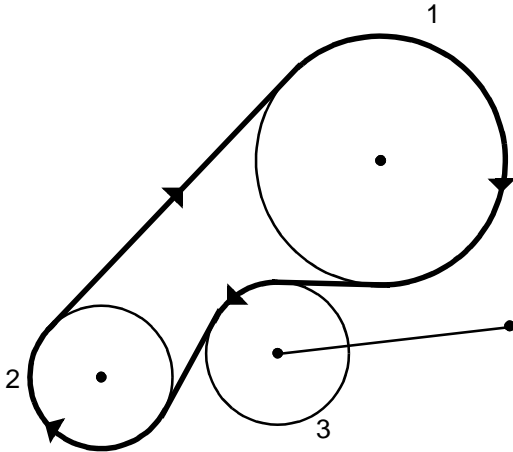
Example

1. Use the *sketcher* action to create the belt drive shown below consisting of three circles, the line to circle 3 and the three tangents, representing the belt, that connect the three circles to one another.



The procedure used to create the objects determines their interdependence. Base your construction upon the original coordinate system so that objects may be changed through iteration based upon the original (or absolute) coordinate system.

Updating Objects



2. Create a contour around the circles and the tangents to form a closed contour sequence (circumferential belt).
3. Click on the *Iteration* icon.



4. Open the alternative parameter and click onto the *contour* icon.



5. Select the contour to be iterated inside the drawing area. (The system marks the first object – or the first value of the object from the group of basic objects that influence the contour length).

The system examines all the basis objects that determine the contour. If this selection takes longer than about 5 seconds, a Geometric tolerances appears to let you cancel the process. The list then can contain objects which do not influence the target value.

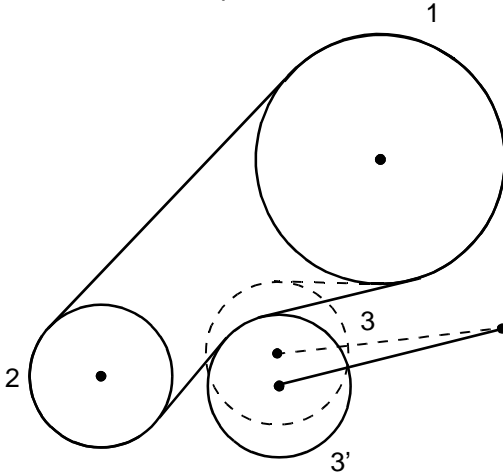
This action can be selected as often as desired. The selection is carried out anew each time (and can change as well).

6. Two alternatives are available for selecting the basic object at whose expense the iteration process shall be carried out:

- Press the <TAB> key until the desired object is marked. Then press the <ENTER> key.
 - Choose in the popup menu of the right mouse key *next* until the desired object is marked.
Choose then *accept* in the same popup menu.
7. The system indicates the calculated value of the contour length in the text entry box. This value can be adapted to the actual circumstances (e.g. the actual belt length) by entering the new value and confirming it with the enter key.



8. The system now tries to display the altered geometry. The system computes (iterates) the basic objects data structure until the start object fits the demanded specific value. In the example, the Y value of the centerpoint of circle 3 has been changed.



There will be times when the desired value cannot be attained immediately. The system then issues a corresponding message. However, you may repeat this action as often as necessary until the specific value is reached. The old value can be reinstated by clicking on the *Reset* icon.

Updating Objects

8.5 Adjusting a Sketched Profile (Adjust Action)

Once you have sketched a component with the sketcher, the qualitative result must be adjusted to the quantitative requirements. Since the dimensions are first determined solely by mouse-click positions, they most likely are fairly inaccurate. The *adjust* action lets you quickly and easily replace these approximate preliminary dimensions with exact ones.

Procedure

1. Click onto the icon *adjust*.



2. Select the viewport in which the geometry of the component is visible. All existing dimensions are deleted first.



3. Use the second parameter *keep* to determine whether the dimensions created are to be retained at the end of the action. If you cancel the action, the dimensions are deleted.



4. Select a dimension and enter a new value for it. Press <ENTER> to confirm or click onto the lower arrow icon. This new value can also be entered via the keyboard or via an implicit action in the fourth icon column. The new value is then depicted in the selected dimension. It is underlined and highlighted in order to show that the dimension has been adjusted. Adjustment takes place, according to the parameter settings, either immediately or upon conclusion of the action.



5. You can immediately check the change in your design brought about by your dimension entry by activating *regen(erate)* and mouse click on the corresponding field in the following Geometric tolerances.



6. When all desired dimensions have been given exact values, you can close the parameter list. At that moment, the original dimensions are deleted (unless you have used *keep* to specify that they are to be retained) and the modified original dimensions are displayed again. All dimensions that have changed since the last regeneration (or since the beginning of the session, if regeneration has not taken place) are now adjusted and the design accordingly regenerated with the new dimensions.

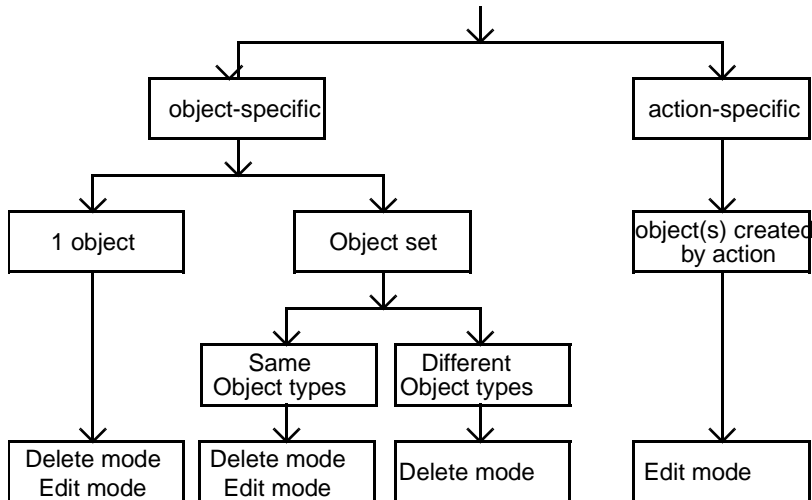
8.6 Undoing the Last Edit

You can undo the last edit action with the menu command *Undo* in the *Undo* menu.

Updating Objects

9 Deletion of Objects

In **EUKLID Design** there are different ways to delete objects:



object-specific The object is deleted in the drawing area. If this object is parameter for creation actions of depending visible objects, it remains in the data structure, for as long as depending objects exist. It is assigned the *deleted* attribute, which can be utilized by AQL programs, for example.

action-specific The object is deleted in the drawing area and data structure. If the deleted object is parameter for creation actions of depending objects, these objects will be redefined absolutely (see main chapter Transformation Techniques, section [„Redefinition of an Object Set to Simple Actions“ on page 8-22](#)).

Deletion of Objects

Example

A line is defined as line through two points. If the start point and/or end point is deleted action-specifically, this point is deleted in the drawing area. In the data structure, the deleted point of the depending line is redefined absolutely.



You may only undo recent delete actions if the deleted object is parameter for creation actions of depending visible objects.

9.1 Deletion of Objects in Delete Mode

In *delete* mode, single objects or object sets may be deleted object-specifically. The creation action of dependent objects remains unchanged. You should delete object sets of heterogeneous object types in this mode.

9.1.1 Deletion of Single Objects in Delete Mode

To delete an single object in *Delete* mode, proceed as follows:

- ☞ Click on the *Delete* icon in the ruler, unless already selected. In the first menu column, the existing object groups are displayed.
- ☞ Click on the icon of the object group to which the single object belongs, in the first menu column.
- ☞ Select the object in the drawing area: it will be deleted in the drawing area. If objects are depending on this object, the deleted object remains in the data structure. The object deleted will not be removed from the data structure, until depending objects have been deleted and the model has been saved.

If you delete an object type *group*, the group itself will be deleted, but not the objects of the group.



You may correct gaps to appear in the display by selecting the *Redraw* icon.

Deletion of Objects

9.1.2 Deletion of Object Sets

An alternative to individual deletion of objects is the deletion of any number of objects of the same type or different types within a rectangle (see main chapter System Handling, section „[Selection in Rectangle](#)“ on page 4-22).

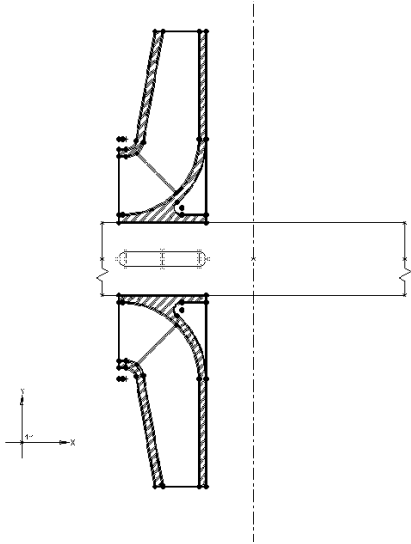
9.1.2.1 Deletion of Same Object Types

To delete the same object types in *Delete* mode, proceed as follows:

- ☞ Click on the *Delete* icon in the ruler, unless already selected. In the first menu column, the existing object groups are displayed.
- ☞ Unless already selected, click on the icon of the object group to which the objects belong, in the first menu column.
- ☞ Select the objects to be deleted in the drawing area via selection in rectangle.
- ☞ Modify the selection set via the single mode, if necessary.
- ☞ Confirm the selection set via the drawing area popup menu. The objects are deleted in the drawing area. If objects depend on this object, the deleted object remains in the data structure. The object deleted will not be removed from the data structure, until depending objects have been deleted and the model has been saved.

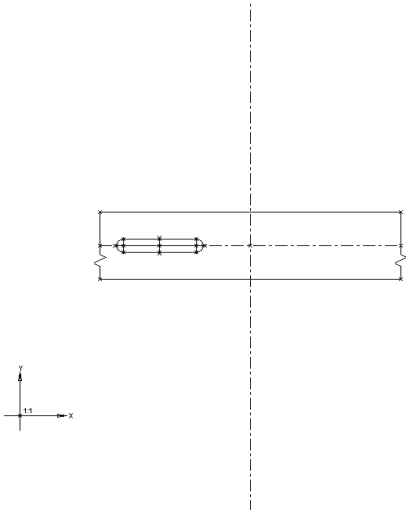
9.1.2.2 Deletion of Different Object Types

The deletion of different object types in *Delete* mode is demonstrated by the lateral view of a pump impeller. All objects except for feather key and groove shall be deleted:



- ☞ Click on the *Delete* icon in the ruler. In the first menu column, the existing object groups are displayed. The default is the last icon selected in the first menu column.
- ☞ Click on the point icon in the first menu column. The point icon is highlighted.
- ☞ Click on the plane icon in the first menu column pressing the <SHIFT> key. All object types from point to plane are highlighted.
- ☞ Select the impeller in the drawing area via selection in rectangle. The impeller is highlighted.
- ☞ Click on the plane icon in the first menu column pressing the <CTRL> key. The highlighting of the plane icon is canceled.
- ☞ Select the points, lines and circles belonging to feather key and groove in the drawing area via selection in rectangle. The highlighting of the corresponding objects is canceled.
- ☞ Choose *delete all* in the popup menu of the drawing area. All objects except for feather key and groove are deleted object-specifically.

Deletion of Objects



If objects depend on this object, the deleted object remains in the data structure. The objects deleted will not be removed from the data structure, until depending objects have been deleted and the model has been saved.

9.2 Deletion of Objects in Edit Mode

In *Edit* mode, you may delete single objects or object sets. You may delete single objects object-specifically or action-specifically.

- In case of object-specific deletion, the creation action of dependent objects remains unchanged.
- In case of action-specific deletion, **EUKLID Design** will redefine depending objects absolutely (chapter „*Redefinition of an Object Set to Simple Actions*“ on page 8-22).



As long as you stay in *Edit* mode, you can perform and undo operations at the object or set selected. This includes deletion. The deletion does not become final until you leave the *Edit* mode.

9.2.1 Deletion of Objects in Edit Mode

By object-specific deletion, you can delete single objects or object sets of the same object type.

9.2.1.1 Deletion of a Single Object in Edit Mode

For object-specific deletion, proceed as follows:

1. Click on the *Edit* icon in the icon bar, unless already selected. In the first menu column, the existing object groups are displayed.
2. Unless already selected, click on the icon of the object group to which the object belongs, in the first menu column.
3. Select the object to be deleted in the drawing area. The second menu column will now display the creation and manipulation actions of the object and the third menu column the properties and parameters of the action.
4. Choose *delete* in the popup menu of the creation action icon in the second menu column (right mouse button on the icon). The object will be deleted in the drawing area. If objects are depending on this object, the deleted object remains in the data structure.

Deletion of Objects

9.2.1.2 Deletion of Same Object Types in Edit Mode

For object-specific deletion, proceed as follows:

1. Switch to the *Edit* mode (see main chapter *Updating Objects*). In the first menu column, the existing object groups are displayed.
2. Click on the icon of the object group to which the objects belong, in the first menu column.
3. Select the objects to be deleted in the drawing area via selection in rectangle.
4. Modify the selection set via the single selection mode, if necessary.
5. Choose *delete all* in the popup menu of the drawing area. The objects are deleted in the drawing area. If objects depend on this object, the deleted objects remain in the data structure. If the depending objects are deleted and the model is saved, the deleted objects are removed from the data structure.

9.2.2 Action-specific Deletion of Objects in Edit Mode

By means of action-specific deletion, you can only delete objects created by a creation action or manipulator action. Proceed as follows:

1. Click on the *Edit* icon in the icon bar, unless already selected. In the first menu column, the existing object groups are displayed.
2. Unless already selected, click on the icon of the object group to which the object belongs, in the first menu column.
3. Select the object to be deleted in the drawing area. The second menu column will now display the creation and manipulation actions of the object and the third menu column the properties and parameters of the action.
4. Choose *remove action* in the popup menu of the creation action icon and/or *remove manipulator* of the manipulator action icon in the second menu column. The action will be deleted. If objects are depending on this action, they are redefined absolutely.

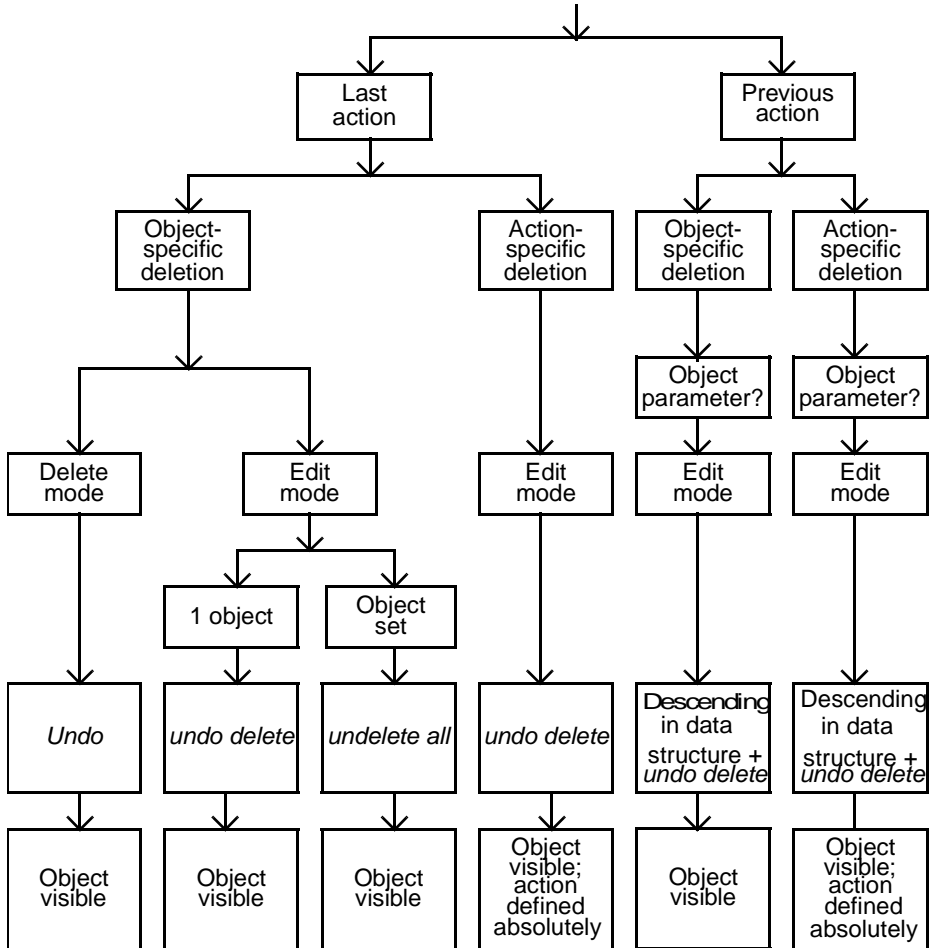
Example

The object set of a pentagon created by the *macro_multicorner* action shall be deleted. Change to the *Edit* mode for one line of the pentagon. From the popup menu in second menu column, select the *remove action* function. All lines, points, and the contour belonging to this pentagon will be removed from the data structure.

Deletion of Objects

9.3 Undoing Delete Actions

In *EUKLID Design* there are different ways to undo delete actions:





After an action-specific deletion, you can make visible the object by descending in the data structure, but the action remains defined absolutely. If you want to re-establish the original creation action, you must redefine the object explicitly (see main chapter Updating Objects, section „*Editing Actions*“ on [page 8-14](#)).

Previous or action-specific deletions can be undone only if there are dependencies on the object or if the object is named.

9.3.1 Undoing the Latest Object-specific Delete Action in Delete Mode

If you object-specifically deleted an object or object set in the *Delete* mode by the latest action, you can undo the deletion action by the *Undo* menu command. Proceed as follows:

☞ After the deletion, choose *Undo* in the *Undo* menu - You may not have left the *Delete* mode yet. The object or object set deleted re-appears in the drawing area.

9.3.2 Undoing the Latest Delete Action in Edit Mode

If you object-specifically or action-specifically deleted an object in the *Edit* mode by the latest action, you can undo the deletion action by the *undo delete* menu command. Proceed as follows:

☞ After the deletion, choose *undo delete* in the popup menu of the creation action. The object deleted re-appears in the drawing area.



You can perform this action until you leave the *Edit* mode in which you deleted the object set.

Deletion of Objects

9.3.3 Recovering Currently Object-specifically Deleted Object Sets in Edit Mode

If you object-specifically deleted an object set by the latest action in the *Edit* mode, you can recover the object set by the *undelete all* menu command. Proceed as follows:

☞ After the deletion, choose *undelete all* in the popup menu of the drawing area. The object set deleted re-appears in the drawing area.



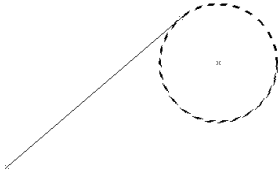
You can perform this action until you leave the *Edit* mode in which you deleted the object set.

9.3.4 Undoing a Previous Delete Action

If you deleted an object in a previous action, you can recover the object, if at least one object depending on this object is still visible. For this purpose, you must descend from a depending object in the data structure (see main chapter Updating Objects, section [„Ascending through the Data Structure“ on page 8-5](#)).

Example

A deleted circle (in the following figure displayed as dashed line) which is used as parameter for the action *Line through a point and tangent to a circle*, shall be made visible again:



1. Click on the icon for the object group *line* in the *Edit* mode, unless already selected.
2. Select the tangent in the drawing area. The second menu column will now display the creation action of the tangent and the third menu column the properties and parameters of the action *Line through a point and tangent to a circle*.
3. Click on the icon of the alternative parameter *el*. The alternative parameter is opened. Default is circle. The circle is highlighted. In the menu extension area, an arrow icon is displayed above the third menu column and the icon of the circle creation action is displayed above the fourth menu column.
4. Click on the arrow icon in the menu extension area. In the second menu column, the icon of the circle creation action is displayed and, in the third menu column, the corresponding properties and parameters are displayed.
5. Choose *undo delete* in the popup menu of the icon of the circle creation action (second column). The circle is displayed in the drawing area.



If the object was deleted action-specifically, the object will continue to be defined absolutely after having undone the deletion action. You can recover the original action by the *redefine* menu command.

Deletion of Objects

10 Manipulating Objects

The following techniques are available:

- Trimming
- Chamfer
- Fillet

Manipulating Objects

10.1 Trimming

The following actions are available:



Trimming of a line up to a line (*trim_lineonline*)



Trimming of two lines up to an intersection point (*trim_2lines*)

The parameters for these actions are described in the Online Help.

10.1.1 Trimming of a Line up to a Line

The manipulator action *trim_lineonline* trims a line up to a line.

EUKLID Design creates an intersection point between both lines and trims the clicked line section of the line to be trimmed to the intersection point. Both end points of the line to be trimmed remain unchanged.

10.1.2 Trimming of two Lines up to an Intersection Point

The manipulator action *trim_2lines* trims two lines up to an intersection point.

EUKLID Design trims the clicked sections of the lines to be trimmed to the intersection point. Both end points of the line to be trimmed remain unchanged.

10.2 Chamfer

The following actions are available:



Chamfer between 2 lines (*line_fase*)



Chamfer between 2 shortened newly created lines (*fase_fase*)

The parameters for these actions are described in the Online Help.

10.2.1 Chamfer between two Lines

The action *line_fase* creates a chamfer between two lines.

The chamfer is stored as an independent action of a line in the data structure and may thus be changed subsequently.

10.2.2 Chamfer between two Shortened Newly Created Lines

The action *fase_fase* creates a chamfer between two lines. ***EUKLID Design*** deletes both of the original lines used in creating the chamfer from the drawing and creates shorter lines. Parallel lines remain unchanged.

The chamfer is stored as an independent action of a line in the data structure and may thus be changed subsequently.

Manipulating Objects

10.3 Fillet

The following actions are available:



Fillet between 2 lines (*circle_tglineline*)



Fillet between 2 shortened newly created lines (*fase_round*)

The parameters belonging to these actions are described in the Online Help.

10.3.1 Fillet between two Lines

The action *circle_tglineline* creates a fillet between two lines.

The fillet is stored as an independent action of a line in the data structure and may thus be changed subsequently.

10.3.2 Fillet between two Shortened Newly Created Lines

The action *fase_round* creates a fillet between two lines. Both of the lines used in creating the fillet are deleted from the drawing and created shorter. Parallel lines remain unchanged.

The fillet is stored as an independent action of a line in the data structure and may thus be changed subsequently.

11 Transformation Techniques

Transformations are working procedures for reproducing objects. The particular relationship between original and transformed objects depends upon the chosen technique.

The following transformations are described in this chapter:

- Copying of objects (vector, mirroring axis)
- Duplication of objects (vector, mirroring axis)
- Moving of objects (vector, mirroring axis)

Transformations are of considerable assistance in designing identical or similar objects such as holes, screws, bolts or threads. These repetitive components no longer need to be constructed anew each time they are needed, since they can be reproduced directly from existing originals or through use of transformation vectors or mirroring axes. The same tools can also help you to create symmetrical components and arrangements of components quickly and easily.



There are several basic differences between the actions of *copying*, *duplication* and *moving* as regards both the data structure created (memory space) and the relation between original and copy, i.e. selection set and resulting objects. Please read the corresponding chapter for basic information.

Transformation Techniques

11.1 Copying Objects

The copy function constructs an identical copy of the original object. When a copy is made, the copy becomes dependent upon the original:

- If the original is altered, the same changes are carried out on the copy as well.

You may copy any object via either of two different methods:

- by specifying a copy vector or
- via selection of a mirroring axis.

Procedure

The principle mode of copying is comprised of four steps:

1. Defining parameters such as the number of copies and the name of the resulting objects.
2. Defining or selecting the copy vector or the mirroring axis in the drawing area.
3. Selecting objects to be copied either by clicking each object inside the drawing area (left and middle mouse key) or by specifying a selection set (selecting within a rectangle).
4. Starting the *copy* action by confirming with <ENTER> or by closing the list parameter (by clicking onto the lower *arrow* icon).

A copied object is created in the data structure as an independent object with its origin specified as *copy of* Its parameters are defined by the corresponding original object and by the copy vector or mirroring axis used in its creation.

Copying is used e.g. for detailing drawings of individual parts consisting of composites. Especially important here is the relation between original and copy. If the original is changed in the composite drawing, e.g. for a design variation, then the copy will automatically be changed in the individual part drawing.

The transformed set can be edited every time. You can also select and transform objects to which no action *copy of* exists.

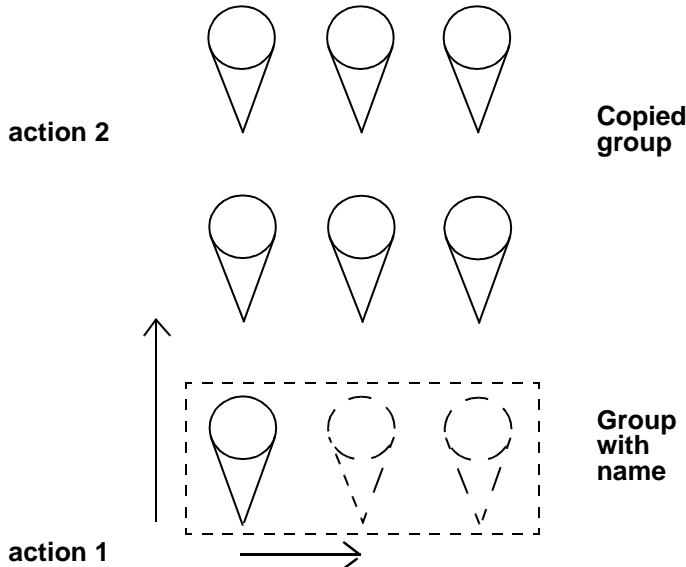


You may designate a name to the object resulting from a *copy* action. All objects involved (original and copy) are then included in the result object *group*. This group then stands at your disposal for further actions such as further copying (using the right mouse key 'select by name' and specifying the name).

Example

The 3 x 3 matrix could be created as follows:

1. Copy a circle with two tangent line segments twice horizontally (i.e. in X direction) and call the result 'Matr_1'.
2. Copy the resulting 'Matr_1' group twice vertically (i.e. in Y direction).
Alternatively, you could copy the three circles with two tangent line segments from the first action twice in Y direction.



Although both alternatives yield the same result, they are distinctly different actions, each with its own affect on the data structure and with its own editing behavior.

Consider the first alternative: if you mark the object at bottom right and specify two copies instead of three in horizontal direction, the number of objects for action 2 also

Transformation Techniques

changes. With the second alternative, however, only the number of objects in horizontal direction changes, while nothing changes in the vertical.

If an original object has a name, the copy is given the same name with the additional index extension, e.g. 'otto_c.5' (or, for a mirrored copy, 'otto_m.5') for the fifth copy of the object named 'otto'.



Names of copied objects should not be changed. They are redesignated according to the name of the original when the next recalculation occurs.

11.1.1 Copying Objects by Defining a Copy Vector



Calling the Action

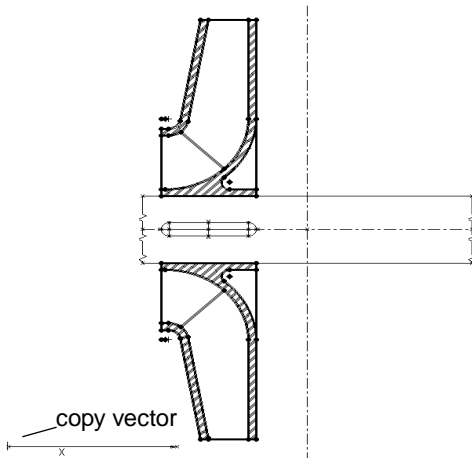
You can copy objects either individually or as a selection set:

- **Individual objects** can be selected from a list of permissible object types such as point, line, circle etc.
- Alternatively, any desired number of objects of identical or different types may be copied at the same time by using a selection rectangle (**selection set**).

Example

You may produce a copy of an original as follows:

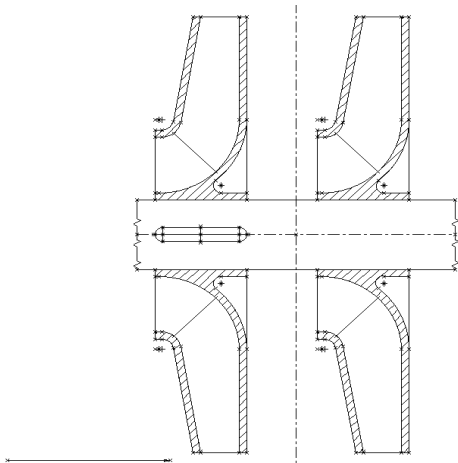
1. Click onto the *copy* icon.
2. Define the numbers of copies or the name of the resulting object if necessary.
3. Define the obligatory parameter *copy vector* by clicking onto it inside the drawing area. If you have not yet created a copy vector, you can make one now in the fourth menu column, then continue with the *copy* procedure.
4. Select the object to be copied in the next step (for individual objects, from the list of permissible object types). In the fanwheel example shown below, select the object to be copied by using the mouse to pull a selection rectangle around it: inside the drawing area, hold down the left mouse key, then pull the mouse to expand the selection area. When the desired objects are within the rectangle, release the mouse key. The objects are now selected.



5. Close the list parameter by clicking onto the lower *arrow* icon. This concludes the copying procedure and carries it out.

Result of copying procedure:

The marked objects have now been copied along the shaft in the direction and distance given by the copy vector.



Transformation Techniques



All newly-created objects are effects of the *copy* action. Their determining parameters are the corresponding original object and the copy vector.

11.1.2 Copying Objects by Defining a Mirroring Axis



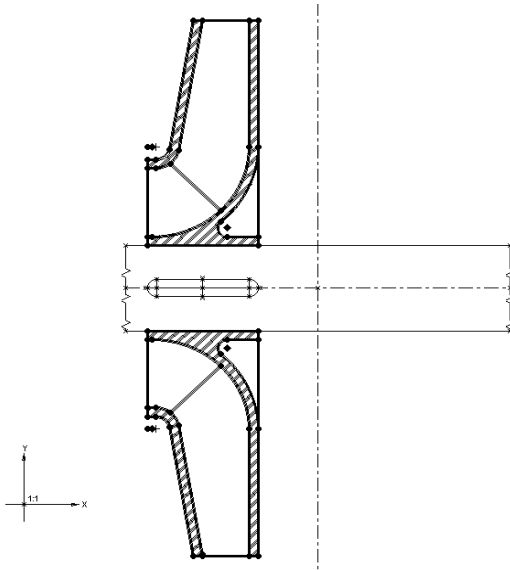
The *mirroring* action is the second form of copying whose operating procedure and action are identical to those of vector copying. It copies the original to any desired mirroring axis.

Select the object to be mirrored either by clicking inside the drawing area or by including the desired object(s) within a selection rectangle. Further selections may be made with the pop-up menu accessed via the right mouse key.

Example

Mirror an original as follows:

1. Click onto the *mirroring* icon.
2. Define the obligatory parameter *mirror axis* by clicking onto it e.g. inside the drawing area. If you have not yet created a mirroring axis, you can make one now in the fourth menu column, then continue with the *mirroring* procedure.
3. Select the object to be copied in the next step. In the fanwheel example shown below, select the object to be copied by using the mouse to pull a selection rectangle around it: inside the drawing area, hold down the left mouse key, then pull the mouse to expand the selection area. When the desired objects are within the rectangle, release the mouse key. The objects are now selected.

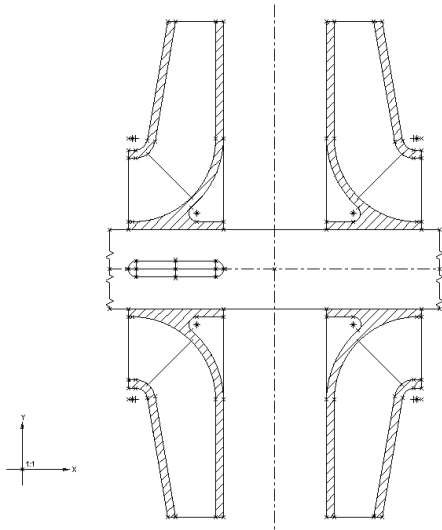


4. Close the list parameter by clicking onto the lower *arrow* icon. This concludes the copying procedure and carries it out.

Result of mirroring procedure

The marked objects have now been mirrored about the specified mirroring axis.

Transformation Techniques



All newly-created objects are effects of the action *mirroring*. Their determining parameters are the corresponding original object and the line specified as the mirroring axis.

11.2 Duplicating Objects

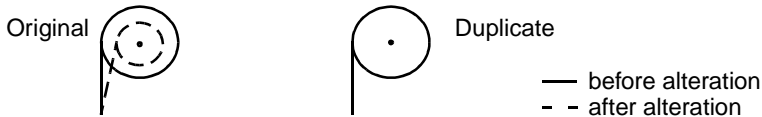
The 'duplication' technique is used when, in contrast to 'copy', no relation to the original is desired. In that case, the duplicate remains unchanged when the original is altered (see the diagram below). This is why the duplication technique is used to design independent, yet similar objects.

The figure below shows the behavior of copy and duplicate when the original is altered.

Copying



Duplication



The result of transformation no longer depends upon on the original object, but rather independent objects are created. The relations between duplicated objects are retained as far as possible, i.e. the actions of duplicated objects within the selection set are not changed.

Example

A 'parallel line' remains a 'parallel line' when duplicated as well.

However, this retention of relations is possible only when all selectable parameters of an object are selected. If not, the object is redefined to simple default actions when duplicated. The additional objects created when redefining (such as beginning and end points of a line) are deleted so that only the duplicated object (here, the line) remains visible.

Transformation Techniques

Procedure

There are two different methods of duplicating objects or mirroring them:

- by specifying a duplication vector or
- through selection of a mirroring axis.

Following method selection, the principle procedure is comprised of the following five steps:

1. Definition of parameters such as number of duplicates, duplication of deleted parameters (yes/no), retention of layer structures (yes/no) or name extension for resulting objects.
2. Definition or selection of duplication vector or of the mirroring axis inside the drawing area (or pop-up selection menu).
3. Definition of destination coordinate system.



Objects assume the transformation from the designated coordinate system. If you want to convert the duplicate to a different coordinate system, then its graphic depiction may change due to different transformation rules of the new coordinate system.

4. Selection of object(s) to be duplicated either via mouse click inside the drawing area or by use of a selection set (e.g. including several objects within a selection rectangle).
5. Execution of the duplication process by confirming with <ENTER> or by closing the list parameter via mouse click onto the lower *arrow* icon.

A duplicate object is created within the data structure as an independent object with the 'transformed' action type.

Procedure

Duplicates may be produced as follows:

1. Creation of a duplicate at the same location as the original.
2. Transformation of the duplicate via manipulator (vector, mirroring axis).

You can also control duplication by using the following **parameters**:

- Number of duplicates
After duplication, the number of duplicates may no longer be edited.
- Duplicate deleted parameters with/without (default = 'with')
If an object has parameters that are deleted (e.g. a circle with center point and circle point; both points are deleted), you normally cannot select the parameters of this object. This object would be redefined via duplication.
'with': the deleted parameters of an object are automatically included in the selection set. These parameters are 'deleted' again from the duplicated object. In the example, the duplicated circle remains a 'circle with center point and circle point'.
'without': the deleted parameters of an object are not automatically included in the selection. In the example, the duplicated circle is redefined to 'circle with center point and radius'.
- Duplication of layer structure of selection set: 'yes/no' (default = no)
'no': the duplicated objects are placed below the layer of the destination coordinate system.
'yes': the original layer structure is duplicated and the duplicated objects are placed into their own layers. The new layer names have the same extensions as the objects.
- Name extension
If a selected object has a name, then the duplicate is given the same name with an extension; the standard extension is '_d'. If more than one duplicate is made, an index number follows the period to indicate the duplicate number (e.g. circle_d.2).

Transformation Techniques



Notes on Positioning

A transformation (vector) is used to position duplicates. This *positioning* action is indicated for a duplicated object in the edit mode by a *transformation* icon (manipulator).

This vector is not identical to the parameter vector that was used to produce the duplicate! Since the duplicated selection must remain independent of the rest of the model, this vector is also duplicated and additionally deleted.

This vector can, however, also be accessed and edited by descending through the data structure of a duplicated object in Edit mode (via manipulator, then further descending). This repositions the entire selection set.

If several duplicates are produced, then an independent vector is produced for each for positioning and set to 'deleted'. This allows each duplicated selection set to be positioned separately.

If an individual object within a selected composite is to be repositioned, then this must be done by editing the parameters of its action.

Calling the Action



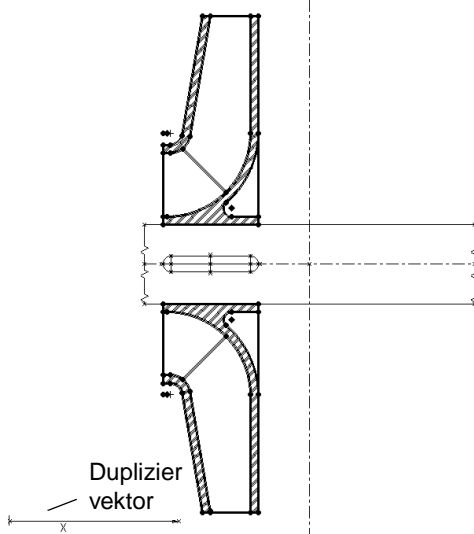
The individual operating steps of the action *duplicate* are the same as those for producing a copy and are carried out in the same fashion. The following description uses an example to illustrate duplication by creating a duplication vector.

You can duplicate objects either individually or together via a composite selection rectangle.

Example

Create a duplicate as follows:

1. Click onto the *duplication* icon.
2. Define the obligatory parameter *duplication vector*. If you have not already done so, create a vector using the fourth menu column, then proceed with the *duplication* action.
3. Select the destination coordinate system.
4. Select the object(s) to be duplicated by using the mouse to pull a selection rectangle inside the drawing area: hold down the left mouse key, then pull the mouse to expand the selection set. When the desired objects are within the rectangle, release the mouse key. The objects are now selected.

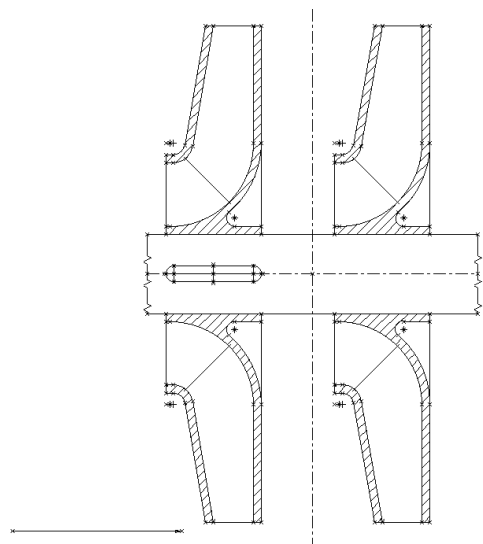


5. Click onto the lower *arrow* icon of the list parameter or confirm using the <ENTER> key. The duplication process is carried out and concluded.

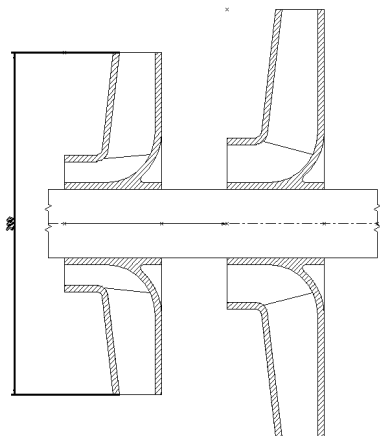
Result Following Duplication

The marked objects are now duplicated along the shaft in the direction and amount specified by the duplication vector. The depiction is identical to that of a copy, but the data structure is not.

Transformation Techniques



If, however, you edit the original drawing, the duplicate – in contrast to a copy – remains unchanged.





- The number of duplicates may not be changed later.
- If an original object has a name, the duplicate is given a variant of that name. Example: 'otto_d' is a duplicate of an object named 'otto'.
- If the transformation includes mirroring (mirroring axis or transformation vector with mirroring component), then the object *coordinate system* may not be duplicated. Objects of this type are automatically removed from the selection set. If groups containing coordinate systems are included in the selection, these objects will also be removed from the selection.
- The following selectable objects are not transformed during duplication because their starting objects include transformation:

Spline

- Groups are duplicated by duplicating the group objects.
- Parameter *Retain layer structure*:

'External layers' are converted during duplication to 'internal layers'.

- If a layer already exists under the new duplicate name, then that layer is used (example: multiple duplication of the same selection with the same name extension). If the layer structure is not duplicated, then all duplicates lie in the same layer as the destination coordinate system.
- If the resulting layer name is too long (i.e. longer than 32 characters), then it is truncated. In that case, the duplicates cannot be separated into separate layers when 'Number of duplicates > 1'.
- Parameter *Destination coordinate system*
The properties of the destination coordinate system influence the position of the duplicates. For example, if all selected objects have the same coordinate system and if the destination coordinate system is rotated with respect to the first system, then the duplicates will also be rotated additionally by the corresponding amount during transformation (parameter vector).

Transformation Techniques

12 Dimensioning and Symbols

Dimensioning and symbols in **EUKLID Design** fulfill national and international standards according to DIN and ISO. The following dimensioning actions are described in this chapter:

- Creating distance dimensions, radius and angle dimensions
- Chain and referenced dimensions
- Dimensioning parameters
- Witness line correction
- Witness line interruption
- Adjusting a sketched profile (*adjust* action)
- Coordinate tables

The section on *symbols* contains explanations of the actions

- General symbols such as *note* and *name lines*, *geometric tolerances*, *reference element symbols*, *surface characteristics*, *section line characteristics*, *conical taper*
- Symbol balloons
- Welding symbols according to DIN 1912

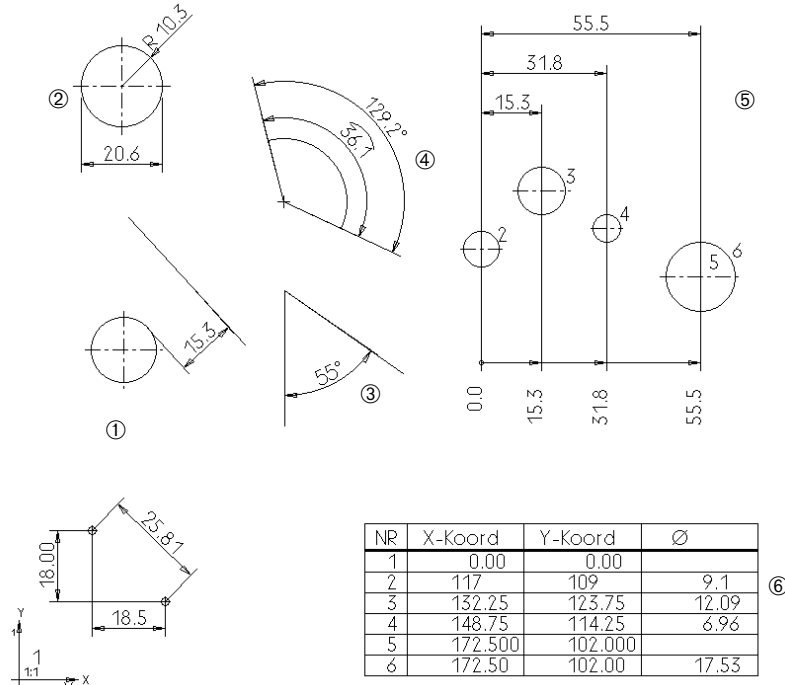
The dimensioning of complex object designs and the symbol actions for the creation of standardized drawings are quite convenient. Interactive operation lets you apply the various actions largely intuitively.

The *adjust* action for sketched profiles lets you quickly specify exact measurements of components whose dimensions were at first only vaguely defined in a sketch (see Chapter *Editing techniques*).

Dimensioning and Symbols

12.1 Dimensioning

The following types of dimensioning may be carried out:



- ① Distance dimensions (of objects of different types such as point, line, circle)
- ② Diameter and radius dimensions
- ③ Angle dimensions for points, individual lines, arcs or pairs of lines
- ④ Arc length dimensions
- ⑤ Incremental dimension (coordinate dimension)
- ⑥ Chain dimensions
- ⑦ Coordinate tables

Furthermore, you may correct and interrupt the witness lines.

Dimensions are displayed using dimension lines and witness lines (where necessary). The objects to be dimensioned (point, line, circle etc.) may be identified by clicking onto them inside the drawing area or other by other selection criteria.

12.1.1 Dynamical Positioning of Measures (Crosshair Guidance)

Dimensions and their position in the drawing area are defined by 'Position' and 'Projection'. Projection means the angle of the measure to the objects. You can switch the projection in the drawing area between horizontal, vertical and object parallel dynamically. If the option *execcursor on* is set, you can select the position by moving the cursor (and the temporary displayed image) with the mouse. Proceed as follows to determine the measurement position.

1. Following selection of the objects to be dimensioned in the drawing area (e.g. for point dimensioning, points 1 and 2), automatic dimension positioning is activated. The position and projection can be set dynamically by moving the cursor with the mouse. Depending upon the crosshair position in different angle segments, the projection switches among horizontal, vertical and parallel to object.



2. Confirm the desired position by pressing the left mouse key.

Fixed projection: you can put the dimension at any desired angle. To do so, click the *projection* (object angle) parameter and freeze it via the fourth column.



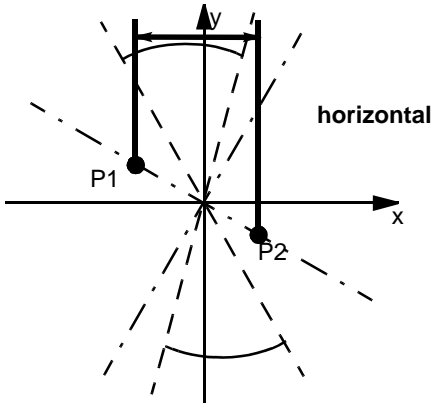
Then select a projection type in the fourth column, e.g. x, y or object parallel. The dimension position can then be dynamically changed only in the chosen projection (angle).

Dimensioning and Symbols

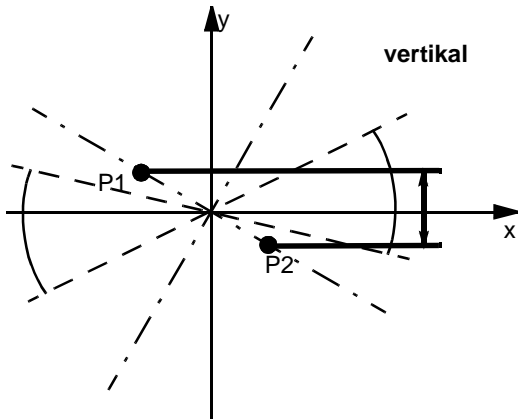
12.1.1.1 Automatic Projection Switching

All actions of the object *angle* are available for projections. Therefore it is possible to set the orientation of the dimension directly by entering a value, or you can make the orientation dependent upon other design parts, e.g. dependent upon variables. The action *automatic projection* makes dimensioning very convenient: the projection switches automatically between horizontal, vertical and object parallel. The position of the cursor relative to the objects determines the type of projection. The area around the objects to be dimensioned is divided into different angle segments that correspond to different projections. If you move the cursor from one segment to the next, the projections change from one dimension depiction to the next.

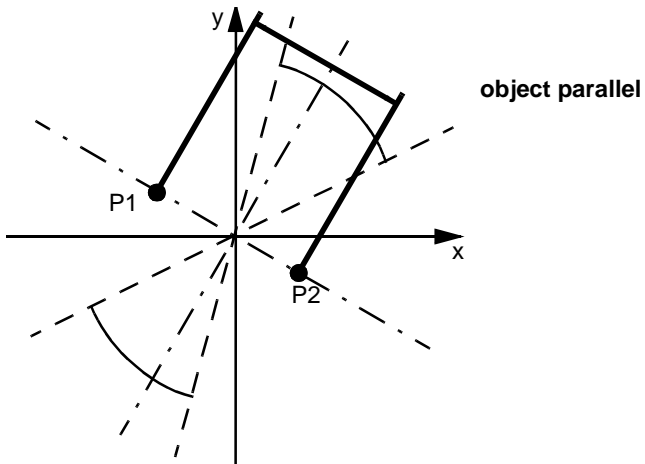
The horizontal projection can be set by moving the cursor near the ordinate (y axis).



Vertical depiction can be set by moving the cursor close to the abscissa (x axis).



The object parallel depiction is shown parallel to the line connecting dimension foot points p1 and p2. Move the cursor within the sketched angle segment to obtain this depiction.



Dimensioning and Symbols

12.1.2 Creating Distance Dimensions

The action *measure_plc1plc2* lets you create any desired distance dimensions as individual dimensions.



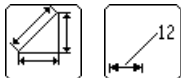
Procedure

The basic procedure for dimensioning a design is as follows:

1. If necessary, define the dimension properties prefix, tolerance, text following dimension value, font (typeface), italic angle, ratio of letter height to letter width, text height.



2. Input the parameters *projection*, *position of dimension text*.



3. Beginning of dimension: selection of the first dimensioned object inside the drawing area or via other identification mechanisms (e.g. by name).



4. End of dimension: selection of second dimensioned object inside the drawing area or by name.



5. The dimension position may be set dynamically by moving the mouse inside the drawing area (crosshair guidance). Click the left mouse key to conclude dimensioning at the final dimension position.



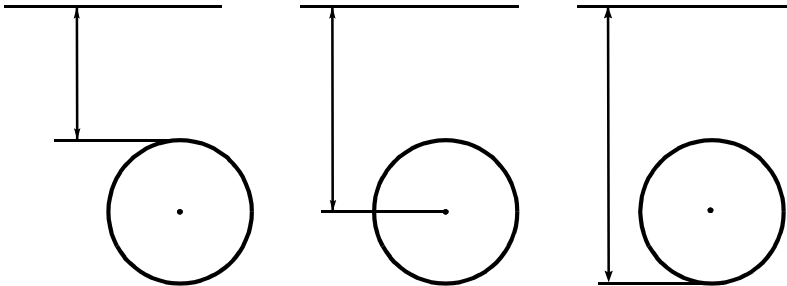
12.1.2.1 Choosing the Dimension Position of Circles (Using the Selector)

The distance dimension to a circle, you can dimension the circle inside, centered or outside. This action is provided by the parameter *selector*, which retains the necessary information and may be altered as desired.

1. In Create mode: the selection position of the object to be dimensioned determines the dimension position. Click onto the circle next to the other object, next to the centerpoint or on the side away from the other object.
2. In Edit mode: select the dimension and then click onto the *selector* icon.



3. Then press the middle mouse key inside the drawing area: the dimension reference point changes with every press of the key as shown in the following diagram:



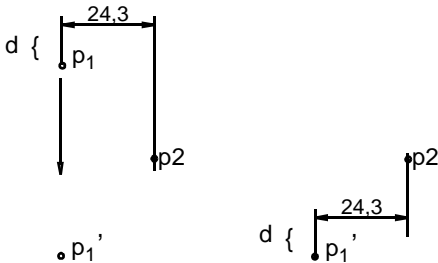
Confirm the setting by clicking the left mouse button.

Dimensioning and Symbols

12.1.2.2 Dimension Positioning with Distance Dimensions

The position of a distance dimension is always entered as the distance relative to the first dimensioned object. The dimension is always retained and the dimension position is automatically adapted to any changes in position of individual dimensioning objects.

The point first identified is decisive here: if the position of point p1 is changed (edited), the distance d to the point p1 or p1' remains the same.



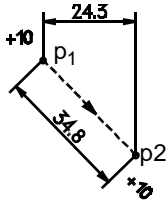
If the distance between the witness lines is too small, the dimension is always placed on the p2 side of extension p1-p2.



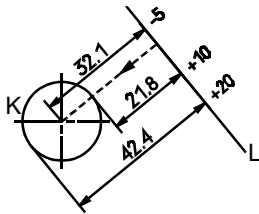
The individual dimension positions:

For x or y parallel dimensioning (points and circles), positive values are placed in positive x or y direction starting at the first object.

For object parallel dimensioning (points and circles), an imaginary connecting line between the objects is used, whereby the point first selected is considered the beginning point of the connecting line. Positive values for the dimension positions are placed at the right, in the direction of the connecting line.



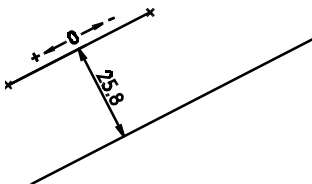
For **circles** the centerpoint is taken as start point for the imaginary connecting line. In the example below, the line 'L' was clicked first.



Concentric circles cannot be dimensioned via distance dimensioning. To do so, select the action *Diameter dimension*.

With **lines** in relation to points or circles, the plumb line through the point or center is what counts.

With two parallel lines of finite length, (or infinite length through two points), the dimension line begins in the middle of the line first dimensioned; positive values are placed to its right. If one line is endless, then the connecting line commences at the middle of the finite line.



With two endless lines each through a point (x or y parallel), the connecting line starts at the point of the first selected line.

Dimensioning and Symbols

12.1.3 Radius Dimensions

Radius dimensions can be specified for circles and arcs at any desired position. The dimension can be positioned inside or outside the object.

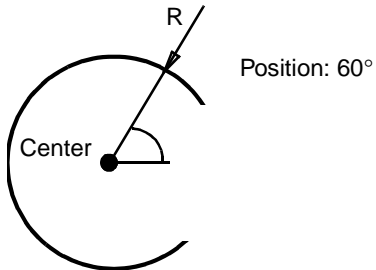


12.1.3.1 Position of Radius Dimensions

The dimension position is measured in degrees, counterclockwise from the horizontal (mathematically positive). With positive values, the dimension is positioned outside the circle or circular arc and vice versa.



With circular arcs, the system automatically adds witness lines for complete dimensioning.



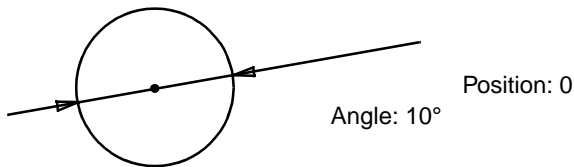
12.1.4 Diameter Dimensions

Diameter dimensions are for dimensioning circles and arcs at any desired position. The dimensions are placed in the drawing area through position and projection. The dimension can be positioned inside or outside the object.



12.1.4.1 Position of Diameter Dimensions

The distance between dimension line to the circle center at the desired angle is used as the dimension position. For dimension position 0, the dimension line passes through the circle center; positive values are placed in negative y-direction, negative values in positive y-direction.



The dimension projection, i.e. the angle position, can be adjusted dynamically by moving the cursor inside the drawing area or through use of the *projection* icon.

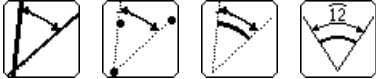


Dimensioning and Symbols

12.1.5 Angle Dimensions, Opening Angle and Length Dimensions of Circular Arcs

The following actions are available:

- angle between 2 lines
- angle between 3 points
- opening angle of circular arc
- length dimension of circular arc



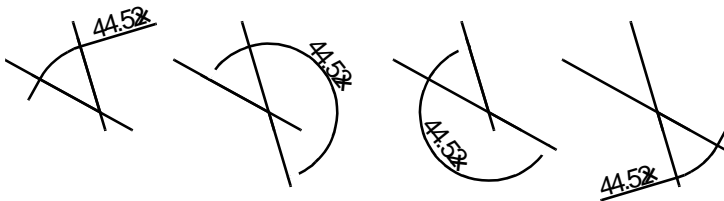
Consider the possibilities of the control box, where you can change e.g. the angle unit.

For convenient handling, you can select the position of dimensions dynamically.

If you are dimensioning two lines, you can change the angle segment in Edit mode: select the dimension and click onto the *selector* icon.



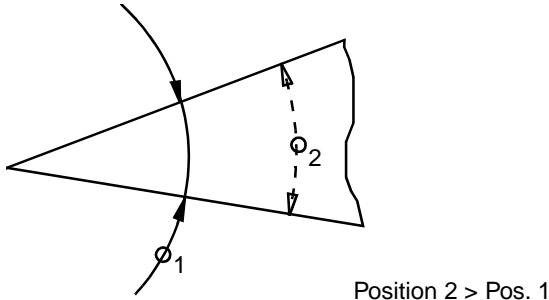
Press the middle mouse key in the drawing area, and the dimension changes as shown below:



Confirm the setting via clicking the left mouse button.

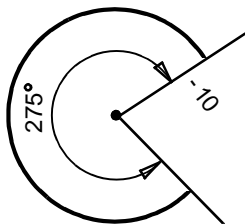
12.1.5.1 Position of Angle Dimensions

The dimension position is placed at a circular radius about the apex (e.g. intersection point of a horizontal line) with the absolute amount of the value entered. For small angles or in case of no space left, the system automatically positions the dimension outside the angle.



12.1.5.2 Dimension Position of Opening Angle and Length Dimensions of Circular Arcs

The dimension position is considered the radius of a circle about the center of the circular arc. The value entered is the difference from the radius of the circular arc to be dimensioned, so for dimension position 0, the dimension line is drawn on top of the arc. For positive position values, the dimension line appears outside the arc; for negative values, inside the arc.

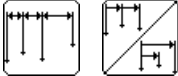


Dimensioning and Symbols

12.1.6 Chain and Referenced Dimensions

Chain and referenced dimensions are conceived strictly for dimensioning points or groups of points. The following actions are available for this:

- Chain dimension
- Referenced dimension with individual dimension lines (parallel dimensioning)
- Incremental referenced dimension



In order to dimension, a selection of points must be made from which pairs of points are given individual dimensions. The points are automatically sorted in ascending order.



Single dimensions consist of effects of the respective action. You can change the properties of each single dimension, e.g. the numbers of decimal places. The parameter of the action, e.g. the *dimension position*, is valid for all effect objects likewise.

12.1.6.1 Deleting Chain- and Referenced Dimensions

There are two possibilities to delete chain- and referenced dimensions:

- Deleting single dimensions (effects) in the Delete mode.
- Deleting the action in the Edit mode. Move the cursor to the *action* icon in the second column and press the right mouse key. Select *delete action* in the popup menu and release the key. The chain- or referenced dimension is deleted.

12.1.6.2 Chain Dimensions

Chain dimensions may be used to create individual dimensions at any angle. Activate this action via the *measure_chain* icon:



Procedure

1. Define the dimension properties in the control box and enter all parameters (prefix, tolerance, etc.)
2. Specify the reference point (zero point) of the chain dimension and all the other points to be dimensioned via selection inside the drawing area.
3. Conclude your selection by clicking onto the lower *arrow* icon.
4. Enter the dimension position.

The distance from the reference point or from the point with the smallest x or y value to the first dimension line (dimensioning of the nearest point) is used as the **dimension position**, so for dimension position 0, the first dimension line passes through the reference point or the first point; positive values are placed in positive x or y direction.

12.1.6.3 Referenced Dimensions

The procedure for creating referenced individual dimensions varies only slightly from creating chain dimensions:

- Definition of the reference point.
If you define the dimension line position to be '0', then the depiction of a coordinate dimension is created.



- The dimension 'zero' to the reference point always retains a shaded spot to mark the dimension line limit. All further dimensions are 'hung onto' this, i.e. they consist of a dimension line and a witness line. Activate this action with the *measure_relative* icon.



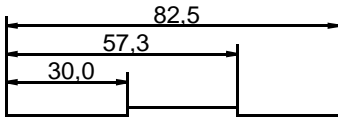
Dimensioning and Symbols

Procedure

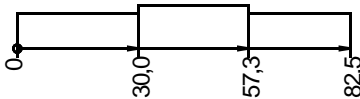
The procedure is identical to that of creating chain dimensions; you only have to define two additional parameters.

1. Define the dimension properties in the control box and enter all parameters (prefix, tolerance, etc.)
2. Define the dimension line spacing.
3. Specify the reference point (zero point) of the referenced dimension via selection inside the drawing area.
4. Select all the other points to be dimensioned.
5. Conclude your selection by clicking onto the lower *arrow* icon.
6. Enter the dimension position.

The **dimension position** is the distance from the reference point or from the point with the smallest x or y value to the first dimension line (dimensioning of the nearest point). For dimension position 0, the first dimension line passes through the reference point or the first point; positive values are placed in positive x or y direction.



Dimension line spacing $\neq 0$



Dimension line spacing = 0

12.1.7 Dimensioning Parameters and Properties

The same properties apply to all actions related to individual dimensions and the most are applied from actions likewise.

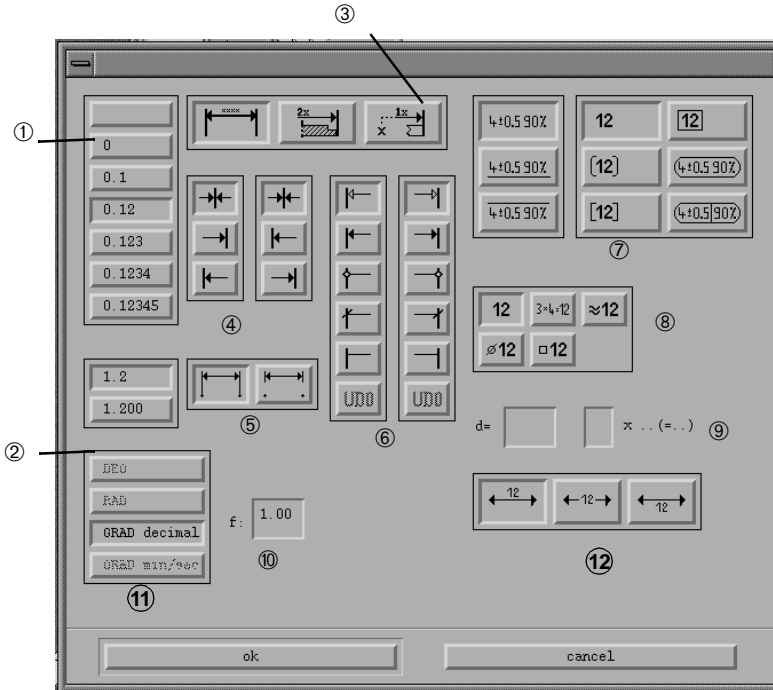
All dimensioning parameters do not have to be entered at the same time:

- They may be set by using global defaults (see the chapter *Parameter configuration*) **before** dimensions are created, and apply from then on to all newly-created dimensions. Prerequisite: the new initiation of an action.
- Dimensioning selection parameters can be set for individual dimensions **during** creation of dimensions in Create mode.
- **After** dimensions have been created, you may change individual dimensions by using the Edit mode.
- The properties and the text height could be set for a desired number of dimensions **after** dimensions have been created (action *inrect_control* in action group *inrect* in standard configuration at delivery).

Dimensioning and Symbols

12.1.7.1 Setting Dimensions

These are chosen in the dimension parameter menu. You can set the following parameters:



1. Number decimal places (0 – 5) or omission of dimension numbering
2. Suppression or display of trailing zeros
3. Selection among symmetry, half-section and normal dimensioning (may be set only for distance and angle dimensions, except for angle dimensions for individual lines)
4. Position of dimension line limit symbol, for both sides separate (forced inside, forced outside or automatic – may not be set for symmetry or half-section dimensioning)
The left button refers to the first, the right button to the second object to be dimensioned. If you click onto the left button, the same setting refers to both sides of dimensioning. If you click onto the right button, only the second object to be dimensioned is affected.
5. Display with long or short witness lines.

6. Selection of dimension line limit (also UDO's, not currently installed)
The left button refers to the first, the right button to the second object to be dimensioned. If you click onto the left button, the same setting refers to both sides of dimensioning. If you click onto the right button, only the second object to be dimensioned is affected.
7. Selection of dimension type (theoretically...)
8. Selection of special characters in front of the dimension number
9. Entry of additional dimension text (e.g. units)
10. Factor for desired reference system (e.g. mm = 1/25.4 inch)
11. Selection among display in gon, radian, decimal degrees and deg/min/sec, (only for angle dimensions)
12. Position of dimension text (above, below or atop dimension line)

You may also set global defaults for dimension properties; see *Parameter configuration*.

12.1.7.2 Prefix (Text Preceding Nominal Dimension)

Enter the prefix text here.

12.1.7.3 Tolerance

Enter the tolerance value here.

12.1.7.4 Text Following Dimension Value

Enter here the text to follow the dimension value.

12.1.7.5 Font, Italic Angle, Ratio of Letter Height to Width, Text Height



The dimension text height is interpreted as an absolute value (not to scale).

Dimensioning and Symbols

12.1.7.6 Determining the Position of Dimensions

The dimension position can be defined by entering directly the desired angle in the fourth column or by positioning the cursor with automatic projection (see chapter *dynamical positioning of measures (crosshair guidance)*).

12.1.7.7 Dimension Text Position

The dimension text can be pulled away from the dimension line by entering position coordinates (via the text entry line) with reference to the original dimension text position. X means parallel to the dimension line, Y is perpendicular to the dimension line. The new position can be sketched in Edit mode. This can become necessary when dimension number overlap.

If the angle between the line of the displaced dimension text and the dimension line is near the dimension line, then the dimension is placed level with the original dimension text position.

12.1.7.8 Position of Dimensions

You can choose 5 actions to position a dimension:

- Dimension position selected by mouse (posmeas_absolute)
The dimension position is sketched by selecting with the mouse within the drawing area. This action is used as default. Alternatively the dimension position may be input via the keyboard using a single number or a group of coordinates (default).



- Dimension position specified by a point (posmeas_onpoint)
The dimension is specified in that way, that the imaginary endless dimension line is passing through the point.



- Dimension position with distance (*posmeas_above*)
Construction of a relation between two dimension lines, one already existing, by specifying a distance between both. The distance of the dimension lines remain unchanged when the relative position of objects is changed. A dimension created with the projection *angle_measauto* (default) uses the dimension angle of the other dimension.



- Dimension position at same height (*posmeas_beside*)
Construction of a relation between two adjacent dimension lines at the same height, one already existing. The height of both dimension lines remains unchanged when objects are changed. A dimension created with the projection *angle_measauto* (default) uses the dimension angle of the other dimension.



Dimensioning and Symbols

12.1.8 Manipulation of Witness Lines

You can vary witness lines, if necessary for viewing reasons. The system adds a parameter at all chosen actions for each single change. You can change or delete the manipulator at any time.

12.1.8.1 Witness Line Correction

Witness line correction lets you correct or 'bend' witness lines, if required for clarity of depiction.

Witness line correction can be carried out only one time for each witness line.

Proceed as follows

1. In create mode, activate the action *witness line correction*.



2. Identify the desired dimension inside the drawing area.



3. Select the witness line to be corrected inside the drawing area. The selection position determines the length of the witness line from the dimensioned object to the bend.



4. Dynamic positioning is now active (if *execcursor on* is set). Move the cursor in the drawing area until the correction is satisfactory.



5. Alternatively, you may perform the correction via the keyboard: after choosing the dimension in the drawing area, select the witness line. Enter 1 for the witness line to the first dimensioned object or 2 for the witness line to the second dimensioned object, then press the <ENTER> key.



Now enter in the following sequence

- the length from the object to the bend



- the angle (positive: counterclockwise from the witness line)



- and the length of the bend via the keyboard, pressing <ENTER> after each entry.



The witness line is now corrected.

Dimensioning and Symbols

12.1.8.2 Clipping Witness Lines

If required for clarity of depiction, you may clip a witness line. In contrast to witness line correction, this operation may be carried out several times on the same witness line. After you have selected the desired dimension,

- click onto the witness line to be changed at the point where you would like to clip it.



- then indicate the length of the interruption or move the cursor inside the drawing area until the interrupted length is reached. Then click the left mouse key.



The witness line is now clipped.

Alternatively, you may specify the interruption via the keyboard.

12.1.9 Adjusting a Sketched Profile

Just like specified end dimensions, this action belongs to editing techniques. It lets you quickly specify the desired exact measurements for designs whose dimensions had been only roughly determined in preliminary sketches (see Chapter *Editing techniques*).

12.2 Coordinate Tables

Coordinate tables can contain the following:

- Coordinates of points and circle centers (in relative or polar coordinate depiction)
- Circle diameters
- Tolerances
- Notes

Besides its heading, every coordinate table contains in its first line the values of the reference coordinate system. The values of every identified object are listed in a line of the table and automatically given an identifying position number. These numbers are also displayed for the corresponding objects in the drawing area.

The entry of **points** and **circle centers** (identified as circles) is made either in ascending order via a list parameter (see the chapter on *list parameters*) or through use of a selection rectangle.

The parameters *tolerance* and *note* are effective only when the corresponding settings have been made in the coordinate table, and must be explicitly specified before the objects are selected.

If circles are contained in the selection the circle centers should be removed from the selection, since they are automatically included in the corresponding circle position lines. In this case the **diameters** should be displayed.

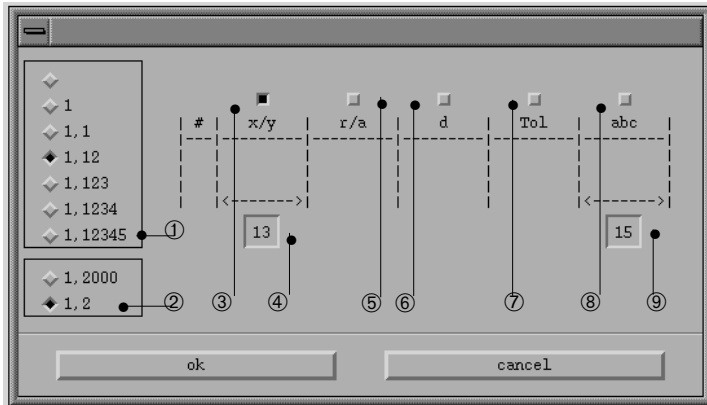
The **depiction** of coordinate tables is handled by corresponding text.



- With extensive coordinate tables, it is recommended to deactivate full depiction while working on drawings, since considerable time is required to compose the full drawing.
- The reference coordinate system to be specified must be of the same proportions and scaling as that of the geometry to be dimensioned.

The list contents as well as the configuration of the coordinate table can be changed in the Edit mode. The latter is controlled via the first parameter; after this is clicked, a menu for setting up the coordinate table appears.

Dimensioning and Symbols



1. Setting value depiction for the entire table
(Number of decimal places, 0 – 5, or no display of values)
This setting can be overridden individually by the selection parameters of individual positions.
2. Zero suppression (implicit production tolerance)
3. Display in form of relative coordinates
4. Column width setting (default: 13 characters)
5. Display in form of polar coordinates
6. Display of circle diameters
7. Display of tolerances (entered via selection parameters of each individual object)
8. Display of comments (entered via selection parameters of each individual object)
9. Setting of column width for comments (default: 15 characters)

12.3 Symbols

Symbols, in the technical drawing sense, are created by conventional drawing means with the aid of a stencil. Their primary action is to convey technological information and to enhance the clarity of a drawing.



The object *symbol* exists as an independent object and serves not only to 'collect' or group geometry. The individual geometric objects which make up a symbol are therefore not available as individual objects.

You may use the following symbols:

- General symbols
- Symbol balloons
- Welding symbols

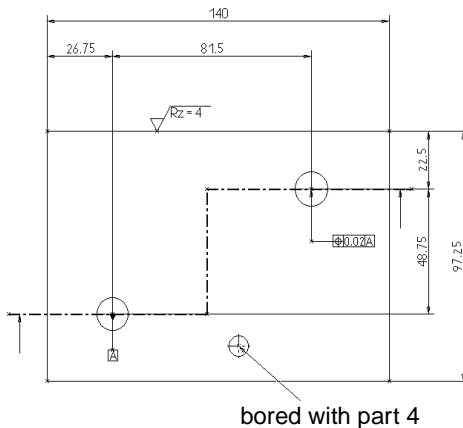
Dimensioning and Symbols

12.3.1 General Symbols

General symbols are used for the following depictions:

- Note and name lines
- Geometric tolerances
- Reference object
- Surface characteristics
- Section line characteristics
- Conical tapers

The depiction of symbols is made for the action in corresponding graphic quality. They are identifiable by this depiction either in the drawing area or via further selection mechanisms such as names.



A complete list of symbols, their actions and parameters may be found in the Online Help.

Procedure

The basic procedure for creating symbols is shown below using the example of section line characteristics.



1. Define the starting point 1 of the section line inside the drawing area.



2. Enter the desired text for this point (in the example, the letter 'A').



3. Enter the text position for this point or use the cursor inside the drawing area to specify the text position.



An arrow is drawn at the position of the beginning and end points.

4. Define further points of the section line (2., 3.). If no text is to appear at these points, then simply press <ENTER> to disregard entry of text and text position.



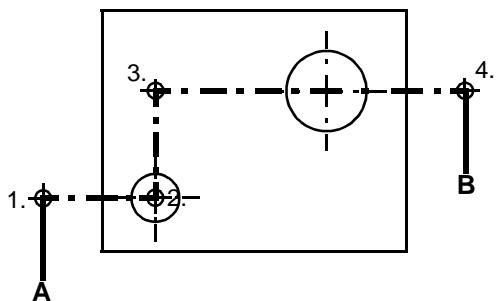
5. Select the end point of the section line 4 and enter its text and text position (in the example, the letter 'B').



6. Conclude by clicking onto the lower *list* icon to depict the section line.



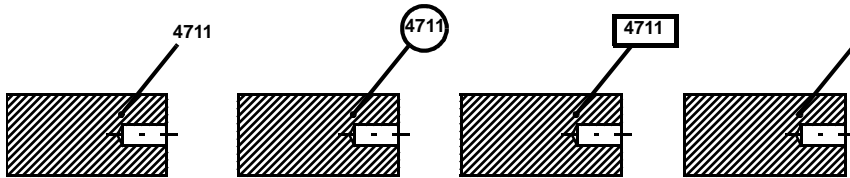
Dimensioning and Symbols



12.4 Symbol Balloons



Symbol balloons serve to record data on components of a model. The following variants of symbol balloons are available:



In the far right variant, the symbol balloon is not plotted, but the component data are stored and may be used to produce parts lists (see the chapter *Production of parts lists*).



When symbol balloons are copied, the data of an existing symbol balloon – including its position and reference number – are copied. This creates a dependency of the copy on its original. The data can be overwritten and expanded.

Dimensioning and Symbols

Procedure

1. Decide whether position or reference number are to be included.



2. Enter the position or identifying reference number along with the number of objects.



3. Determine the attributes (version, units, name, material, parts list number, comments, supplier, weight units, additional comments) and enter these into the Geometric tolerances.



4. Use the cursor inside the drawing area to locate the reference point of the symbol balloon.



5. The symbol balloon is then depicted and you may position it using the cursor.



6. Conclude positioning by clicking the left mouse key.

12.4.1 Welding Symbols According to DIN 1912

DIN 1912 welding symbols are implemented as an additional action of the object *symbol* and are identifiable as such in the drawing area or via further selection mechanisms such as names. They are kept as general as possible in order to permit any desired combination of graphic symbols and texts.

DIN 1912 welding symbols consist of

- One arrow line per weld
- One reference line (sometimes with dashed line)
- One or more graphic symbols
- Several dimensions and
- customary (supplementary) notes

Welding is often performed in several layers so that the graphic symbols appear above one another in drawings. For this reason, you may freely select the position of symbols relative to the reference line (through definition of a parameter) and therefore combine all graphical symbols. Welding symbols are displayed so that the graphical symbols are aligned to one another depending upon the longest previous text.

Dimensioning and Symbols

Procedure

1. Define the path and type of the weld, e.g. assembly weld.



2. Enter the position at which welding is to be performed. In doing so, specify whether the weld is to lie above or below the reference line. The dashed marking line does not cover any graphic symbols.



3. You may select any number of reference points (but at least one) from the resulting list. An arrow line will then be drawn to these points.



4. Enter the position of the reference line.



The entry coordinates are relative to the first point of the defined group of points. The x distance to this first point determines the fold direction, i.e. if the distance ≥ 0 , then the welding symbol will be drawn to the right; if the distance < 0 , it is drawn to the left.

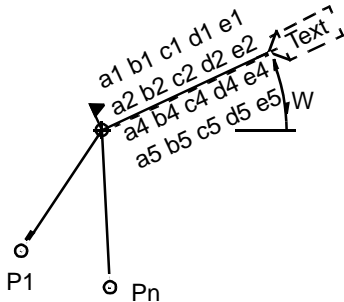
5. Enter the angle of the reference line to the x axis of the active coordinate system.



6. Define the position of the symbol, e.g. below the reference line, if the arrow line points to the upper workpiece surface.



7. Enter the following information for clear depiction of symbols:



Weld thickness (a1 – a5 in the figure), e.g. weld thickness s in mm; for fillet-welds, the weld thickness a in mm.



Symbol for the weld (b1 – b5), e.g. fillet weld. Input via selection from the geometric tolerances.



Weld length (c1 – c5), e.g. number of weld lengths x weld length



Offset symbol (d1 – d5),



Dimensioning and Symbols

Weld spacing (e1 – e5)



Fork: Additional information such as evaluation group, welding position, additional welding material with the *fork* icon.



You can choose between no fork, closed fork with one line and a fork with n lines as a list of strings (text entry). Each text entry is put onto its own line.

13 Standardization

The following standardization possibilities can be used:

- User-defined objects and user-defined actions
- Construction and calculation of variants by means of variables
- Supply of parameters by means of tables



The existing version of user-defined values (UDV) has not be released.

13.1 User-defined Object Types and User-defined Actions

UDOTs or UDAs can be defined in ***EUKLID Design***. They are created interactively as models or defined via AQL. Standard geometry, standard calculations, standard functionality as well as their nesting and linking can be carried out with their aid. Their use rationalizes construction and production as construction solutions do not have to be developed each time.

UDOs and UDAs can replace procedures out of conventional CAD systems as they can be controlled via the parameterization of their environment (other objects of the current model).

Example

A user-defined object, *tapped hole*, can stand vertically on any entrance surface by means of a corresponding action, as functionally required.

As is the case with system-defined object types and actions, UDOTs and UDAs are introduced into the models via their own icons in the menu (instance creation). These icons can be designed with the icon editor.

A straight forward representation and simple processing of the UDOs is possible with the *structure* representation type. In it all UDOs and layers are displayed in the hierarchy and can be identified. (see chapter „[Visualization of Layers](#)“ on page 14-18)

User-defined objects can be queried and edited in the data structure and UDAs stored in the data structure via the result or effect object.

User-defined Object Types (UDOT)

With the use of UDOTs new object types can be introduced into ***EUKLID Design***. Every UDOT has a clear identification number and is therefore not interchangeable with other UDOTs. UDOTs are available via their names in AQL. You can identify UDOs like other objects in models.

UDOTs are used for structuring, e.g. parts and module grouping or grouping according to media flow in the case of pipelines. The meaning of a UDOT is determined by the

constructor. Grouping need not be stipulated beforehand and can be introduced afterwards.

Example

Screw

It is constructed out of the existing object types which can be system-defined or user-defined (nesting). These individual objects can also be identified in entities of UDOTs. The object specification can also include connections between these objects. It is thus possible to freely define internal parameterization for user-defined objects.

Example

Screw head made of circle and polygon with 6 corners, center point of the circle, angle 0° and 0.6 times radius of the circle.

The properties specified for a UDOT do not have to be additionally specified as a parameter in the case of UDAs which produce this UDOT as a result object.

Example

Nominal diameter and length of the screw.

The properties are queried independent of the importing in UDA during the creation of entities for UDOTs. These properties are entered in the same way as properties for system-defined objects.

A UDOT can inherit the properties of existing UDOTs so that it does not have to be defined again.

Example

The UDOT *screw* can inherit the properties from the UDOT *part*. The UDOT-class *screw* is thus derived from the UDOT-class *part*.

Aspects which were not known during the definition of the UDOT can also be subsequently introduced. In this way production aspects (e.g. NC parameters) can be subse-

Standardization

quently introduced into the model without preparation work on the part of the constructor.

Example

A UDOT *screw* can be subsequently declared as a UDOT *clamping screw*. Centrifugal-moment is the property of a clamping screw. The UDOT *screw* then belongs to the UDOT classes *part*, *screw* and *clamping screw* and can be identified under these three name types.

UDOTs are available in the model in which they were defined. They can be created in all models by being saved.

UDOTs can be created in models as an absolute instance. However, UDAs can also be defined to place UDOTs. The UDOT is thus a result object of these UDAs.

Example

Screw at center point (*screw_point*) and at intersection (*screw_intersection*)

UDOs are evaluated by the action introduced into the model and are recalculated if the interactive parameter is changed or a change is effected via AQL.

User-defined Actions (UDA)

Principally, UDAs do not differ from system-defined actions. They have the same characteristics which are as follows:

- Name
- Icon
- Attributes (e.g. *drop*)
- Result object
- Effect objects
- Parameter
- Parameter attributes (*execcursor*,...)

AQL programs:

- Parameter types (*star*, *alternative*)
- *Pre-/Post* action routines
- *Pre-/Post* parameter routines

UDAs can be used as follows:

- for the layout of frequently used user actions
- for placing UDOs
- to manipulate parameter objects and UDOs

Placing UDOs

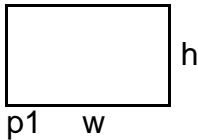
Example

A new object type *rectangular* with the property *color* is to be created. There are different possibilities for creating entities for this object type:

Action: *box create_box_wh*

Parameters:

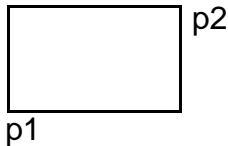
- Point *p1*
- Length *w*
- Length *h*



Action: *box create_box_p1p2*

Parameters:

- Point *p1*
- Point *p2*

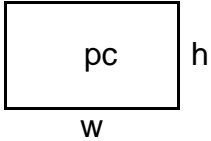


Standardization

Action: *box create_centerbox*

Parameters:

- Length w
- Length h
- Point pc



Manipulation of parameter objects and UDOs

Owing to the possibility of linking UDAs with AQL programs, it is possible to solve complex applications, e.g. layout variants (see volume *Open Architecture*).

13.1.1 The Creation of User-defined Objects in Models

There are two possibilities of creating UDOs in constructions:

- absolute
- with the aid of a user action (UDA)

UDOs can be interactively created in models as is the case with other supplied object types, when the following preconditions are fulfilled:

- The UDOT icon is in the main menu.



If the configuration does not contain a required UDOT, this can be explicitly reloaded and is then available for instance processing.

- An action is available which creates the UDO as an effect or result.
- The UDOT is loaded.

If the UDO is a result of a loaded UDA, then the UDOT is automatically reloaded.

Creating an absolute Instance of a UDOT

Only the properties have to be entered to create an absolute instance. No entries are necessary, if there are no defined properties or if they are selectable properties.

1. Click the corresponding menu icon in the *Create* mode, if the menu is not already set.
2. Click the corresponding action group icons in the first column of the menu.
3. Click the corresponding UDOT icon in the second column of the menu.
4. Enter the properties which have been requested.

Creating a UDO with the Aid of a UDA

The required parametric behavior is guaranteed during the creation of UDOs with UDAs:

1. If the menu is not already set, click the menu icon in the menu special range in the *Create* mode under which the UDOT is located.
2. Click the required action group in the first column of the menu.
3. Click the required UDA icon in the second column of the menu.
4. Enter the requested properties.

Standardization



- To terminate the creation of a UDO, click another icon.
- UDOs can be created with AQL via an appropriate UDA or absolute with the following action (see volume *Open Architecture*):

```
<name>_absolute(<Properties>)
```

13.1.2 Changing User-defined Objects in Models

As is the case with system-defined objects, parameter values of a UDO in a model can be changed in the *Edit* mode via the user icon or quick access (see main chapter „[Updating Objects](#)“ on page 8-1).



UDOTs can be accessed via base object or top object:

- ☞ Choose *sub objects* in the popup menu in the second menu column of the corresponding base object icon. The subobjects are displayed in the second menu column.

To return to the previous hierarchy level, select *base object* in the popup menu of the subobject icon in the second menu column.

To return to the top hierarchy level, select *top object* in the popup menu of the subobject icon in the second menu column.

13.1.3 Using User-defined Action in Models

UDAs can be used as all supplied actions interactively in models when the following preconditions have been fulfilled:

- The UDA icon is located in the main menu.
- An action is available which the UDA uses.
- The UDA is loaded.

If the UDO is the result of the loaded UDA, then the UDOT is automatically reloaded.

Proceed as follows:

1. Click the corresponding menu icon in the *Create* mode if the menu is not already set.
2. Click the corresponding action group icon in the first menu column.
3. Click the corresponding UDA icon.
4. Enter the parameters.



Each UDA out of AQL can be called as follows:

`<name> (.)`

Pay attention to name collisions!

It is sensible to follow a convention when giving names. The convention upon which the whole system is build is as follows:

`<what>_<how> (...)`

`<what>` What is being created/processed e.g. the object type name *line* for line

`<how>` How is something being created/processed e.g. *pointpoint* for line between two points

The action is called *line_pointpoint*.

Standardization

13.1.4 Conception of Object Types

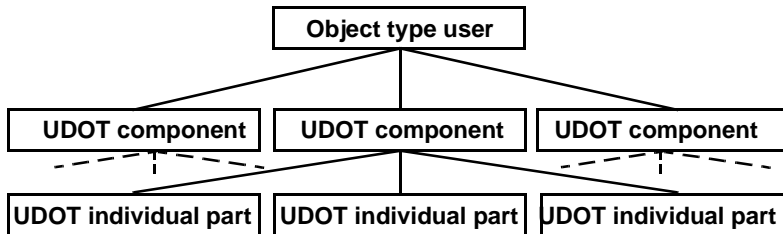
UDOTs should be created by experienced CAD constructors as high demands are placed upon the design functionality.

The following aspects should be taken into consideration during the conceptual formulation of UDOTs:

- class affiliation
- structure
- properties

Class Affiliation

The top class of the hierarchy is the system-defined object type *user*. Every UDOT is a subtype of this class.



Every UDOT can be defined as a subtype of an existing UDOT. The objects, actions and characteristics of the base class then become objects, actions and characteristics of the new UDOT, e.g. properties and views. A subtype must be an extension of a base type.

Class allocation is not only possible during the definition of the UDOTs for all instances, but also subsequently for every individual instance. The objects of the base and subtypes must be the same type and have the same identification number.

Structure

The following points must be observed:

- correct and generally valid definition of dependencies
- optimization of loading time through efficient design

- planned property allocation
- library structure in the file system
- clear tree structure of nested user objects

Properties

Properties are characteristics of UDOTs which all objects of this type have, e.g. line mode of a line. You control information which is required to create an instance of the UDOT. The following types can be assigned as a property:

- objects
- parameters
- properties



Properties can be filed in variables, e.g. for directed access after breaking up UDOs.

13.1.5 Defining Object Types

The following list gives the development process of UDOTs. Certain development steps can be left out in individual cases during the execution:

- allocation in class structure
- designation
- designing the UDOT
- creation of representation groups (optional)
- determination of the UDOT icon (optional)
- appending of properties (optional)
- allocation of attributes (optional)
- saving the UDOT (optional)
- appending the UDOT icon in the menu (optional)
- testing
- saving the configuration (optional)



A UDOT definition can be determinated at any time. To do so, choose *abort definition* in *Udo* menu.

Standardization

13.1.5.1 Allocation in Class Structure and Designation

Here there are several possibilities:

- A UDOT can be defined which is a subtype of the object type *user*.
- A UDOT can be defined which is a subtype of the system-defined object type *user* and which is restarted on a loaded model in the construction.
- A UDOT can be defined which is a subtype of a loaded UDOT.

Subtype of the System-defined *user* Object Type

- ☞ Choose *init definition>new* in *Udo* menu. The prompt *Name: ""* appears in the text area for entering the name of the UDOT.
- ☞ Enter the required name in the text area. An empty window with the UDOT name is opened.

Subtype of the System-defined *user* Object Type out of the Model

- ☞ Choose *init definition>from model* in *Udo* menu. The dialog box for entering the UDOT name and the original model is opened. The active model is preset.



- ☞ Enter the UDOT name in the text area.
- ☞ Click the required original model in the list field if the active model is not desired as original.
- ☞ Click *ok*. The model window now becomes a UDOT window.

Subtype of a Loaded Base Type

- ☞ Choose *init definition>from udot* in *Udo* menu. The action *udo_master_of_udo* is displayed in the main menu and the prompt *Name: ""* appears in the text area.
- ☞ Enter the name of the subtype in the text area. The model box is opened.



- ☞ Click the base type in the list box.
- ☞ Click *ok*. A window with the base type is opened.



UDOTs which have already been defined can adopt type characteristics from UDOTs (see section [„Adopting Object Type Definitions for User-Defined Objects“ on page 13-22](#)).

Standardization

13.1.5.2 Designing Object Types

After determining the class allocation and the name, **EUKLID Design** opens a window with the UDOT name which, depending on the initialization, is either empty or contains the base type or the original model.

- ✎ Design or complete the UDOT like every model with the aid of the predefined actions.
- ✎ Create a group out of the objects to be represented for each view if required in addition to the standard representation *full*.
- ✎ Close the design phase with the menu command *close definition* in *Udo* menu, if you only want the complete representation of the UDOTs in models. The UDOT definition box is opened:

The screenshot shows a dialog box titled "UDOT definition" for an object named "screw". The window is divided into several sections:

- Name:** A text field containing "screw".
- UDOT-Icon:** A small square icon with the text "screw" inside.
- Attributes:** A list of checkboxes: ☐ *implicit*, ☒ *selectable*, ☒ *normal*, ☒ *graphical*, and ☐ *init_uda*.
- Properties:** A large text area containing "< no properties >".
- Visibilities:** An empty rectangular box.
- Is_Subclass_of:** A list box containing the text "user".
- Buttons:** "ok" and "cancel" buttons at the bottom.

The *Name* field is already occupied by the UDOT name. The field *UDOT icon* is already occupied by the UDOT name. The attributes *selectable* and *normal* are preset. The base class is displayed in the list box *Is_Subclass_of*.

This dialog box has the following tasks:

- changing the UDOT name
- calling up the icon editor
- appending properties
- selection and display of attributes
- display of views
- display of the base class
- immediate forming of an instance in a UDA



If the UDOT name is to be changed, please correct it in the *Name* field.

13.1.5.3 Defining Views for User-defined Object Types

The possibility to represent UDOTs in models is given by the views, e.g. all dimensions for UDOTs can be blended out. The degree of detailing during the design phase can thus be kept small (and thus usable for redraw etc.) and the design can be made more detailed for a plot by changing the views.

The views are described by grouping objects during the UDOT definition. As is the case with layers, there is a standard view *structure* for UDOTs in the structure view port.

If a view is not defined, the system uses the *full* view of the base class *user*. It displays all objects defined in the original model as far as they are visible.

- ☞ Create a group or a layer out of the objects to be represented for each view. The name of the group or layer is used for the designation of the respective view.
- ☞ Close the construction phase with the menu command *close definition* in *Udo* menu. The UDOT definition box is opened. This dialog box lists all group or layer names in the list box *Visibilities*.
- ☞ Click the groups which are to become views in the list box *Visibilities*.

Standardization

13.1.5.4 Determining Icons for User-Defined Object Types

EUKLID Design allocates an icon for each UDOT with the UDOT name which is displayed in the menu after the icon has been appended. Instead of using a system-defined, you can design an icon which is only used for this UDOT.

- ☞ Choose *icon* in the popup menu field *UDOT Icon* in the UDOT definition box. The icon editor is opened.
- ☞ Create the desired icons (see section „*Creating User Icons*“ on page 13-46).

13.1.5.5 Appending Properties

Any number of properties of any type can be appended. New properties are added at the end of the list. Both objects and also (enumeration) values can be determined as properties and can be set in an optimum way. It might be necessary to go down the data structure until the object or (enumeration) value to be appended has been reached. If the property to be appended is an object, there are then two possibilities:

- The parameter becomes a property.
- The object becomes a property.

If the property to be appended is a (enumeration) value, only the parameter can be turned into a property.



An appended property can be removed from the property list by using the menu command *delete* in the popup menu of field *Properties*.

Appending Objects as Properties

Proceed as follows:

1. Choose *append* in the popup menu of field *Properties*. The definition box is closed and **EUKLID Design** changes to *Browse* mode. The data structure can be searched in this mode, as in the *Edit* mode, but values cannot be changed.
2. Click the icon of the object group to which the object which is to be appended as a property belongs, in the menu column.
3. Identify the object which is to become a property in the UDOT window. The creation action and, if applicable, the manipulation actions appear in the main menu.
4. Choose *set property* in the popup menu of the action icon. The UDOT definition box is opened. The property is now appended.

Appending Properties and Parameters as Properties

Proceed as follows:

1. Choose *append* in the popup menu of field *Properties*. The definition box is closed and **EUKLID Design** in *Browse* mode.
2. Click on the object group icon to which the object is allocated, its property or parameter to be appended as property in the main menu.
3. Identify the object whose property or parameter are to become property in the UDOT window. The creation action and, if applicable, the manipulation actions of the object are displayed in the main menu.
4. Click on if not default, the desired action icon.
5. Click on the desired property or parameter icon.
6. Choose *set property* in the popup menu the parameter icon. The UDOT definition box is opened. The property is now appended.



An appended property can be removed from the property list with the menu option *delete* in the popup menu of the field *Properties*.

Determining Property Icons

EUKLID Design allocates the corresponding system-defined icon for each property which can occur more than once depending on the constellation. Instead of using the system-defined, you can design an icon which is only used for this property.

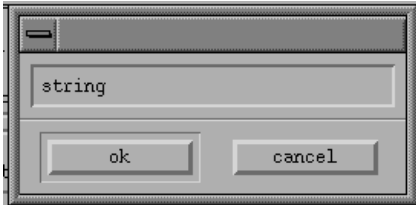
- ☞ Choose *icon* in the popup menu the corresponding property icon in field *Properties*. The icon editor is opened.
- ☞ Create the desired icon (see section „[Creating User Icons](#)“ on page 13-46).

Standardization

Designation of Properties

EUKLID Design allocates the corresponding system-defined name for each property which can occur more than once depending on the respective constellation. In this case the properties must be given their own names so that, e.g. AQL programs can access the individual properties.

☞ Choose *name* in the popup menu of the corresponding property icon in field *Properties*. The following dialog box is opened:



The text area is already occupied by the system-defined name.

☞ Enter the required name.

☞ Click on *ok*.

Optional Setting of Properties

(Enumeration) value properties and simple object properties can be set optionally.

☞ Choose *set optional* > *yes* in the popup menu of the corresponding property icon in field *Properties*. The corresponding icon in field *Properties* is displayed shaded.

Changing the Position of Icons

It is possible to change the sequence in which the properties for the UDO instantiation are requested by the system.



This can lead to incompatibilities.

☞ Choose *move* in the popup menu of the property icon in field *Properties*. The property icon is highlighted and the message *CHOOSE NEW POSITION* appears in the dialog box.

☞ Click on the icon in whose position the icon is to be added.

Combining Properties in dialog boxes

The entry of value properties of UDOTs which follow each other can be combined in a dialog box.

☞ Choose *combine>yes* in the popup menu of the icon of the first property to be combined in the field *Properties*.

The system opens a dialog box for the entry of properties for the creation of a corresponding UDO in a model.

13.1.5.6 Allocation of Attributes for User-defined Object Types

The attributes determine the instance type resulting out of the UDOT. The following types are possible

implicit	In this case it is an implicit object
selectable	The UDO is selectable in the graphic range
normal	Normally only the UDOT itself becomes an instance. If a UDO is changed in a model or the contents are extended, the object and the action become an instance. This is an instance type by default.
graphical	This instance does not copy any objects into the instance. The UDOT is transformed for the representation of the instance. The instance only contains one object, which determines the transformation.
init_uda	A UDA is to be created immediately which contains the UDO as an instance. This is only possible when no properties have been defined. The UDA contains the UDO as a result object.

Standardization

13.1.6 Saving User-defined Object Types

If a UDOT is not only to be made available in an active model (model-internal UDO) then the UDOT definition must be saved in an external file (see section *„Saving User-defined Object Types and User-defined Actions“ on page 13-54*).

13.1.7 Appending Object Type Icons in the User Interface

UDOs can only be created in models if the appropriate icons are integrated in the user interface (see section *„Appending Object Type Icons and Action Icons to the User Interface“ on page 13-55*).

13.1.8 Saving the Configuration

UDOs can only be created in models if the configuration is saved with the appended UDOT icon (see section *„Appending Object Type Icons and Action Icons to the User Interface“ on page 13-55*).

13.1.9 Loading User-defined Object Types

UDOTs must be loaded before models can be created. There are different possibilities for doing this:

- Automatic loading during the start
- Automatic loading with the model file
- Manual loading

By quering the subobjects of the user object type it is possible to check whether the object is loaded. If an object type is loaded, it appears in the tree structure of the base element *user*. Precondition for this is a action group which contains the object type *user*. The user object can be directly accessed by acceptance in an action group.

Automatic Loading at the Start

When the system is started the ***EUKLID Design*** loads the menus and action groups which contain the configuration file. ***EUKLID Design*** only loads the actions and objects which were integrated into the user interface and only then when these (by clicking the action group) have been requested. This reduces the starting time of the system when the menus contain a large number of UDOTs or UDAs.

Automatic Loading with the Model File

EUKLID Design automatically loads a UDOT with the model file in instances where UDOTs exist.

Manual Loading

If a desired UDOT is not available in the main menu, the corresponding UDOT definition file can be loaded in the main memory:

- ☞ Choose *Load* in the *Udo* menu. The file box is opened.
- ☞ Enter the required file name.

Another alternative is the action *user_short_create* which loads the UDO and simultaneously appends it under UDO in the menu and creates an instance.

Standardization



- The AQL function *user_symbol* contains this action for compatibility reasons (*imp*).
- The UDOT icon can be integrated into the user interface with the menu command *edit action group*.

13.1.10 Adopting Object Type Definitions for User-Defined Objects

With the relation *is a* the definition of another loaded UDOT can be added to a UDO in the model. The UDO thus becomes a subclass of the UDOT and can be identified as such.

EUKLID Design copies the objects from UDOT into UDO. The properties of the UDOT are queried additionally.

- ☞ Choose *Is a* in *Udo* menu. The *udo_is_a* action is represented in the menu.
- ☞ Select the UDO in the drawing range. The UDO selection box is opened.
- ☞ Click on UDOT name in list box.
- ☞ Click on *ok*.
- ☞ Enter the properties for the UDOT. The UDO is now a subclass of the UDOT.



Layers can also be defined as a subclass of UDOT (see main chapter Working with Models, chapter [„Adding a User-defined Object Definition to a Layer“ on page 5-35](#)).

13.1.11 Editing Object Type Definitions

UDOTs can be modified. Attention must be paid that every modification has an effect on the models which contain a UDOT instance which is not broken up. These models contain the updated UDO when reloaded.

- ☞ Load the UDOT which is to be edited.
- ☞ Choose *Edit definition* in *Udo* menu. The model box is opened.
- ☞ Choose the name of the UDOT to be edited in the list box.
- ☞ Click on *OK*. A window with the corresponding UDOT design is opened.
- ☞ Change the UDOT design if necessary.
- ☞ Choose *Close definition* in *Udo* menu. The UDOT definition box is opened.
- ☞ Change the characteristics of the UDOT if necessary.



- Objects must not be deleted as every model which the UDO contains, could contain references to them.
- If file and link structures are changed, then it is possible that the file locator tables have to be adapted in order to load a model which contains the UDO.

In these cases a new UDOT must be created with another name.

EUKLID Design refuses certain changes to the definition, e.g. appending from properties if an instance has already been created in a model. This lock-out can be avoided during the development phase of a UDOT by means of the start option *-admin*. Depending on the modification, however, models which contain the instances of this UDOT cannot be loaded.

13.1.12 Access to Objects of User-defined Objects

Models only contain instances of UDOs in the data structure as a reference to the UDOT. It is therefore not possible to access the individual objects of the UDO directly.

The procedure differs depending on the conceptual formulation:

- Individual objects can be accessed in the short term via selection in user objects (see main chapter System Handling, section „[Selection in User Defined Objects](#)“ on [page 4-26](#))
- All objects can be permanently accessed after converting the UDO into a layer. This action is reversible, see section „[Converting User-defined Objects into Layers](#)“ on [page 13-24](#)).

After the layer conversion of nested user objects, the objects are separated into UDOs and existing individual objects of the respective lower nesting level. Nesting levels which follow each other can be converted into layers until only individual objects are left over.

13.1.13 Converting User-defined Objects into Layers

UDOs can be converted into layers in models. In this way information can be passed on to the converted UDO and then the layer can be converted back into a UDO.

A UDO can be converted into a layer in the following way:

- ☞ Choose *convert in layer* in *Udo* menu.
- ☞ Identify the UDO in drawing area. A layer box is opened.
- ☞ Click on *active* if the layer is to be active.
- ☞ Click on *ok*.

The resulting layer is given the name *<UDO name>_instance* and is allocated in the hierarchy of the layer in which the UDO was.

13.1.14 Conception of Actions

As is the case with UDOs, UDAs should be created by experienced CAD constructors.

The following aspects should be taken into consideration during the conception of UDAs:

- structuring
- parameterization

Structuring

The following points must be observed:

- method of behavior description
- correct and generally valid definition of dependencies
- optimization of the loading time through efficient design
- carefully planned allocation of parameters
- machine-independent path names/file locator

There are 3 possibilities of describing the UDA behavior:

- via AQL (see section [„Appending AQL Programs to User-defined Actions“ on page 13-33](#))
- via model
- as a combination of both

As is the case with every model, this UDA model consists of a network of objects and actions which determine the interactions of the UDA with its parameters and the result object/effect objects. The actions determine the dependencies between the objects. As a result of the description of actions or objects, the behavior of a UDA can be defined quicker and more specifically for influencing the result objects or parameter objects.

Standardization

Parameterization

Parameters control the information which is required to create an instance of the UDA.

Parameters can be used to modify the result object or parameter or can be used as an input for an external AQL program.

The following types of parameters can be selected:

- ☐ objects
- ☐ parameters
- ☐ properties

13.1.15 Defining Actions

The following list gives the development process of UDAs. Certain development steps can be left out in individual cases:

- determination of the original model and designation
- design of the UDA
- design of the UDA icon
- determination of result object
- determination of effect objects
- appending of parameters
- allocation of attributes
- saving the UDA
- appending the UDA in the menu
- testing
- saving the configuration



A UDA definition can be terminated at any time. Choose *abort definition* in *Uda* menu.

13.1.15.1 Determining the Master Model and Designation of User-defined Actions

There are several possibilities during construction:

- A UDA can be redefined. A new UDA model is used, e.g. as an interface to an AQL program.
- A UDA can be defined which has been derived from an existing model.
- A UDA can be defined which has been derived from an existing UDOT. All objects and actions of the UDOT are adopted in the newly created original model.

Redefinition

- ☞ Choose *init definition>new* in *Uda* menu. The prompt *Name: ""* appears in the text area for entering the name of the UDA.
- ☞ Enter the required name in the text area. An empty window with the UDA name is opened.

Definition from model

- ☞ Choose menu command *init definition>from model* in *Uda* menu. The dialog box for entering the UDA name and the original model is opened. The active model is default.



- ☞ Enter the required UDA name in the text area.
- ☞ Click the required original model in the list box if the active model is not required as the original.
- ☞ Click on *ok*. The model viewport is UDA viewport.

Standardization

Definition from UDOT

- ☞ Choose *init definition>from udot* in *Uda* menu. In menu the action *uda_master_of_udo* displayed and in text area the Prompt Name : "".
- ☞ Enter the desired name of the UDA in the text area. The model box is opened.
- ☞ Click the name of the desired UDOT in the list box.
- ☞ Click *ok*. A window with the UDOT model is opened.
The model name is changed to the UDA name.

13.1.15.2 Designing User-defined Actions

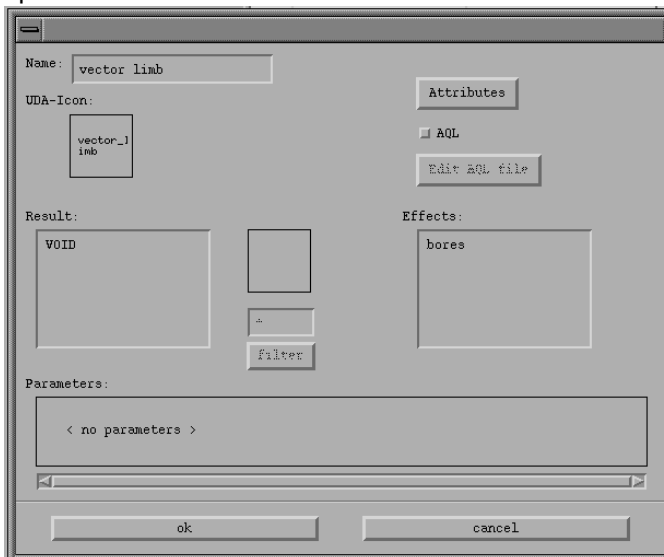
After determining the original and the name, **EUKLID Design** opens a window with the UDOT name which, depending on the initialization, is empty, contains the UDOT or the original model.

- ☞ Design or complete the UDA like every model with the aid of the main menu.
- ☞ If effect objects are also to be created, then a group out of the objects must be created to be used as an effect object or objects.

Example

A UDA *vector_partial_circle* is to be defined to bring in a flange which creates a variable number of bores as effect objects.

- ☞ Create 2 variables for the copy. $n=5$ and the copy angle $\alpha=360/(n+1)$.
- ☞ Create the flange and the first circular bore.
- ☞ Create the copy vector as *vector_rotator* with rotation point *center point of flange* and angle from variable *alpha*.
- ☞ Copy the circular bore with the action *group_copy* with copy, variable *n*, group name *bores*, vector *vector_rotator*. The group name *bore* appears in the list box *effects*.
- ☞ Close the design with *close definition* in *Uda* menu. The UDA definition box is opened:



Standardization

The field *Name* is already occupied by the UDA name. The field *UDA Icon* is already occupied by the UDA name icon. The field *Result* is already occupied by *void*.

This dialog box is used for the following tasks:

- Changing the UDOT name
- Calling up the icon editor
- Providing Online help
- Setting result objects
- Appending parameters
- Selecting attributes (to control the behavior)
- Determining effect objects



If the UDA name is to be changed, correct it in the field *name*.

13.1.15.3 Determining Icons for User-defined Actions

EUKLID Design gives an icon with the UDA name for every UDA, which is represented after the icon has been appended in the main menu. Instead of using the system-defined, you can design an icon which is only used for this UDA.

☞ Choose *icon* in the popup menu of field *UDA Icon* in UDA definition box. The icon editor is opened.

☞ Create the desired icon (see section „[Creating User Icons](#)“ on page 13-46).

13.1.15.4 Determining Result Objects for User-defined Actions

Depending on the creation of the result object, there are two possibilities of determining the result object:

- The result object is object of UDA model.
- The result object was created by the AQL program.

Result Object in UDA Model

1. Choose *set result object* in the popup menu of result icon field. The UDA definition box is closed and **EUKLID Design** in *Browse* mode.
2. Click on the requested icon in the first menu column.
3. Identify the result object in UDA model viewport.
4. Choose *Set result object* in the popup menu of the action icon in second menu column. The result object icon is displayed in result icon field.

AQL Program-created Result Object

1. Click on the AQL field.
2. Click on the name of the desired objects in *Result* field.

13.1.15.5 Determining Effect Objects for User-defined Actions

Depending on the creation of the result objects, there are two possibilities of determining the effect object:

- as object of UDA model (see section „*Designing User-defined Actions*“ on page 13-29)
- by selecting the effect group
- via AQL program (see section „*Appending AQL Programs to User-defined Actions*“ on page 13-33)

Standardization

13.1.15.6 Allocation of Attributes for User-defined Actions

The UDA attributes determine the type of instance of the UDA. The following types can be selected:

drop The UDA is not taken up in the data structure after being carried out.

tmp The UDA can be carried out during the execution of another action. The active action is not terminated.
Only for actions which do not change the data structure.

AQL An AQL program is called up before and after the execution of an UDA, i.e. before and after the UDA model is/was supplied with parameters.

Edit AQL file A text editor indicates the supplied pattern AQL program (see section [„Appending AQL Programs to User-defined Actions“ on page 13-33](#)).



A different editor can be called up by modifying the file *#/basic/aql/edit.aql*.

no_protocol To prevent double protocolling during certain actions.

suppress_properties To suppress certain properties of the result object (only with AQL).

pre action An AQL *pre_action* function is run (only with AQL).

post_action An AQL *post_action* function is run (only with AQL).

name_by_action The name function is not to be possible on the user interface as the action names its own result (e.g. variable, tables).

13.1.15.7 Appending AQL Programs to User-defined Actions

By coupling UDAs with AQL programs the UDA behavior can be determined. Apart from the interactive characteristics to be entered, the following characteristics of UDAs can be defined:

- Parameter types (*star*, *alternative*)
- Parameter attributes (*draft*,...)
- Pre /Post actions and parameter routines

Proceed as follows:

- ☞ Choose *init definition>new* in *Uda* menu.
- ☞ Enter the desired UDA name in the text area. A new UDA model view port is opened.
- ☞ Choose *close definition* in *Uda* menu. The UDA definition box is opened.
- ☞ Determine the UDA icon.
- ☞ Click on *AQL*. The field *Edit AQL file* is selectable and in *result* list box the system and user defined objects displayed.
- ☞ Click on *Edit AQL file*. An editor viewport with the following AQL file is opened:


```
// this is a template aql file for a UDA
// it describes the parameters and their attributes
// it should have the name "test.aql"
// if the UDA has a declared result object, the result object
// can be reached using the variable action.result
// -----
function pre_action( action )
    'Pre action for UDA test' nl
end
-----
function execute( action )
    'Execute for UDA test' nl
    ok = true
    return (ok)
end
-----
function post_action( action )
    'Post action for UDA test' nl
end
```
- ☞ Complete the AQL program (see volume *Open Architecture*).
- ☞ Save the file.

Standardization

☞ Click on the name of desired result objects in *Result* list box.



The list of the result object names can be limited. To do so, enter the desired search pattern in the text area and click the *filter* field. The desired result object name is displayed in *Result* list box.

☞ Append the parameters (see section „[Appending Parameters to User-defined Actions](#)“ on [page 13-35](#)).

☞ Click on *ok*. The AQL file is generated.



A written AQL program can be edited via the menu command *Edit AQL file* in the popup menu of the corresponding UDA icon in the main menu (see Volume *Open Architecture*).

13.1.15.8 Appending Parameters to User-defined Actions

Any number of parameters of any type can be defined.

A parameter and its auxiliary structures (i.e. structures necessary for creation, e.g. the coordinate system) are removed out of the UDA as they exist in the target model. Attention must be paid to the sequence especially when defining parameters which serve each other as auxiliary structures.

Example

A point is defined as a point on a line. Both point and line are to become parameters. As the line serves as an auxiliary structure for the point in this case, the point cannot be defined as a parameter before the line. The line (which is also an auxiliary structure) would be deleted and will not be available for definition as a parameter.

Both objects and (enumeration) values can be determined as parameters and can be set optionally. It could be necessary to go down to the object or (enumeration) value to be appended in the data structure.

If the parameter to be appended is an object, then there are two possibilities:

- The parameter becomes a parameter.
- The object becomes a parameter.

If the parameter to be appended is a (enumeration) value, the parameter can only be appended as a parameter.



Before defining a UDA it is recommended to create a model and to store it, then to define the UDA out of the model so that errors can be corrected simply without having to rebuild the whole structure.

Standardization

Appending Objects as Parameters

Proceed as follows:

1. Choose *append* in the popup menu of field *Parameters*. The UDA definition box is closed and **EUKLID Design** in *Browse* mode. In this mode, as in the *Edit* mode, it is possible to search the data structure but values cannot be changed.
2. Click on the icon of the object group to which the object which is to be appended as a parameter belongs, in the first menu column.
3. Identify the object which is to become a parameter in the UDA view port. The creation action and, if applicable, the manipulation actions of the objects are represented in the second menu column.
4. Choose *set parameter* in the popup menu of the action icon (second menu column). The UDA definition box is opened. The parameter is now appended.

Appending Properties and Parameters as Parameters

Proceed as follows:

1. Choose *append* in the popup menu of field *Parameters*. The UDA definition box is closed and **EUKLID Design** in *Browse* mode.
2. Click on the object group icon which is allocated to the object, its property or parameter which is to be appended as parameter in the first menu column.
3. Identify the object, its property or parameter which is to become a parameter in the UDA model view port. The creation action and, if applicable, the manipulation actions are represented as objects in the second menu column.
4. Click on if not default, the desired action icon.
5. Click on the desired property or parameter icon (3rd column).
6. Choose *set parameter* in the popup menu of the parameter icon. The UDA definition box is opened. The parameter is now appended.



An appended parameter can be removed from the parameter list with the menu command *delete* in the popup menu of the field *Parameters*. The removal can lead to incompatibilities if there is an instance of a UDA.

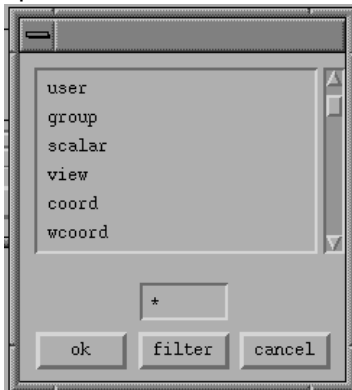
Appending parameters used by AQL

Some UDA parameters are not displayed on objects or values of the original model. They only appear in the third menu column and are used by AQL programs (see section „[Appending AQL Programs to User-defined Actions](#)“ on page 13-33). The procedure differs according to the parameter type which is to be appended:

- Object parameter
- Value parameter
- Enumeration value parameter

Appending Object Parameters

☞ Choose *append types>objects* in the popup menu of the corresponding parameter icon in the *Parameters* area. The dialog box for selecting the UDA parameter type is opened:



☞ Click on the name of desired objects.



The list of object names can be limited. To do so, enter the desired search pattern in the text area and click the *filter* field. The desired object parameter is displayed in the *Parameters* area.

☞ Click on *ok*.

After completing the UDA definition, the icon of the appended object parameter is displayed in the third menu column assuming that the UDA icon was appended to the vertical menu bar.

Standardization

Appending Value Parameters

☞ Choose *append types>values* in the popup menu of the corresponding parameter icon in the *Parameters* area. The dialog box for selecting the UDA parameter type is opened.

☞ Click on the name of desired values.



The list of the value names can be limited. To do so, enter the desired search pattern in the text area and click on the *filter* field. The desired value parameter is displayed in the *Parameters* area.

☞ Click on *ok*.

After completing the UDA definition, the icon of the appended value parameter is displayed in the third menu column assuming that the UDA icon was appended to the main menu.

Appending Enumeration Value Parameters

☞ Choose *append types>enums* in the popup menu of the corresponding parameter icon in the *Parameters* area. The dialog box for selecting the UDA parameter types is opened.

☞ Click on the name of desired enumeration values.



The list of enumeration value names can be limited. To do so, enter the desired search pattern in the text area and click the *filter* field. The desired enumeration value parameter is displayed in the *Parameters* area.

☞ Click on *ok*.

After completing the UDA definition, the icon of the appended enumeration value parameter is displayed in the third menu column assuming that the UDA icon was appended to the main menu.

Determining Parameter Icons

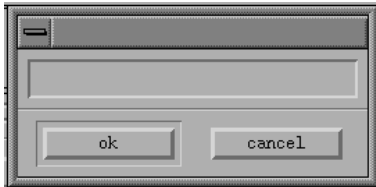
EUKLID Design gives the corresponding system-defined icon for each parameter for the main menu which can occur several times in the case of a UDA depending on the constellation. Instead of using the system-defined, you can design an icon which is only used for this parameter.

- ☞ Choose *icon* in the popup menu of the corresponding parameter icon in the *Parameters area*. The icon editor is opened.
- ☞ Create the desired icon (see section „*Creating User Icons*“ on page 13-46).

Designation of Parameters

EUKLID Design gives a corresponding system-defined name for every parameter which can occur several times in the case of a UDA depending on the constellation. In these cases, the parameter must be given its own respective name, so that, e.g. AQL programs can access the individual parameter directly.

- ☞ Choose *name* in the popup menu of the corresponding parameter icon in the *Parameters area*. The dialog box for entering UDA parameter names is opened.



The text area is pre-allocated by the system-defined name.

- ☞ Enter the desired name.
- ☞ Click on *ok*.

Standardization

Optional Setting of Parameters

(Enumeration) value parameters and simple object parameters can be set optionally during the definition phase.

☞ Choose *set optional* in the popup menu of the corresponding parameter icon in the *Parameters* area. The corresponding icon is displayed shaded in the *Parameters* area.



Complex parameters can only be set optionally in models.

Changing the Sequence of Icons

The sequence in which the system requests the parameters during UDA creation can be changed.



This change can lead to incompatibilities if there is an instance of a UDA.

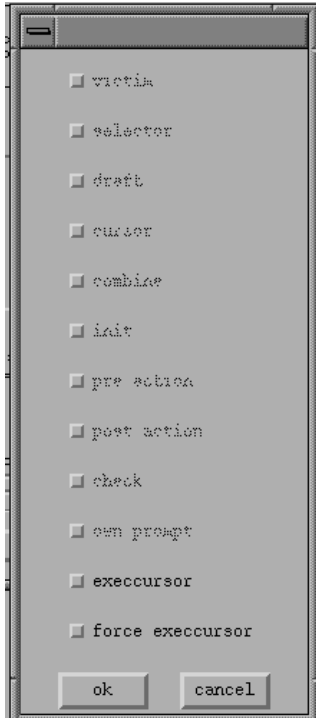
☞ Choose *move* in the popup menu of the parameter icon in the *parameters* area. The parameter icon is highlighted and the message *CHOOSE NEW POSITION* appears in the dialog box.

☞ Click on the icon in whose position the icon is to be inserted.

Allocation of Parameter Attributes

Parameter attributes can be allocated.

- ☞ Choose *attributes* in the popup menu of the parameter icon in the *Parameters* area.
The dialog box for allocating UDA parameter attributes is opened:



Depending on the type, situation and combination the following can be selected:

- ☐ victim
- ☐ selector
- ☐ draft
- ☐ combine
- ☐ initialize
- ☐ pre_action
- ☐ post_action
- ☐ check
- ☐ own_prompt

Standardization

- execcursor
- force execcursor



Information concerning this subject is only of interest to BEA application developers and is given in courses. Please contact the office of *CAD/CAM Strässle Informationssysteme GmbH*.

Combining Parameters in dialog boxes

The entry of successive value parameters for UDAs can be combined in a dialog box.

☞ Click on *Combine* in dialog box for allocating of UDA parameter attributes.

When carrying out a UDA, **EUKLID Design** opens a dialog box in which the entry areas of the three parameters are combined.

Example

A UDA has three combined value parameters:

- *weight* from data type *real*
- *name* from data type *string*
- *quantity* from data type *integer*

When carrying out the UDA, **EUKLID Design** combines the entry areas of these parameters in one dialog box.

Switching on the Execcursor

The execcursor can be switched on for entering parameters which can be sketched in the drawing range.

☞ Click on *execcursor* in the dialog box for allocating of UDA parameter attributes.

Determining Victim Parameters

This attribute determines that an action is allowed to change this parameter. If no AQL program is connected to the UDA, then **EUKLID Design** determines whether a parameter is a modified parameter or not.

☞ Write an AQL program (see section [„Appending AQL Programs to User-defined Actions“ on page 13-33](#)).

13.1.16 Saving User-defined Actions

If a UDA is not only to be made available in the active model (model-internal UDA) then the UDA definition must be saved in an external file (see section *„Saving User-defined Object Types and User-defined Actions“ on page 13-54*).

13.1.17 Appending User-defined Action Icons to the User Interface

UDAs can only be carried out in models when the corresponding icons are integrated in the user interface (see section *„Appending Object Type Icons and Action Icons to the User Interface“ on page 13-55*).

13.1.18 Saving the User-defined Action Configuration

UDAs can only be carried out permanently when the configuration is saved with the appended UDA icon (see section *„Appending Object Type Icons and Action Icons to the User Interface“ on page 13-55*).

Standardization

13.1.19 Loading User-defined Actions

UDAs must be loaded before they can be used in models. There are different possibilities for doing this:

- automatic loading during the start
- automatic loading with the model file
- manual loading

Automatic Loading during the Start

During the system start **EUKLID Design** loads the menus and action groups which contain the configuration file. **EUKLID Design** only loads the actions and objects which are requested by the user interface. This reduces the time for the system start when the menus contain a large quantity of UDOTs and UDAs.

Automatic Loading with the Model File

EUKLID Design automatically loads a UDA with the model file in which instances of the UDA are saved.

Manual Loading

If a desired UDA in a model is not available in the main menu, then the corresponding UDA definition file can be loaded in the main memory:

- ☞ Choose *load* in *Uda* menu. The file dialog box is opened.
- ☞ Enter the desired file name.



- The AQL function *user_symbol* contains this action for compatibility reasons (*imp*).
- The UDA icon can be integrated in the user interface with the menu command *edit action group*.
- The AQL program must make sure that the UDA is loaded for application in an AQL program.

13.1.20 Editing Action Definitions

UDAs can be changed. Attention must be paid that every modification has an effect in the models which contain an UDA instance. These models contain the updated UDA when next loaded.

- ☞ Load the UDA which is to be edited.
- ☞ Choose *edit definition* in *Uda* menu. The model dialog box is opened.
- ☞ Click the name of the UDA to be edited in the list box.
- ☞ Click *Ok*. A window with the corresponding UDA model is opened.
- ☞ Change the UDA model if desired.
- ☞ Choose *close definition* in *Uda* menu. The UDA definition box is opened.
- ☞ Change the characteristics of the UDA if desired.
- ☞ Save the modification with the menu command *save definition* in *Uda* menu.



- Objects must not be deleted as references to them can exist in every model which contains the UDA.
- If file and link structures are changed, then it must be ensured that models which contain UDA instances can be reloaded. This is done by means of an adapted file locator table.
- **EUKLID Design**, refuses certain changes to the definition, e.g. appending of parameters, if an instance has already been created in a model. This lockout can be avoided during the design phase of a UDA with the start option *-admin*. However, depending on the modification, models which contain instances of these UDAs cannot be reloaded.

- ☞ Save the definition in an external file.

Standardization

13.1.21 Creating User Icons

With the icon editor of **EUKLID Design**, icons can be designed for the menus, UDOTs and UDAs and their properties and parameters. As a basis, existing icons can be accessed.

The call for the icon editor depends on the icon to be designed:

UDOT

☞ Choose *icon* in the popup menu of the UDOT icon in the UDOT definition box.

UDA

☞ Choose *icon* in the popup menu of the UDA icon in the UDA definition box.

UDOT property

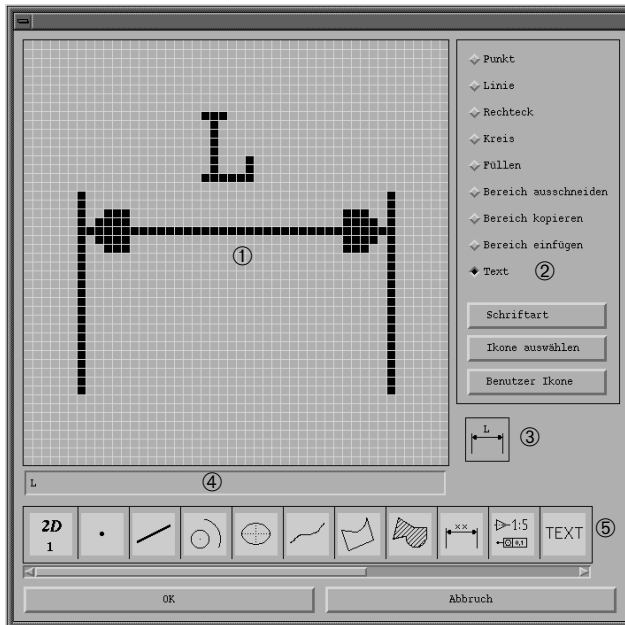
☞ Choose *icon* in the popup menu of the property icon in the UDOT definition box.

UDA parameter

☞ Choose *icon* in the popup menu of the parameter icon in the UDA definition box.

The icon editor is opened. It has the following areas:

- ① Working area: The icons can be created here.
The icon pictogram is processed on the level of the smallest pixel. This working area is enlarged, whereby pixel raster limits are made clearer by a line grid.
The pixels are displayed in the background color.
- ② Action bar: Actions can be called up here.
- ③ Icon display area: The current icon is represented here in its original size.
- ④ Text area: Text can be entered here.
- ⑤ Icon bar: The loaded icon file is represented here out of which icons can be copied into the working area.



The working area is pre-allocated by the following texts or pictograms depending on the call-up environment:

- UDOT name
- UDA name
- property name/icon
- parameter name/icon

To create an icon, proceed as follows:

- ☞ Copy an existing icon as a working basis into the working area.
- ☞ Edit the icon with the existing actions and drawing modes in the working area.
- ☞ Click on ok. The icon editor is closed and the icon is displayed in the respective definition box.

Standardization

13.1.21.1 Copying an Existing Icon

An icon can be copied as follows:

- ☞ Select the desired icon file if not default.
- ☞ Load the icon out of the icon file into the icon bar if not displayed.
- ☞ Click the desired icon in the icon bar. The icon pictogram is copied into the working area.

13.1.21.2 Editing an Icon

The following drawing modes of the Icon Editor are available:

left-hand mouse key	drawing in the foreground color
center mouse key	toggle mode
right-hand mouse key	drawing in the background color

An icon can be edited as follows:

- ☞ Click the desired action in the action bar.
- ☞ Press the mouse key corresponding to the desired drawing mode in the working area at the desired starting position.
- ☞ Release the depressed mouse key in the working area at the desired final position.
The editing result is displayed in the working area and in the icon display area.

Actions of the Icon Editor

The following actions are available:

<i>Point</i>	Drawing points
<i>Line</i>	Drawing lines
<i>Rectangle</i>	Drawing rectangles
<i>Circle</i>	Drawing circles
<i>Filling</i>	Filling closed contours (filling/removing)
<i>Cut area</i>	Cutting out a rectangular area
<i>Copy area</i>	Copying a rectangular area into the intermediate memory
<i>Paste area</i>	Inserting a copied or cut out area
<i>Text</i>	Creating text
<i>Select font</i>	Setting the font
<i>Selecting icon</i>	Setting the icon file
<i>User icon</i>	Loading an icon file into the icon bar

Drawing Points

Pixels can be drawn with this action as follows:

- ☞ Click the action *point* in the action bar.
- ☞ Click the pixel to be drawn in the working area (starting position = end position). The point is drawn in the desired drawing mode. A free-hand drawing can thus be obtained if the mouse key is kept depressed.

Drawing Lines

Lines can be drawn with this action in the following way:

- ☞ Click the action *line* in the action bar.
- ☞ Press the corresponding mouse key at the desired starting point of the line in the working area. The course of the line with regard to the current cursor position is represented by a rubber band line.
- ☞ Release the mouse key at the desired end point of the line. The line is created in the desired drawing mode.



If the mouse key is released outside of the working area, the corresponding margin point of the working area becomes the end point of the line.

Drawing Rectangles

Rectangles can be drawn with this action as follows:

- ☞ Click the action *rectangle* in the action bar.
- ☞ Press the corresponding mouse key at the desired anchor point of the rectangle in the working area. The current cursor position becomes the transverse corner point of the rectangle. The current outline of the rectangle is represented by a rubber band rectangle.
- ☞ Release the mouse key at the desired corner point of the rectangle. The rectangle will be created in the desired drawing mode.



If the mouse key is released outside of the working area, only the part of the rectangle inside the working area is drawn.

Standardization

Drawing Circles

Circles can be drawn with this action as follows:

- ☞ Click the action *circle* in the action bar.
- ☞ Press the corresponding mouse key at the desired center point of the circle in the working area. The current cursor position becomes a point on the circumference of the circle. The current circumference of the circle is represented by a rubber band circle.
- ☞ Release the mouse key at the desired circumference of the circle. The circle is created in the desired drawing mode.

Filling Closed Contours

Closed contours can be filled with this action as follows:

- ☞ Click inside the contour to be filled with the mouse key in the working area. All pixels of the contour are created in the desired drawing mode.



The working area is treated as its own contour.

Cutting an Area

Rectangles can be cut out of the working area with this action and placed in the intermediate memory as follows:

- ☞ Click the action *cut area* in the action bar.
- ☞ Press any mouse key at the desired anchor point of the rectangle to be cut out in the working area. The current cursor position becomes the transversal corner point of the rectangle cutout. The current outline of the rectangle is represented by a rubber band rectangle.
- ☞ Release the mouse key at the desired corner point of the cutout rectangle. The cutout rectangle is saved in an intermediate memory. The area where the rectangle was cut out is colored over with the background color.

Copying an Area into the Intermediate Memory

Rectangles can be copied out of the working area into the intermediate memory with this action as follows:

- ☞ Click the action *copy area* in the action bar.
- ☞ Press any mouse key at the desired anchor point of the copied rectangle in the working area. The current cursor position becomes the transversal corner point of the rectangle copy. The current outline of the rectangle copy is represented by a rubber band rectangle.
- ☞ Release the mouse key at the desired corner point of the rectangle copy. The rectangle copy is saved in an intermediate memory.

Inserting a Cutout or Copied Area

Cutout or copied rectangles can be inserted in the working area with this action as follows:

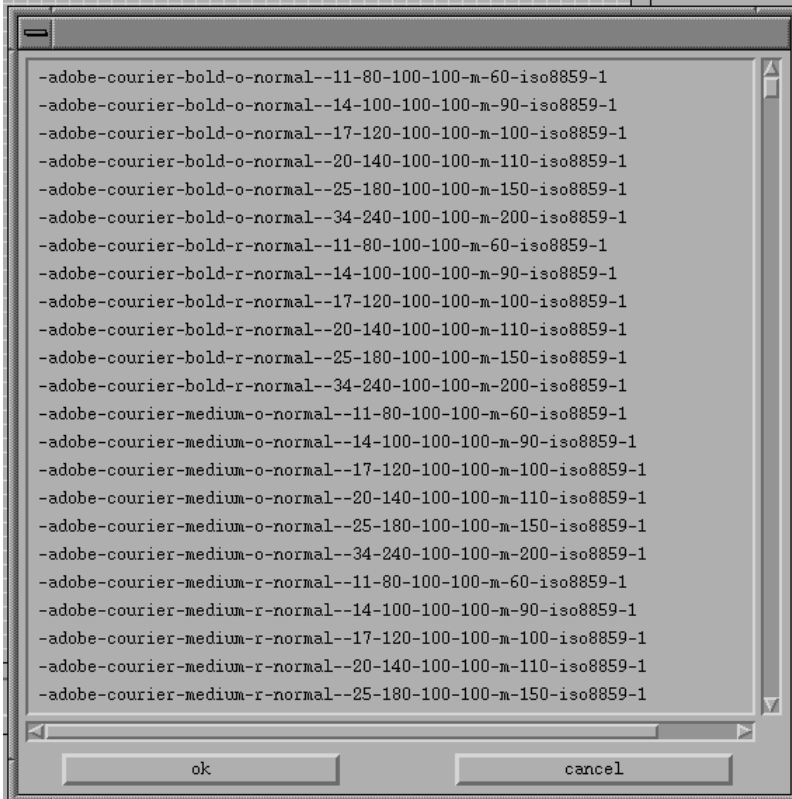
- ☞ Click the action *insert area* in the action bar.
- ☞ Press any mouse key at the desired anchor point of the cutout or rectangle copy in the working area. The outline of the cutout or rectangle copy is displayed.
- ☞ Release the mouse key. The cutout or rectangle copy is inserted.

Standardization

Setting a font

A font can be set with this action as follows:

- Click the action *select font* in the action bar. The active command is interrupted. A dialog box with all available fonts is opened in the working area:



- Click the desired font in the dialog box.
- Click *ok*. The popup menu is closed. The selected font is set. The previous command becomes active again.



A font selection only remains active for setting a pictogram.

Creating Text

Text can be created with this action as follows:

- ☞ Click the action *text* in the action bar.
- ☞ Type a text into the text area and press the <RETURN> key.
- ☞ Press the corresponding mouse key in the working area. The outline of the text rectangle is displayed.
- ☞ Release the mouse key. The text is displayed.



- The font can be set with *select font*.
- The placing of the text can be terminated by releasing the mouse key outside of the working area.

Setting an Icon File

An icon file can be set with this action as follows:

- ☞ Click the action *select icon* in the action bar. The file dialog box is opened. The installation directory and the file name transmission *.bitm* are default. The system-defined icons can be found in the following directories:

#/basic

#/d23bs

#/d2con

#/d2drw

- ☞ Enter the desired icon file. The file dialog box is closed. The icon file is loaded into the icon bar in which the first 10 icons are represented. The available icons can be searched with the scroll bar.

Copying an Icon of the Icon Bar into the Working Area

An icon can be copied out of the icon bar into the working area with this action as follows:

- ☞ Click the action *user icon* in the action bar.
- ☞ Click the desired icon in the icon bar. The icon is copied into the working area.



Icons which are not displayed can be taken out of the loaded icon file and brought into the view port of the icon bar with the aid of the scroll bar.

Standardization

13.1.22 Saving User-defined Object Types and User-defined Actions

If a UDOT or UDA is not only to be available in the active model (model-internal UDOT or UDA), then the UDOT or UDA definition must be saved in an external file. Attention must be paid to path names which can be subsequently reproduced. This can only be ensured by relative path names. Relative path names can only be attained by entering file names concerning the work directory or via relative links. Further subdirectories can be located under relative links.

- ☞ Choose *save definition* in UDO and/or UDA menu. The model box is opened:
- ☞ Click the UDOT/UDA to be saved in the list box.
- ☞ Click *ok*. The model box is closed and the file box is opened. The file name is pre-allocated by the UDOT and/or UDA name.
- ☞ Click *ok*. The file box is closed.



The menu command *save definition* as in *Udo* and/or *Uda* menu does not only change the file name but creates another type identification. This can be useful for editing a UDOT/UDA definition.

13.1.23 Appending Object Type Icons and Action Icons to the User Interface

In order to create UDOs in models or to use UDAs in models, the corresponding icons must be integrated in the user interface of **EUKLID Design**. The description is divided into the following sections:

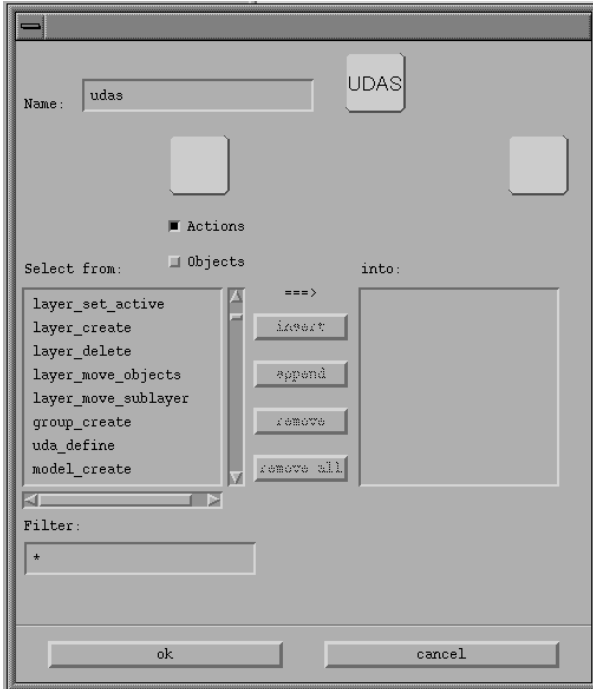
- „*Appending Object Type Icons and Action Icons to Existing Action Groups*“ on page 13-56
- „*Appending Object Type Icons and Action Icons to New Action Groups*“ on page 13-58
- „*Appending Object Type Icons and Action Icons to New Menus*“ on page 13-61
- „*Moving Object Type Icons and Action Icons*“ on page 13-63

Standardization

13.1.23.1 Appending Object Type Icons and Action Icons to Existing Action Groups

UDOT and/or UDA icons can be appended to existing action groups as follows:

- ➞ Choose *edit action group* in the popup menu of the action group icon. A message box is opened.
- ➞ Click *continue*. The action group box is opened.



- ➞ Click *objects* and the default *Actions* if objects are to be taken up in the action group.
- ➞ Click an action and/or an object to be taken up in the action group in the left-hand list box.



In order to keep the listed actions and/or objects clear, enter a search key in the *filter* field and close with <RETURN>.

Example

Search key *line_** for all actions which have a line as a result.

☞ Click *insert* if the action and/or object is to be located above the selected action and/or object in the action group.

Click *append* if the action and/or object is to be located below the selected action and/or object in the action group.

The action and/or object is taken up in the right-hand list box and the allocated icon is taken up in the action group.



If an action and/or object is to be removed from the action group, click the action and/or object to be removed in the left-hand list box and the field *remove*.

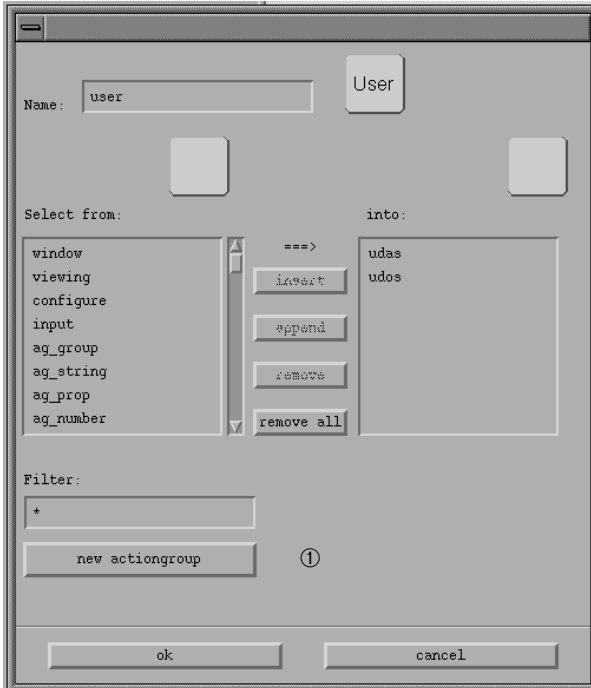
☞ Click *ok*.

Standardization

13.1.23.2 Appending Object Type Icons and Action Icons to New Action Groups

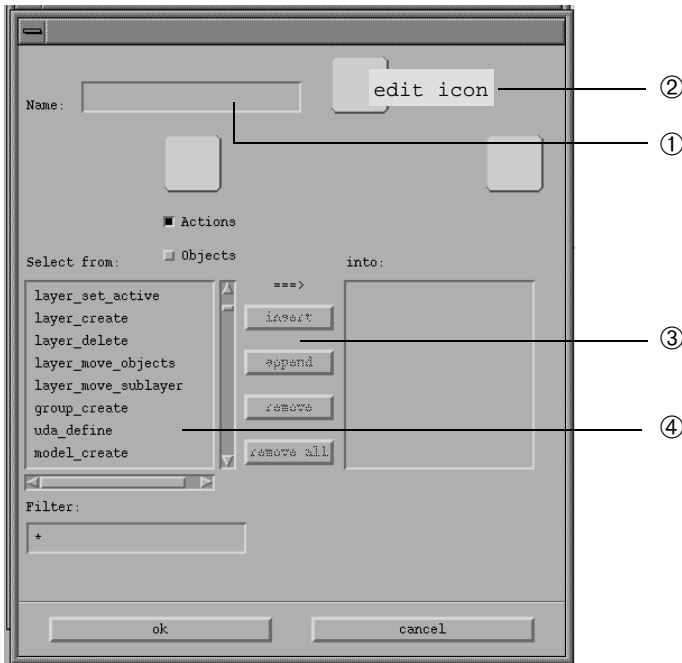
UDOT and/or UDA icons can be appended to new action groups as follows:

- ☞ Choose *edit menu* in the popup menu of the menu icon in *Create* mode. A message box is opened.
- ☞ Click on *continue*. The menu box is opened:



- ☞ Click on *new actiongroup* ①.

The action group box is opened:



- ☞ Enter the desired action group name in the *Name* field ①.
- ☞ Choose *edit icon* in the popup menu of icon field ②. The icon editor is opened with which the action group icon can be designed (see section „[Creating User Icons](#)“ on page 13-46).
- ☞ Click on *Objects* and the default *Action* field ③ if objects are to be taken up in the action group.
- ☞ Alternately click the action to be taken up in the action group or the object in the left-hand list box and the field *insert* or *append* ④.
insert hangs the action and/or object above the action selected in the right-hand list box and/or the object in the action group.
append hangs the action and/or object under the action selected in the right-hand list box and/or the object in the action group.

Standardization



- To keep the listed actions and/or objects clear, enter a search key in the *filter* field and close with <RETURN>, e.g. search key *line_** for all actions which have a line as a result.
- To remove an action and/or object from the action group, click the action and/or object to be removed in the left-hand list box and the field *remove*.
- If all actions and/or objects are to be removed from the action group, click the field *remove all*.

☞ Click *ok*. A dialog box is opened.

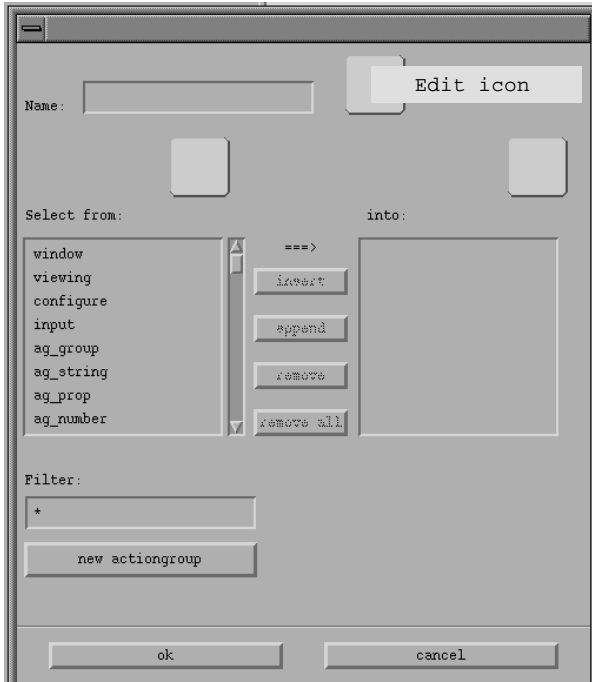
☞ Click whether the new action group is to be inserted above the currently selected action group or appended under the currently selected action group. The action group is displayed at the selected position in the menu.

Now the user actions and/or objects can be appended to the new menu as described in the section „[Appending Object Type Icons and Action Icons to Existing Action Groups](#)“ on page 13-56.

13.1.23.3 Appending Object Type Icons and Action Icons to New Menus

UDOT and/or UDA icons can be appended to new menus as follows:

- ☞ Choose *insert menu* in the popup menu of the menu icon in *Create* mode. The menu box is opened.



- ☞ Enter the desired menu name in the *Name* field.
- ☞ Choose the *edit icon* in the popup menu of the icon field. The icon editor is opened with which the menu icon can be designed (see section „[Creating User Icons](#)“ on page 13-46).
- ☞ Alternately click the action group to be taken up in the menu in the left-hand list box and the field *insert* or *append*.
insert hangs the action group in the menu above the action group selected in the right-hand list box.
append hangs the action group in the menu under the action group selected in the right-hand list box.

Standardization



- If a new action group is to be adopted in the menu, click the field *new action group*. The action group box is opened with which the new action group can be created (see section „[Appending Object Type Icons and Action Icons to New Action Groups](#)“ on page 13-58).
- To keep the listed action groups clear, enter a search key in the *filter* field and close with <RETURN>, e.g. search key *line_** for all actions which have a line as a result.
- To remove an action group from the menu, click the action group to be removed in the left-hand list box and the field *remove*.
- To remove all action groups from the menu, click the field *remove all*.

☞ Click *ok*. The new menu is inserted in front of the currently selected menu.

Now user actions and/or objects can be appended to the new menu as described in the section „[Appending Object Type Icons and Action Icons to Existing Action Groups](#)“ on page 13-56.

13.1.23.4 Moving Object Type Icons and Action Icons

System-defined and user-defined action groups and menus can be changed but application-defined action groups and menus cannot. System-defined action groups and menus are only allowed to be changed with the corresponding rights (see chapter Configuring the System).

- ☞ Choose *edit* menu in the popup menu of the icon of the menu to be changed in *create* mode or choose *edit action group* in the popup menu of the icon of the action group to be changed. A message box is opened.
- ☞ Click *continue*. The menu and/or action group box is opened.

The menu and/or action group box can now be operated as described in the section „[Appending Object Type Icons and Action Icons to Existing Action Groups](#)“ on page 13-56 and „[Appending Object Type Icons and Action Icons to New Action Groups](#)“ on page 13-58.

Standardization

13.1.24 Saving the Object Type and Action Configuration

The configuration must be saved after the UDO and/or UDA icons have been appended to the user interface. There are two possibilities of doing this:

- global system saving
- user-specific system saving

Global saving

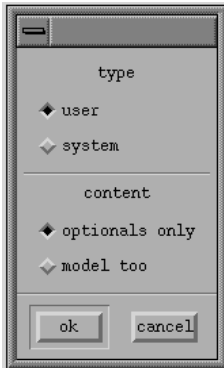
The global system configuration provides the optional values for the complete installation. This is saved in the directory *#/design_config*.



Global system saving should only be carried out by authorized users with the corresponding access rights.

Proceed as follows:

☞ Choose the *Save configuration* in the *File* menu. The configuration box is opened:



Saving of the user-specific configuration and the model-specific saving of optional values are default.

☞ Click the switch field *system*.

☞ If the model is to be saved as an original model, click the switch field *model too*.

☞ Click *ok*.

User-specific Saving

User-specific saving provides the optional values for all models of the user. They are saved in the directory `~/design_config`. The user-specific configuration overwrites the system configuration.

Carry out user-specific saving as follows:

- ☞ Choose *save configuration* in the *File* menu. The configuration box is opened. Saving of the user-specific configuration is default.
- ☞ If the first model is also to be saved, click the switch field *model too*.
- ☞ Click *ok*.

Model-specific Saving

The model-specific configuration is valid for one model of a user and is automatically left for the model by *save model*. It is left in the model in the form of the optional values. A setting carried out in the model overwrites all other configurations with the exception of session data, e.g. dialog.

Proceed as follows:

- ☞ Choose *save model* in *File* menu. The model box is opened:



- ☞ Click the name of the model whose configuration is to be saved.

Standardization

☞ Click *ok*. The following dialog box is opened:



Keeping write access is default.

☞ Click *ok*. The optional values and the global defaults of the current model are saved.

AQL Program-specific Saving

The settings out of the system configuration or the settings of the current AQL program are valid for AQL programs. AQL programs thus produce data which can be reproduced without being dependent on the configuration. AQL programs are responsible for the availability of existing UDAs and UDOs and must load them before use.

13.1.25 Nesting User-defined Object Types and User-defined Actions

A nested UDOT includes one or more UDOs, which can be nested themselves. Tasks to be subdivided into several similar subtasks and their combinations can be properly handled by nesting of UDOTs.

The same analogously applies to user-defined actions. A UDA may in turn include UDAs and UDOs.

Example

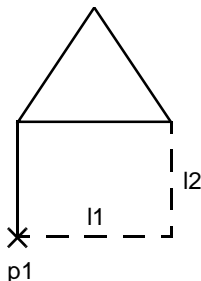
A UDOT *village* shall be created as nested user-defined object. The village consists of several UDOTs *house_group* in turn consisting of several UDOTs *house*. The nesting is subdivided into the following steps:

- Definition of a UDOT *house*
- Definition of a UDA *position house*
- Definition of a UDOT *house_group* consisting of two UDOs *house*
- Definition of a UDA *position house_group*
- Definition of a UDOT *village* consisting of two UDOs *house_group*
- Definition of a UDA *position village*

Merely the construction steps essential for nesting are performed.

Definition of the User-defined Object Type *house*

This UDOT shall have a variable base line and a variable height.

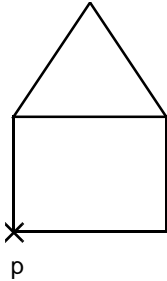


Standardization

- ☞ Construct the UDOT *house 1* by the *rectangle* macro with the parameter p as point with x and y distances to the origin of the coordinate system for the basic walls and sketched roof lines.
- ☞ Append the following properties:
 - Length I1 of rectangle
 - Length I2 of rectangle
- ☞ Save the UDOT.
- ☞ Append the UDOT to the menu.

Definition of the User-defined Action *position house*

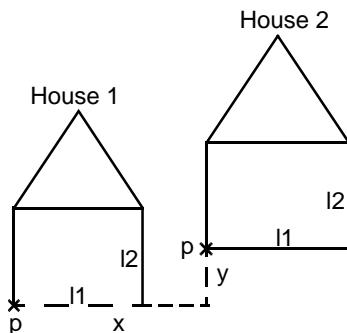
This UDA shall position the UDOT *house*.



- ☞ Construct the UDA *position house* from a UDO *house* with parameter p as point with x and y distances to the origin of the coordinate system.
- ☞ Append the parameter p .
- ☞ Save the UDA.
- ☞ Append the UDA to the menu.

Definition of the User-defined Object Type *house group*

This UDOT shall have a variable distance and variable dimensions of base lines and heights of the houses.

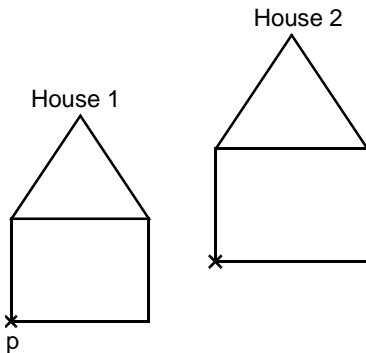


Standardization

- ☞ Create two UDOs *house* with the UDA *position house* with
 - *p* of house 1 as point with x and y distances to the origin of the coordinate system
 - *p* of house 1 as point with x and y distances to *p* of house 1
 - arbitrary *l1*
 - arbitrary *l2*
- ☞ Append the properties in the following sequence:
 - *l1* of house 1
 - *l2* of house 1
 - *p2*
 - *l1* of house 2
 - *l2* of house 2
- ☞ Save the UDOT.
- ☞ Append the UDOT to the menu.

Definition of the User-defined Action *position house group*

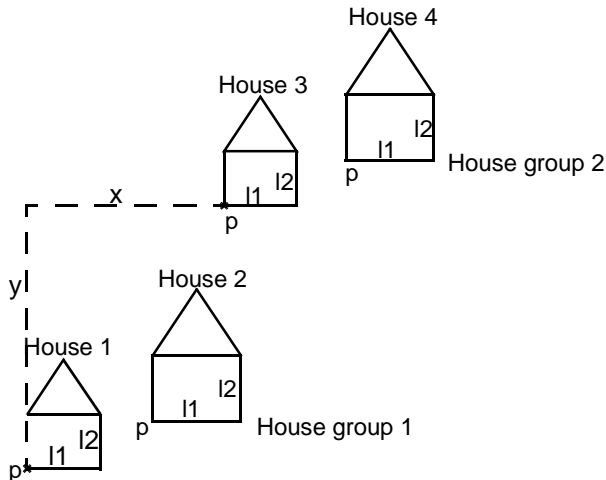
This UDA shall have a variable position of the house group



- ☞ Construct the UDA *position house group* from a UDO *house group* with parameter *p* of house 1 as point with x and y distances to the origin of the coordinate system.
- ☞ Append the parameter *p* of house 1.
- ☞ Save the UDA.
- ☞ Append the UDA to the menu.

Definition of the User-defined Object Type *village*

This UDOT shall have a variable distance of house groups and variable dimensions of base lines and heights of houses.



☞ Create two UDOs *house group* with

- p of house 1 as point with x and y distances to the origin of the coordinate system
- p of house 2 as point with x and y distances to p of house 1
- p of house 3 as point with x and y distances to p of house 1
- p of house 4 as point with x and y distances to p of house 3
- arbitrary $l1$
- arbitrary $l2$

☞ Append property p of house 3.

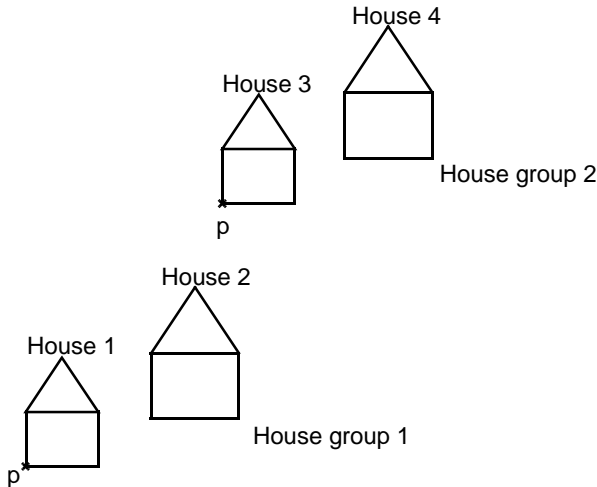
☞ Save the UDOT.

☞ Append the UDOT to the menu.

Standardization

Definition of the User-defined Action *position village*

This UDA shall have a variable position.



- ☞ Construct the UDA *position village* from the UDO *village* with parameter *p* of house 1 as point with x and y distance to the origin of the coordinate system.
- ☞ Append parameter *p* of house 1.
- ☞ Save the UDA.
- ☞ Append the UDA to the menu.

You could alternatively have placed a set of houses by one UDOT *house* and one action *house_village* in order to represent a village. In this case, the data structure would neither include an object *village* nor *house_group*. Nevertheless, the village could be modified and addressed by the house-creating action. The alternative selected depends on the characteristics desired.

13.1.26 Definition Example for User-defined Object Types and User-defined Actions

The UDO and UDA technique is described step by step using the example of an object type *component* and a component *screw*. The screw is to be created in models by two user actions and represented symbolically as an axis intersection. Both UDOTs are to be externally available. The example is clearly divided into task fields.

13.1.26.1 Defining the Component

Initializing the Definition

1. Choose *init definition>new* in *Udo* menu. The action *udo_create* is displayed in the main menu and the prompt *name: ""* appears in the text area.
2. Enter *part* in the text area. The view port named *part* is opened.
3. Create a text string *classification*, start point *point of origin of the coordinate system* and text height 0 using the action *text_absolute*.

Closing the Definition

1. Choose *close definition* in *Udo* menu. The UDOT definition box is opened.

Designing the UDOT Icon

1. Choose *icon* in the popup menu of field *UDOT Icon*. The icon editor is opened. The working area is pre-allocated by *part*.
2. Design an icon with the two-line text *part class*. The icon editor is closed.

Appending the Properties

1. Choose *append* in the popup menu of field *Properties*. The UDOT definition box is closed. **EUKLID Design** changes to *Browse* mode.
2. Click on the *TEXT* icon.
3. Identify the text in the UDOT viewport. In the main menu, the creation action is displayed.
4. Click on the *string* icon.
5. Choose *set property* in the popup menu of the string icon. The UDOT definition box is opened. The property is appended.
6. Click on *ok*. The UDOT definition box is closed.

Standardization

Saving the Definition

1. Choose *save definition* in *Udo* menu. The model box is opened.
2. Click on the UDOT to be saved in list box.
3. Click on *ok*. The model box is closed and the file box is opened. The file name *part.udo* is default.
4. Click on *ok*. The file box is closed.

Appending the UDOT Icon to a New Menu

1. Choose *insert menu* in the popup menu of the menu icon in *Create* mode. The menu box is opened.
2. Enter *part* in the *Name* field.
3. Choose *edit icon* in the popup menu of icon field. The icon editor is opened.
4. Design an icon with the text *part*. The icon editor is closed.
5. Click on the field *new action group*. The action group box is opened.
6. Enter *part* in the *Name* field.
7. Choose *edit icon* in the popup menu of icon field. The icon editor is opened.
8. Design an action group icon with the text *part*. The icon editor is closed.
9. Click on *Objects* and the default field *Actions*.
10. Click on *part* in left-hand list box and *insert*. The action group is appended above the action group selected in the right-hand list box.
11. Click on *ok*. A dialog box is opened.
12. Click on *insert*. The action group is displayed in the right-hand list box.
13. Click on *ok*. The new menu is inserted before the currently selected menu.

Saving the Configuration

1. Choose *save configuration* in *File menu*. The Configuration box is opened.
2. Click on *ok*. The configuration is saved.

13.1.26.2 Defining the Screw

Initializing the Definition

1. Choose *init definition>from udot* in *Udo* menu. In the main menu, the action *udo_master_of_udo* is displayed and the prompt *name: ""* appears in the text area.
2. Enter screw in the *text* area. The model box is opened.
3. Click on *part* in list box. A view port named *screw* is opened.
4. Create a circle without an axis intersection with the action *circle_centerradius*.
5. Create a polygon with 6 corners, center point of the circle, angle 0° and 0.6-times radius of the circle, with the action *macro_multicorner*.
6. Create a geometric axis intersection out of lines (*multi_circleaxes*).
7. Create a group *simple_screw* out of the lines of the axis intersection.

Closing the Definition

1. Choose *close definition* in *Udo* menu. The UDOT definition box is opened.
2. For simpler representation, choose the group *simple_screw* by clicking it in the list box.

Designing the UDOT Icon

1. Choose *icon* in the popup menu of field *UDOT Icon*. The icon editor is opened. The working area is pre-allocated by *screw*.
2. Design an action group icon with a screw representation consisting of a circle and 6 lines. The icon editor is closed.

Appending the Properties

1. Choose *append* in the popup menu of *Properties* field. The UDOT definition box is closed. **EUKLID Design** changes to *Browse* mode.
2. Click on the circle icon.
3. Identify the circle in the UDOT view port. In the second menu column, the creation action is displayed.
4. Click on the icon (third column).
5. Choose *set property* in the popup menu of the radius icon. The UDOT definition box is closed. The property is appended.
6. Click on *ok*. The ODOT definition box is closed.

Standardization

Saving the Definition

1. Choose *save definition* in *Udo* menu. The model box is opened.
2. Click on UDOT to be saved in the list box.
3. Click on *ok*. The model box is closed and the file box is opened. The file name defaults to *screw.udo*.
4. Click on *ok*. The file box is closed.

Appending the UDOT Icon to an Existing Action Group

1. Choose *edit action group* in the popup menu of the *part* action group icon. The action group box is opened.
2. Click on objects and the default button *actions*.
3. Click on *screw* in the left-hand list box.
4. Click on *append*. The object is transferred to the right-hand list box.
5. Click on *ok*. The *screw* icon is transferred to the *part* action group.

Saving the Configuration

1. Choose *save configuration* in the *File* menu. The configuration box is opened.
2. Click on *ok*. The configuration is saved.

13.1.26.3 Defining the First Screw Action

1. Choose *init definition>new* in *Uda* menu. In the main menu, the action *uda_master* is displayed and the prompt *name: ""* appears in the text area.
2. Enter *screw_point* in the text area. A view port named screw point is opened.
3. Create a UDO *screw*.

Closing the Definition

1. Choose *close definition* in *Uda* menu. The UDA definition box is opened.

Designing the UDA Icon

1. Choose *icon* in the popup menu of field *UDA Icon*. The icon editor is opened. The working area is pre-allocated by the text *screw_point*.
2. Cut the text.
3. Load the screw icon into the working area. The screw representation is displayed in the working area.
4. Create a circle as center point of the screw head.
5. Click on *ok*. The icon editor is closed.

Determine the Result Object

1. Choose *set result object* in the popup menu of *Result* field. The UDA definition box is closed.
2. Click on the *User* object icon in the first menu column.
3. Identify the UDO *screw* in UDA model viewport.
4. Choose *set result* in the popup menu of the keyboard icon in second menu column. The *screw* icon is displayed in result icon field.

Appending Parameters

1. Choose *append* in the popup menu of the *Parameters* area. The UDOT definition box is closed. **EUKLID Design** changes to *Browse* mode.
2. Click on the point icon in the first menu column.
3. Choose *select in user* in popup menu of the *screw_point* viewport.
4. Select the UDO *screw* in *screw_point* viewport.
5. Select the center point of the UDO.
6. Choose *set parameter* in the popup menu of the action icon (second column). The UDA definition box is opened.
7. Click on *ok*. The UDA definition box is closed.

Standardization

Saving the UDA

1. Choose *save definition* in *Uda* menu. The configuration box is opened.
2. Click on *ok*. The configuration is saved.

Appending the UDA Icon to an Existing Action Group

1. Choose *edit action group* in the popup menu of the *part* action group icon. The action group box is opened.
2. Click on *screw_point* in the left-hand list box.
3. Click on *append*. The action is transferred to the right-hand list box.
4. Click on *ok*. The *screw_point* icon is appended to the part action group.

Setting Optional Parameters

1. Click on the *screw_point* icon in the second menu column.
2. Choose *optional on* in the popup menu in the string icon.
3. Choose *optional on* in the popup menu of the radius icon.
4. Create the instance with corresponding values.

Saving the Configuration

1. Choose *save configuration* in the *File* menu. The configuration box is opened.
2. Click on *ok*. The configuration is saved.

13.1.26.4 Defining the Second Screw Action

1. Choose *init definition>new* in *Uda* menu. In the main menu, the action *uda_master* is displayed and the prompt *name: ""* appears in the text area.
2. Enter *screw_intersection* in the text area. A viewport named *screw_intersection* is opened.
3. Create a UDO *screw*.
4. Choose *convert to layer* in *Udo* menu.
5. Select the UDO *screw* in *screw_intersection* viewport. The UDO conversion box is opened.
6. Click on *active*.
7. Click on *ok*. The UDO *screw* is converted into the layer *screw_instance*.
8. Create 2 lines which intersect each other as lines between 2 points with a distance of *x* and *y* from the origin of the coordinate system in the layer.
9. Pay attention that the lines are independent. Switch to *Edit* mode.
10. Click on the point icon in the first menu column.
11. Select the center point of the screw head in *screw_intersection* viewport.
12. Choose *redefine* in the popup menu of the action icon.
13. Click on the *point_intersection* icon in the second menu column.
14. Select as line parameter the two intersecting lines in *screw_intersection* viewport. The intersection of the lines is the center point of the screw head.
15. Choose *convert to user* in *Layer* menu. The layer conversion box is opened. The layer *screw_instance* is default.
16. Click on *ok*. The layer is converted into the UDO *screw*.

Closing the Definition

1. Choose *close definition* in *Uda* menu. The UDA definition box is opened.

Creating the UDA Icon

1. Choose *icon* in the popup menu of field *UDA Icon*. The icon editor is opened. The working area is pre-allocated by the text *screw*.
2. Cut out the text.
3. Load the *screw_point* icon into the working area. The screw representation is displayed in the working area.
4. Create two diagonal lines from corner point to corner point of the working area through the center point of the circle.
5. Click on *ok*. The icon editor is closed.

Standardization

Determining the Result Object

1. Choose *set result object* in the popup menu of field *Result*. The UDA definition box is closed.
2. Click on the *User* object icon in the first menu column.
3. Identify the UDO *screw* in *screw_intersection* window.
4. Choose *set result object* in the popup menu of the keyboard icon (2. column). The screw is displayed in *result* field.

Appending Parameters

1. Choose *append* in the popup menu of the *Parameters* area. The UDA definition box is closed. **EUKLID Design** changes to *Browse* mode.
2. Click on the lines icon in the first menu column.
3. Choose *Selection in user* in the popup menu of *screw_intersection*
4. Select the UDO *screw* in *screw_intersection* viewport.
5. Select the first diagonal line in *screw_intersection* viewport.
6. Choose *set parameter* in the popup menu of the action icon. A notification box is opened with which **EUKLID Design** asks whether the marked objects in *screw_intersection* viewport can be deleted.
7. Click on *yes*. The UDA definition box is opened. The line icon is displayed in the *Result* field.
8. Click on *ok*. The UDA definition box is closed.

Appending the UDA Icon to an Existing Action Group

1. Choose *edit action group* in the popup menu of the *part* action group icon. The action group box is opened.
2. Click on *screw_intersection* in the left-hand list box.
3. Click on *append*. The action is transferred to the right-hand list box.
4. Click on *ok*. The *screw_point* icon is appended to the *part* action group.

Setting Optional Parameters

1. Click on the *screw_intersection* icon in second menu column.
2. Choose *optional on* in the popup menu of the string icon.
3. Choose *optional on* in the popup menu of the radius icon.
4. Create an instance with the corresponding values.

Saving the Configuration

1. Choose *save configuration* in *File menu*. The configuration box is opened.
2. Click on *ok*. The configuration is saved.

Saving the UDA

1. Choose *save definition* in *Uda* menu. The configuration box is opened.
2. Click on *ok*. The configuration is saved.

Appending the UDA Icon to an Existing Action Group

1. Choose *edit action group* in the popup menu of the *part* action group icon. The action group box is opened.
2. Click on *screw_point* in the left-hand list box.
3. Click on *append*. The action is transferred to the right-hand list box.
4. Click on *ok*. The *screw_point* icon is appended to the *part* action group.

Setting Parameters

1. Click on the *screw_point* icon in second menu column.
2. Choose *optional on* in the popup menu of the string icon.
3. Choose *optional on* in the popup menu of the icon.

Saving the Configuration

1. Choose *save configuration* in *File menu*. The configuration box is opened.
2. Click on *ok*.

As result your menus might look like this:



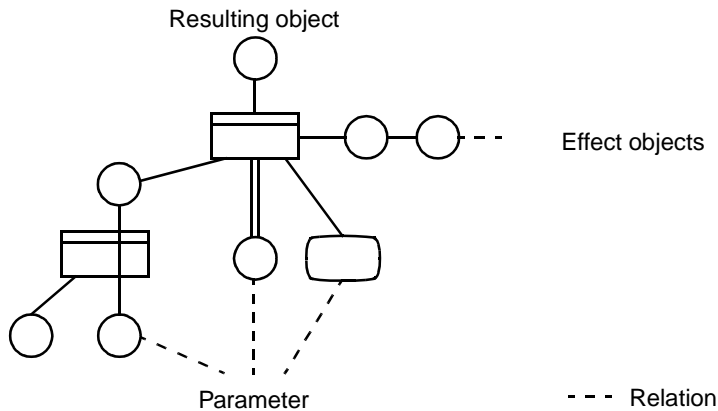
Standardization

13.1.27 Notes on the Data Structure

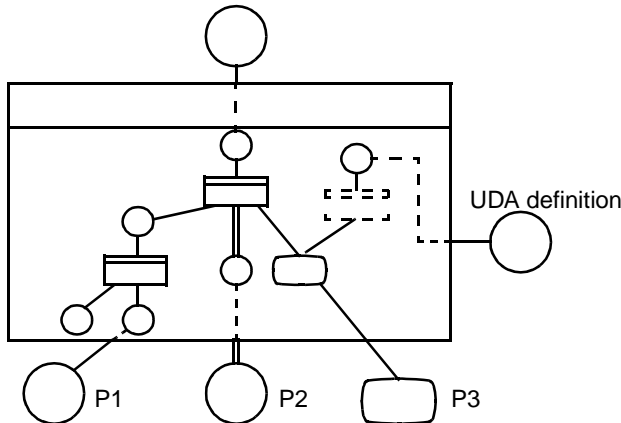
This chapter is made for a better understanding and is addressed to expert users developing UDOs/UDAs.

A UDA behaves as if the UDA model would stand between the parameter and the result/effects:

UDA Definition



UDA Characteristics



When an action is updated, the current parameters are copied into the corresponding object or action. The depending objects are not recalculated. UDA-internal objects belonging to the resulting object and to modify parameters are updated. The internal result and the modified parameters are copied to the current parameters.

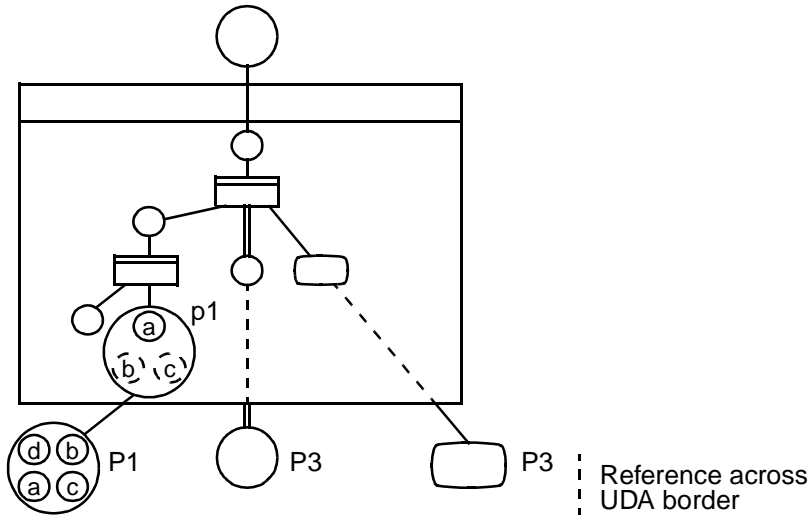
If parameters and resulting objects are UDOs, the situation is more complicated. The actions of the UDA definition layer cannot treat the UDO parameter as entity. They refer to a subobject of the UDO.

In the following cases, the possible relations of an UDA to an UDO are examined:

- UDA with UDO parameter
- UDA with UDO result
- UDA extends resulting UDO
- UDA with modified UDO parameter
- UDA extends a modified UDO
- UDA creates modified objects

Standardization

Relationship of an UDA to an UDO Parameter



The current parameter P1 may include more information than the declared parameter type *p1*. Not all subobjects of *p1* are used. If the UDA is saved, the UDA definition file merely includes the subobject *a*.

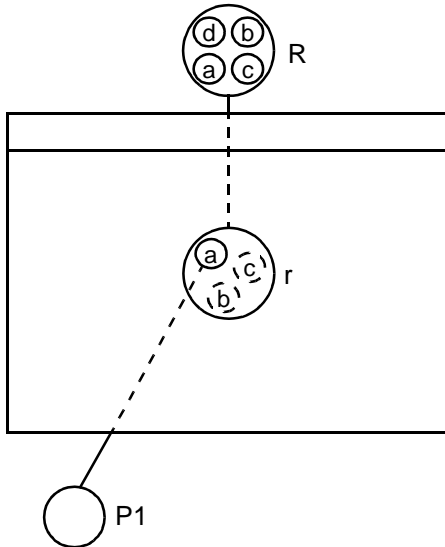


If you copy the current parameter P1 to the internal parameter *p1*, you just need to copy the required subobject.

Subsequently, you create an instance *p1* of the parameter UDO in the UDA model. By selection in UDO in the popup menu of the drawing area, you can access the subobject *a* from the instance *p1*.

Relationship of an UDA to an UDO Result

The easiest way to determine the result UDO of a UDA, is to declare an object of UDO as parameter of UDA.



The current parameter *P1* may include more information than the declared parameter type *p1*. Not all subobjects of *p1* are used. If the UDA is saved, the UDA definition file merely includes the subobject *a*.



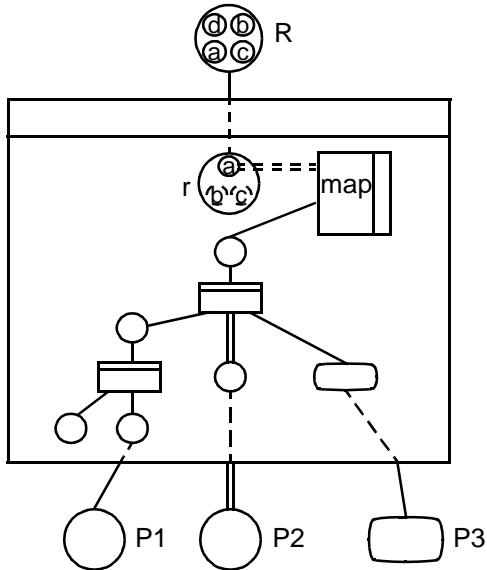
If you copy the current parameter *P1* into the external result *R*, you need to copy the modified subobject *a* only:

- ☞ Choose *set parameter* in the popup menu of the second/third menu column. The dialog box is closed and the system changes the *browse* mode.
- ☞ Choose *selection in user* in the popup menu of the drawing area.
- ☞ Select the object *a* in UDO *r*.

In this case, the UDA parameters may be indirectly connected to the resulting object.

Standardization

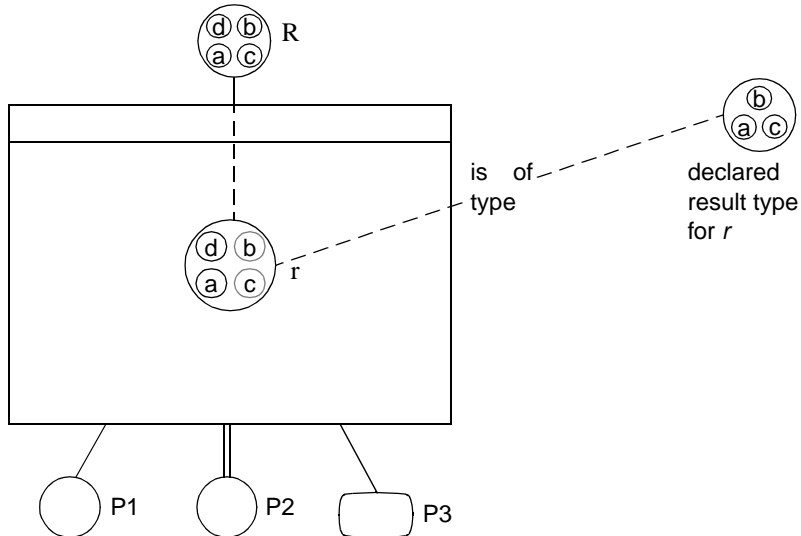
By means of a predefined manipulator, you can map an object within the resulting object to an object within the UDA layer.



The resulting object R may include more information than the declared result type r . Not all subobjects r are determined by the UDA contents. If the UDA is saved, the UDA model file merely includes the subobject a . If the internal result r shall be copied to the external result R , you need to copy the modified subobject a only.

Extending the Resulting UDOs by a UDA

In the following figure, the object type for the result r describes a minimum contents of 3 objects a , b , c . The instance r within the UDA model can be extended by the object d .



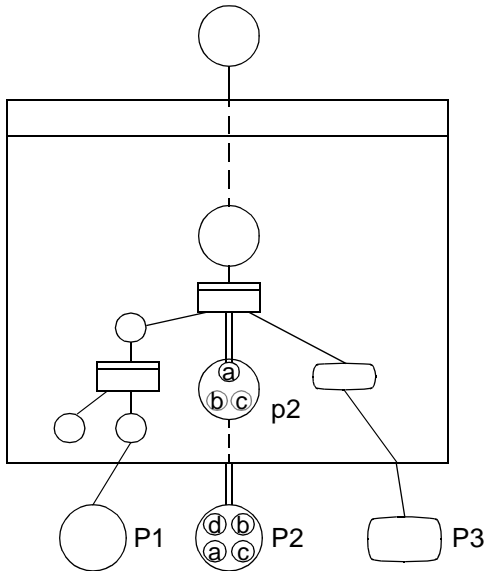
Proceed as follows to extend an UDO instance:

- ☞ Choose *edit definition* in the *Udo* menu. The model box is opened:
- ☞ Click on the name of the UDO to be extended in the list box.
- ☞ Click on *ok*.
- ☞ Add some objects (here: the object d).
- ☞ Choose *close definition* in the *Uda* menu.
- ☞ Click on *ok*. The UDA execution method exports this subobject to the current result.

The internal result object r may include more information than the declared result type.

Standardization

Relationship of an UDA to a Manipulated Parameter of an UDO

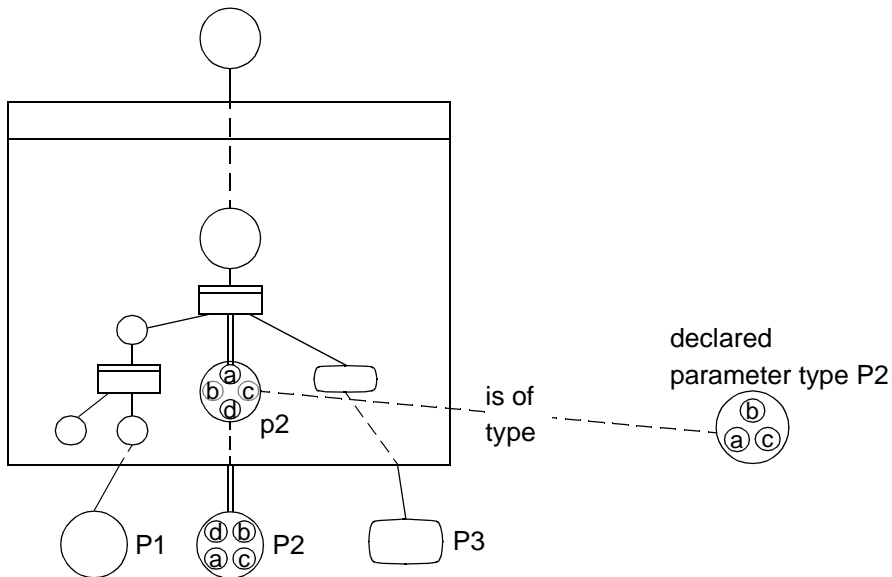


The manipulated parameter *P2* may include more information than the internal parameter *p2*. Not all subobjects or *p2* are manipulated by the UDA contents. If the UDA is saved, the UDA model file merely includes the subobject *a*.

If the internal, manipulated parameter *p2* is copied to the external manipulated parameter *P2*, you need to copy the modified subobject only.

Extending a Manipulated UDO by an UDA

The parameter type for the manipulated parameter $p2$ specifies the minimum contents of 3 objects a , b , c . The instance $p2$ within the UDA model was extended by the object d .



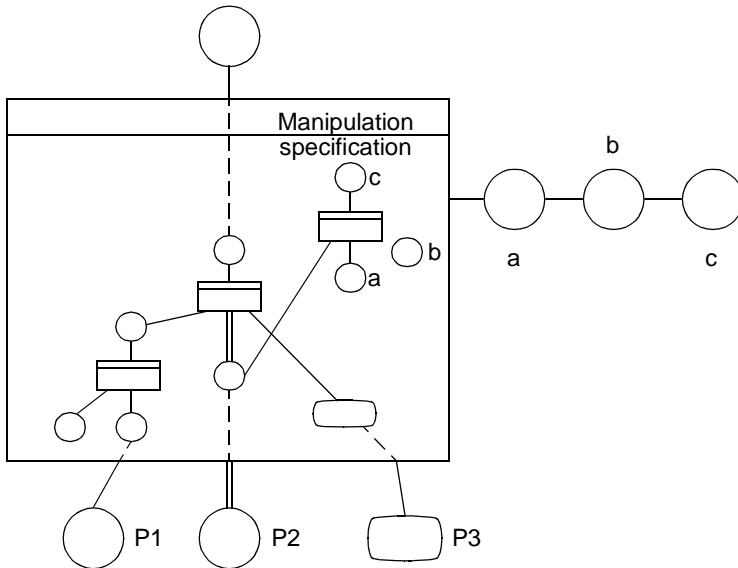
Proceed as follows to extend an UDO instance:

- ☞ Choose *edit definition* in the *Udo* menu. A window is opened with the corresponding UDOT model.
- ☞ Add some objects (here: the object d).
- ☞ Choose *close definition* in the *Udo* menu.
- ☞ Click on *ok*. The UDA execution method exports the additional subobjects to the current parameter $P2$.

The internal, manipulated parameter $p2$ may include more information than the manipulated parameter type declared in advance.

Standardization

Creating Manipulated Objects by an UDA



The action may include a specification enabling objects to be created in the call environment.

Manipulated objects can only be created by means of an AQL program belonging to the UDA.

13.2 Working with Variables

The object *variable* provides functions outside the scope of CAD proper. The important point is that this does not involve elements of a procedure language but objects within the actual data structure. Used in conjunction with user defined objects and actions, this results in a range of options for standardized calculations, the calculation of specification data etc, which may not only be derived from the CAD geometry but also be used to influence it.

The principal applications of the object *variable* are:

- Storage of key design data
These variables may be referred to during design work in such a way that a change to the data causes the design to be adjusted accordingly. If, for example, a gasoline engine is being designed, the key design data will include the power output, nominal rpm etc.
- Calculations within the model
You may execute complex mensuration with nested variables and mathematical functions. This facility concerns other design data derived from the key design data. In the example of the gasoline engine this would include the torque, swept volume, bore, stroke etc.
- Reproduction of a table of figures within the drawing area and translating of the resulting figures to a geometric display.
The use of calculated tables makes it easier for you to comprehend design contexts and thus to arrive at an optimum solution to the problem in hand.
- Memory function for temporary "design figures"
You may delete such variables after use.
- Visualization in numeric form of geometrical data which otherwise would be difficult to estimate.

Example

It is possible to relate the actual swept volume and the actual combustion volume as derived from the geometry. The result is the compression ratio, of which any changes in value may be monitored as the geometry is modified.

Standardization

A variable is represented by its

- Text size (optional)
- Value
- Locating point
- Name

The variable is displayed in the following form by means of a line of text:

Variable name = value

In *Edit* mode, variables resulting from calculations, are displayed the algebraic formula by which behind the value was arrived at in brackets following the value itself.

Example

Product = 386 (v1 * v2)

Variables are located left-justified on an identified or recursively created object *point*.



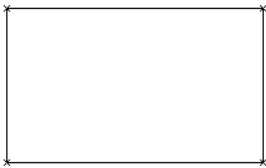
- A variable cannot be identified as an object type *text*.
- Variable of type *length* and *angle* must have a unit specified, and should therefore be used where the unit is known. A variable of type *length* has the unit *mm* or *inch*, according to the system default. If two variables of type *length* are multiplied together, the product should not be applied to another variable of type *length*, since this is an area. It is possible for careless use of variable types to cause errors which are very difficult to detect, since they only show up when converting units.

Example

A rectangle is defined with the key data *length* and *width* which are stored in variables. The area of the rectangle is to be calculated and stored in a variable. A change to the variables *length* and *width* will affect the geometry and the area variable.

\times length = 50

\times width = 30



☞ Create the variable *length* as a variable of type *length*.

☞ Create the variable *width* as a variable of type *length*.

Standardization

- ☞ Create the geometry with the action *macro_rectangle* with the length and width from the variables.

\times length = 50 \times area = 1500

\times width = 30



- ☞ Click on the icon of the object group *variable*.
- ☞ Click on the action *variable from formula*. (see chapter [„Formula Processing“ on page 13-97](#))
- ☞ Enter parameter formula in the text area: $a*b$
- ☞ Click on the variable icon of the opened alternative parameter in the third menu column for input of argument a .
- ☞ Identify the variable *length* in the drawing area.
- ☞ Click on the variable icon of the opened alternative parameter in the third menu column for input of argument b .
- ☞ Identify the variable *width* in the drawing area.
- ☞ Identify locating point in the drawing area.
- ☞ Input variable name *area* in the text area.

The variable *area* contains the area of the rectangle being displayed in square millimeters.

13.2.1 Simplification of Construction Modifications

The search for the right parameter and the descent through the data structures necessary to make changes to an existing design, can be involved (see chapter „*Editing Properties and Parameter in a Complex Design Environment*“ on page 8-9). Provided that the nature of the subsequent variant remains comprehensible, it is worthwhile to make the parameters to be modified dependent on one variable, guaranteeing direct access.

A useful application of this facility is to define the key data by means of variables from the start, before actually starting design work. The geometry will then be generated dependent on the variables.

Example

In the example in the section „*Editing Properties and Parameter in a Complex Design Environment*“ on page 8-9, the rotation of the entire wheel is dependent on the orientation of the center line of the one "genuine" spoke. The center line of the spoke is dependent on an infinite line, whose action has polar coordinates. If the angle is changed, the orientation of the infinite line and, in turn, of the entire wheel, changes accordingly.

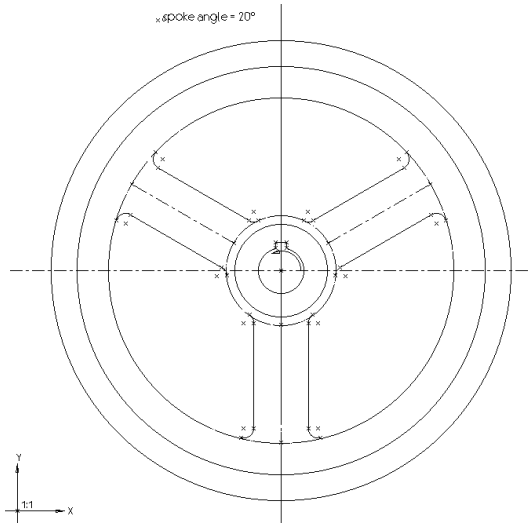
If the angle of the infinite line, as defined under polar coordinates, is made dependent on an angle variable, it is always possible to access this parameter directly.

It is also possible to introduce variables into the design at a later date.

Example

The action of the implicit object *angle* is to be changed from *angle from keyboard input* to *angle from variable*.

Standardization



1. Generate a variable *spoke angle* of type *angle*.
2. Identify the angle of infinite spoke center line to be changed in the drawing area.
3. Select the angle parameter icon in the third menu column. It may be seen from the highlighted keyboard icon in the fourth menu column that input was via the keyboard.
4. Select variable icon in the fourth menu column.
5. Select variable *spoke angle* in the drawing area. The angle parameter of the infinite line has now been made dependent on the variable *spoke angle*.

13.2.2 Formula Processing

EUKLID Design permits complex calculations to be carried out with the help of formulae:

- When inputting via the text area merely the result is saved (see section „[Editing of Text Inputs](#)“ on page 3-31).
- When input as action *variable from formula*, the result is recalculated when a parameter value changes. The calculation rule of the variable forms the formula string (i.e. the existing object type *string*).

The formula processing facility includes the following functions:

- Standard mensurations may be carried out with user defined objects and actions without the need for programming effort.
- Values and formulae may be taken from tables.
- Formulae may be linked via the object *variable* since the coefficients of a formula may in their turn contain formulae; value ranges for functions may for example be defined by a separate formula.
- The use of logical variables allows (with restrictions) the definition of logical relations within a model.
- It is possible to apply spreadsheet programs by showing the formula with the implicit object *string*.

Standardization

13.2.2.1 Formula Input

EUKLID Design requires the input of following data for the calculation of formulae:

- Formula string
- Arguments
- Location point
- Variable name

In addition, you can modify the following default:

- Text height



The formula string entered is called *actual_formula*. This name will not be deleted until the action will have been completed successfully. The name is used to avoid the necessity to re-enter the formula string when an error occurs and the action is aborted. The formula string *actual_formula* can be selected by name.

The formula string consists of operators and arguments.

Operators

Operators are used to define how one or more expressions are to be related or processed. The following operators are available in **EUKLID Design**:

- Fundamental arithmetical operations and functions
- Logical relations

A detailed list of the available operators is given in the following table. The abbreviation *expr* stands for *expression*, which may in turn include operators and arguments. An expression is defined as follows:

expression = argument + operator + argument

An expression may in turn consist of several simpler expressions.

<i>Operator</i>	<i>Interpretation</i>	<i>Pri-ori-ty</i>	<i>Value Range</i>	<i>Remarks</i>
$\text{expr1} + \text{expr2}$	plus	4		
$\text{expr1} - \text{expr2}$	minus	4		
$\text{expr1} * \text{expr2}$	times	5		
$\text{expr1} / \text{expr2}$	divide	5	$\text{expr2} \neq 0$	
$\text{expr1} ** \text{expr2}$	expr1 to the power of expr2	6	$\text{expr1} > 0$	
$\text{expr1} \text{ MOD } \text{expr2}$	expr1 modulus expr2	5		expr1 and expr2 are floating point numbers, i.e. $6.3 \text{ MOD } 3.1$ is permissible and gives the result 0.1.
$\text{sqrt}(\text{expr})$	square root of expr	8	$\text{expr} > 0$	
$\text{exp}(\text{expr})$	e to the power of expr	8		
$+ \text{expr}$	sign	7		More than one sign before expr causes an error, e.g. $\text{expr1} + - - \text{expr2}$ is not permissible; $\text{expr1} + - (- \text{expr2})$ is permissible)

Standardization

<i>Operator</i>	<i>Interpretation</i>	<i>Pri-ori-ty</i>	<i>Value Range</i>	<i>Remarks</i>
- expr	sign	7		Same principle as above
(expr)	brackets	8		Nest to any depth
expr1 < expr2	less than	3		Result is 1 if true, otherwise 0; only one of the following operators may occur in a given nesting: <, >, <=, >=, =, !=
expr1 <= expr2	less than or equal to	3		Same principle as <
expr1 = expr2	equals	3		
expr1 >=expr2	greater than or equal to	3		
expr1 > expr2	greater than	3		
expr1 != expr2	not equal to	3		

<i>Operator</i>	<i>Interpretation</i>	<i>Pri-ori-ty</i>	<i>Value Range</i>	<i>Remarks</i>
NOT <i>expr</i>	logical NOT	7		Result is 0 if <i>expr</i> not equal to 0; result is 1 if <i>expr</i> equals 0; NOT may not occur more than once in succession (NOT NOT causes an error message)
<i>expr</i> 1 AND <i>expr</i> 2	logical AND	2		Result is 0 if at least one of <i>expr</i> and <i>expr</i> 2 equals 0, otherwise result is 1
<i>expr</i> 1 OR <i>expr</i> 2	logical OR	1		Result is 0 if <i>expr</i> 1 and <i>expr</i> 2 both equal 0, otherwise result is 1
sin(<i>expr</i>)	sine	8		<i>Expr</i> is given as an angle
cos(<i>expr</i>)	cosine	8		As sine
tan(<i>expr</i>)	tangent	8	<i>expr</i> = 90+k*180k= - 3,-2,- 1,0,1,2,3,	As sine

Standardization

<i>Operator</i>	<i>Interpretation</i>	<i>Pri-ori-ty</i>	<i>Value Range</i>	<i>Remarks</i>
asin(expr)	arc sine	8	$-1 \leq \text{expr} \leq 1$	Result is an angle; $-90 \leq \text{result} \leq 90$
acos(expr)	arc cosine	8	$-1 \leq \text{expr} \leq 1$	Result is an angle: $0 \leq \text{result} \leq 180$
atan(expr)	arc tangent	8		Result is an angle: $-90 < \text{result} < 90$
log(expr)	base 10 logarithm	8	$\text{expr} > 0$	
ln(expr)	natural logarithm (base e)	8	$\text{expr} > 0$	
abs(expr)	absolute value of <i>expr</i>	8		
int(expr)	whole number from <i>expr</i>	8		Result is the integer part of <i>expr</i>

<i>Operator</i>	<i>Interpretation</i>	<i>Pri-ori-ty</i>	<i>Value Range</i>	<i>Remarks</i>
min(expr1.... exprn)	smallest value from set (expr1...exprn)	8		The comma must be followed by a blank (e.g. min(2,3) =2.3 min(2,3) 2)
max(expr1.... exprn)	greatest value from set (expr1...exprn)	8		The comma must be followed by a blank.
pos(string1, string2)	the position of the first occurence of <i>string2</i> within <i>string1</i>	8		Both <i>string1</i> and <i>string2</i> may be input literally (in quotes) or as arguments; result is 0 if <i>string2</i> does not occur in <i>string1</i>
len(string)	number of characters in string	8		A string which is input literally must be enclosed in quotes e.g. len(hugo)

Standardization

Expressions are processed with standard mathematical priorities. Expressions of the same priority are processed in sequence from left to right. Operators with a higher number have higher priority.



Exceeding a value range causes an error message to be output.

Arguments

After input of the formula string the system calls for input of the arguments. These are used to furnish the coefficients with concrete values. The following arguments are available under ***EUKLID Design***:

- Constants
- Standard constants
- Variable
- Lengths
- Angles
- Alphanumeric character strings

Constants may be input in the following ways:

- 6
- 6.123
- 6,123
- 6e4
- 6e+4
- 5e-7
- 8.9e2
- 8,9e2
- .387
- ,387

The following standard constants may be input:

$\pi = 3.14159256$

$e = 2.7182$



Formulae without arguments, i.e. consisting entirely of constants, may not be used.

Lengths may be given in *mm* or *inches*; the value of the object *length* is stored in the predefined unit.

Arguments and argument names are explicit, sequential parameters which are always specified in combination. In other words, the object *variable* contains any number of arguments consisting of one name and one value.

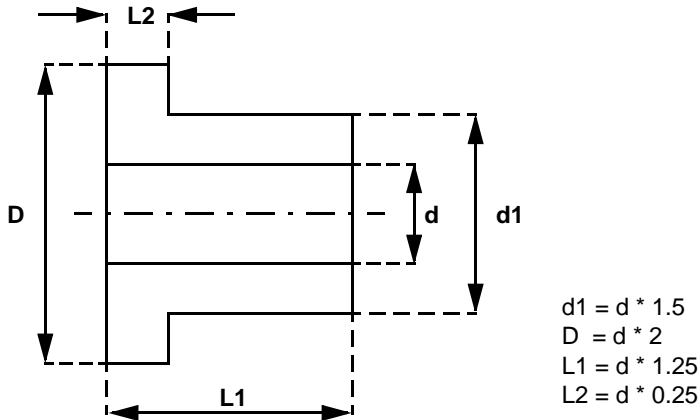
The argument names are specified by the system in *Create* mode in the sequence in which the arguments concerned occur within the formula string. You may specify any number of additional arguments not occurring in the formula. You have to select the parameter argument name for this purpose explicitly. The corresponding value is then called for by the system.

You may input argument values called for by the system via the text area or by implicit construction after opening of the alternative parameter and selection of the corresponding parameter object icons with the mouse. Argument values are based on the objects *length*, *angle*, *string* and *variable*.

Standardization

Example

A design office uses a special type of plain bearing with the following parameters and geometry:



The weight of the bearing is to be calculated.

The following formulae are required:

☞ The cross section areas of the three cylindrical components of the design (flange, body and bore) are calculated with the formula:

$$F = d^2 * \pi / 4$$

☞ The volume of each component is calculated with the formula:

$$V = d^2 * \pi / 4 * h \text{ (cross section area * height of cylinder)}$$

☞ The results of the separate volume calculations are added to give the total volume of the bearing:

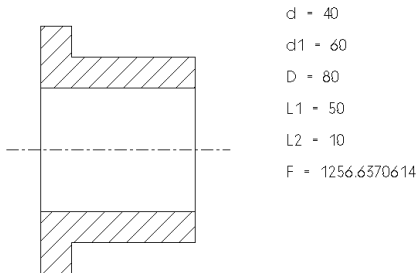
$$V = \text{volume_of_flange } V_1 + \text{volume_of_body } V_2 - \text{volume_of_bore } V_3$$

☞ The weight of the bearing is calculated by multiplying the volume by the density of the raw material used:

$$G = V * \rho$$

Creation of Formula String Variable

The formula string variable for the calculation of the cross section area of the three cylinders is generated by the following steps (these are repeated for each cylinder in turn):



- ➡ Click on the object group icon *variable* in *Create* mode.
- ➡ Click on the action icon *formula variable*.
- ➡ Input formula string $d^2 \cdot \pi/4$ via text area.
- ➡ Click on the alternative parameter argument value d in the third menu column.
- ➡ Click on the variable icon in the third menu column.
- ➡ Identify the variable in the drawing area.
- ➡ Input location point of variable.
- ➡ Input the variable names *cross_section_area_of_bore*, *cross_section_area_of_body* and/or *cross_section_area_of_flange* via text area.
The variable is generated after input of the last parameter.

Standardization

Change Formula String

If you augment the formula string by further arguments which are not in the argument list, you must edit the arguments and the missing ones added. The following example demonstrates how the argument list for the volume calculations with the formula $V = d^2 \cdot \pi / 4 \cdot h$ may be expanded. The processing steps are to be repeated for each of the three formulae in turn.



d = 40
d1 = 60
D = 80
L1 = 50
L2 = 10
F = 1256.6670614

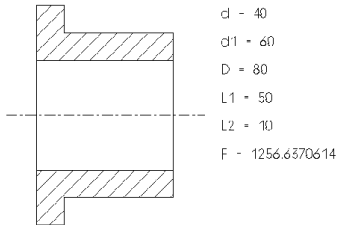
×cross_section_area_of bore = 62831.8530718
×cross_section_area_of body = 113097.3355292
×cross_section_area_of flange = 50265.4824574



- ☞ Click on the object group icon *variable* in *Edit* mode.
- ☞ Identify variable in drawing area.
- ☞ Click on formula string icon.
- ☞ Change formula displayed in text area.
- ☞ Confirm message *Undefined variable type* by clicking on the button OK.
- ☞ Click on lower arrow icon of sequential parameter arguments. The icon for extending the parameter list is displayed in the fourth menu column (see section [„Extending and Shortening of Parameter Lists in Edit Mode“ on page 4-13](#)).
- ☞ Click on this icon. The value of the argument *h* is requested via the text area.
- ☞ Input value of argument *h*.
- ☞ Click on the parameter variable name. The relevant variable name *cross_section_area_of_bore*, *cross_section_area_of_body* or *cross_section_area_of_flange* is displayed in the text area.
- ☞ Change variable name in text area: *cross_section_area_of_bore*, *cross_section_area_of_body* or *cross_section_area_of_flange*.

Linking of Formulae

The object *variable* may be used to link formulae, since variables used in one formula may consist of other formulae. The example shows how this principle is used to calculate the volume and weight of the bearing:



\times cross_section_area_of_bore = 62831.8530718

\times cross_section_area_of_body = 113097.3355292

\times cross_section_area_of_flange = 50265.1621571

$\times V = 100550.9649149$

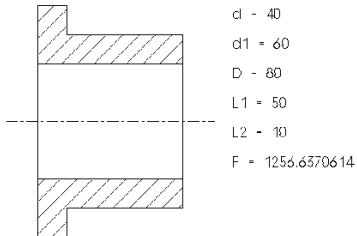


- ➞ Click on the object group icon *variable* in *Create* mode.
- ➞ Click on the action icon *variable from formula*.
- ➞ Input the parameter formula string $V1 + V2 - V3$ via text area.
- ➞ Click on the alternative parameter *argument value* in the third menu column.
- ➞ Click on the variable icon in the third menu column.
- ➞ Identify the variable *cross_section_area_of_bore*, *cross_section_area_of_body* and *cross_section_area_of_flange* in the drawing area.
- ➞ Input the location point of variable.
- ➞ Input the variable name *bearing_volume* via text area.

Standardization

The weight of the bearing is calculated in the same way with the following parameters:

- Formula string $V * Rho$
- Argument value V from variable *bearing volume*
- Argument value Rho from variable to be generated with action *real number* in the fourth menu column
- variable *bearing_weight* via text area



„cross_section_area_of bore = 62831.8530718
„cross_section_area_of body = 113097.3355292
„cross_section_area_of flange = 50265.4624571
„bearing_volume = 100330.9649149
„bearing_weight = 402.23.8596595.



13.2.2.2 Logical Links

The use of logical variables permits (subject to certain restrictions) logical links to be carried out within a model. It is also possible in this way to define value ranges for functions. The following operators are permissible:

<, >, =, <=, >=, !=, AND, OR, NOT

Logical variables are interpreted as follows:

- *TRUE* corresponds to 1.
- *FALSE* corresponds to 0.

Example

The following formula is applicable to a boring in a steel plate:

$l/d \leq 6$

If the depth of the boring exceeds 6 times the diameter a warning message is to be output:

Warning: the depth of the boring exceeds specified limits.

Solution:

This is achieved as follows:

The warning text is stored in a text object. The optional text height is constructed implicitly from a formula string variable:

$3.5 * (l/d > 6)$

The 3.5 corresponds to the height of the warning text.

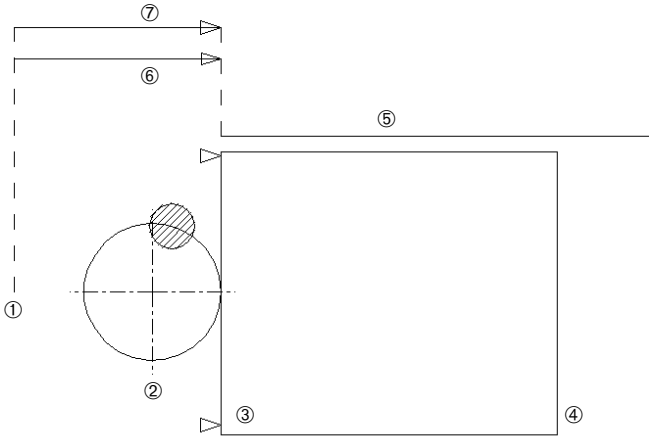
As long as the depth of the boring does not exceed six times the diameter, the expression in brackets is false and returns a 0. This multiplied by the text height of 3.5 gives also 0; thus no text appears.

If the ratio in brackets is greater than 6, a 1 is returned. This multiplied by the text height of 3.5 gives 3.5; thus the text is displayed.

Standardization

13.2.2.3 Formula Processing Exemplified by Kinematic Simulation

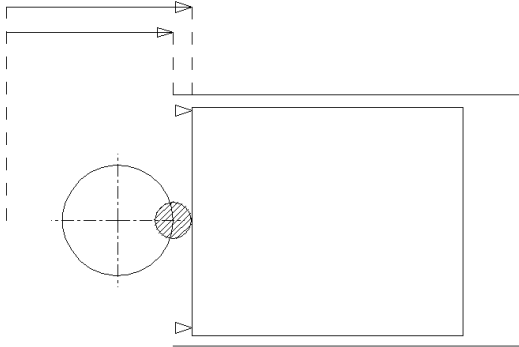
A further example of the scope of the formula processing facility is illustrated by an eccentric drive mechanism:



$$v = 17.5231989$$

- ☞ ① Reference point
- ☞ ② Eccentric sheave
- ☞ ③ Piston face
- ☞ ④ Piston
- ☞ ⑤ Guide
- ☞ ⑥ Dimension line L1 from reference point to backstop
- ☞ ⑦ Dimension line L2 from reference point to eccentric

An eccentric is to be designed to push a piston into a guide. As soon as the follower of the rotating eccentric meets the piston face, the piston is pushed into the guide. The right dead center position of the eccentric corresponds to the furthest movement of the piston. Once the eccentric passes this position, the piston returns to and remains at its starting position. The eccentric follower then leaves the piston face.



$$v = 23.4620882$$

The following is to be borne in mind while designing the eccentric:

- The center of the eccentric sheave is the starting point for the design and forms also the basis for the dimension reference point, the piston face reference point, and the center point and epicenter of the eccentric.
- The center point and epicenter of the eccentric are constructed as points with polar coordinates.
- The piston face is generated as a line with length and angle of orientation through a point with distance along the x axis from the center point of the eccentric sheave.
- The dimension line L1 gives the distance between dimension starting point and the piston face.
- The dimension line L2 gives the distance between the dimension starting point and the epicenter of the eccentric.
- A formula string variable *max (L1, L2)* is used to compare the length of the two dimension lines and to measure the longer of the two.
- The distance of the piston face is taken from the variable.

Standardization

13.2.3 Calculation for Planes and Closed Contours

In **EUKLID Design**, the following calculations can be performed for planes and closed contours:

- Perimeter
- Area
- Volume
- Mass
- Moment of inertia for x axis and y axis resp.
- Polar moment of inertia
- Mass moment of inertia
- Section modulus for x axis and y axis resp.
- Statical moment for x axis and y axis resp.
- Centrifugal moment
- Inertial radius

Any value calculated is stored as object *variable* in the model and can be edited.

EUKLID Design requires the input of the following data for the area calculation:

- Coordinate system, in which the area values shall be calculated
- Density of material
- Length of workpiece in z direction
- Object for which the values shall be calculated (default: contour)
- Location of variable

In addition, you can modify the following defaults:

- Height of variable text (default: 3.5 mm)
- Values to be calculated (default: all)
- Variable name (default: see dialog box)
- Calculation based on contour or plane (consideration of negative areas, default: plane)

In order to enter the values to be calculated and the variable names **EUKLID Design** opens the dialog box shown below. The calculation of all values and the variable names shown are the defaults.

calculation	name
<input checked="" type="checkbox"/> circumference	U
<input checked="" type="checkbox"/> area	A
<input checked="" type="checkbox"/> volume	V
<input checked="" type="checkbox"/> mass	m
<input checked="" type="checkbox"/> xmomentofinertia	I _x
<input checked="" type="checkbox"/> ymomentofinertia	I _y
<input checked="" type="checkbox"/> polarmomentofinertia	I _p
<input checked="" type="checkbox"/> massmomentofinertia	I _z
<input checked="" type="checkbox"/> xmomentofresistanc	W _x
<input checked="" type="checkbox"/> ymomentofresistanc	W _y
<input checked="" type="checkbox"/> xstaticalmoment	M _x
<input checked="" type="checkbox"/> ystaticalmoment	M _y
<input checked="" type="checkbox"/> centrifugalmoment	M _c
<input checked="" type="checkbox"/> radiofgiration	R _g

☐ reset all

OK cancel

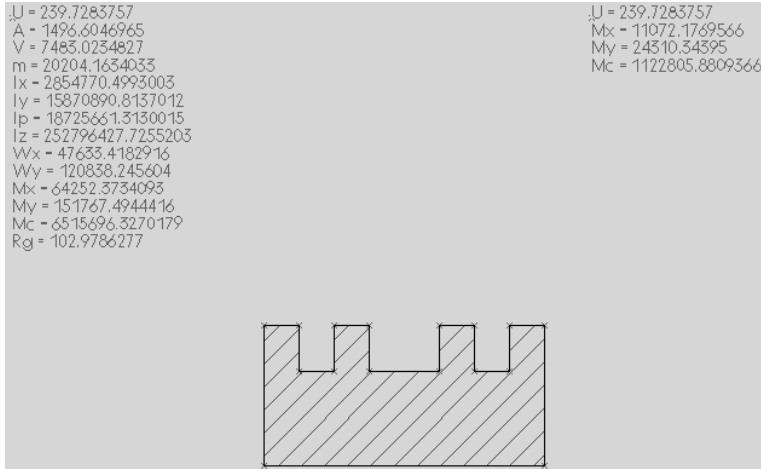
- If you do not want to calculate some values of the selected set, click on the button of the corresponding value(s). Clicked values are removed from the selected set.
- If you want to calculate just some values, click on *reset all*. The selected set is reset. Subsequently, click on the buttons of the value(s) to be calculated. Clicked values are added to the selected set.
- If you want to edit variable names, click on the text box of the corresponding value and enter the value desired.

☞ Complete the input by clicking on *OK*.

Standardization

Example

The contour shown is calculated once based on *contour* and once based on *area*.



Proceed as follows:

1. Click on the variable icon in the first menu column in *create* mode, unless already set.
2. Click on the icon of the *variable from calculated values* (*variable_massprops*) in the second menu column.
3. Select the coordinate system in the drawing area.
4. Enter the density in the text box.
5. Enter the length in the text box.
6. Click on the text height icon in the third menu column and enter 1.5.
7. Select the contour in the drawing area.
8. Select the reference point in the drawing area.
9. Repeat steps 3 to 6.
10. Click on the icon for the calculation base in the third menu column.
11. Click on the contour icon in the fourth menu column.
12. Select the contour in the drawing area.
13. Select the reference point in the drawing area. The values calculated are output in the drawing area.

13.3 Tables

Tables are a tool with a level of usefulness for a wide range of applications accessing external data or databases. Examples are the standardization of components, design variations, standard calculations, transformations, standard production methods and search operations, etc.

- Saving frequently-used parameters for designs.
- Specification and grouping of dimension combinations for parameters of a variant design.
- Access to external data and data sources, e.g. from databases.
- Explicit limitation of variant flexibility in order to reduce the variety of parts for purposes of forming part families. Examples the standardization of components, design variations, standard calculations, transformations, standard production methods, search operations, etc.
- External control of a design via a file table.

Job-oriented design can be automated on the basis of a product model which contains all technological and functional relations. The data from a job are entered into a file table and can be used for automated selection of components from the parts spectrum and to generate complete quote and production documents. This procedure can also be initiated by a connected PPS system.

Tables are useful repositories for parameter values which are used frequently in models. The type and content of the data to be held in the table is defined by the user. Table data may be invoked by almost any function, since the derivation uses the implicit objects *length*, *angle* and *string*.

Depending on the actual data's storage location, tables can be created by different actions. You can choose to store tables exclusively in the model (**internal table**), to store data in a file (**file table**), or to take data from a database (**database table**)

It is possible to nest tables hierarchically by means of references from one table to another (**nested tables**). This allows large amounts of data to be given a clear structure and keeps data redundancy to a minimum.

Certain rows of an arbitrary table can be selected and combined to a new table (**subtable**) by a single action. The subtable merely includes rows matching a specified condition (specified as formula).

Standardization

Tables are implemented as *tab* object. Data are accessed by means of a separate object *tab_instance*, also called index.

Table operation is divided into two phases:

- Preparatory phase Table generation
- Design phase Table implementation

The **preparatory phase** consists of the following steps:

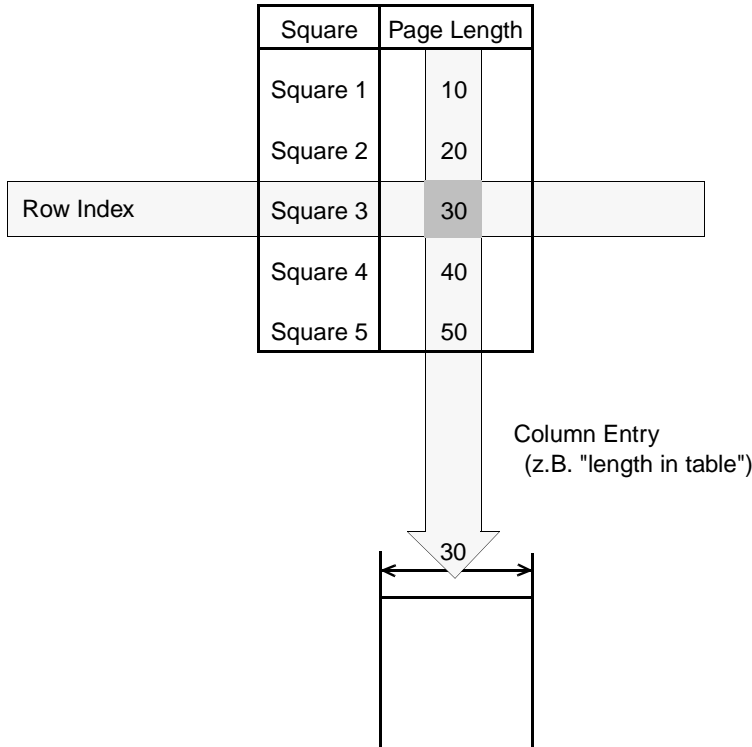
- | | |
|----------------|--|
| table design | The task here is to specify which data will be derived from which tables and where table data shall be stored. |
| table setup | Once the design has been finalized, the tables may be set up and furnished with data. The scope of effectivity of the subtables is also defined at this stage. |
| table indexing | The access methods for data derivation are defined. |

During the **design phase** one value, and one value only, in a table is relevant to a design value. This unique value is determined by combining the row index with the column entry.

Standardization

Two steps are necessary to read a value from a table:

- Input of the table row (*index object*)
- Input of the column



The index is determined by a combination of table row and column, e.g. by clicking on a data field. This defines, at the same time, the column from which the selection is to be made and preselects the table row from which values are read afterwards. The table row can also be determined using comparative operators (relations to design objects).

The column is determined via an action ... *from table*, e.g. *Angle from table* (fourth column): instead of a value entered e.g. from the keyboard, the index is set to the table and the column is selected from which the value is to be taken.

The design geometry is allocated to the values by means of the column entry, while the variants are formed by displacing the valid line (*index*).

Several column entries can be set in a model for the index, and several indexes can be set for each table.

The following description of the table facility adheres to the phase breakdown given above. The generation of tables is normally carried out centrally, once for the whole system, allowing just the necessary data to be retrieved by each user during everyday working (see „[Data Extraction from Tables](#)“ on page 13-139).

Standardization

13.3.1 Table Design

The objective of table design is to specify which tables are to hold which types of data, and determines the options to be available within the system later on. The table structure and indexing (data normalization) must be carefully thought through in view of the wide ranging implications.

The following points must be borne in mind:

- Data redundancy is to be avoided. Data must be input only once as far as possible. Duplicated input is time-consuming, storage-intensive and results in maintenance inconsistencies.
- The standardization of data affects every model which uses data from the tables concerned. Changes made at a later date are difficult to keep track of.

When furnishing several parameters with data from a table, the number of design options is governed by the number of tables. If, for example, a rectangle is to be parametrized by table, one table may be generated, containing length and width. The rectangle may then be parametrized in one step for both these values. Then the number of design options corresponds to the number of rows in the table.

	Rectangle	Length	Width
1st design option →	Rectangle1	10	20
2nd design option →	Rectangle2	20	30
3rd design option →	Rectangle3	30	40
4th design option →	Rectangle4	40	50
5th design option →	Rectangle5	50	60

An alternative is to store lengths and breadths in separate tables. Such a table matrix allows more combination options ($n = \text{entries in table 1} \times \text{entries in table 2}$) with fewer entries, however length and breadth must be maintained separately by the user.

Length table			Width table		
	Rectangle	Length		Rectangle	Width
1st entry →	Rectangle1	10	1st entry →	Rectangle1	20
2nd entry →	Rectangle2	20	2nd entry →	Rectangle2	30
3rd entry →	Rectangle3	30	3rd entry →	Rectangle3	40
4th entry →	Rectangle4	40	4th entry →	Rectangle4	50
5th entry →	Rectangle5	50	5th entry →	Rectangle5	60
5 entries			5 entries = 25 design options		



In view of the wide ranging implications of data storage in tables, formal training in their design and standardization is very much to be recommended.

Standardization

13.3.2 Creating Tables

After the definition which data has to be stored in tables, the required tables can be created.

A *table* resp. *index* object can be created or modified by the following actions independent of the type of data storage and data input desired.



Table index (*tab_instance_row*)



Internal table (*tab_absolute*)



File table (*tab_file*)



Database table (*tab_db*)



Subtable (*tab_sub*)



Convert file table into internal table (*tab_import*)



Export table to file (*tab_export*)



Convert external table into internal (*tab_convert*)



Synchronize file table (*tab_sync*)



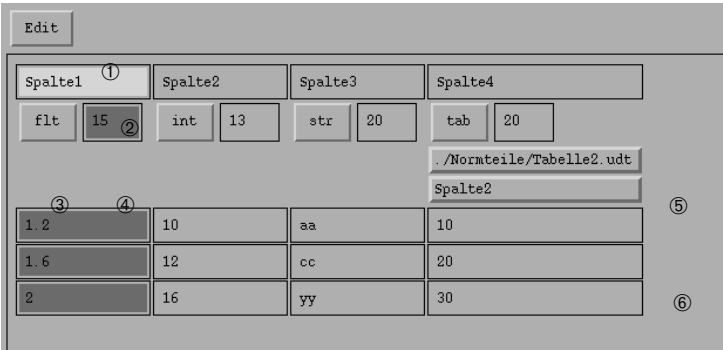
Join table indices (*tab_instancejoin*)

These actions are described in the Online Help.

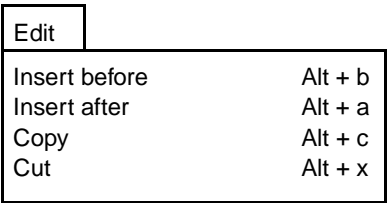
13.3.2.1 The table editor

The table editor allows you to define or edit the contents of tables – with database table, only reading is allowed since the data may be edited only via the database.

A table can contain any number of columns and rows.



① The following popup menu appears when the left mouse button is pressed:



All these functions relate to the selected column or row. If, for example, a column is selected (*Column 1* in the figure above), then an *Insert after* will add a new column. If, on the other hand, a row is selected, then a new row would be added.

Insert before

A new row or column is created before the one selected.

Standardization

Insert after

A new row or column is created after the one selected.

Copy

The selected row or column is written to the copy buffer.

Cut

The selected row or column is written to the copy buffer.

Paste

The row or column in the copy buffer is inserted before the one selected.

②

Column name

The maximum name length is 31 characters; no spaces are allowed.

③

Data type

flt (float)	real number
int (integer)	whole number
str	string
tab	reference to table

④

Column width

The column width is defined as *<Column name>* + 6, whereby 6 characters, for the data type, are always added by the system (the data type is recorded in the column when writing to a data table).

⑤

With the *tab* data type, the reference table and its column must be specified (see section „[Nesting of Tables“ on page 13-142](#))

⑥

A field may be edited by clicking the left mouse button onto it.

A field can be selected for overwriting by double-clicking onto it.

13.3.2.2 Internal Tables

Internal tables do not have external references and store all data in the model. Thus they can only be changed by the model. You can enter data directly via the table editor by the *tab_absolute* action. This is recommended for all smaller tables only.

In addition, you can read data from a file by means of the *read data into model* (*tab_import*) action. Please note, that this does not create any file table.

Furthermore, any table can be converted by the *import to model* action (*tab_convert*) into an internal table.



An internal table is no absolute object and the redefinition to absolute does not correspond to the *import to model* action (*tab_convert*). This redefinition does not change the table type, but merely separates the tab object from the action and hence from its parameters, but the external data access is maintained with its momentary settings, which will result in errors in any case.

13.3.2.3 File Tables

File tables have only one external reference to a file containing all data of the table. They can be created by the *table from file* action (*tab_file*).

File tables can be used just like internal tables. If the user has write access to the file, he may edit them by the table editor.

You should use file tables only if these data shall be changeable independent of the model referring to these data. It should be noted that any change of these data may change the dependent model.

The *export to file* action (*tab_export*) merely writes the data of a table to a file. This does not create a file table. The file, however, may be used to create a file table.

This change does not become visible until a recalculation of the model, e.g. after loading the model or after calling the *synchronize table* action (*tab_sync*).

Standardization



Furthermore, it is to be noted that the file table properly functions in a model only if the file is found by the program. For this purpose, a unique identification is entered into the file to enable it to be identified and found by a file search rule, if it has been physically shifted.

Files declared as read only when the file table is initially created, must not be "moved". If these tables are nevertheless moved, all models using this file must be modified.

Marked table files can be managed by suitable data management without any problem.

In order to ensure compatibility with previous versions, the file name extension was changed from *.tab* to *.udt*. Files with the extension *.tab* are converted to *.udt* and marked, if permitted by the rights. Subsequently, the *.udt* file is read instead of the *.tab* file.

Table files created or read by other programs should have their own extension (not *.tab* or *.udt*). These files are not marked.

Please note, that the *export to file* action (*tab_export*) does not enter an identification into the table file or even deletes an existing identification, if the action overrides a table file.



The identification line must be deleted. This has the same effect as the deletion of the complete file.

13.3.2.4 DataBase Tables

Database tables provide a connection to Oracle or Ingres databases (optional). They can be created by the *table from database* action (*tab_db*) and can be used in the same way as internal tables. They, however, cannot be edited. Individual data fields can be changed according to the user's rights in the database.

Database tables enable the access to large data sets even across computer limits. With regard to usability, they are subject to even more severe limitations than file tables. The access to the database depends on more factors than on the mere file

access, e.g. on computer name, access rights, passwords, etc. If models are located in an instable environment, they cannot be loaded under certain circumstances.

Database tables should be located in a stable environment (computer name, passwords, user names, database names, etc.) with frequently changing data as objects in models.

If data are rarely changed in the database, you can create database tables converting them by the *import to model* action (*tab_convert*) and continue to work with these tables (e.g. standard component databases).

The access to data in the databases mentioned and to the computers used may require passwords which cannot be kept secret. Thus these should be limited with regard to their rights (e.g. no interactive log-in, merely *rexec* of a certain file, access by a certain program only, etc.).

13.3.2.5 Subtables

Subtables are created from the previous table types by accessing to the already existing table.

Since this is a database function, those operations must not use database tables. It is recommended to create the table in the database (e.g. as view) and to use the views as database tables.

The subtable is created by the *tab_sub* action providing a table including only those lines of an arbitrary original table matching the given condition.

Standardization

13.3.2.6 Converting tables into other types

Convert file table into internal table (*tab_import*)

This action converts a file table into an internal table. Tables from a previous version (*.tab*) as well as those with identification numbers are accepted. In any case, a new *tab* object is created.



The action is intended to transform tables from the file system, especially those from previous versions, into internal ones.



Model references to this table (by the action *tab_file*) are maintained. The file may not be deleted, as otherwise the model could no longer be loaded. In other cases, e.g. when a file is transformed from a previous version, the remaining "orphaned" file may be deleted.

Export table to file (*tab_export*)

This action write the table data to a file. This can allow access e.g. to internal tables by other models. The reference to the original table, e.g. a database table, is maintained.



The action is intended to transform write internal tables into the file system.

If the file name extension *.tab* is specified, no identification number is written along into the file.

Convert external table to internal (*tab_convert*)

This action converts external tables already in use by a model into internal ones and transfers the relations. If an index was set to such an external table, it points to the new internal table following conversion.



This action allows conversion of entire hierarchies of tables (switched tables). The hierarchy is maintained, provided all tables are converted at the same time.

Synchronize file table (*tab_sync*)

This action is needed to transfer along to models any changes that were made to files in the file table.

Background: a copy of the file content is held in working memory. If, for example, a line is added to a file using an editor, then the copy in working memory remains unchanged at first. Not until the model is recalculated, e.g. upon loading or when this action is called, does the change become apparent.

Standardization

13.3.2.7 Join table index (*tab_instancejoin*)

Through use of this action you may join different, completely independent tables with one another. By changing only a single index, you may e.g. change all indexes of standardized parts of a model and thus produce different variants very quickly.

To do so, a "master" index must be created to control the indexes. This index could be the index to a "master" table, for example. The tables to be joined must all contain a common column, and an index must be set to each table. Then, through *tab_instancejoin* action, the master table is linked with one of the subordinate tables each time, with the identical columns specified.

Example

Customer-specific variants of a product are to be depicted by a model. The customer-specific characteristics are represented by the following three tables:

				specialrequests.udt			
		int CustNo		int Features		str IR conduit	
		970003		1		yes	
		970004		3		yes	
technology.udt						no	
		int CustNo		int Size		str Material	
		970003		1		PU/PA	
		970004		3		PU/POM	
		970005		6		Color	
design.udt							
int CustNo		str Farbe		str Design		str Type label	
970003		rt-ws		oldstyle		xxx.uda	
970004		gn-bl		standard		yyy.uda	
970005		sw-ge		avantgarde		zzz.uda	

master.udt			
int CustNo	str Shortname	str CoName	str Address
970003	fgrz	A GmbH	80333 Munich
970004	grtq	B KG	70111 B City
970005	awqu	C AG	90111 C City

In individual steps, the master table is joined with one of the other tables, specifying the common column each time.

The remaining indexes are automatically changed correspondingly when the index is edited to *master.udt*.

Master = 970004

Design: standard
 Lichtleiter: ja
 Material: PU/POM

int Kunden-Nummer	str Kurzname	str Firmenname	str Anschrift
970003	fgrz	A GmbH	80333 Muenchen, Abc-Strasse 5
970004	qrtg	B KG	70111 Bstadt, Efz-Strasse 6
970005	awqu	C AG	90111 Cstadt, Ijk-Strasse 7
> []	= []	= []	= []
< []			


☐ Wert aus Tabelle ☐ Filter
☐ Wert aus Objekt ☐


OK Abbruch


Standardization

13.3.3 Example for an Internal Table

In the following example, various rectangles shall be created in a model, the lengths and widths of which shall be read from a table. An internal table is created for this purpose.

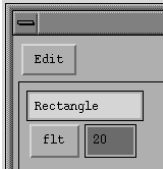
1. Click on the *table* action group icon  in the *Create* mode.

2. Click on the *table in model* icon  .

3. Click on the *name* icon  and enter the name `rectangle` in the text field. Confirm your input with <RETURN>.

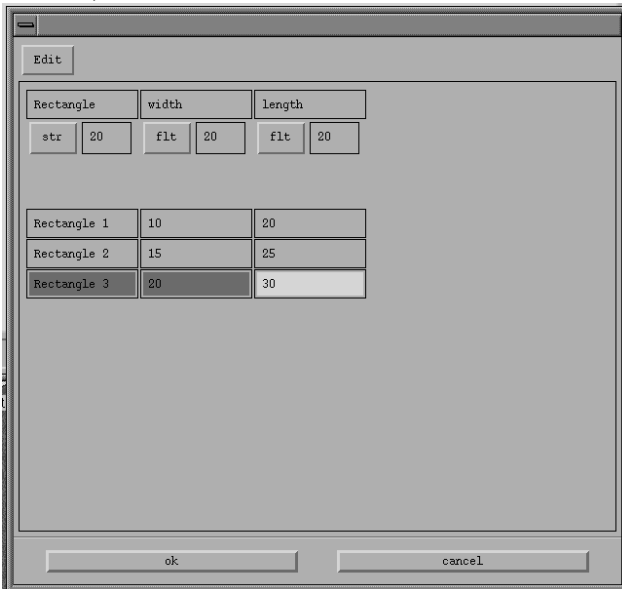
If you do not enter the name, the name is created from the current date and time. Subsequently, the table editor is opened, with which you can create the table.

4. In the table editor, click on the editable column (below the *Edit* button) and enter the `name` rectangle.



5. Now click on the button below and select the column type *String* in the popup menu.
6. Expand the table by one column, e.g. using the key combination <ALT> <a>. Repeat this process once more. The table now has three columns.

7. Change the column names to `width` and `length` and the column type to `flt` (real number).



The screenshot shows a dialog box with an 'Edit' button at the top left. Inside the dialog is a table with two columns and three rows. The first row has headers 'width' and 'length'. The second row has values '10' and '20'. The third row has values '15' and '25'. The fourth row has values '20' and '30'. Below the table are 'ok' and 'cancel' buttons.

	width	length
Rectangle 1	10	20
Rectangle 2	15	25
Rectangle 3	20	30

8. Edit the columns of the table. Enter `rectangle 1` in the rectangle column, 10 in the width column and 20 in the length column.
9. Copy the first line using `<ALT> <c>` and paste it by `<ALT> <v>`. Repeat this procedure for the third line.
10. Now edit the rectangle column in the second line and enter `rectangle 2` and the values 15 (width) and 25 (length).
Edit the third line of the table correspondingly.
11. Confirm your entries with `OK`.
The table is now complete (see figure above) and can be used within the model.

Standardization

13.3.4 Creating an Index

In order to extract data from a table, the system must be informed as to which data from the table shall be output to objects. Since several values are generally required from one row, this is first done by means of a pointer to a row.

The *Index* object (*tab_instance*) is used as the pointer. It specifies the row of the table to be used for the data extraction. At the same time, the column is defined for the selection.

An index is created by the *row index* action (*tab_instance_row*) on an arbitrary table.



To ensure a unique search for the index even after a possible extension of the table, the key entry of the selected column is saved as representation of the row. Thus, the column selection is of great importance for determination of the index. Key entries must exist exactly once only. Thus you may wish to consider inserting a column into each table to contain a unique key entry (e.g. code number).

A table must include at least one index to enable data extraction. If several data rows are required from the table, several index pointers are required likewise.

By editing the index, you can extract further data. This enables e.g. complex geometry modifications for variant constructions.

An index may be defined by two different types within the action:

- value from table (direct access) and
- value from object (calculated index).

Selection by Direct Access

If you select by direct access, the first table row is searched where the value of the selected table column matches the index value - even after a table modification. The row found is highlighted.

Example

An index shall refer to the example table „[Creating Tables](#)“ on page 13-124. For this purpose, an index shall be created with direct access.



1. Click on the action group icon *Table* in *Create* mode.



2. Click on the *Index* icon .
3. Select the name of the table in the dialog box *Name* and confirm with *OK*.
4. Click onto the entry *Rectangle 1* of the column *Rectangle* in the index menu and confirm with *OK*.

This sets the index to the table row *Rectangle 1*. At the same time, you have defined the selection column. (It is irrelevant for direct access)



By default, the index is given the same name as the table to which it points, but this may be edited if desired.

Standardization

Calculated index

By means of the **calculated index**, a component may depend on its construction environment and nevertheless be a standard component. For this purpose, the index is provided with a conditional relation to a reference object, e.g.

screw length \geq gripping length + minimum thread reach

The value matching this condition best is taken from the table. If the gripping length is changed, the screw is automatically adjusted according to the standard: through the conditional relation, the value is taken from the table which is most suitable following the change. In contrast, with direct access there is no relation between screw and gripping lengths, and the screw must be changed explicitly via its screw index.


Example: Calculated Index

From the "Rectangle" example table, a rectangle shall be selected with a length smaller than the reference length.



1. Click on the action group icon *Table*  in *Create* mode.



2. Click on the *Index* icon .
3. Enter the name of the table in the *Name* box.
4. Click onto the *length*.
5. Select the *value from object* option in the *index* box.
The dialog box is closed.
6. Now you can select an object to which a relation is produced in conjunction with a condition.
Following selection, the index box is reopened.
7. Activate the access criterion $<$.
8. Confirm with *OK*.

The index is set to the table row whose *length* column contains the maximum value smaller than the length of the reference object.

13.3.5 Data Extraction from Tables

After having created the necessary tables and defined at least one index per table (refer to chapter „*Creating an Index*“ on page 13-136), you can extract data from the existing tables. Any parameter based on the object *length*, *angle*, *number*, *proportion* and *string* can be supplied by the table.

The parameter to be supplied by the table is to be entered by the 4th menu column. For this purpose, select the *from table* action.



After having specified the index name and the column, the value is read from the table by means of the index located in the specified column in the valid row.



The use of external tables requires stable tables, i.e. their column architecture may no longer be changed. If, for example, an index is set to a particular column and the column were to be deleted, then the corresponding model could no longer be loaded.

Example for Data Extraction from Separate Tables

In this example, length and width of a rectangle are supplied by an index from two separate tables. This requires the following steps:

1. Create a table with the *rectangle* and *length* columns and a table with the *rectangle* and *width* columns as shown on Page 13-122.
2. Use direct access to create an index for each of the two tables, e.g. *lLength* and *lWidth*. Specify the *rectangle* column and an arbitrary entry as index contents for each (refer to chapter *Creating an index* starting on Page 13-136).
3. Create a *Rectangle with angle and rounded corners*.



Click on the action icon *rectangle with angle and rounded corners*.



z value (optional)

Standardization



Line mode and thickness (optional)



Rounding radius (default: 0.0)



Supply the *position* parameter.



Supply the *angle* parameter.



Supply the *length* parameter:



Click on the action *length from table* in the fourth menu column.



Subsequently, the second, third and fourth column of the *length from table* action are modified.



Enter the index name *lLength* by the *name* box.



Select the *length* column by the value selected in the dialog box. Subsequently, **EUKLID Design** returns to the creation of the rectangle. The second, third and fourth columns are adjusted to the action *rectangle with angle*.



Supply the *width* parameter:



Again, click on the *length from table* action in the fourth menu column.



Subsequently, the second, third and fourth column of the *length from table* action are modified again.



Enter the index name *lWidth* by the *name* box.



Select the column with the current value in the dialog box. The rectangle is created with the values extracted from the table.

In this example, the indices can or must be changed (by Edit Index *Length* or *Width* from the menu) in order to obtain other rectangles.

Example of Data Retrieval from one Table

In this example, the length and width of a rectangle are obtained in one step via an index. The procedure is as follows:

1. Generate a table with three columns, *Rectangle*, *Length* and *Width*, as shown on Page 13-122.
2. Generate an index *Irectangle* with direct access to this table. The index contents to be input are the column *Rectangle* and any required entry (see „[Creating an Index](#)“ on page 13-136).
3. Create a rectangle (see Online Help, *Creation of Polygons*). Furnish the parameters *Length* and *Width* via column 4 as definition type *Length from table*. Use the index *Irectangle* previously created and the columns *Length* and *Width*.

If the index *Irectangle* is altered by this example, it is possible to toggle between fixed, pre-selected rectangles, the dimensions of which are defined by the table.

Both these examples illustrate that the particular application determines the composition of tables and indexes: if a rectangle always has fixed characteristics, then it makes sense to use only one table. An example of this would be a standardized part such as a screw.

If length and width of the rectangle can assume values independent of one another, then several tables should be used. An example of this would be a series of plates that allows combination of all standardized widths to any standardized length.



If it is intended to furnish input parameters for a user element from one table, i.e. in one step, the index must be set to the required table and to the column which links the values (e.g. column *Rectangle* in the table used as an example), and it must be defined as an input parameter.

Standardization

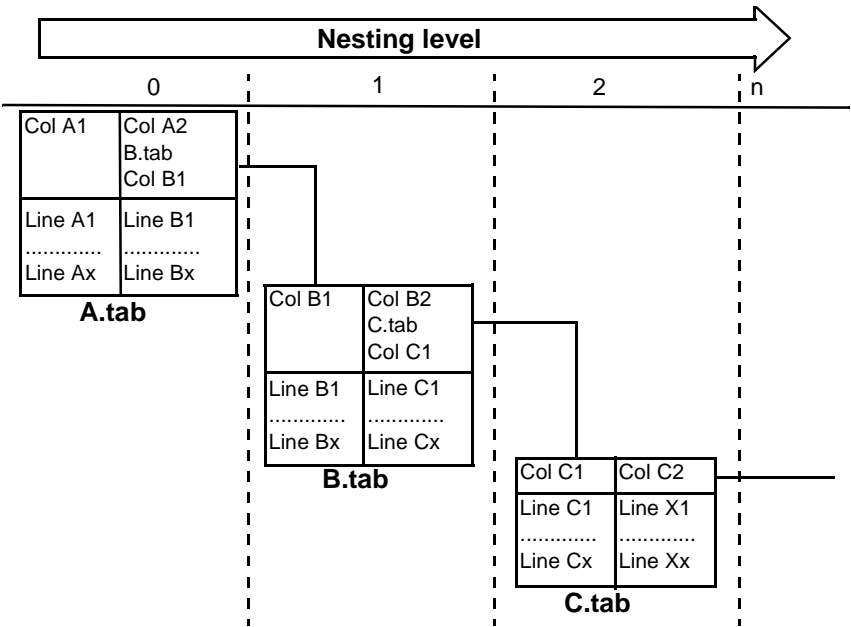
13.3.6 Nesting of Tables

Nested tables are useful where table entries are used often or where tables are to be kept small for the sake of clarity.

The index column establishes a link to another table. Each entry in the index column must exist as a key entry in the linked table. The number of key entries used in the index column must be \leq the number of rows in the linked table (this corresponds to the number of bi-unique key entries).

The data in the linked table can be accessed via the row index of the master table (the effect is the same as with one large table).

The tables are nested via "table-implicit" indexes. The column which is to refer to a column in another table must therefore be of the data type *table*. The table and column names to which reference is to be made must be entered in the table header.

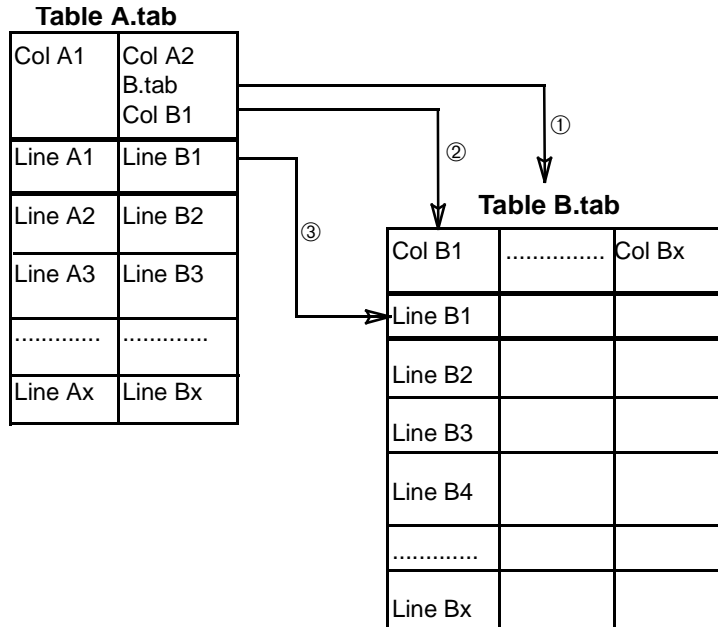


Implementation

How to nest tables is described using the example of two tables called "A.tab" and "B.tab":

1. Generate the table at the deepest nesting level first (*B.tab* in figure overleaf).
2. Set up a column of column entry data type *table* (index column) for each table column to which reference is to be made, in each referring table, from which the table being generated is to be accessed (*A.tab* in figure overleaf).
3. You must set the type of column to be referred to in advance.
Two fields appear directly below the data type in the dialog box.
Confirm the upper field and select the table to be referred to (destination table) in the dialog box displayed.
The lower three fields have special meanings:
 - ... display further tables
 - <internal> create internal table with *table in model* (*tab_absolute*)
 - <extern> create file table with *table from file* (*tab_file*)
4. Now you are prompted to enter the column name:
Confirm the second field located directly below and choose the column name desired in the dialog box.
5. Enter the intended indexes in the index column. They must be locatable in the destination table.
It is not necessary to generate explicitly an index to a table which is to be accessed directly. The index is set to point to the referring table and the table referred to is accessed automatically from there via the implicit index. Unless otherwise specified, the search for the table being referred to is run in the directory containing the associated referring table.

Standardization



- ① Reference to table
- ② Reference to table column
- ③ Implicit index pointing to table row

Example

A table of screw dimensions does not, for the sake of clarity, contain the metric threads; these are held in a separate table. The link between the tables should be achieved via a column *Thread* in the screw table, which points to the thread table.

The following must already have been carried out:

- Generation of the thread table
- Generation of the table for screw dimensions with a column named *thread* of data type *table*.

The indexes are generated in the following steps:

1. Generate thread table.
2. Generate screw table with columns *Nominal*, *Head diameter*, *Head height*, *SL* and *Thread*. The column *Thread* is of data type *Table* and points to the column *Type* in the table *Thread.tab*.
3. Input the name of the table referred to from the index column into the table header using the editor.
4. Input the name of the column referred to from the index column into the table header using the editor.
5. Input the corresponding values into the index column. These must be accessible in the table being referred to.

13.3.7 Application of Tables to Standard Components

The furnishing of parameters with values from tables is particularly relevant to standard components. By using tables it is possible to ensure that components and production methods generated as UDO/UDA, are provided with standard dimensions.

Example

A design department wishes to use only a limited number of standard screws with fixed dimensions. The screw diameter is to be taken directly from a table of available screws. The screw length is to have a minimum value depending on the fixture required (e.g. shank length $+1.5*d$). A user element is generated for this purpose with the parameters *nominal diameter* and *screw length*.

The input parameters are taken from tables. The fixed dimensions for the necessary screws must be saved in data tables for this purpose.

The first table contains all nominal diameters and dependent dimensions (e.g. head diameter). The second table contains all the screw lengths and dependent dimensions (e.g. thread length).

One index points to the table row currently being read and furnishes the parameters with data from the tables (see figure on Page 13-147).

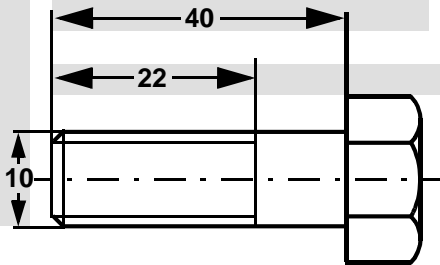
The index to the diameter table is used as an input parameter of the user element, and the index to the length table is chained to a compare object *length* in the design environment, which is defined as an actual input parameter, via the access criterion \Rightarrow . The system retrieves from the length table the standard length which satisfies the condition and displays the screw with the corresponding geometry.

Table „Diameter.tab“		
Description	Norminal	Reference to Thread.tab Norminal
M6	6	6
M8	8	8
M10	10	10
....
Mn	n	n

Index
(Directly accessible)

Table „Length.tab“	
Screw Length	Thread Length
10	10
12	12
20	20
30	30
40	22
...	...
n	n

Index
(Directly accessible
e.g. Screw Length >=32)



14 Visualization Techniques

14.1 Windows

14.1.1 The Desktop

In the working area (desktop) it is possible to have up to ten different model windows open, and to position them freely. You can determine which model should be displayed in which window. Different views of the same model may be displayed in several windows.

The number, size, position and display of windows are saved in the model, and restored after loading.

The desktop can extend beyond the application window, model windows can be of any size. In this case the view can be moved using scrolbars.



Activate the scrolbars in the Options-Menu → *Window-Layout* → *Scrolled Working Window*

Iconized Windows

Model windows which have been iconized are located in the upper left corner of the desktop. Via double click or by using the pop-up menu of the right mouse button it is possible to return a window to its original size and position.

Tooltips

When the cursor is held above an iconised model window, the model name or the file name will appear as a tooltip.

Visualization Techniques

14.1.2 Types of Windows

You can switch between the various model windows at will, and work in any combination of these windows. You can e.g. set a layer to active in a structure window, select a reference point in a geometric window and in a second geometric window complete the action.

Geometric Window

This contains a geometric design, your drawing.

Structure Window

This displays the layer structure of your building parts symbolically.

By means of this type of window you can e.g. enter the product structure of your design, before creating any kind of geometry. The individual structure items can easily be assigned, switched or moved.

Further information is contained in the section „[Visualization of Layers](#)“ on page 14-18.

14.1.3 Basic Elements of Model Windows

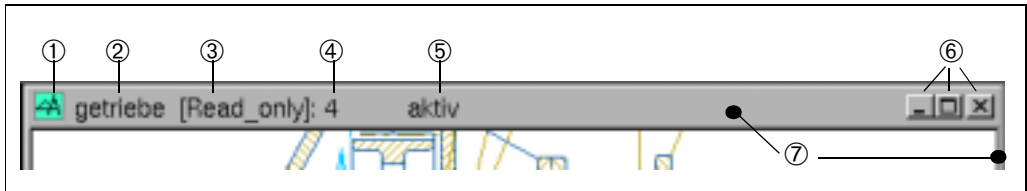
In the **drawing area** of the model window the model is displayed, positions can be defined, and construction elements identified as parameters.

A context sensitive pop-up menu can be opened using the right mouse button. (see „[Popup Menus](#)“ on page 3-26).

The content and **visualization methods** of a window are saved in an implicit object *view*, which can be configured for special displays.

The window frame

- contains information on the model.
- can be modified in size dragging with the left mouse button or command buttons.
- enables a popup menu to be opened using the right mouse button.



<i>Position</i>	<i>Meaning</i>
①	Different color for each model
②	Model name
③	Additional information, when model is opened in read only status
④	Current window number
⑤	Additional information when model is active
⑥	Minimize, maximize, close
⑦	Open a popup menu using the right mouse button

14.1.4 Open a Model Window

New model windows are always opened in a standard size, the display uses a separate object *view*.

The model of the topmost window is the active model.

A new model window can be opened as follows:

- Window-Menu → *New Window*, Choose window type.
Content of window is the active model. A standard display is used, in which for example all layers are made visible.
- Use the action *New Window* in the action group *Windows* (2nd Menu).
Select via mouse click a window which defines the model and view.
The view of the new window takes the values of the selected one.

Visualization Techniques

14.1.5 Change size and position

Size and position of model windows can be changed as follows:

<i>What is your goal?</i>	<i>Steps to be taken</i>
Change to any size	Bewegen Sie die Maus auf einen Fensterrand oder eine Ecke, bis der Cursor seine Form ändert. Ziehen Sie, bei gedrückter linker Maustaste, das Fenster auf die gewünschte Größe.
Change to any position	Klicken Sie mit der linken Maustaste auf die Titelleiste des Fensters und bewegen Sie es mit gedrückter Taste zur gewünschten Position.
Minimize / maximize	<ul style="list-style-type: none">● Button Minimize ☯: Minimize a window to icon● Button Maximize ☯: The window uses the whole desktop. Clicking again returns to size before.

14.1.6 Activate Models and Model Windows

Concealed or iconised model windows can be brought to the foreground at any time.

<i>If you ...</i>	<i>the following will happen:</i>
click on a window frame	<ul style="list-style-type: none">● The model window moves to the foreground● The selected model becomes active
select a model in the Window-Menu	<ul style="list-style-type: none">● All model windows of the selected model move to the foreground● The model becomes active



Should the activation of a model window involve a change in the active model, the current action will be cancelled.

14.1.7 Arrange Model windows

Model windows can be arranged quickly and clearly as follows:

<i>Arrangement</i>	<i>Steps to be taken</i>
Tiled horizontally, Tiled vertically, Overlapping	<ul style="list-style-type: none">● Click on the appropriate icon in the ruler.● Select the corresponding entry in the Window-Menu.
Standard	Select the entry <i>Standard</i> in the Window-Menu. If possible, all windows are arranged such that they fill the working area.
foreground, background	<ul style="list-style-type: none">● Select the corresponding entry from the popup-menu of the right mouse button on the window frame● Use the corresponding action from the action group <i>Window</i> (2nd menu).● Foreground: click on the frame of the desired window.

14.1.8 Close Model Windows

Click the button *Close* (Ⓒ see diagram on Page 14-3) in the top right-hand corner.



When closing the last window of a model it is also possible to close the entire model (automatic question popup)

Visualization Techniques

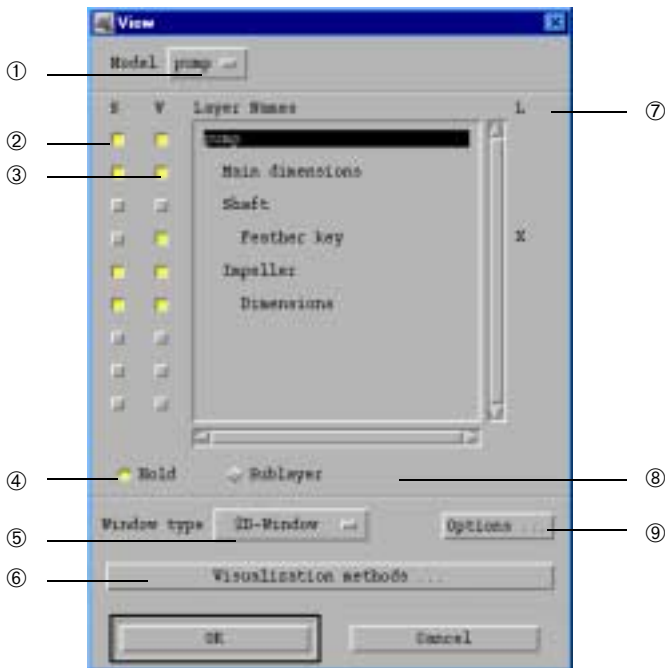
14.1.9 The View of a Window

Individual settings can be used for each window, e.g. particular layers or certain object types can be set to invisible. The settings are attached to an implicit *view* object and saved together with the model windows in the model.

Should you wish to use this view object for other model windows, please use the action *Save view* and *Assign view to window* of the action group *Window* (2nd menu).

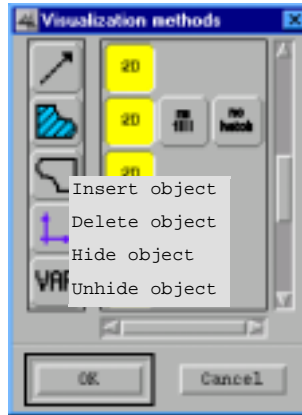
View objects can be selected either via their names or directly in the appropriate window.

In order to modify a view, select the entry *View* in the popup menu of the right mouse button in the drawing area.



<i>Position</i>	<i>Bedeutung</i>
①	Model selection in pulldown menu Display or change the current model
②	Switch on/off selectable Layer
③	Switch on/off the visibility of the Layer
④	Switch on/off immediate window update
⑤	Window type : Geometric- or Structure-Window Selection in pulldown menu.
⑥	Visualization methods, see next page
⑦	Layer is locked
⑧	Applies changes in visibility to all sublayers
⑨	<ul style="list-style-type: none">● Hiddenline● Display line thickness: Real thickness or colored display Selection in separate dialog box

Visualization Techniques



In this dialog box you can define the visualization method for each object type. Visualization of each object type can also be disabled.

<i>What is your goal?</i>	<i>Steps to be taken</i>
Select visualization method	Click on an available visualization method of the right-hand list.
Display/Hide object type	<ul style="list-style-type: none">● Click on object type in the left-hand list.● Select <i>Hide object</i> / <i>Unhide object</i> from the popup menu of the right mouse button on the object type
Insert / Delete object	Add or remove User Defined Object types to this dialog box.

14.1.10 View using the function keys F5 - F9

You can set the function keys <F5> to <F9> to keep views.

<i>Goal</i>	<i>Steps to be taken</i>
Define view	<ul style="list-style-type: none">● Pan and zoom the desired view● Press <Shift> and <F5> ... <F9>
Display view	Press <F5> ... <F9>

14.2 Grid

You can switch on a background grid to provide assistance while sketching (see section „[Sketching a Contour – contour_sketch](#)“ on page 15-27).

This action resides in the action group *Window* (2nd menu).

The following settings can be made:

- Switch grid visible
- Snap sketched points to grid points
- Define the grid point spacing

For more information please refer to the Online-Help.

Visualization Techniques

14.3 Line Style

You can set the line style of simple lines, circles and arcs, ellipses and elliptical curves:

<i>When?</i>	<i>How can you set the line style?</i>
before creation	Via action group <i>Defaults</i> (see Online-Help)
during creation	Property of the respective object
after creation	<ul style="list-style-type: none">● single: Property of the respective object in Edit-Mode● multi: Via action group <i>In Rectangle</i> (see Online-Help)

You can enter the line style via the following dialog box:

Line style

Solid line, not visible (0.50 mm)

Long-dashed line (0.25 mm)

Dash-dotted line (0.25 mm)

Dash-double-dotted line (0.25 mm)

Short-dashed line (0.25 mm)

automatic line style (0.50 mm)

Break line (0.25 mm)

Line thickness

0.18 mm

0.25 mm

0.35 mm

0.50 mm

0.75 mm

Entry in mm

Each line style is allocated a standard thickness, which can be changed.



The *Break line* can only be used for finite lines. Therefore it only appears in this dialog box for line properties, and not for the *Defaults* and *In Rectangle* actions.

Automatic line style

The automatic line style is to be used for hidden lines.

<i>Use for</i>	<i>Display</i>
Visible line	Solid line, thickness selectable
Hidden line	Line style and thickness as you like

You define the line style for the visible and hidden objects in separate dialog boxes.

For further information on hidden lines please refer to section „[Suppression of Hidden Lines](#)“ on page 14-14.

Visualization Techniques

14.4 Colors of Objects

Certain objects may have a color as property. You can define the color at creation or retrospectively.

Color, used as parameter or property, is selected via the following dialog box:



The first six colors (in the first row of the dialog box from left to right) are used as standards shown below:

<i>Color</i>	<i>Standard color for</i>
1st color	Background of the 2D geometric window
2nd color	Infinite lines, coordinate systems
3rd color	Thin line thickness, Dimensions, Symbol balloons
4th color	Points, medium line thickness
5th color	Standard line thickness for geometry
6th color	Cursor and selection color

Allocate colors to objects

<i>When?</i>	<i>How can you allocate the color?</i>
during creation	Property of the respective object
after creation	<ul style="list-style-type: none"> ● single: Property of the respective object in Edit-Mode ● multi: Via action group <i>In Rectangle</i> (see Online-Help)



The individual object color is only be displayed, if the layer these objects belong to, is not active.

Reason: In the active layer the objects will be displayed in the standard colors (2-5) to enable line weighting.



If you want to reset an individual color of an object to its standard color please select the background color (=1st color).

Allocate colors to layers

Please click in the dialog box Layer-Menu → *Status* in the column C the field beside the requested layer and select a color in the following dialog box.

All objects and plane border objects respectively of this layer get the selected color. Fill colors are not changed.

Allocate colors to structure elements

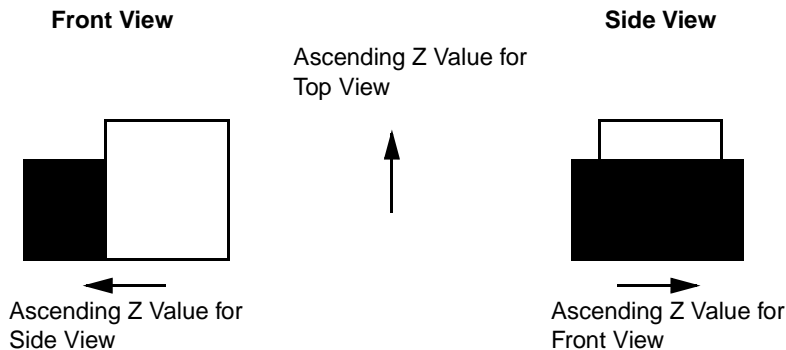
Please click in the dialog box Edit (popup menu of the right mouse button in the structure window) on the button color and select a color in the following dialog box.

All objects and plane border objects respectively of this layer (= structure node) get the selected color. Fill colors are not changed.

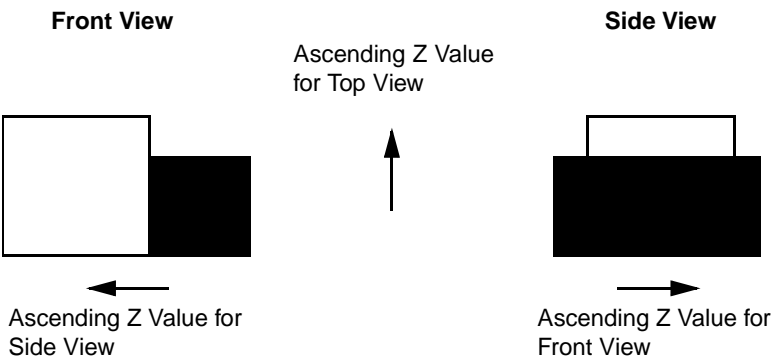
14.5 Suppression of Hidden Lines

The generation of different views and elevations will often cause one component to lie in front of or behind another. The hidden contours are automatically suppressed by the system without the user needing to split them up or delete them. To achieve this, the components lying in front must be redefined as filled areas with a larger Z value than the components lying behind. The display is adjusted automatically according to the direction of viewing. The effect of this facility is shown in the two illustrations below.

Display before Shifting Components



Display after Shifting Components



The **Z value** is an implicit object of the type *length*. The length is measured from an imaginary zero point along the Z axis.

- Larger Z values have a higher display priority, i.e. a object with a high Z value will conceal one with a low Z value.
- If the Z values are the same, the object types have different priorities, e.g. a line is located on a plane with the same Z value. The priority of planes is lower here than that of other objects.
- Objects without a Z value (dimensions, texts, symbols, ...) have the highest priority and are fully visible.

The following **object types** may be furnished with a Z value:

- Line
- Circle
- Ellipse
- Spline
- Contour
- Surface

<i>When?</i>	<i>How can you set the Z value?</i>
before creation	Via action group <i>Defaults</i> (see Online-Help)
during creation	Property of the respective object
after creation	<ul style="list-style-type: none">● single: Property of the respective object in Edit-Mode● multi: Via action group <i>In Rectangle</i> (see Online-Help)

Visualization Techniques

The **input** may be made as follows:

- Via text area
- By implicit construction
- Sketching in the drawing area

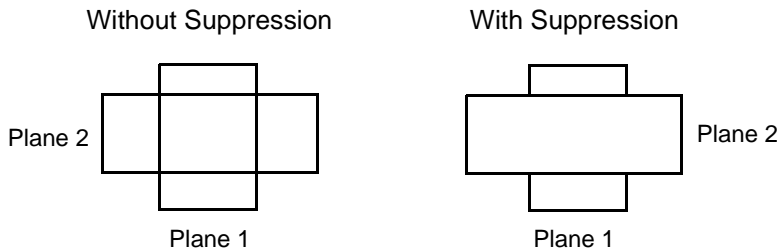


For sketching input the Z value of the identified object is used.

Example

The following illustration compares the visibility of planes. In the first case, **without suppression** of hidden lines, the two surfaces lie in one plane. The prerequisite is that the planes have either the same Z value or none at all, and that the property *Hiddenline* has not been selected for the fill style.

In the second case, **with suppression** of hidden lines, surface 2 has a bigger Z value than surface 1. The *Hiddenline* property is set for fill style. A part of plane 1 is therefore "hidden" via surface 2.



The plane border objects are only suppressed in the display and not actually clipped, i.e. an object which is partially hidden by another is not split into other objects, but **one** object.



Please notice the following when you enter Z values for rotational parts:

- The **sectioned planes**, e.g. bearing, must have a higher priority, the closer they are to the middle line.
- The **non-sectioned planes** must have a higher priority, the further away they are from the center line.

During appropriate design of models the control of the visibility of objects with the Z value affords the following options:

- The visibility of objects may be changed automatically by linking of Z values and other objects via the implicit object *length*.
- The visibility of entire assemblies within a model may be controlled by making the objects of the assembly dependent on a single Z value. To change the visibility one object from the assembly is to be selected and the corresponding Z value adjusted as required. The display of all objects referred to by this Z value is changed accordingly.
- The visibility of UDOs/UDAs (standard components, production methods etc.), may be controlled by defining Z values as input parameters via the object *length*. The input parameters may be defined as optional.



- The priority of visibility is rearranged by *Redraw*.
- Drawing models on the screen begins with the object of the lowest priority and ends with that of the highest priority. Since this is a time-consuming operation, it is possible to disable the suppression facility (see chapter on *Display and suppression of object types*). The action *Redraw* then runs unsorted.
- Lines with automatic line style are displayed as either a full line (visible) or in the specified second line style (hidden) according to line visibility (see section „[Line Style](#)“ on page 14-10).
- Since dimensions, symbols and texts cannot have Z values, their drawing sequence is purely random, i.e. texts may overlap other texts.

Visualization Techniques

14.6 Visualization of Layers

14.6.1 Symbolic Visualization of Models

EUKLID Design contains a tool for the visualization of your product model:

- It can be used to display parts symbolically without any definition of their geometry in advance.
- These graphs can be plotted via Postscript and are used as an abstract means of structuring in the concept phase.
- Components, assemblies etc. can be identified in the finished design by either their symbolic or geometric representation.

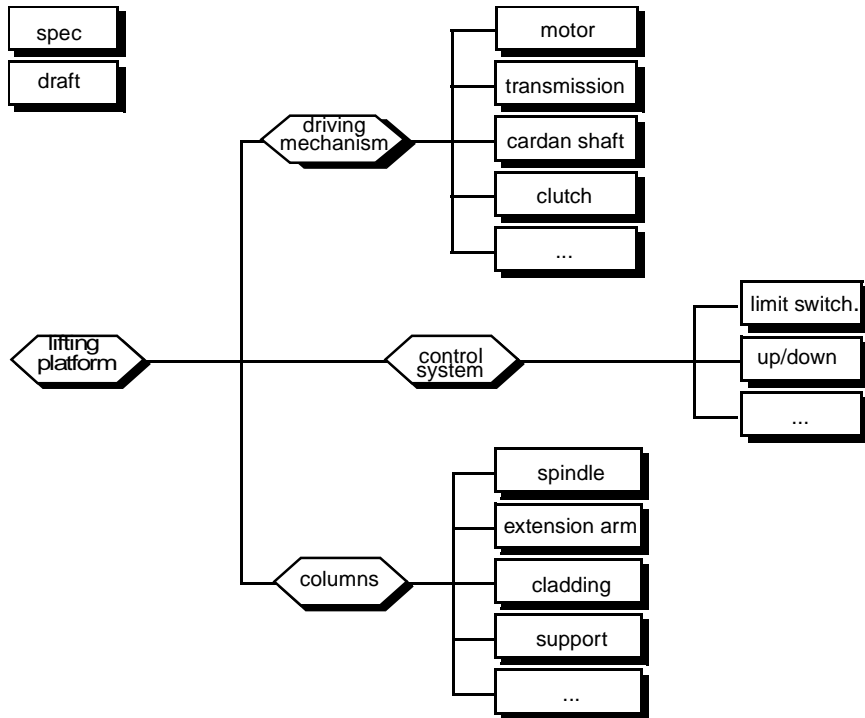
A direct relationship exists between these forms of visualization. The layer structogram is thus just another “view” of the product model. The structuring remains the responsibility of the user and is not determined by the program.

EUKLID Design provides the window type *Structure* for improved visualization and processing of the structures.

This shows all UDOs and layers within their hierarchy. You may identify them directly.

Example

The following diagram shows the structure for the lifting platform:



Visualization Techniques

14.6.2 Creating and Changing a Structure Model

The following example shows how to

- visualize your product model
- modify the visualization of your product model
- change the product model you have created.

As an example, the structure of a lifting platform as shown at the beginning of this chapter will be created.



If not described otherwise the actions of the action group *Layer visualization* of the 2nd menu are used.

14.6.2.1 Creating a Structure

☞ Open a window of type *Structure*,
e.g. in this way: Window-Menu → *New window* → *2D-structure-Window*.

☞ Create a new layer with *Create Layer*
Enter the name *lifting platform* of the topmost structure element in the text field.
The structure element *lifting platform* will be shown in the structure window.

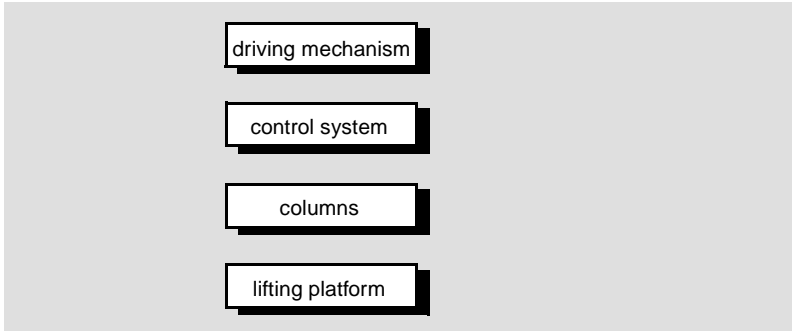
When adding the further substructures attached to the *lifting platform* structure element, you can proceed in two different ways.

- A. Create all dependent structure elements and insert them into the hierarchy.
- B. Set the structure element *active* under which further substructures are to be created.

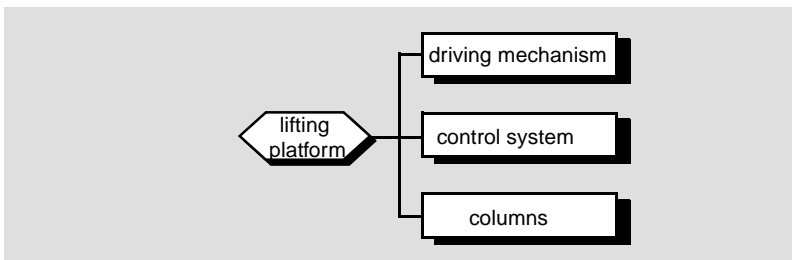
Both methods are shown on the following pages.

A. Create structure element and insert into hierarchy

- ☞ Create all the other elements of the *lifting platform* structure model, such as *columns*, *control mechanism*, *driving mechanism*, as described above. The elements are displayed one above the other in the window.



- ☞ To place the elements in their correct hierarchical structure, please use the action *Create structure*.
- ☞ Click on the *driving mechanism* structure element that is to be inserted under *lifting platform*.
- ☞ Then click on the *lifting platform* element.
The *drive mechanism* component is now inserted below *lifting platform*.
The *lifting platform* structure element is now shown graphically as a hexagon. This indicates that the element has a substructure.
- ☞ Add the other elements under the *lifting platform* structure element.
The elements are now arranged in the structure window as follows.



Visualization Techniques

B. Set the structure element active and create further substructures

☞ Set the *lifting platform* structure element *active* by placing the cursor on the *lifting platform* structure element and click *Edit* in the popup menu of the right mouse button.

The dialog box *Properties* is opened (see section „[Change a Structure Element](#)“ on [page 14-24](#)).

☞ Click the *active* button in the dialog box.

The *lifting platform* structure element is now active. This is indicated by giving the structure element a colored background.

☞ Now create the *drive mechanism*, *control mechanism* and *columns* structure elements.

These structure elements are automatically inserted under the *lifting platform* structure element as they are created.

Now create the remaining structure elements until the visualization corresponds to the example at the beginning of this chapter.

14.6.2.2 Modifying the Structure Visualization

When designing complex product models, it may be that not all structure elements can be displayed in the structure window.

The following actions allow the view of the structure elements to be changed, or individual structure elements to be hidden:

- All **zoom** and **pan** actions are also available in the structure window. Some of these actions can be invoked via the popup menu of the right mouse button.
- The action *Center structure element in window* allows the structogram to be **centered** at the click position.
- The action *Display substructures* allows all the subordinate **substructures** to be hidden.

Do this by clicking on a hexagonal structure element.

By clicking again on the selected structure node, the complete substructure will be shown again.

- If you want to show only **one level** of the substructure again, please use the action *Add substructure (one level)* and click on the desired structure node.
- The action *Structure visualization horizontally/vertically* allows the visualization of the structure to be toggled between **vertical** and **horizontal**.

- To show a **detail** of the structure in another window:

☞ Create another window of type *Structure*.

Position the window so that the window first created (main window) is not hidden. The same structogram as is in the main window is now also displayed in the new window (subwindow).

☞ Click on the action *Show structure detail in another window*.

☞ Drag the selection rectangle (press and hold left mouse button) in the main window over the portion of the structure view that you wish to show in the subwindow and release the mouse button.

☞ Then click in the subwindow with the left mouse button.

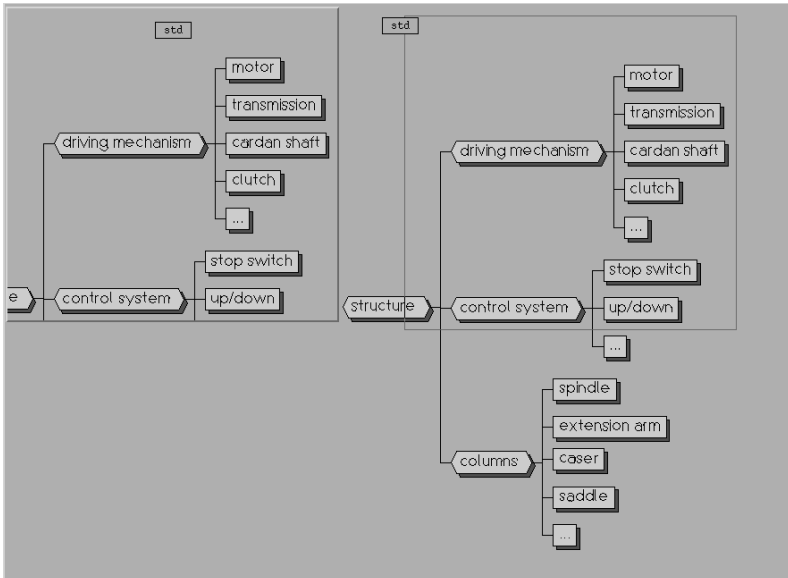
The selected portion of the structure will now be displayed in the subwindow.



The view in the main window can be changed independently of the subwindow and vice versa.

If, for example, you hide substructures in the main window, they will still be visible in the subwindow, as shown in the following diagram.

The portion of the main window that is displayed in the subwindow has a border around it.



Visualization Techniques

14.6.2.3 Delete a Structure Element

A structure element can be deleted as follows:

- ➡ Change to the Delete-Mode and click on the *User* icon in the first column.
- ➡ Click on the structure element that you want to delete.
The selected structure element is deleted.



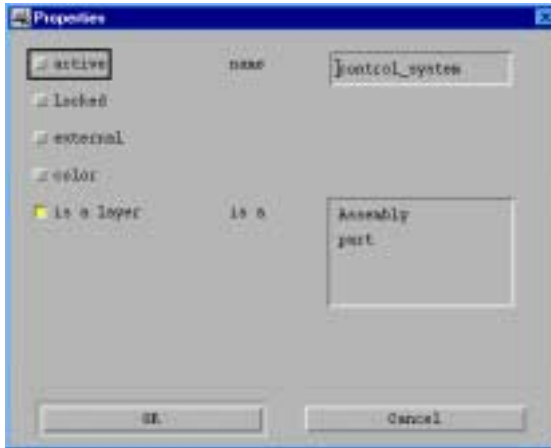
If you delete a structure element that has a substructure, this will be deleted as well.

14.6.2.4 Change a Structure Element

Any structure element can be changed as follows:

- ➡ Move the cursor close to the desired structure element and choose *Edit* in the popup menu of the right mouse button.

A dialog box appears in which the following settings can be made for the selected object:



<i>Button</i>	<i>Meaning</i>
active	Set structure node (=Layer) active / inactive
locked	The selected node cannot be changed or deleted
extern	Is the Layer external or do you want to make it external? (See section „ External Layers “ on page 5-27)
color	Choose a color, see „ Colors of Objects “ on page 14-12
is a layer	Convert to UDO, using the <i>is a</i> -Relation (see below)
name	Name of the structure element
is a	Hierarchy definition, e.g. „Assembly“, see section „ Adopting Object Type Definitions for User-Defined Objects “ on page 13-22.

14.6.2.5 Graphical Display of Structure Elements

The visualization a structure element depends on its properties:

<i>Representation</i>	<i>Meaning</i>
Color of node	Color of layer / UDO
Element on colored back-ground	Layer / structure element is active
{ }	External layer
□	Layer cannot be changed or deleted

Visualization Techniques

14.6.2.6 Allocate Objects to Individual Structure Elements

When designing models, objects can be allocated to individual structure elements.

Allocate directly

Proceed as follows:

- ☞ Set the structure element to which you wish to allocate objects *active*.
- ☞ Create the objects.

The objects are allocated to the selected layer / structure element.

If you have assigned a particular color to the structure element, the allocated objects also get this color.

Allocate retrospectively

Objects can also be allocated to structure elements retrospectively.

- ☞ Select Layer-Menu → *Layer-Status*.
Select in the following dialog box the layer to which you wish to allocate objects.
- ☞ Click on the button *Move Objects*.
- ☞ In the drawing area, select the objects that you want to move to another layer, for example using the selection rectangle.
- ☞ Confirm the selection with *OK*.

The objects are now allocated to the selected layer or structure element.

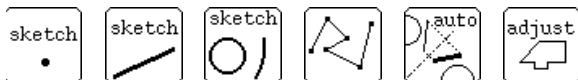
If you have assigned a particular color to the structure element, the allocated objects also get this color.

15 Construction Aids

The construction aids section describes actions simplifying the work with **EUKLID Design** and hence enabling the construction of complex models or components by a limited number of actions. The user advantage is an increased comfort by a simplified operation of the system and, in particular, time savings, since the number of operations and mouse clicks required to generate a model can be essentially reduced.

The following actions are summarized under the term "Construction aids":

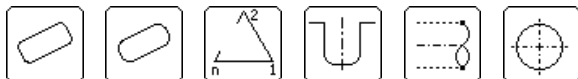
- Sketcher



- Generation of contours and planes



- Macros



- Drawing frames



Construction Aids

15.1 Sketcher

The section on the Sketcher of **EUKLID Design** includes the basic application features and properties of this convenient construction aid in the first part and the practical application based on a construction example in the second part.

15.1.1 Introduction

The principle procedure for the use of the Sketcher includes the following steps:

1. Activate the desired action group in the first menu column in order to generate points, lines, circles, contours or planes (or add the action *automatic mirroring*, if required).
 2. Execute the action by clicking the mouse in the drawing range
- to create elementary objects related to an already existing object located in the search range of the cursor (e.g. construction of a point with x and y distance to another point, construction of lines and circles – see following application example);
 - to create contours and planes with more important relations to other objects of the contour or plane than to the objects in the window.
 - to create a symmetric profile or
 - to adjust a sketched profile (*adjust* action).

The advantage and attractivity of the Sketcher in the practical construction is that the "system intelligence" is used to construct complex structures; i.e. the system offers the actions suitable for the respective processing step. In most cases, these are sufficient to create the model to be sketched. Hence you can create a complex construction in a few working steps by intuition. Designer and system interactively cooperate for the creation: the system offers the required construction relationships and construction steps, which are merely confirmed by the designer. Hence the user has more time for the actual construction and design.

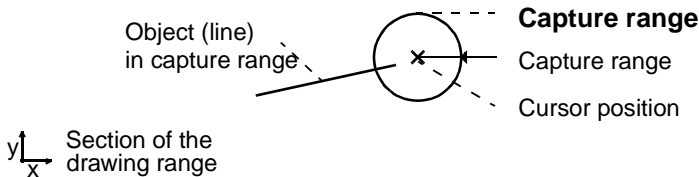
15.1.2 Application Features

As far as possible, the Sketcher provides a relation between an object to be created and existing objects in the search range with a minimum of mouse clicks and a maximum of construction intelligence.

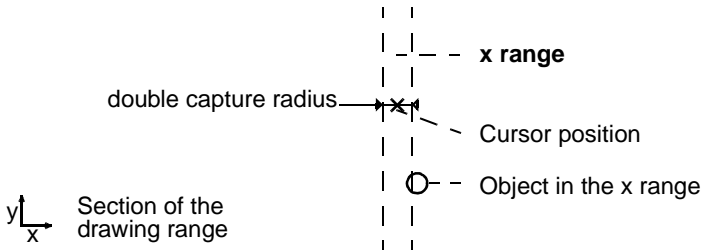
In the practical application of the action, the "search range", i.e. the range around the cursor during the selection of objects in the drawing range is of decisive importance:

Three search ranges are distinguished:

Capture range: The capture range is a circular range around the cursor position with a capture radius corresponding to a number of pixels. An object is located within the capture range, if the distance between the object and the cursor position is smaller than the capture radius in the drawing range.

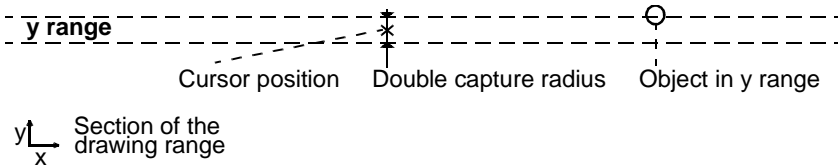


x range: The x range is a vertical range at the left and at the right of the cursor with a total width of twice the capture radius. Vertical means parallel to the y axis of the coordinate system at which the object is located. An object is located in the x range as soon as the horizontal distance of this object to the cursor position is smaller than the capture radius.



Construction Aids

y range: The *y range* is a horizontal range above and below the cursor, the total width of which corresponds to twice the capture radius. Horizontal means parallel to the *x* axis of the coordinate system, in which the object is located. An object is located in the *y range* as soon as the vertical distance of this object to the cursor position is smaller than the capture radius.



Objects in the capture range

The Sketcher first searches for objects in the capture range. If an object is located in the capture range, e.g. a circle, it is marked and the symbols next to the cursor are masked. The circle can be used to create the following action, e.g. tangent at circle.

No objects in the capture range

The Sketcher first searches for objects in the capture range. If no object is located in the capture range, one of the following possibilities applies depending on the action:

- The position itself is used for the Sketcher, e.g. to create a line at a certain distance parallel to another line.
- A point is created at this position based on an object in the *x* or *y* range.

The point created is used as parameter for the Sketcher, e.g. to create a line through a point parallel to another line.

Priority of objects in the search range

Objects in the search ranges are sorted by the following priority independent on their distances to the cursor within the search range:

<i>Object</i>	<i>Priority</i>
Point	High
Line, circle (arc)	Low

If several objects with a maximum priority are located in the search range, the closest is used to create the action unless the objects can be connected to create a new point (e.g. intersection).

Virtual points

If objects are located in the search range, virtual points are used: these are imaginary points which can be created at a proper position on objects. On a straight line, its center is e.g. offered as possible point. Those points do not exist in the data structure. As soon as reference is made to a virtual point while sketching, this point is created as point on the respective object in the data structure.

<i>Object</i>	<i>Virtual points</i>
Point	- - -
Line	Start point, center point, end point, hot point
Circle	Every 45°
Arc	Start point, center point, end point

Please refer to „[Properties of the Sketching Action](#)“ on page 15-29 in which you can define all system defaults.

Construction Aids

15.1.3 Sketching an Elementary Object

This action creates an elementary object in relation to visible objects in the window. There are the following possibilities:

- Sketching an object with a position:
Sketching a point (`point_sketch`)
- Sketching an object with two positions:
Sketching a line (`line_sketch`)
Sketching a circle/arc (`circle_sketch`)

Sketching an Object with one Position

Sketching a point














This action creates a point at the cursor position by an action resulting from the closest other objects.

Procedure:

1. Click on the action group of points in the Create mode. The Sketcher is active as uppermost action.
2. Choose the action to be used to create the object *point* depending on the closest objects by moving the mouse in the drawing area (please note the symbols shown next to the cursor).
Possible actions are listed as icons in the fourth column. The action which would be activated by a click of the left mouse button at the cursor position in the drawing area is shown as symbol next to the cursor.
3. Confirm the sketched point by a mouse click. It is created with the selected action as result of the Sketcher.



Possible actions depending on the closest objects are listed in the following table.

<i>Objects in the capture range</i>	<i>Note Action</i>	<i>Sketched action/object</i>			
2 or more objects		Intersection of the 2 closest objects (circle/line)			
1 object 		Point on object (circle/line)			
No object A point is created based on the object in the x and/or y range		Allocation to the action sketched			
same object in x and y range, possibly under the angle		Object in x/y range	Priority	Virtual points	Auxiliary relation
different objects in x and y ranges		.	high	----	
merely 1 object in the x range			low	Start, center, end point	
merely 1 object in the y range			low	every 45°	
no object in the x and y ranges			low	Start, center, end point	

Construction Aids

Sketching an Object with two Positions

This action creates elementary objects (line, circle, arc) with two click positions in relation to objects shown or selected in the window.

The system needs two separate positions having a minimum distance of a certain number of pixels. Otherwise, no object is created.

There are two possibilities:

- Sketching a line (line_sketch)
- Sketching a circle/arc (circle_sketch)

Sketching a Line



This action creates a line in relation to objects shown or selected in the window.

Procedure:

1. Click on the action group of lines in the Create mode. The Sketcher is active.
2. Move the mouse in the drawing area. The system dynamically captures the object being closest to the cursor position in the search range. It is marked. If there is no object in the search range, a point is created by the Sketcher for *points* by using one of the actions shown in the 4th column.
3. Select the first object for the Sketcher by a mouse click.




















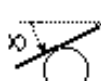










The symbol of the action just created is shown next to the cursor. A temporary line, which would be created, is dynamically shown.

4. Select the second object of the Sketcher by a mouse click. A *line* object is created in relation to the objects of the first mouse click (position 1) and of the second mouse click (position 2).



The following table shows a summary of possible actions for the creation of a line:

Position 1 Position 2		Object in the capture range					
							
Object in capture range		2 points + 			1 point + 		
		1 point + 					
		1 point + 					
		1 point + 					
							

Examples

If you selected a line as object of the first mouse click and if there is no other object in the capture range, a parallel line is created by the second mouse click.

If you selected a line and a circle as objects of the action, a tangent is created with the slope of the line at the circle (parallel through the tangent point). The length of a tangent results from the length of the line. The point at which you click on the circle is decisive for the creation of the tangent, since several tangents are possible at the circle.



The functions of the Sketcher cannot be used across window boundaries.

Construction Aids

Sketching a Circle/Arc



This action creates a circle or arc in relation to objects shown or selected in the window.

Procedure

1. Click on the action group of circles in the create mode. The Sketcher is active.
2. Move the mouse in the drawing area. The system dynamically captures the object being closest to the cursor position in the search range. It is marked. If there is no object in the search range, a point is created by the Sketcher for *points* by using one of the actions shown in the 4th column.

3. Select the first object for the Sketcher by a mouse click.




























The symbol of the action just created is shown next to the cursor. A temporary circle (arc), which would be created, is dynamically shown.

4. Select the second object of the Sketcher by a mouse click. A *circle (arc)* object is created in relation to the objects of the first mouse click (position 1) and of the second mouse click (position 2).



The following table shows a summary of possible actions for the creation of a circle/arc:

<div>Position 1</div> <div>Position 2</div>		Object in the capture range				
						
Object in capture range		1 point + 		Tangent 	Tangent 	
		1 point + 		Tangent 	Tangent 	
		Tangent 	Tangent 			
		Tangent 	Tangent 			
						

Examples

If the object of the first mouse click is a point and if there is no other object in the capture range, a circle is created with a radius around the selected point.

If you selected a line as object of the first mouse click and if there is another line within the search range intersecting the first line, a rounding arc appears as symbol next to the cursor. When you press the mouse button, the rounding circle is created and the two lines are trimmed.

Construction Aids

If you selected a full circle as object of the first mouse click and a point as object of the second mouse click (or if there is no object in the search range), a circular arc is created touching a point on the circle with a point as starting point. The point at which you select the circle is decisive for the creation of the arc and the tangential angle at the starting point.



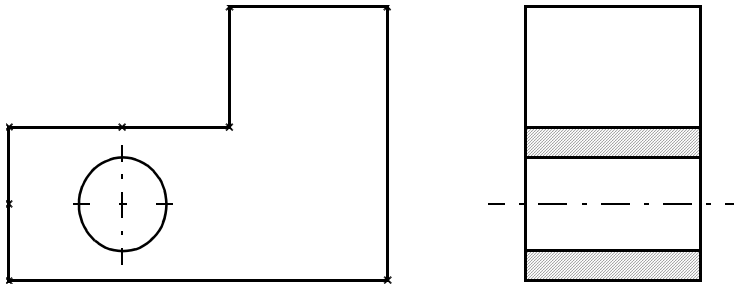
The functions of the Sketcher cannot be used across window boundaries.

15.2 Designing with the Sketcher

Complex models and components can often be designed using a limited number of interrelated actions. These are contained in the Sketcher of **EUKLID Design** to let you design intuitively: following certain design steps, the Sketcher automatically suggests the next step in the logical sequence of working. This frees you from having to switch constantly back and forth by the Sketcher between various actions, so that you can better concentrate on your design.

The Sketcher also allows you to create geometrically interdependent units, such as two views of the same object, so that when one parameter is changed (e.g. the *length of a line* or a *bore diameter*), the depiction is altered in both views.

The following examples show just how easy and convenient Sketcher is to operate.



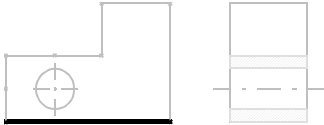
Set the line weight depiction in the system settings to *off* (default). Screen depiction on a refresh will then be somewhat quicker during designing.

Construction Aids

15.2.1 Constructing a Closed Sequence of Line Segments

Begin with a new, empty model. In the first step, we shall sketch a closed line sequence comprising a plan view. Proceed as follows:

Constructing the First Line



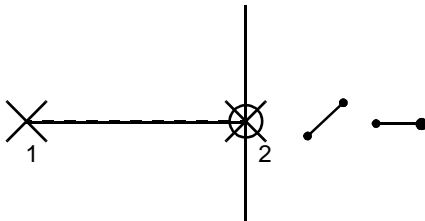
1. Click onto the *line_sketch* icon.



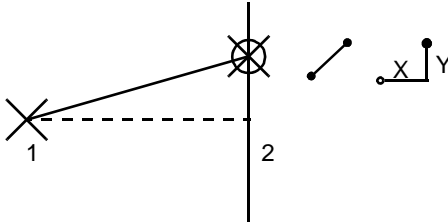
2. Inside the drawing area, click the left mouse key onto the position for one end of the line (*point relative to the original coordinate system*) and move the mouse horizontally to the right.



The Sketcher displays the symbols *line between two points* and *point with same Y value*.



If you move the crosshair too far off of the horizontal axis, then the symbol *point relative to a point* appears.

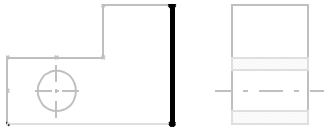


3. Move the crosshair horizontally to the right, then click the left mouse key again at the end of the horizontal line. This sketches the baseline of the model.



The x-value of the end-point is displayed in the status field in the lower right corner of the screen.

Constructing a Vertical Second Line



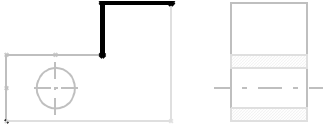
4. Without moving the crosshair, click the left mouse key again at the last point: as soon as you bring the crosshair near this point, it is marked. Then click the left mouse key. The *vertical line* symbol now appears next to the crosshair.



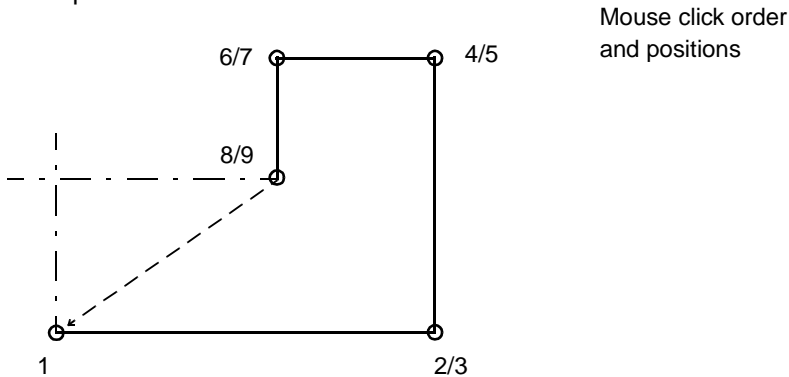
5. Now move the crosshair upwards in the drawing area and click the left mouse key at the new endpoint of the vertical line.

Construction Aids

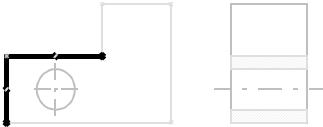
Constructing the Third and Fourth Lines



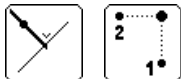
6. Proceed as described in 4 and 5 above. Set the click positions as shown in the example.



Constructing the Fifth and Sixth Lines



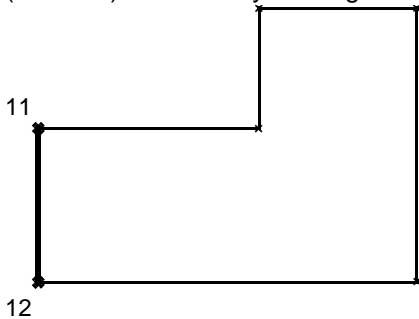
7. Now construct the final horizontal line such that its final endpoint is located exactly on top of the first point: click a second time onto the last created point 8 (as shown above), then move the crosshair horizontally to the left. Depending upon system defaults, the symbols *vertical line* and *point relative between two points* will appear. Then click the left mouse key (10th click position) to place the endpoint of the horizontal line exactly on top of the first point.



If these symbols do **not** appear, proceed as follows: pull the cursor to the first click position (the first point of the drawing) until this position appears highlighted. Then move the crosshairs upwards. At first, the symbol *point with same x value* appears

until you have reached the y value of Point 5. Then the symbol changes to the *vertical line* symbol with the second symbol *point relative to two points*. Then click onto the left mouse key (10th mouse click), and the endpoint of the horizontal line is placed exactly on top of the first point.

8. Close the line sequence by clicking again on the endpoint of the horizontal line (Point 11) and then by clicking on the point 12.



Closed line
sequence

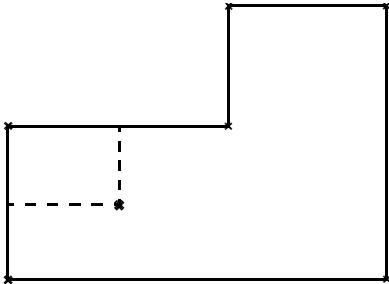
Construction Aids

15.2.2 Constructing the Bore Hole

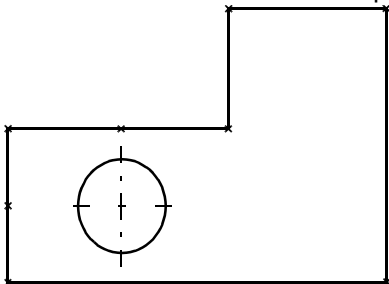
The bore hole is to be placed exactly in the center of the left portion of the part. It is drawn using the *circle_sketch* icon.



1. Center the crosshair between the last two straight lines until the symbol *point relative between two points* appears, then click the left mouse key. The center of the circle is then placed exactly in the middle, and the system marks two additional points on the straight lines.



2. If this symbol fails to appear, then move the crosshair toward one line, then the other, then place them directly between the two lines until the symbol *point relative between two points* appears. Click the left mouse key, and the center of the circle is established.
3. The Sketcher activates the symbol *radius* next to the cursor and now you can move the crosshair to define the radius of the circle. Click the left mouse key at the desired radius. The bore hole appears.



Bore hole centered
between two lines

15.2.3 Constructing the Side View

In the following section we shall construct the corresponding side view, which is dependent upon the plan view just drawn.

Constructing the Parallel Lines



1. Click onto the *line sketch* icon.
2. With the cursor inside the drawing area, click onto the vertical line of the plan view (between Points 3 and 4). A temporary line appears next to the position of the cursor and the *parallel line* symbol appears next to the cursor.



Move the temporary line to the right with the cursor and click the left mouse key when the final position is reached. A vertical line appears parallel to the first vertical line.

3. Now click onto this new parallel vertical line and move this temporary line toward the right until its position corresponds to the desired depth of the part.

Constructing the Connecting Lines

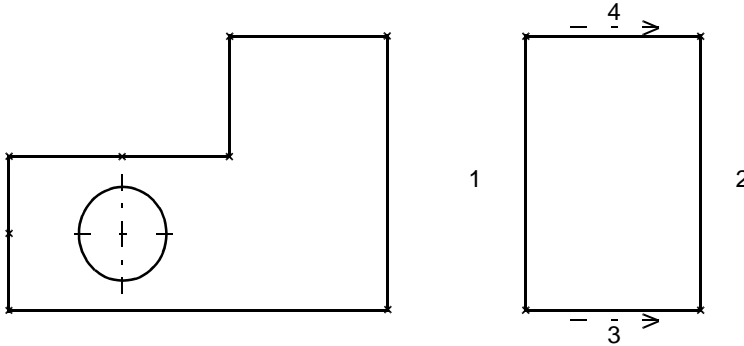


4. Connect the open ends of the parallel lines in the side view. Click on the end points of the parallel lines, and the symbol for a *line between two points* appears.



Construction Aids

Your drawing should now resemble that shown below:



Constructing the Offset Edge



The offset edge must depend upon the plan view.

5. Click onto the baseline 3 of the side view – the *parallel line* symbol appears – and move the temporary line upwards.

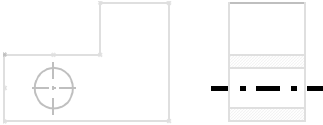


Move the crosshair to Point 8/9 of the plan view until this point appears marked – the symbol *parallel line through point* appears – and click with the left mouse key.



This creates the edge of the offset in the side view. The offset edge is dependent upon the length of Line 4, i.e. if the line is changed, the offset edge will automatically change correspondingly.

Constructing the Bore Centerline



6. Click onto the *line type* icon and select *centerline with overhang*.



7. Click onto the baseline (3) in the side view. The *parallel line* symbol reappears.



8. Move the temporary line with the mouse upwards and specify the relation to the center of the circle in the plan view: place the crosshair on the center of the circle to mark it, then press the left mouse key.



This makes the position of the centerline in the side view dependent upon the centerpoint of the circle in the plan view.

Constructing the Bore Edges



The bore edges also must depend upon the plan view:

9. Click onto the *line type* icon and select *line with endings*.
10. Click onto the *bore centerline* in the side view, and move it vertically by any desired amount; press the left mouse key to confirm.
11. Enter the Edit mode by clicking the right mouse key and selecting *parameter input* from the pulldown menu. The action *line parallel length* appears in the second column with the corresponding parameters in the third column.

The system creates a line using the displacement value from step 10. Nevertheless you wanted to create a dependency on the radius of the circle. Therefore the system offers the Edit mode with which you can proceed as described.

Construction Aids

12. The *length* icon is active in the third column.



13. Click onto the *length radius* icon in the fourth column.



14. Inside the drawing area, click onto the circle in the plan view. This designates the circle radius as the distance from the parallel line (upper edge of bore) to the dashed centerline.

15. Enter the Create mode.

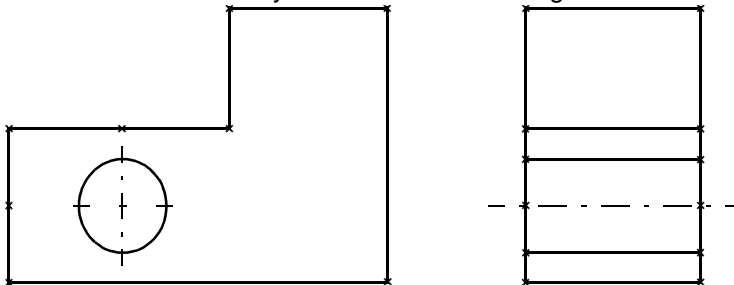
16. Click once more onto the line just created (upper edge of bore). The parallel line symbol reappears next to the cursor.



17. Move the crosshair to the bore centerline. The *line_mirrorline* symbol appears next to the cursor and a temporary line below the centerline.



Click the left mouse key to create the lower edge of the bore by mirroring.



15.2.4 Shading a Cross Section

In the following section we shall construct a section view from the bore side view and shade the cross section.

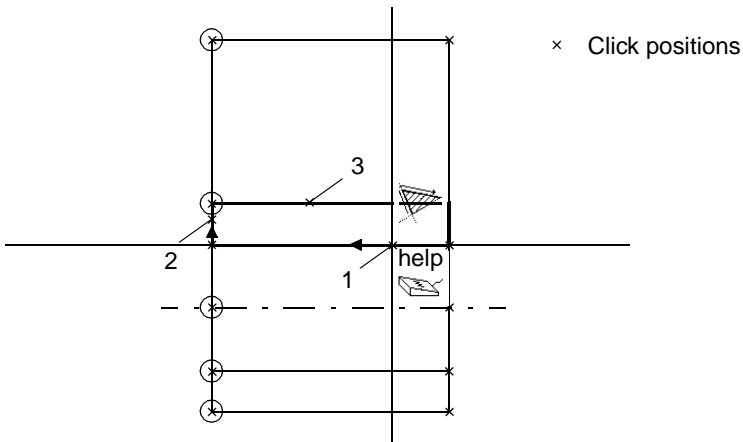


15

1. In the Create mode, click onto the *plane* icon to activate automatic surface creation in the second menu column.



2. Define the plane to be shaded by selecting an object of the surface to be shaded, e.g. one edge of the bore in the drawing area (click position 1).



The *plane* and *help* icons appear near the crosshair. The system searches for intersection points as close as possible to the starting click position (1) and marks these

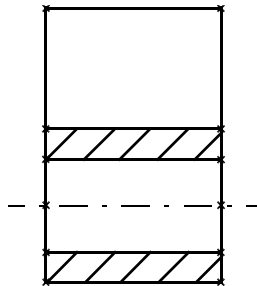
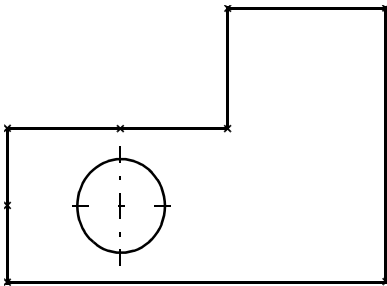
Construction Aids

with circles. The system also attempts to mark a line sequence corresponding to the surface area to be shaded.

3. Move the crosshair along the contour to be shaded. Select the individual objects (2, 3) that enclose the area by clicking onto them with the left mouse key.
4. When you have marked enough objects, both symbols near the crosshair disappear. Now you have created a closed plane.
5. Follow steps 2 through 4 above to shade the lower portion of the bore cross section. Finally click onto the lower *arrow* icon of the list parameter.



The shading of the cross section appears with the predefined values for line distance and -angle.



15.3 Sketching of Contours and Planes

These actions create contours and planes from the series of successively created, elementary objects (point, line, circle, arc). The principle procedure for the creation of contours and planes is identical; it will be explained by the example of contour creation.

The *sketching* action relates basic, elementary objects unless there are points of the surrounding construction at a closer position within the search range. Based on the cursor position, it detects the desired continuation of the contour: **tangential** (in case of arcs) or the same direction (in case of lines) or **vertical** (in case of arcs vertical to the imaginary tangent at the start and end points). In these cases, corresponding symbols are shown next to the cursor.

In this connection, the capture angle is of importance:

This is a preset angle (*parameter of action*) between the normal or tangent to the previous project and an imaginary, axis-parallel line through the previous point.

If the cursor is located within the capture angle, the continuation is considered as tangent or normal to the previous position.

A new, elementary object will be created in relation to the previous or first object as far as possible. If the *sketching* action is not able to create a relation to these objects, it tries to create a relation to a visible object in the window:

1. The point is created in relation to the previous or first object:
 - The cursor position is vertical to the previous object:
 - If the direction of the previous object is horizontal, a point is created vertical to the previous point.
 - If the direction of the previous object is vertical, a point is created horizontal to the previous point.
 - If the direction of the previous object is tangential, a point is created polar to the previous point at an angle vertical to the previous object.
 - The cursor position is tangential to the previous object:
 - If the direction of the previous object is horizontal, a point is created horizontal to the previous point.
 - If the direction of the previous object is vertical, a point is created vertical to the previous point.

Construction Aids

- If the direction of the previous object is tangential, a point is created polar to the previous point at an angle tangential to the previous object.
- 2. The system is not able to create a relation to the previous or first object respectively:
 - If there is at least one object in the search range, the procedure is similar to that for the sketching of points.
 - If there is no point in the search range, a point is created.

Points created are connected to the previously created point by the following actions:

- by a *line through two points*
- by an *arc through two points and an tangent angle*
- by a *circle through three points*

A *line through two points* is the default. On pressing of the center mouse button, the *arc through two points and tangent angle* action is defined by a tangent angle to the previously defined object. If you press the center mouse button again, this position is selected as second point and the action *circle through three points* is set.

15.3.1 Sketching a Contour – contour_sketch

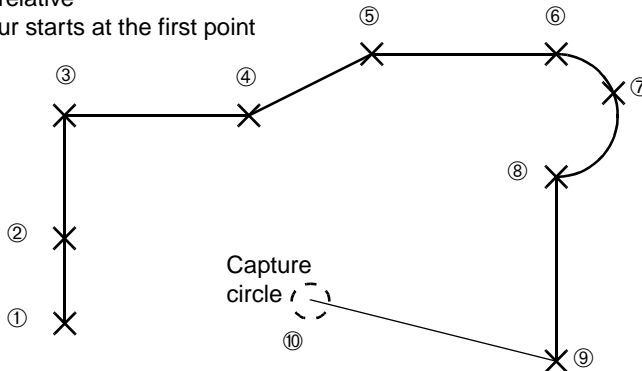


By means of this action used as tool for the creation of simple contours, you can create open or closed contours from lines and arcs by continuously sketching. The creation of the contour is facilitated by the automatic indication of a "rubber band" to the next click position.

15

Defaults:

- point relative
- contour starts at the first point



- ①: First sketched point = reference point for all points with relative coordinates. At the same time, used as start point of the contour in this example.
- ②: Point with vertical distance to ①
- ③: Point with vertical distance to ②
- ④: Point with horizontal distance to ③
- ⑤-⑨: Points with relative coordinates referred to ①. The circle point is to be sketched by the center mouse button (2x).
- ⑩: Capture circle (= current clicking position); in order to close the contour, click on the position ① within the (invisible) capture circle.

Construction Aids

The creation of the contour is **completed**

- by clicking on the start point of the contour;
- by clicking on another icon in the menu bar in case of an open contour.

Procedure

1. Click on the *contour_sketch* icon (in the standard configuration delivered in the *contour* action group).



2. Define the type of contour points (point relative or polar) by the open dialog box and click on the OK button. *Point relative* is the default. The dialog box is closed.



3. Enter the capture angle by the keyboard or the fourth menu column, if required.



4. Specify whether the contour starts at the reference point (default), if required.



5. Click on the desired position in the drawing range. A start point is created.



6. Create another point by pressing the left mouse button at another position. This point is connected by the action *line through two points* to the previously created point.
7. Create further points at desired positions by pressing the left mouse button.
8. Reselect the start point in order to complete the action and to create the final contour.

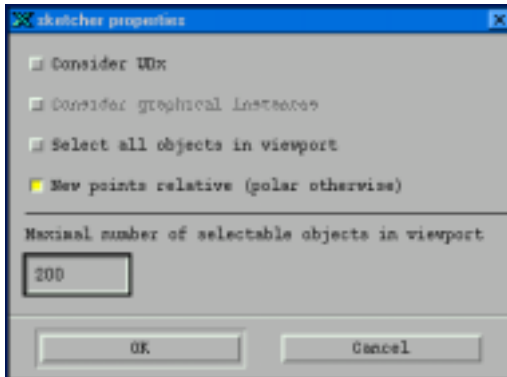
15.3.2 Notes on the Application of the *Sketching* Action

- Here you can automatically create an action symmetric to the action selected. Enable the automirror mode prior to selecting the *sketching* action.
- You can abort the *sketching* action by pressing the <ESC> key. Objects sketched during the action (temporary graphics) will not be created.

15

15.3.3 Properties of the *Sketching* Action

The defaults of properties can be defined by the *set_sketchprops* action (you can find this action in the default configuration under the *preference* action group in menu sheet 2).



In the dialog box, you can select the following settings by clicking with the mouse – active settings are marked with a black box:

- *Consider UDx*
Objects within user-defined objects (UDOs) are searched, too.
Default: on
- *Consider graphical instance*
Objects are searched in graphical instances.
Default: off (not possible in this version)

Construction Aids

- *Select all objects in viewport*

Default: off

On: The *sketching* action automatically searches all objects in the viewport.

Off: The *sketching* action searches in the window for objects located within the search range of the cursor in the drawing area.

- *New points relative (polar otherwise)*

Default: on

On: If possible, a point is marked relatively (x, y)

Off: If possible, a point is marked by polar coordinates (by distance and angle).

- *Maximum number of selectable objects in viewport*

Specifies the maximum number of objects to be searched in the window.

Default: 200

If this number is exceeded, the symbol appears next to the cursor (exclamation mark in triangle). You should reset the presentation or increase this number.



The higher the number is set, the more sluggish is the *sketching* action, since more objects have to be searched.

15.3.4 Sketching a Symmetric Profile – automirror



In addition to the creation of a simple geometric object (line, circle, ...), this action creates objects automatically mirrored at an axis selected (the *automirror* option does not only affect the sketching action, but also all other actions).



Compared to subsequent mirroring, this method has the advantage that you can see all results while sketching. You need not reselect the objects created and thus save one step.

Procedure

1. Click on the *automirror* action. (You can find this action in the standard configuration under the *preferences* action group in menu sheet 2)



2. Select *automirror* in the open dialog box.
3. Define the mirror axis by selected a line in the drawing area.



4. Change to the first menu sheet and continue with your construction. In addition to the sketched object, another object is created symmetric to the axis selected.

15.4 Automatic Creation of Contours and Planes

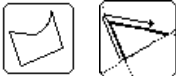
The automatic creation of contours and planes facilitates the generation of open or closed contours or planes in all cases with successive order objects. It is required that two successive objects have a common intersection. You can only create closed planes.



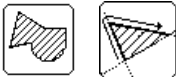
Prior to creating a contour, reduce the window section to include all objects required as border objects in the window.

The following two actions are available (see Online Help, *contour_tracing* and *plane_tracing*):

Contour



Plane



The following **parameters** are provided:

- z value
- fill color (plane)
- fill properties (plane)
- line mode of border objects
- window section
- possible contours (contour)
- type of contour (contour)
- border objects (list parameters)
- hatch line distance (plane)
- hatch line angle (plane)

15.4.1 General Notes on the Creation of Contours and Planes

You can assign a *z* value parameter to contours and planes in order to control the visibility of the plane or contour respectively (see chapter „[Suppression of Hidden Lines](#)“ on [page 14-14](#)).

In case of the automatic creation of planes, you can select the desired *fill color* by the fill color parameter. The input is made by color setting menus (see chapter See „[Colors of Objects](#)“ on [page 14-12](#)).

Further filling properties of planes (hatching and masking properties) are selected by the *plane filling* parameter in the *fill plane* box (see chapter See „[Colors of Objects](#)“ on [page 14-12](#)).

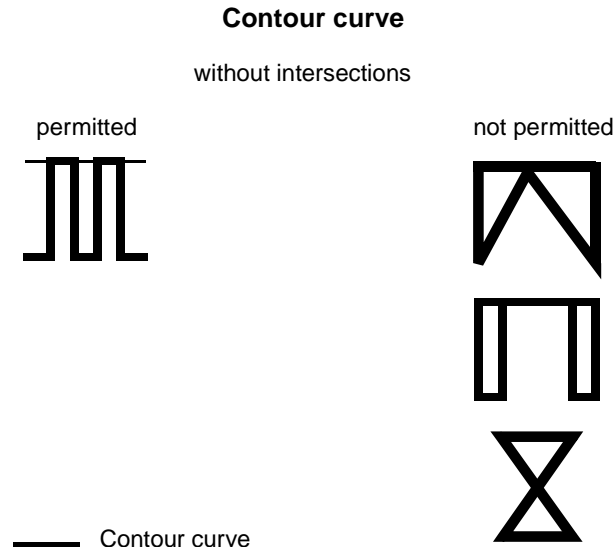
The *Line mode of border objects* parameter determines which border objects are used: merely solid lines (default) or also objects with other line modes.

By means of the *window section* parameter, you can define whether objects are considered within the visible window section only (default) or also outside. The default is to be preferred for performance reasons.

The *permitted contour curves* (contours only) is referred to overlappings and intersections: the default prevents those (same behavior as for the plane creation).

Construction Aids

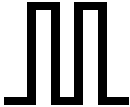
Example



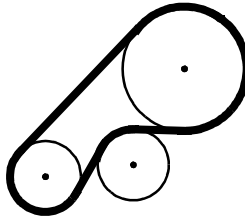
You can also permit overlappings and intersections by means of this parameter, i.e. contour curves not permitted in the default will be permitted (see examples in the right column of the figure above).

In case of the automatic creation of contours, you can select open or closed contours by means of the *contour type* parameter.

Open contour



Closed contour

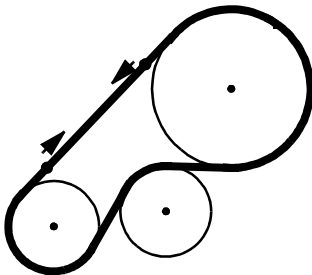


— Contour curve

By means of the list parameter, you can enter **border objects** of external contours and – if available – islands for the contour or plane to be created. Any individual island is to be considered in a contour. Islands must be completely within the external contour, i.e. they must not intersect the external contour.

In order to enter the **start point**, click on the first border object of the contour desired. The orientation of the contour is determined by the clicking position on the first border object:

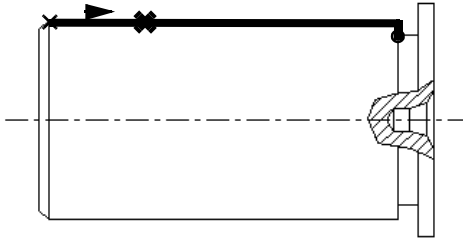
Contour orientation



● Clicking position

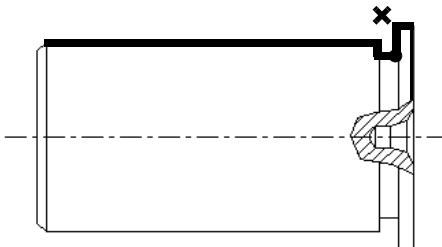
The contours border objects selected are marked up to the first branching point in the drawing area. A possible branching point is marked by a circle. The system needs a decision.

Construction Aids



- ✕ Clicking position on 1st border object
- ✕ Start point (calculated by system based on the clicking position)
- ➔ Orientation
- Identified border objects
- Branching point

The next **border objects after the branch** are highlighted depending on the current cursor position to the next branching point in the drawing area. When the cursor position is changed, the highlighting is dynamically adjusted. When the display matches the desired contour, click to the current cursor position (**auxiliary point**).



- Branching point
- ✕ Possible clicking position (auxiliary point)
- Identified border elements

In this way, the curve of the contour is determined by your inputs until the border objects of the contour are completely identified. Subsequently, the input of border objects is completed by clicking the lower *arrow* icon of the list parameter.

In case of the automatic creation of open contours, enter the desired **end point** of the contour.

In case of the automatic creation of planes, you can define the optional parameters *hatch line spacing* and *hatch line angle* as well as *fill properties* and *colors*.



In case of automatic plane and contour creation, you cannot create any contour if successive border objects have no point in common.

If there is a variety of branching possibilities at a branching point, solutions might be offered which do not lead to the target (closing the contour). Since the number of possible branches depends on the size of the window, the latter should be enlarged prior to creating planes or contours until the window rectangle just encloses the contour desired. The window should be chosen as selection rectangle.

Contours and planes may also consist of several partial contours or planes. These may also be created by the contour tracer. Enter arbitrarily many further partial contours/planes on completion of the first partial contour.

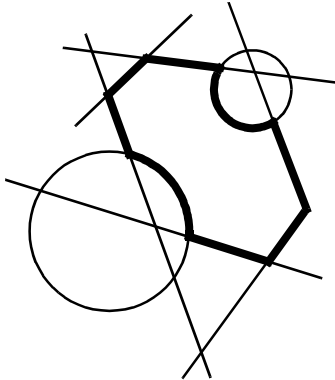


Contour tracing cannot be continued beyond window limits.

Construction Aids

15.4.2 Procedure

A closed contour shall be created from the arrangement of objects as shown in the figure.



Proceed as follows:

1. In the drawing area, construct a similar arrangement of 2 circles and 6 lines.
2. Activate the *contour* icon in the create mode. The *automatic contour creation* action is automatically activated.



3. Enter the following optional parameters or accept the default parameters and continue with step 4.



z value (optional parameter): can be entered by the text field or by constructing by means of the 4th menu column.



specifies whether solid lines or dotted lines can be used for contour tracing (4th column). Default: solid lines



Total view/screen view (optional parameter): default: screen view
Enter by selecting in the fourth menu column.

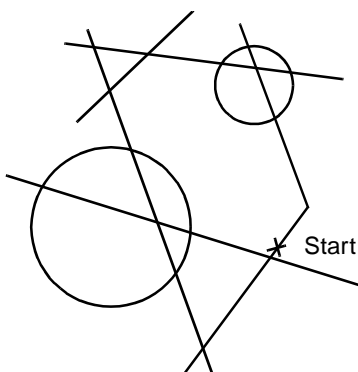


Permitted contour curves (optional parameter)
Default: overlappings and intersections are not permitted.
Enter by selecting in the fourth menu column.



Contour type: open/closed (optional parameter)
Enter by selecting in the fourth menu column.

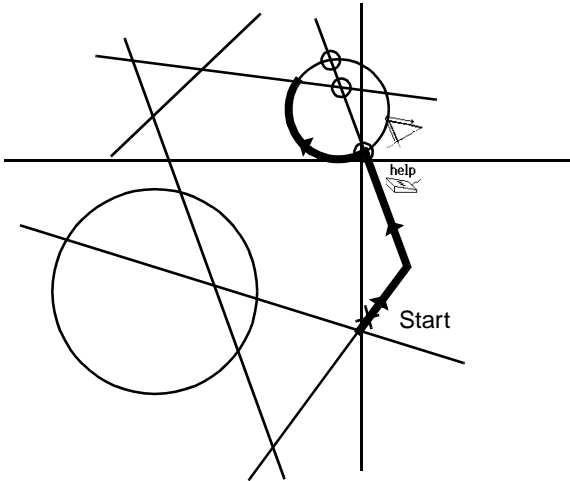
4. Start "tracing" the contour by marking the start point.



Now the system already calculates the possible course of the contour and highlights it.

Construction Aids

- At intersections with ambiguous course, auxiliary points appear with circular marks. Select border objects by the mouse in order to specify the course desired.



The orientation must be maintained when you click on objects! The selection positions on circles/arcs determine the selection of sectors. The selection positions of individual contour segments should be selected close to the latest branch in order to avoid errors. In addition to the cross-hair, the symbols of the action and of the current parameter are displayed as shown in the above example. If you shift the mouse, the recalculated course of the contour is drawn by the system.

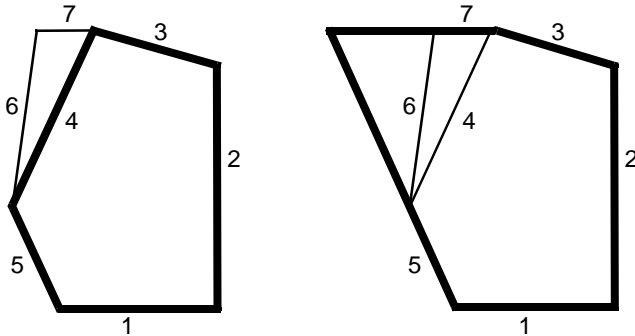
- Determine the course of the contour by selecting further objects until the contour is closed (the symbols disappear).
- Complete the creation of the contour or supplement the partial contour by further partial contours by clicking on the lower *arrow* icon of the list parameter.



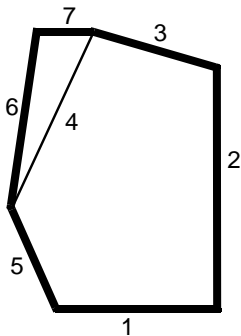
In edit mode, the orientation of the contour or plane is shown first. On selection of the list parameter, the orientation of the individual border objects is displayed. This may deviate from the orientation of the contour.

If you change the list of a closed contour in edit mode, this may cause results appearing slightly strange at first glance.

In the following example, the contour (left figure, bolt) shall be changed by replacing line 4 by lines 6 and 7 in the list. If you change in edit mode from line 4 to line 7, the system tries to close the contour automatically. Lines 5 and 7 remain unchanged, merely the contour gets another appearance not matching the original objects in the list at this moment.



If you add line 6 to the list – behind line 7 – the graphic appearance of the contour matches that of the original objects in the list.



Construction Aids

15.5 Macros

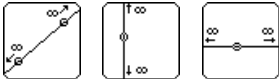
The following macros are available in the *Auxiliaries* action group:



- Indefinite lines (between two points, vertical, horizontal)
- Creating a rectangle with rounded corners
- Creating an oblong hole
- Creating a polygon
- Creating a groove
- Creating a shaft end
- Creating a reticule for circles

The various actions and their parameters are described in the *Functions* volume.

15.5.1 Indefinite Lines



Actions used to create indefinite auxiliary lines

15.5.2 Rectangle (with rounded corners)



Starting from a point, a rectangle (with rounded corners) is created. The angle to the x-axis determines the rectangle's position. A contour is generated as resulting object; all points, lines, and circles created are effects of the action.

15.5.3 Oblong Hole



Starting from a point, an oblong hole is generated. The angle to the x-axis determines the hole's position. A contour is generated as resulting object; all points, lines, and circles created are effects of the action.

15

15.5.4 Polygon



A polygon is created by means of an imaginary circle around a point. The angle to the x-axis determines the position of the first corner point. A contour is generated as resulting object; all points, lines, and circles created are effects of the action.

15.5.5 Groove



This action is used to create a groove in an already constructed component. An existing object (line, circle, ellipsis) is interrupted; e.g. a starting line is deleted and two new ones are created. All points, lines, and circles created are effects of the action.

Construction Aids

15.5.6 Shaft End



By means of the *shaft end* action, you can create a shaft end between two parallel lines by specifying two points.



The shaft end consists of three circular arcs and one ellipsoidal arc put together. These are effects of the action.

15.5.7 Reticule of a Circle



This command is used only if lines are actually required as objects. If the reticule shall merely be represented, it is sufficient to use the *circle* or *circular arc* action with the *reticule* property. Lines and points are effects of the action.

15.6 Drawing Frames

Drawing frames are supplied as UDAs (user defined actions) in the directory *#symbols*.

A drawing frame can be created as follows:

1. Click on the action group icon representing the desired drawing frame:



Creates a drawing frame in DINA0-A5 format



Creates a drawing frame in ANSI 11" format



Creates a drawing frame in ANSI 12" format

2. Then choose the desired action or drawing frame in the 2nd menu column and enter any necessary parameters.

A comprehensive description of the actions and parameters used in the creation of a drawing frame can be found in the *Functions* user manual.

Please use the UDA/UDO technique to modify the label field and the form of the drawing frame.

Construction Aids

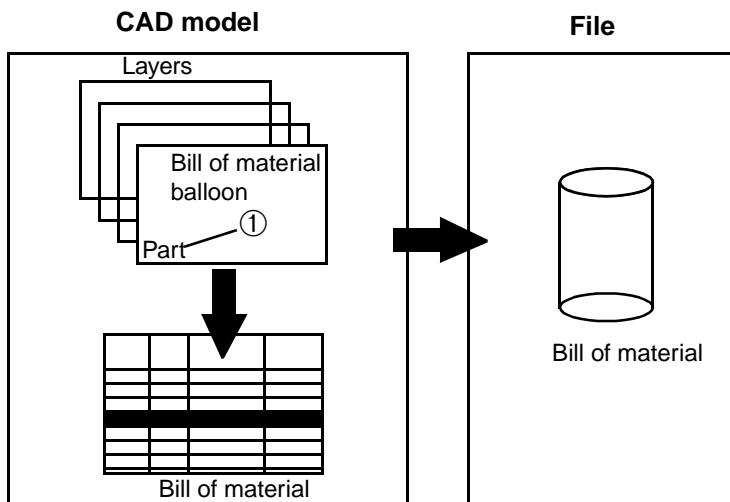
16 Bill of Material Generation

A bill of material is to be generated in two stages:

- Storage of bill of material data in the model
- Output of the bill of material to model or file

Bill of material data is assigned to a component or an assembly or function category in a higher level, and is stored in the model in a balloon in a generalized form. The balloon lies in the same layer as the component referred to.

The bill of material may be displayed graphically within the model or output to a file, and contains data from selected balloons lying within visible layers (see chapter [„Changing the Layer Status“ on page 5-22](#)).



EUKLID Design contains a standard bill of material composed of user objects, which may be generated by a user action in models. You can configure it according to your

Bill of Material Generation

demands on the graphical representation. You may also create own bills of material as user objects.

The bill of material is subordinated to the component data, i.e. is dependent on the bill of material data of the model, guaranteeing the consistency of the bill of material data with the bill of material in the model.

16.1 Storage of Bill of Material Data in the Model

Bill of material data is attached in the form of protected attributes to balloons in the model. You generate these balloons in *Create* mode with the object *symbol*. Two actions are available:



Bill of material balloon



Copied bill of material balloon

The action *bill of material balloon* is used to generate the balloon, where data concerning a given component is being stored in the model for the first time.

With the action *copied bill of material balloon*, data including the item and load sequence numbers is taken from an existing balloon, whereby a relation is set up between the original and copied balloons. This avoids the need for error-prone second input of data for identical components. The copied data may be overwritten or augmented by the user.

The following properties and parameters are available under both actions:

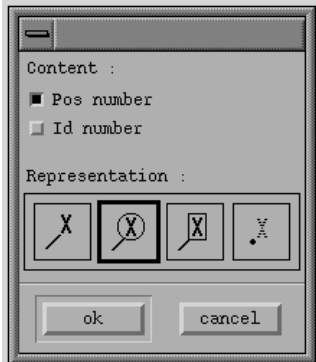
- ☐ Font
- ☐ Oblique angle
- ☐ Size-to-width ratio
- ☐ Text size
- ☐ Display characteristics of the balloon
- ☐ Load sequence number of component
- ☐ Item number of component
- ☐ Quantity per of component
- ☐ Attributes (defined by user)
- ☐ Balloon reference point
- ☐ Balloon position

With action *copied bill of material balloon*, it is also necessary to specify the balloon to be copied. The copied balloon has no item number.

Bill of Material Generation

The properties *font*, *oblique angle*, *size-to-width ratio*, and *text size* are to be provided as for text objects.

The optional parameter *optical characteristics* is input via the following dialog box:



Via this dialog box, you may specify the optical characteristics and contents of a bill of material balloon.

- ☞ Check the default of the representation contents (pos. number) and change it, if required:
 - If you click on an unselected button, the corresponding representation content is set.
 - If you click on a selected button, the respective representation content is removed from the selection.
- ☞ Check the default of the representation shape (reference line with number enclosed in a circle) and change it, if required, by clicking on the corresponding menu area:
 - Reference line with number not enclosed
 - Reference line with number enclosed in a rectangle
 - Reference line not displayed (e.g. for standard components)
- ☞ Click on OK.

The values for the parameters, which define the optical characteristics of the balloon, are requested. It is also possible if so desired to select the corresponding parameter icon directly with the mouse. With the action *copied bill of material balloon* the parameter *load sequence number* is optional.

If *Pos. number* is set as representation content, the Id number of the component is set to " ".

The optional parameter *quantity* per has a standard default value of 1. Any other value may be input via the text area or by implicit construction.

The attributes which are stored in the balloon are input via the following dialog box, which is displayed automatically when generating the balloon under action *bill of material balloon*:

Name	Value	-> str	att
version	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
unit	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
name	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
material	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
stock number	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
remark	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
supplier	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
mass	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
mass unit	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
remark2	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>

ok cancel

Bill of Material Generation

The dialog box contains a list of attribute names which are defined in the files *<language>.bom* of the directory *#/design_config* according to the interactive language specified (see chapter „*Bill of Material Configuration*“ on page 16-22).

☞ Input the attribute values directly into the corresponding value fields or take them from reference objects:

- By clicking on *str*, you may construct the corresponding value implicitly as a *string* object via the fourth menu column. For this purpose, the dialog box is closed. The dialog box is opened again after the input is completed.
- By clicking on *att*, you may take the corresponding attribute from another balloon. The balloon is to be selected in the drawing area. For this purpose, the dialog box is closed. The dialog box is opened again after the input is completed.

☞ Conclude the input of attributes via *OK*.

When generating under action *copied bill of material balloon*, the values from the balloon specified are normally used. It is also possible, if so desired to select the corresponding parameter icon directly with the mouse.

☞ Select the *balloon reference point* in the drawing area or construct it implicitly.

☞ Sketch the balloon position in the drawing area or construct it implicitly. For implicit construction the distance from an existing balloon along x and y axes may be specified.

With the action *copied bill of material balloon*, the balloon to be copied must also be selected within the drawing area.

16.2 Bill of Material Output

The bill of material data may be transferred from the balloons into the model or an ASCII file. You may configure the bill of material structure and file format.

The following actions are available:

- Output of bill of material in a model (*bom_tomodel* and *part_to_list*)
- Output of bill of material to a file (*bom_tofile* and *part_to_file*)

16.2.1 Extracting Bills of Material to Models

16.2.1.1 Action bom_tomodel



Bills of material in models are realized as texts (effects of action) of the user defined action (UDA) *bom_tomodel*. The corresponding texts are created by an AQL program *#/d2drw/udx/bom_tomodel.aql* belonging to the UDA.

The following parameters are available:

- Format of bill of material
- Font
- Oblique angle
- Character proportions
- Ratio of frame height and text height
- Text height
- Reference point
- Element set of balloons, parts and/or UDOs containing bill of material balloons

Bill of Material Generation

A dialog box including the following format defaults is opened for the entry of the bill of material format:

output attribute	output name	length
Pos number	Pos number	45
Id number	Id number	45
quantity	quantity	45
version	version	0
unit	unit	0

output direction: ☒ up ☐ down

sort direction: ☒ up ☐ down

sum up: ☒ yes ☐ no

10 lines per block

ok cancel

You can change these defaults as follows:

- For system-defined output attributes, you can change output names and column length in mm (extracting a file to characters). Spaces entered in output names are used as separators if the column length is too short for the output name. Attributes with column length 0 are not output. Attributes with negative length are left-aligned.
- You can change the default output direction from column head up to column head down.
- You can change the default sorting direction from minimum position number up (increasing) to maximum position number down (decreasing).
- Instead of summing the numbers of parts with equal position and item number you can have numbers entered individually.



Bills of material in which the quantities of components with the same load sequence or item numbers are summarized, are not updated by parameter changes.

- You can change the maximum number of 10 successive lines for the default bill of material. The bill of material is continued with the next line at the left (or at the right if the output direction is downwards) with the proper spacing.

The default font is *Helvetica*. You can change it via the font box.

The default oblique angle is 0.0. You can change it in the text field.

The default character proportion is 1.0. You can change it in the text field.

The default ratio of frame height to text height is 1.7. You can change it in the text field.

The default text height is 3.5. You can change it by the text field.

☞ Select the lower right corner (or upper left corner in case of downward output direction) of the standard bill of material as reference point in the drawing, or construct it implicitly.

☞ Input the elements whose data is to be carried over into the bill of material. In the present version, you can select or implicitly construct bill of material balloons or UDOs containing those balloons.

After input of all parameters the bill of material is displayed. Any further lines are displayed to the left and/or right one line lower at a suitable distance.

Bills of material extracted to models can be edited by the *text* object group (see main chapter *Objects*, section *Changing properties and parameters of a generation action*). For this purpose, you must select a line of the bill of material to be changed in the drawing area. In the third menu column, the properties of the line followed by the parameters of the complete bill of material are shown from top to bottom.

You may only change the data of a bill of material via the corresponding balloons (object group *symbol*), since the bill of material has a relation to the bill of material data in the model.

Bill of Material Generation

16.2.1.2 Action part_to_list



This action is designed for models which are converted from SIGGRAPH-Design. For new constructions please use the [„Action bom_tomodel“ on page 16-7](#).

Bills of material in models are to be realized as user defined objects (UDOs) with a header and a number of lines and user defined actions (UDAs). The UDO *bom-header.udo* and the UDA *bomheader.uda* for the header and *bomline.udo* and *bomline.uda* for the lines, are included in the standard system. You may generate additional UDOs and UDAs (see chapter [„Bill of Material Configuration“ on page 16-22](#)).

The included UDAs for bill of materials are invoked via the action *Output parts list to screen – part_to_list* (see Online Help).

The following parameters are available:

- Summarize
- User defined action
- Reference point
- Selected set of balloons
- Any other user-defined parameters

The optional parameter *summarize* is used to specify whether the quantities of components with the same load sequence and item number are to be given summarized or separately for each occurrence. The default is *summarized*.



Summarization is only possible when the quantity per is numeric.

The optional parameter *User defined action* is used to specify the name of the user defined action, describing the structure of the bill of material. The name of the user element is input by invoking the file selection menu (see main chapter Working with Models, chapter „*File Operations*“ on page 5-39). If no user defined action is input, the standard configuration of *bomheader.uda* for the bill of material header and *bom-line.uda* for the lines is taken from the directory *#/symbols*.

When using the standard configuration, the reference point is identified in the drawing as the lower right corner of the bill of material, or constructed implicitly.

The parameter *selected set of balloons* is furnished with data by invoking the selection menu. The balloons whose data is to be carried over into the bill of material, are selected in a rectangle (see main chapter System Handling, section „*Selection in Rectangle*“ on page 4-22).

It is also possible to use data from the drawing frame in the bill of material header. It is essential that the data concerned is identical in both the drawing frame and the bill of material balloons in the user elements (see chapter „*Bill of Material Data Configuration*“ on page 16-22). The drawing frame from which the data is to be derived, is selected together with the bill of material balloons concerned.

After input of all parameters the bill of material is displayed. When using the standard user elements a maximum of 5 lines is displayed one above the other. Any further lines are displayed to the left one line lower at a suitable distance. This arrangement may be configured (see chapter „*Bill of Material Configuration*“ on page 16-22).

Bill of Material Generation



The contents of a bill of material may only be changed via the balloons, since the bill of material has a relation to the bill of material data in the model. Changes may be made either directly in *Edit* mode via the object *symbol* or indirectly via the object *User* after identifying the line concerned.

When making changes via the object *User* it is possible to descend through the structure to the balloon required and edit it as needed.



Bills of material in which the quantities of components with the same load sequence or item numbers are summarized, are not updated by parameter changes.

16.2.2 Extracting a Bill of Material to a File

16.2.2.1 Action bom_tofile



Bill of material data can be output by the AQL program `#/d2drw/udx/bom_tofile.aql` belonging to the UDA `bom_tofile` to an ASCII file.

The following parameters are available:

- Format of bill of material
- Name of output file
- Element set of bill of material balloons, components or UDOs containing bill of material balloons

In order to input the bill of material format, the dialog box for the setting of the bill of material format is opened. You can change its defaults (see section *Extracting bills of material to models*).

☞ Input the name of the ASCII file to which the bill of material is to be output via the opened file dialog box (see main chapter Working with Models, chapter „*File Operations*“ on page 5-39).

☞ Input the elements whose data is to be carried over into the bill of material. In the present version, you may select or implicitly construct bill of material balloons or UDOs containing those balloons.

After input of all parameters the bill of material is output to the file specified.



Model data must be output to file after every update, otherwise component and bill of material data will not be consistent.

Bill of Material Generation

16.2.2.2 Action part_to_file



This action is designed for models which are converted from SIGGRAPH-Design. For new constructions please use the „[Action bom_tofile](#)“ on page 16-13.

With the aid of an AQL program, you may output bill of material data to an ASCII file. The generation of the AQL program is described in the chapter „[Configuration of Bills of Material in Files](#)“ on page 16-30. This function is invoked with the action *output bill of material to file*.

The following parameters are available:

- Summarize
- Name of extracting AQL program
- Name of ASCII file
- Bill of material balloon

The optional parameter *summarize* is used to specify whether the quantities of components with the same load sequence and item number are to be given summarized or separately for each occurrence. The default is *summarized*.

The name of extracting AQL program is input via popup menu. Default is *bom....aql*.

The name of the ASCII file to which the bill of material is to be output is input via the file box (see main chapter Working with Models, chapter „[File Operations](#)“ on page 5-39).

The parameter *bill of material balloon* is furnished with data by invoking the selection menu. The balloons whose data is to be carried over into the bill of material are selected in a rectangle (see main chapter System Handling, section „[Selection in Rectangle](#)“ on page 4-22).

It is also possible to use data from the drawing frame in the bill of material header in the same way as when incorporating the bill of material in a model (see chapter „[Extracting Bills of Material to Models](#)“ on page 16-7). The drawing frame from which the data is to be derived, is selected together with the bill of material balloons concerned.

Bill of Material Generation

After input of all parameters the bill of material is output to the file specified.



Model data must be output to file after every update, otherwise component and bill of material data will not be consistent.

Bill of Material Generation

16.3 Creating a Bill of Material

The procedure for bill of material generation is illustrated by a simple example demonstrating the use of the actions *bom_tomodel* and *bom_tofile*. The model shown consists of two plates lying one on top of the other and held together by two screws.

The bill of material is to conform to the standard configuration and contain the following information:

- Position
- Item number
- Load sequence number
- Quantity
- Unit of measure
- Name
- Material
- Remark

The example given here includes the following procedural steps:

- Creation of the bill of material balloons for the plates and the screw 1 with action *bill of material balloon*
- Creation of bill of material balloon for the screw 2 with action *copied bill of material balloon*
- Extracting the bill of material to the model
- Extracting the bill of material to a file

16.3.1 Creation of Bill of Material Balloons

Bill of material balloons with all parameters are to be generated explicitly for the two plates and one of the screws with action *bill of material balloon*:

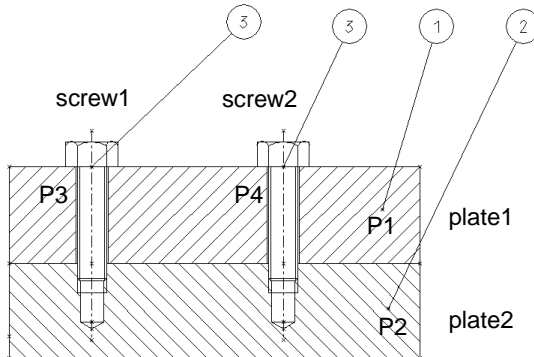


Plate 1

1. Click on the action *bill of material balloon* in the second menu column in *Create* mode (default configuration: menu *2D 1*, action group *symbols*).
2. Input the value 1 for the parameter *load sequence number*.
3. Click on the parameter *identification number* and input the value S1.
4. Input the following attribute values in the corresponding text fields of the opened dialog box:

Unit of measure	Pieces
Name	Plate 1
Material	ST 37
5. Select the point P1 as reference point in the drawing area.
6. Sketch the position of balloon in drawing area.

The bill of material balloon is generated for plate 1.

Bill of Material Generation

Plate 2

To generate the balloon for plate 2, mode, action group and action have already been specified.

7. Input the value 2 for the parameter *load sequence number* via text area.
8. Click on the parameter *identification number* and input the value S2.
9. Click on the *att* button behind each value fields *unit* and *material* in the open dialog box in order to enter the bill of material attributes and click on the bill of material balloon of plate 1 in the drawing area.
10. Enter *plate 2* in the value field *name*.
11. Select point P2 as reference point in the drawing area.
12. Sketch the position of balloon in the drawing area. The bill of material balloon is generated for plate 2.

Screw 1

To generate the balloon for screw 1 only properties and parameters have to be specified.

13. Input the value 3 for the parameter *load sequence number* via text area.
14. Click on the parameter *identification number* and input the value S3.
15. Click on the *att* button behind each value field *unit* in the open dialog box in order to enter the bill of material attributes and click on the bill of material balloon of plate 1 in the drawing area.
16. Input *screw* in the value field *name* and *ST 60* in the value field *material*.
17. Select point P3 as reference point.
18. Sketch the position of balloon in the drawing area. The bill of material balloon is generated for plate 2.

The bill of material balloon is generated for screw 1.

16.3.2 Creation of a Bill of Material Balloon as a Copy of an Existing one

When creating the bill of material balloon for the screw 2 it is possible to take data from the balloon for screw 1:

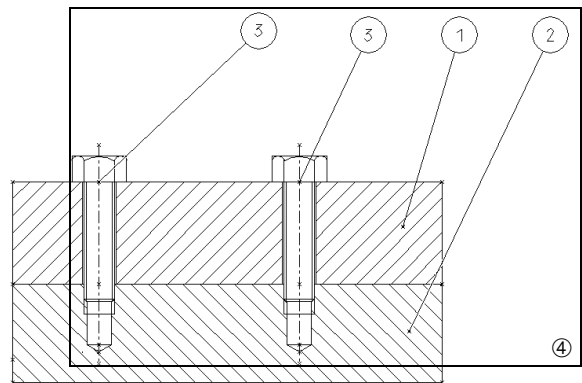
1. Click on action *copied bill of material balloon* in the second menu column in *Create* mode (default configuration: menu *2D 1*, action group *symbols*).
2. Select balloon for screw 1 in the drawing area as reference balloon.
3. Select point P4 in the drawing area as reference point.
4. Sketch the position of the balloon in the drawing area.

After specification of the last parameter the bill of material balloon for screw 2 is created.

Bill of Material Generation

16.3.3 Extracting the Bill of Material to the Model

The bill of material is now displayed using the data stored in the balloons.



3	S3	2	Stk	ST 60
2	S2	1	Stk	ST 37
1	S1	1	Stk	ST 37
Position number	Item number	Quantity	Unit	Material

1. Click on action *incorporate bill of material in model* in the second menu column in *Create mode* (default configuration: menu *2D 2*, action group *output*).
2. Enter the desired attributes for the bill of material into the dialog box.
3. Enter the reference point for bill of material header: sketch position in drawing area.
4. Select all balloons in rectangle.

After specification of the last parameter the bill of material is displayed.

16.3.4 Extracting a Bill of Material to a File

The bill of material data is also stored in the file *stueckl.bom*.

1. Click on the action *output bill of material to file* in the second menu column in *Create* mode (default configuration: menu *2D 2*, action group *output*).
2. Enter the desired attributes for the bill of material into the dialog box.
3. Enter the name of the output file *stueckl.bom* via file box.
4. Select all balloons in rectangle.

After specification of the last parameter the bill of material is written out to the file.

Bill of Material Generation

16.4 Bill of Material Configuration

You may configure the bill of material type and data type to tailor them to the requirements of the application. This task should only be undertaken by experienced users well practiced in the creation of UDOs, UDAs and AQL programs.

16.4.1 Bill of Material Data Configuration

The quantity and type of bill of material data stored in the balloons are defined by the user, with the exception of three system defined attributes. The following attributes are defined by the system for all bill of material balloons:

- *\$_Position* (load sequence number)
- *\$_Identification* (item number)
- *\$_Quantity* (quantity per)

The user-defined attributes are to be described in the language files of the directory *#/design_config*, and are proposed via a dialog box for furnishing the parameter *attributes* when creating the bill of material balloon.

This directory contains for each definable language a file *<language>.bom*. For example, when the language defined is *English*, the file *En.bom* is used.

The files *<language>.bom* contain the list of those attributes which may be defined as defaults for all bill of material balloons.

A line of the *<language>.bom* file is subdivided into 2 parts:

- 1st part internal attribute name
English names are used only. This part is the same for any *<language>.bom* file
- 2nd part language-specific attribute name
This name is offered in the dialog box in the user dialog. The language-specific attribute name is a translation of the internal attribute name.

The procedure for adding new attributes to the user dialog is as follows:

- ☞ Switch to directory *#/design_config*.
- ☞ Open defined language file, e.g. *En.bom*.
- ☞ Add corresponding internal and attribute names to the file. The file is saved and the dialog box for value input for balloon attributes is updated.

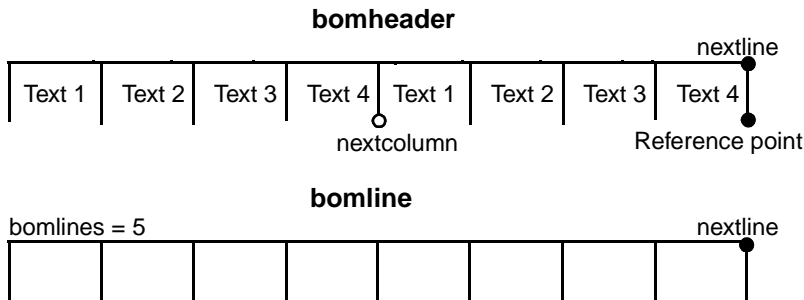
Bill of Material Generation

16.4.2 Configuration of Bills of Material in Models

You must create an UDO and an UDA in the directory *##/symbols*, to create a bill of material for the current application:

- One UDO and UDA for the header or print format of the bill of material
- One UDO and UDA for one line of the bill of material

bomheader.udo and *bomheader.uda* for the header and *bomline.udo* and *bomline.uda* for one line are supplied with the system and you may adapt them to suit your requirements.



The standard configuration has the following structure:

- The reference point specifies the position of the bill of material within the model. This in the standard configuration is the lower right corner of the bill of material header, and may be positioned anywhere within the drawing area via UDA.
- The points *nextline* define the points of attachment for the following line of the bill of material. In the standard configuration the header occupies the lowest line. The bill of material lines are arranged in sequence above it.
- The variable *bomlines* defines the maximum number of lines which may be arranged one above the other. In the standard configuration this is 5, the load sequence numbers being in ascending order from bottom to top.
- The bill of material is continued at the point *nextcolumn* if the limit defined in *bomlines* is exceeded by the actual number of lines. In the standard configuration the continuation is to the left, a new header being created in each case.

Creating of Bill of Material Header

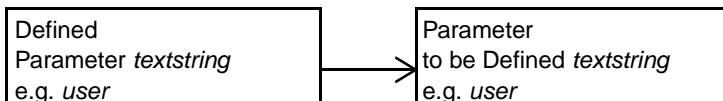
The UDO for the bill of material header must contain the following named elements (see main chapter System Handling, section „[Assigning of Names“ on page 4-17](#)):

- Point *nextline*, at which the first line is to begin
- Variable *bomlines*, which defines the maximum number of lines which may be arranged one above the other. If the limit is exceeded, the continuation is positioned at one side. The sort sequence of the bill of material lines (ascending or descending) may be governed by the leading sign:
If the load sequence numbers are to be in ascending order from bottom to top, the variable *Zeilenzahl* must carry a value greater than zero.
If the load sequence numbers are to be in ascending order from top to bottom, the variable *Zeilenzahl* must carry a value less than zero.
- Point *nextcolumn*, at which the bill of material is to be continued if the limit defined in *bomlines* is exceeded.

If a text element from the text area of the drawing frame is to be used in the bill of material header, the corresponding entries in the frame and header must be identical.

Drawing Frame

Bill of Material Header



All parameters for which values are to be requested must be defined as input parameters to the UDA for the *bill of material header*. The parameter defined first should be the reference point of the bill of material header.

Bill of Material Generation

Creating of the Bill of Material Line

The UDO for the bill of material header must contain a named point, at which the next line is to begin.

If the bill of material header occupies a consistent position as footer or header within the bill of material model to be generally applied, the bill of material line need only be created with one horizontal demarcation line.

The UDO requires the balloon whose data is to be output in this bill of material line, as parameter. The UDA requires the reference point of bill of material line as parameter. If further parameters are defined, values for them will be requested during bill of material setup.

After creation UDOs and UDAs must be saved under any desired file name.



Before creating UDOs and UDAs under the model names *bomheader.udo* and *bomheader.uda* and *bomline.udo* and *bomline.uda*, you must rename the files supplied containing the standard bill of material.

The bill of material is invoked within a model by reading the UDOs and UDAs from the directory *#/symbol* via the file box (see chapter „[Extracting Bills of Material to Models](#)“ on page 16-7). The reference point of the first bill of material line is positioned at the point *nextline* of the UDO *bomheader.udo*. A further UDO *bomline.udo*, referenced to the point *nextline*, plus one of the specified balloons, is then read into the model for each actual bill of material line.

Example

A bill of material is to be created and listing the following component data:

- Load sequence number
- Item number
- Quantity per
- Description
- Material
- Remarks

The bill of material is to be of the same width as the text area of the drawing frame.

L/S no	Item number	Qty per	Descr	Material	Remarks	L/S no	Item number	Qty per	Descr	Material	Remarks
next line						next line					
						bomlines = 4					

Bill of Material Generation

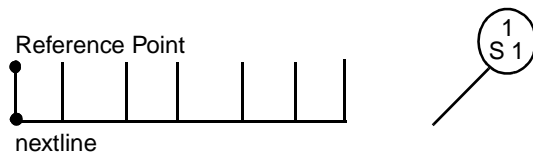
Creating of the UDO for the Bill of Material Header

1. Create a new model.
2. Create the geometry for the bill of material header, including the points *nextline* and *nextcolumn*.
3. Assign the name *nextline* to the point at which the first bill of material line is to be positioned and assign the name *nextcolumn* to the point at which the next bill of material box is to be positioned.
4. Create the required text elements in the bill of material header as a *text* object.
If text from the drawing frame is to be used, the corresponding strings are to receive the same contents as in the drawing format (original input in UDO, see file *titlecorn.udo* in the installation directory *#/symbols*).
5. Create variable *bomlines* for sort sequence and number of bill of material lines.
6. Transfer variable *bomlines* to a non-visible layer.
7. Append the object names *nextline*, *nextcolumn*, and *bomlines*.
8. Store the UDO under the name *bomheader.udo*.

Creating of the UDA for the Bill of Material Header

1. Create a new model.
2. Create an UDO *bomheader.udo*.
3. Create the reference point.
4. Append the *reference point* parameter.
5. Store the UDA under the name *bomheader.uda*.

Creating of the UDO for the Bill of Material Line



1. Create a new model.
2. Construct geometry, including point *nextline*.
3. Create bill of material balloon.
4. Create required text elements in bill of material line (string from balloon: input appropriate attributes via text area), e. g. *\$_Quantity* for column *Quantity*).
5. Append the object name *nextline*.
6. Store the UDO with the name *bomline.udo*.

Creating of the UDA for the Bill of Material Header

1. Create a new model.
2. Create an UDO *bomline.udo*.
3. Create the reference point.
4. Append the parameter *reference point*.
5. Append the parameter *bill of material balloon*.
6. Store the UDA under the name *bomheader.uda*.

Bill of Material Generation

16.4.3 Configuration of Bills of Material in Files

Bills of material may be output to file by an AQL program in table form, for example.

To output bills of material to a file, two AQL programs are required in the directory *#/aql.aql* (see Volume *Open Architecture*):

<name>.aql extract one bill of material line with summarize *no*
<name>sum.aql extract one bill of material line with summarize *yes*

Two frame AQL files (*bom.aql* and *bomsum.aql*), which contain the user-configurable part in includes (*bom.inc* and *bomsum.inc*), are provided to allow the user to configure the bill of material without an AQL developer's license. This license is not necessary to modify these two files.

Which AQL program is actually used depends on *summarize*:

summarize *no* *<name>.aql*
summarize *yes* *<Name>sum.aql*

It is possible to access the set of selected balloons with the name *bomgroup* and the selected drawing frame with the name *Drawing*. This set has also the attribute *file-name*. The attribute value is name of the file to be defined.

An example file *bom.inc* can be found in the *#/aql.aql* directory.

17 Configuring the System

As **EUKLID Design** is a user-friendly system, you can configure it to your precise requirements. This is supported by a multistage configuration concept.

Data that may be specified when configuring the system includes:

- appearance of the user interface following program start (e.g. colors, dialog language, menus. etc.)
- configuration data to handle the interface to other systems
- default system parameters (e.g. response of the sketching tools, file search rules, etc.)
- optional default parameters and properties

These settings may also be defined as temporary, in other words just for the duration of a session, without saving them in the configuration files. This is done at several levels, each of which can be assigned different priorities:

- system configuration
- user-oriented configuration
- model-oriented configuration

Model-oriented configuration data is always stored with the model when the model is saved. No special actions are required on the part of the user.

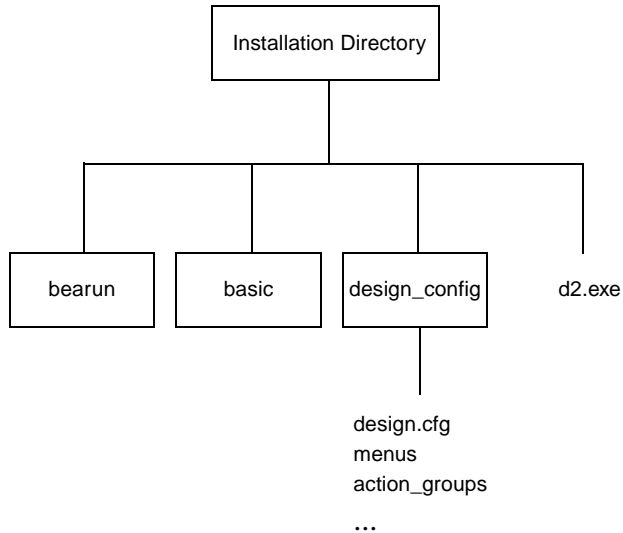


You may store the contents of a model as part of the configuration. This enables you, for example, to start a session always using a specific standard model.

Configuring the System

17.1 The System Configuration

The configuration applies across the whole installation and should only be edited by authorized persons. System-wide settings are stored in the subdirectory called *#/design_config*.



The system configuration contains following information:

- Menus
- Viewports
- Optional parameters and properties
- Model data
- Application specific options

Menu Configuration Specification

The *menu configuration specification* is a list of menus that must be loaded. Each menu is a list of action groups. Menus and action groups reside in separate files. If the configuration file contains user menus, these will also be stored, together with the associated action groups, in the `#/design_config` directory. The following naming conventions are used:

- `<menu_name>.mnu` menu files
- `<action_group_name>.agp` action group files
- `<object_group_name>.ogp` object group files

EUKLID Design will use the file locator module to find the required menu and action group descriptions. For this reason, the file locator table will contain the `#/design_config` path.

Viewport Configuration

This section of the configuration file contains a specification of the position, size and content of the viewports that are opened when the system is started.

Optional Parameter (and Property) Specification

This part of the configuration defines which parameters and properties are optional and what their default values will be.

Optional value parameters apply to all loaded models.

Optional object parameters are dependent on the active model. If one defines an object parameter to be optional, the option will only be set for the model in which the object is stored.

Configuring the System

Model Section

This part of the configuration defines what objects and actions must be created each time a new model is created.

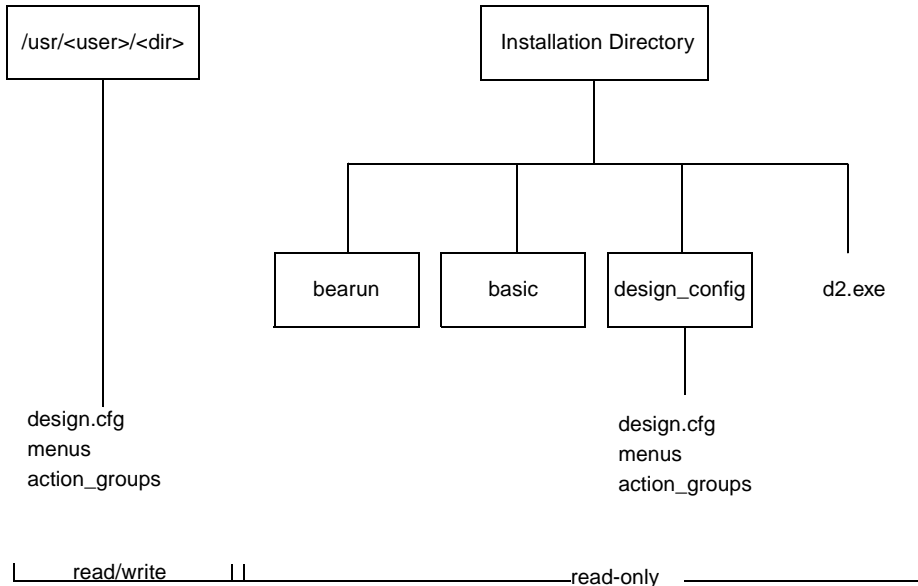


- You can use the start option `-config_dir <config_path>` to specify the directory that you want to search for the system configuration files.

Example

If the start option `-config_dir /usr/meier/config` has been specified, the configuration file pathname is `/usr/meier/config/design_config/design.cfg`.

- You are able to split the installation directory from the directory in which the configuration information is stored. This makes it possible to consider the installation directory as a read-only file system. By separating the distribution files from the configuration files, we make sure that the configuration files will not be overwritten when a new **EUKLID Design** version of the application is installed.



17.2 The User-oriented Configuration

The user-oriented configuration applies to the models and working environment of a user and overrides the system configuration. It determines the model configuration data when a new model is started. The user-oriented configuration and all associated menu and action group files will be stored in the subdirectory with the default name *design_config* of your home directory.

The user configuration file contains the following information:

- Menus
- Viewports/views
- Optional parameters and properties
- Model data
- Application specific options

The naming conventions for *user configuration files* are the same as for system configuration files.

17.3 The Model-oriented Configuration

The model-oriented configuration applies to a model and overrides all other configuration settings with the exception of session data such as dialog language, etc. It is saved in every model file.

The model configuration contains the following information:

- Viewports
- Model data (objects and actions)
- Application specific options

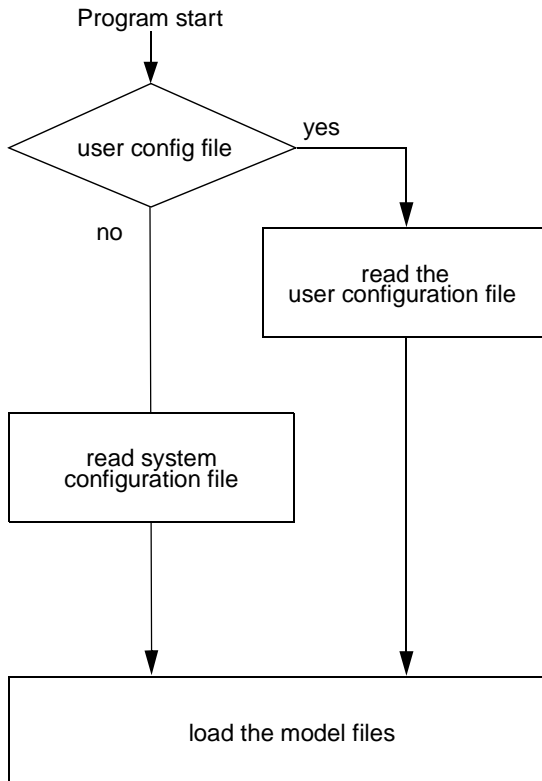
Configuring the System

17.4 Loading the Configuration

The configuration information is read by **EUKLID Design** from the following files:

- System configuration file
- User-oriented configuration file
- Model file

When a new model or session is started, the settings are reset and reloaded in the following sequence:



If there is no user configuration file, the system configuration file `#/design.config` delivered with the system is used.

When **EUKLID Design** is loaded, the system checks the existence of a current configuration file. If an old file (with previous name *grivak.config* or with old version number) is found, it is converted appropriately.

17.5 Saving the Current Configuration

The following types of saves can be performed

- save across the entire system
- a user-specific save
- a model-oriented save



You must have the appropriate permissions to be able to save a configuration.

17

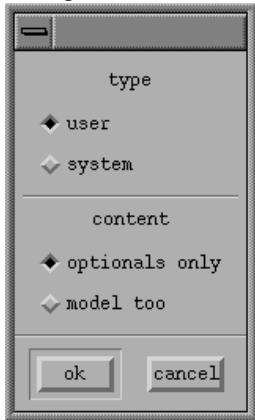
17.5.1 Saving System

If the configuration data for the active model is to be saved across the entire system, the saved configuration will apply to all users that do not have their own configuration files.

Preceed as follows:

Configuring the System

- ☞ Choose *Save configuration* in the *File* menu. The following dialog box for setting the configuration will be opened:



- ☞ Click on *system*.
- ☞ Click on *model too* if you want to save the objects and actions of the model as well as the optional values in the system configuration file.
- ☞ Click on *OK*.

17.5.2 User Specific Saving

If the configuration is saved on a user-specific basis, it will apply to all models of the user that saved the configuration.

Proceed as follows:

- ☞ Choose *Save configuration* in the *File* menu. The dialog box for setting the configuration will be opened. Default is the user specific saving.
- ☞ Click on *model too* if you want to save the objects and actions of the model as well as the optional values in the system configuration file.
- ☞ Click on *OK*.

17.6 Configuring the Interprocess Communication (IPC)

The required settings are held in the file `#/d23bs/ipc.config`.

This file is searched for in the following sequence:

- Current directory
- \$HOME
- \$HOME/user_data
- #/d23bs

It consists of a sequence of a keyword, a space, a text string and line end.

Text following `#` or `//` is treated as a comment until the end of the line.

Where the character `#` appears within a text string, it must be paired with the `\` character if it is not to be interpreted as the start of a comment.

pat5	pathname for <i>Sigraph-Ratio</i>
arg5	name and arguments for <i>Sigraph-Ratio</i>
pat8	pathname of the program for <i>Ingres</i>
arg8	name of the program
pat9	pathname of the program for <i>Oracle</i>
arg9	name of the program
timo	time out period for establishing the connection

The pathname for the installation directory must begin with `\#`. The `\` is required so that the `#` is not interpreted as the start of a comment. The pathname must not contain any more `#` or `\#` strings.



Note that when a remote user logs in, the system will attempt to find the pathnames for the database interface programs. The necessary start procedures and programs must be copied to the database server or the NFS link specified.

Configuring the System

Example of ipc.config

```
# Inter Process Communication Config File
# 2 Oktober 1995
# -----
#
# pathname for Sigraph-Ratio
pat5 /usr/sigraph/sit/sit.start
# arguments
arg5 sit -ipc design
#
# ingres parameters
# Be aware rexec() uses names relative to home of remote login
#
pat8 \#/d23bs/start_ingres
arg8 start_ingres
#
# Be aware rexec() uses names relative to home of remote login
#
pat9 \#/d23bs/start_oracle
arg9 start_oracle
#
# timeout for remote ipc
# times are given in seconds
# after timo seconds of a request to connect to a remote host
# the request is considered as failed
timo 10
#
```


17.7 Configuring the User Interface

17.7.1 Switching the Cursor Icon On and Off

How the cursor is displayed depends on the mode:

- In *Create* mode, the icon for the active action and below it the icon for the required parameter will be shown attached to the cursor.
- In *Edit* and *Delete* modes, the icon(s) of the action(s) affecting the object will be displayed as the cursor touches the objects.

The relevant object is highlighted.

This property can be specified as follows:

☞ To switch the cursor on or off, choose *Cursor icon on* or *Cursor icon off* respectively from the *Options* menu.



This property can be made permanent by selecting *Save configuration*.

17.7.2 Switching the Exec Cursor On and Off

This provides a preview of the object that is being created and can be used with most parameters/properties:

- In *Create* mode, the object that would be created is displayed temporarily at the cursor while the left mouse button is held down.
- In *Edit* mode, the current value corresponding to the cursor position for the selected parameter is displayed while the left mouse button is held down.

The value is accepted by the system when the button is released. The relevant object is highlighted.

This property can be specified as follows:

☞ To switch the cursor on or off, choose *Exec Cursor on* or *Exec Cursor off* respectively from the *Options* menu.



This property can be made permanent by selecting *Save configuration*.

Configuring the System

17.7.3 Setting the Dialog Language

The dialog language is the language that will be used for all **EUKLID Design** messages, input prompts and documentation. You can change the language as follows:

- ☞ Choose *Select language* from the *Options* menu. The language dialog box will appear.
- ☞ Click on the required language in the listbox.
- ☞ Save the configuration.
- ☞ Restart **EUKLID Design**.



This property can be made permanent by selecting *Save configuration*.

17.7.4 Setting the Colors

To permit a certain degree of modification to the user interface, a number of color palettes are provided that can be used on either a light or dark background.

You can select a color scheme as follows:

- by specifying the following switch when **EUKLID Design** is started
-scheme <Name>
- by choosing *Select color palette* from the *Options* menu:
 - ☞ Choose *Select color palette* from the *Options* menu.
 - ☞ Click on the required color palette in the listbox.
 - ☞ Save the configuration.
 - ☞ Restart **EUKLID Design**.

17.7.5 Define Default viewport type

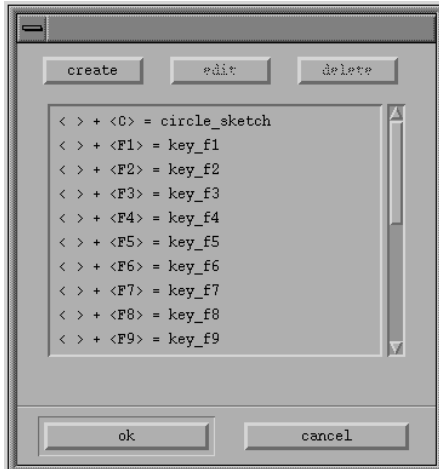
This allows you to specify which viewport type should be used by default when the dialog box *View* is opened.

A description of the dialog box can be found in chapter *Visualization methods*, section *View*.

17.7.6 Define Function keys

Actions can be assigned to keys on the keyboard. The action will be performed when the appropriate key is pressed. To change these assignments

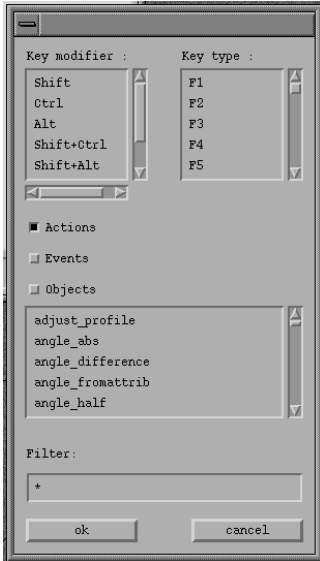
☞ choose *Define function keys* from the *Options* menu. The following dialog box will appear:



This dialog box shows the current keyboard assignments.

Configuring the System

Choosing the *create* option opens the *Define key assignments* dialog box.



- The keyboard modifiers <SHIFT>, <CTRL> and <Alt>, and combinations thereof, may also be used. When creating key assignments, make sure they are unique and note that some keys are predefined.
- The key assignments can be made permanent by selecting *Save configuration*.

17.7.7 Configuration of the Main Menu

Irrespective of the mode, you may create your own menus from system defined and user defined action groups:

Create mode	action group menu
Temp mode	temp action group menu
Edit and Delete mode	object group menu



System defined menus can only be modified by users with the appropriate access rights.



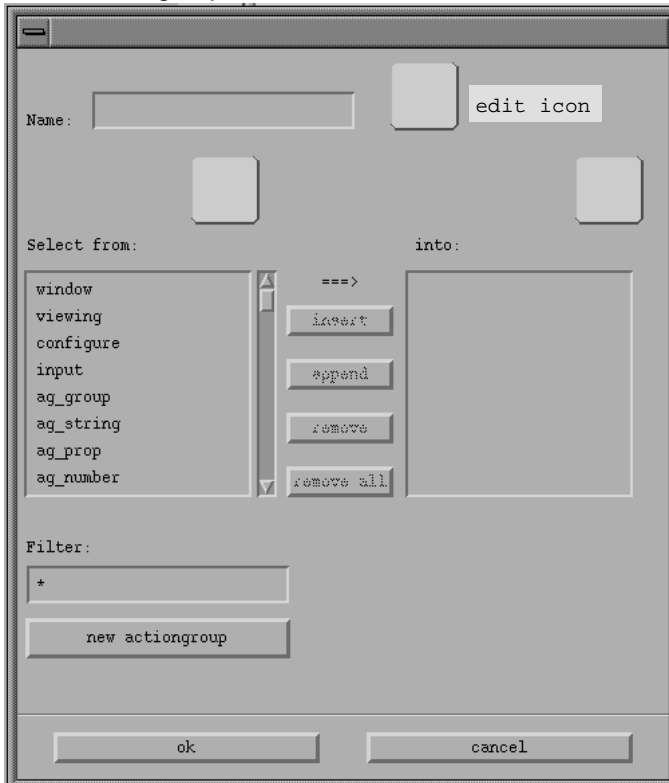
The main menu configuration can be saved (See „[Saving the Current Configuration](#)“ on page 17-7.).

17.7.7.1 Action Group Menus

Inserting Action Group Menus

You can create your own action group menus in *Create* mode and *Temp* mode.

☞ In the required mode, choose *insert menu* from the popup menu of menu icons. The menu dialog box, in which you can enter the name, the icon and the contents of the action group menus, will appear. You may then add new and existing action groups to the action group menu.



Configuring the System

- ☞ Enter a name for the menu in the *Name* field.
- ☞ Choose *edit icon* from the popup menu of icon fields. The edit icon dialog box, in which you design the menu icons, will appear (see chapter Standardization, section „[Creating User Icons](#)“ on page 13-46).
- ☞ In the listbox on the left, click on the action group to be added to the action group menu and then on the button *insert* or *append*.
insert inserts the action group into the action group menu before the action group selected in the listbox on the right.
append adds the action group into the action group menu after the action group selected in the listbox on the right.



- If you want to add a new action group to the action group menu, click on the *new action group* button. The action group dialog box, with which you create the new action group, will appear (see chapter Standardization, section „[Appending Object Type Icons and Action Icons to New Action Groups](#)“ on page 13-58).
- To keep the list of action groups to manageable proportions, enter a search expression in the *Filter* field and press <RETURN>, e.g. search expression *line_** for all actions that produce a line.
- To remove an action group from the action group menu, click on the relevant action group in the listbox on the left and then the *remove* button.
- To remove all action groups from the action group menu, click on the *remove all* button.

- ☞ Click on *OK*. The new action group menu will be inserted before the one currently selected.

Editing Action Group Menus

You can modify system defined and user defined menus in *Create* and *Temp* modes, but not application defined ones.

☞ In the required mode, choose *edit menu* from the popup menu of the icon for the menu to be edited. If this is a system defined menu, an appropriate stick-up message will appear that can be removed by clicking on *Continue*. The dialog box, in which you can edit the name, the icon and the contents of the menus, will appear (see the section „*Inserting Action Group Menus*“ on page 17-15 for details of the inputs to this dialog box).

Removing Action Group Menus

You can remove system defined and user defined menus in *Create* and *Temp* modes, but not application defined ones.

☞ In the required mode, choose *remove menu* from the popup menu of the icon for the menu to be deleted. If this is a system defined menu, an appropriate stick-up message will appear that can be removed by clicking on *Continue*. The menu will be removed.

Hiding Action Group Menus

You can remove menus from the user interface in *Create* and *Temp* modes.

☞ In the required mode, choose *hide menu* from the popup menu of the icon for the menu to be deleted. If this is a system defined menu, an appropriate stick-up message will appear that can be removed by clicking on *Continue*. The menu will no longer be visible.

17.7.7.2 Action Groups

Inserting Action Groups

You can insert your own action groups in *Create* and *Temp* modes.

☞ In the required mode, choose *Insert menu* from the popup menu of menu icon in which you want to insert the new action group. The menu dialog box will appear.

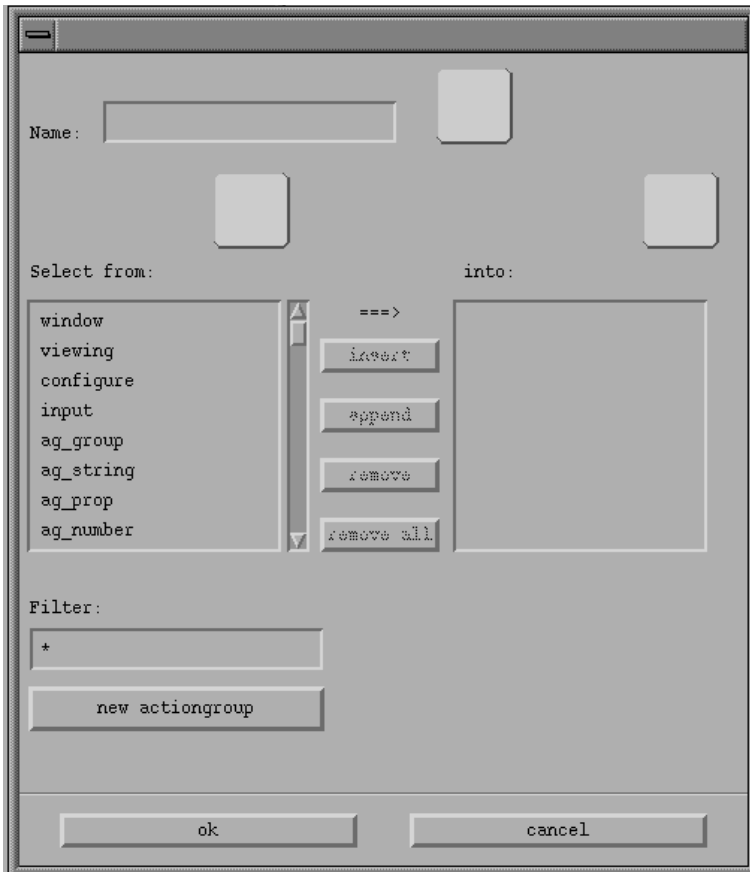
Configuring the System



In temp mode, action groups may only contain actions that are also valid temporary actions. As a rule, this will mean those that do not create a data structure.

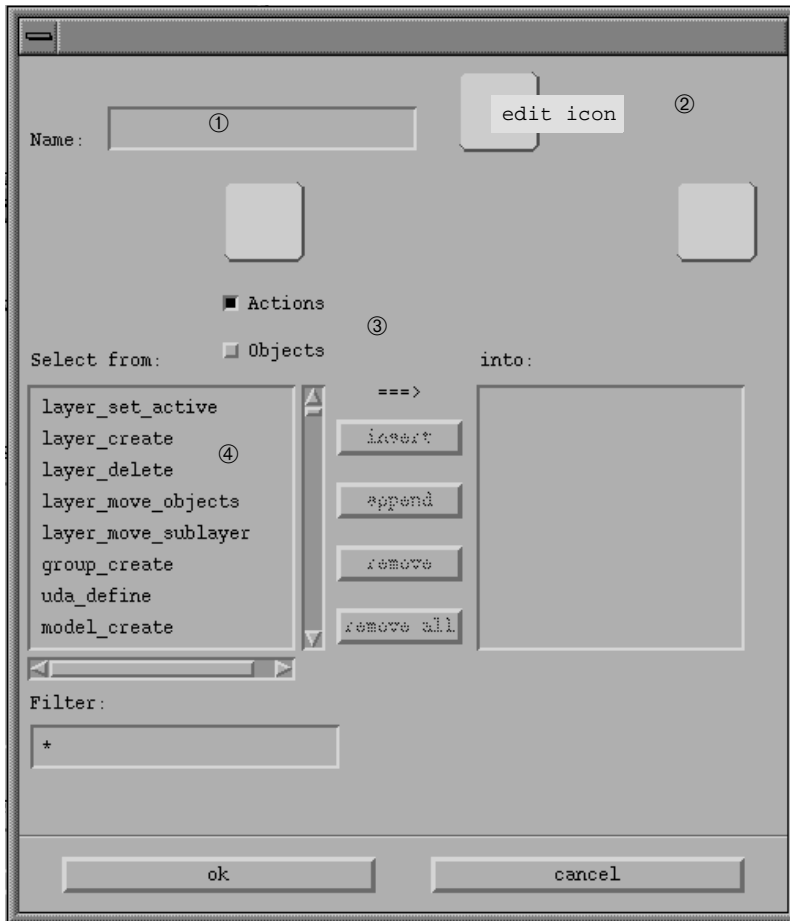


You can also open the menu dialog box using *Edit menu* from the popup menu for the corresponding menu icon. If this is a system defined menu, an appropriate stick-up message will appear. Click on *Continue* to open the dialog box.



☞ Click on *new actiongroup*.

The action group dialog box will appear:



- ➡ ① Enter a name for the action group in the *Name* field.
- ➡ ② Choose *edit icon* from the popup menu of the icon field. The icon editor, in which you can design the action group icon, will appear (see chapter Standardization, section „[Creating User Icons](#)“ on page 13-46).
- ➡ ③ Click on *Objects* and the predefined *Action* field to add an object to the action group.

Configuring the System



④

In the listbox on the left, click on the action or object to be added to the action group menu and then on the button *insert* or *append*.

insert inserts the action or object into the action group before the action or object selected in the listbox on the right.

append adds the action or object into the action group after the action or object selected in the listbox on the right.



- To keep the list of actions or objects to manageable proportions, enter a search expression in the *Filter* field and press <RETURN>, e.g. search expression *line_** for all actions that produce a line.
- To remove an action or object from the action group, click on the relevant action or object in the listbox in the left and then the *remove* button.
- To remove all actions or objects from the action group, click on the *remove all* button.



Click on *OK*. A dialog box will appear.



Click as appropriate to indicate whether the new action group is to be inserted above or below the one currently selected. The action group will be shown at the selected location in the menu.

Editing Action Groups

You can modify system defined and user defined action groups in *Create* and *Temp* modes, but not application defined ones. System defined menus can only be modified by those with the appropriate permissions.



In the required mode, choose *edit action group* from the popup menu of the relevant action group icon. If this is a system defined action group, a stick-up message will appear that can be removed by clicking on *Continue*. The action group dialog box, in which you can edit the name, the icon and the contents of the action groups, will appear (see the section „[Inserting Action Groups](#)“ on page 17-17 for details of the inputs to this dialog box).

Removing Action Groups

You can remove system defined and user defined action groups in *Create* and *Temp* modes, but not application defined ones. System defined menus can only be modified by those with the appropriate permissions.

☞ In the required mode, choose *remove action group* from the popup menu of the icon for the action groups to be deleted. If this is a system defined action group, an appropriate stick-up message will appear that can be removed by clicking on *Continue*. The action group will be removed.

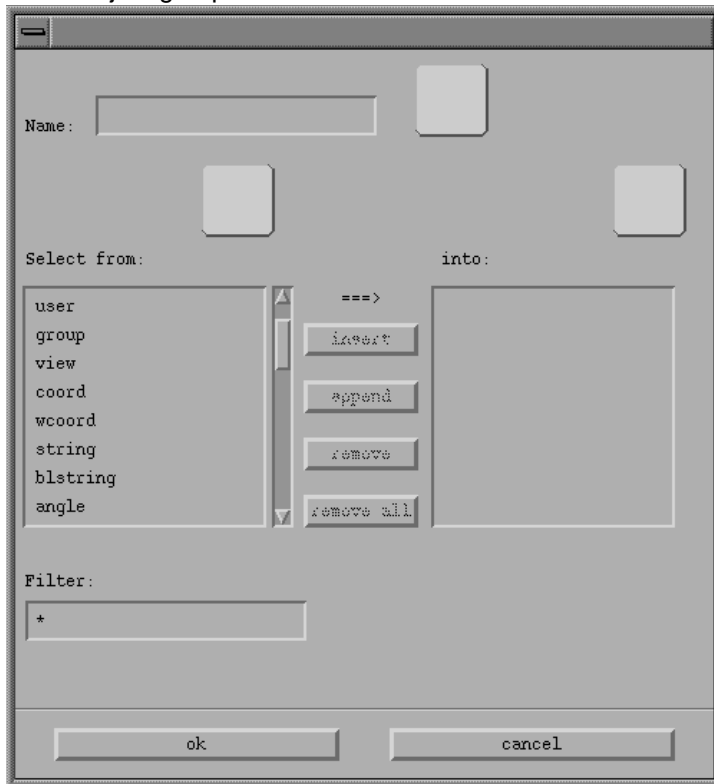
Configuring the System

17.7.7.3 Object Group Menus

Inserting Object Group Menus

You can insert your own object groups in *Edit* and *Delete* modes.

- ➡ In the required mode, select *insert objgrp* from the popup menu of menu selection icons. The object group dialog box, in which you can enter the name and content of the object group, will appear. System and user defined object types may be added to the object group.



- ➡ Enter a name for the object group in the *Name* field.
- ➡ Choose *Edit icons* from the popup menu of the icon field. The icon editor, in which you can design the object group icon, will appear (see chapter Standardization, section „[Creating User Icons](#)“ on page 13-46).

☞ In the listbox on the left, click on the object type to be added to the object group and then on the button *insert* or *append*.

- *insert* inserts the object type into the object group before the object type selected in the listbox on the right.
- *append* adds the object type into the object group after the object type selected in the listbox on the right.



- To keep the list of object types to manageable proportions, enter a search expression in the *Filter* field and press <RETURN>, e.g. search expression *t** for all object types starting with a t.
- To remove object types from the object group, click on the relevant object group in the listbox in the left and then the *remove* button.
- To remove all object types from the object group, click on the *remove all* button.

☞ Click on *OK*. The new object group will be inserted after the one currently selected.

Editing Object Groups

You can modify system defined and user defined object groups in *Edit* and *Delete* modes, but not application defined ones.

☞ In the required mode, choose *edit objgrp* from the popup menu of menu selection icons. If this is a system defined object group, a stick-up message will appear that can be removed by clicking on *Continue*. The object groups dialog box, in which you can edit the names, the icons and the contents of the object groups, will appear (see the section „*Inserting Object Group Menus*“ on page 17-22 for details of the inputs to this dialog box).

Removing Object Groups

You can remove system defined and user defined object groups in *Edit* and *Delete* modes, but not application defined ones.



If only one object group is present, it cannot be removed.

☞ In the required mode, choose *remove objgrp* from the popup menu of menu selection icons. If this is a system defined object group, a stick-up message will appear that can be removed by clicking on *Continue*. The object group will be removed.

Configuring the System

Hiding Object Groups

You can remove object groups from the user interface in *Edit* and *Delete* mode.

☞ In the required mode, choose *hide objgrp* from the popup menu of menu selection icons. If this is a system defined menu, an appropriate stick-up message will appear that can be removed by clicking on *Continue*. The menu will no longer be visible.



If only one object group is present, it cannot be hidden.

17.8 Configuration of the file system

17.8.1 File search rule (location table)

EUKLID Design identifies models and files belonging to them, such as UDx, external layers and tables, internally through use of unique identification numbers and not via the file name (symbolic name).

In models, external references to **EUKLID Design** files (UDx, layer) can appear. The user enters the pathnames (absolute or relative) where they can be found. In the interest of flexibility and to guarantee location, a file search rule is used: this assists in maintaining a table with pathnames where **EUKLID Design** searches for such files. The search is continued until the specified identification number is found.

A search for **EUKLID Design** files is conducted as follows: **EUKLID Design** searches first for files with the pathnames stored in the model. Only if the search was unsuccessful the search will be continued in the file search rule directories.

The file search rule is also stored as a file in the directory ~/design_config (*user-specific*) or #/design_config (*system-wide*). They are also read again when the program is restarted.

File name memory

In order to avoid redundant search processes the **EUKLID Design** files found are noted in the file name memory. This is also stored in a file in the directory ~/design_config. This procedure greatly accelerates the search procedures: file names available in the file name memory must be checked only for conformance to the file search rule. Therefore, if frequently used files are registered in the file name memory the file search rule must only play the role of a filter. In such cases it may even be possible to simplify the rule so that it contains only a few 'recursive' directories, e.g. "~".

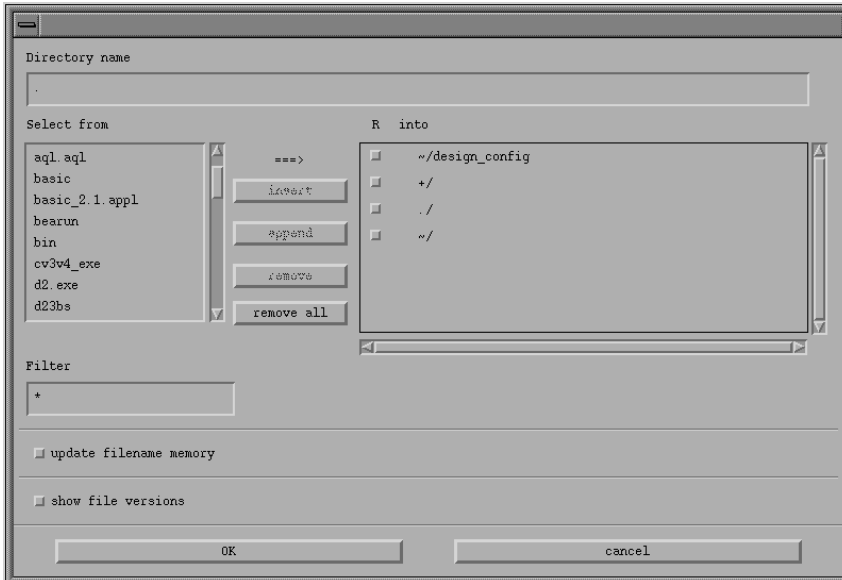
The file name memory is always extended automatically whenever an **EUKLID Design** file is encountered, e.g. during loading or search procedures. This 'learning' behavior of the file name memory can be forced by the parameter *Update filename memory*.

Configuring the System

Updating the file search rule (location table)

The file search rule can be updated by adding or removing search paths as follows:

- ☞ Select *Update location table* in the *File* menu and click onto the desired updating mode: *user-specific* or *system-wide*. The file search rule dialog box is opened:



Adding search paths

- ☞ In the left list field, click onto the directory to be added to the file search rule.
- ☞ In the right list field, click onto the directory before or after which the additional directory is to be added or appended.
- ☞ Click on *Insert* or *Append*:
 - Click on *insert* if the directory is to be searched before the selected directory.
 - Click on *append* if the directory is to be searched after the selected directory.

The directory is added to the right list field and thereby to the file search rule.



if you want to add all subdirectories of a selected directory to the file search rule, activate the option *Recursive* in front of the corresponding directory.

- ☞ *Updating filename memory*

When this option is active, all **EUKLID Design** files from all (sub-) directories of the file search rule table are stored in the file name memory.

Show file versions


This option affects how models are loaded: by copying (at operating system level), **EUKLID Design** shows identical 'Versions' of (sub-) models in a selection list. Through careful selection, you can determine the structure of your design. When the model is stored, this selected path is stored along with it. In this manner, the paths originally stored in the main model can easily be edited as well.

You may use this mechanism for simple versioning tasks, e.g. for test versions of UDO's.

 Click on **OK**. **EUKLID Design** stores all directories of the file search rule.


Removing search paths

 In the right list field, click onto the directory to be removed from the file search rule.

 Click on the *remove* field. The directory disappears from the right list field.



If you want to remove all directories from the right list field, click the box *remove all*.

 Click on **OK**. **EUKLID Design** removes all directories deleted from the list, including their subdirectories (with *Recursive* option), from the file search rule and stores it.



- If the name of the desired directory does not appear in the left list field, you must
 - change to the parent directory (by clicking onto the line '..')
 - enter the corresponding directory into the *Directory namebox*.The left list field will be updated with the corresponding subdirectories or files.
- You may enter a search expression into the *filter* field in order to reduce the entries of the list. Please confirm with <RETURN>.
- Updating of the file search rule is restricted during the start phase and in batch mode.
- The option *Show file versions* remains active until the program is ended. This dialog box may be used to turn it off explicitly.

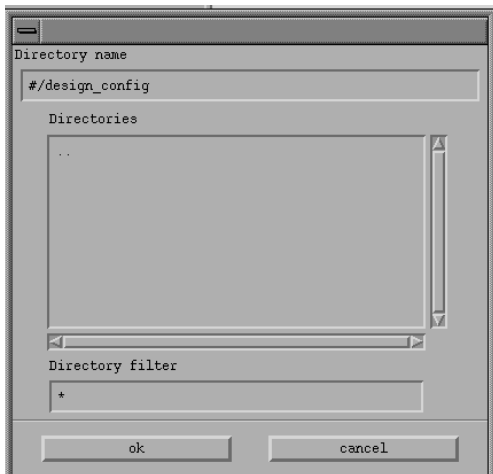
Configuring the System

17.8.2 The Standard Library "+"

In the predecessor version the standard library defined the search path of **EUKLID Design** model files to be independent from absolute pathnames. This mechanism will be replaced by the *command search rule* (see section „[File search rule \(location table\)](#)“ on page 17-25).

Nevertheless you may use the standard library furthermore and define the path as follows:

☞ Choose *Set library* in the *File* menu. The directory dialog box will be opened.



The default is the installation directory.

☞ Click on the desired directory name in the list box.



You can reduce the list by entering a search expression in the *Directory filter* field.

☞ Click on OK. The symbol "+" at the beginning of a pathname will be substituted by the specified path. "+" is a component of the file name and will be saved in the model.



You may save the standard library in a configuration file (See „[Saving the Current Configuration](#)“ on page 17-7.).

18 Plotting

EUKLID Design supports plot outputs for color PostScript (cps), black-and-white PostScript (ps), HP-GL (hpg), character-coded CGM as well as I-DEAS Picture File Format (pff).



The *pff* plot format does not support planes which are filled in color, i.e. the intersection takes place and filled planes always appear in the color of the paper.

Before direct output to a plotter is possible, the shell script `#/d23bs/plot.cmd` first has to be adapted to the customer-specific environment (see release notification). In order to output CGM format, the add-on product PLOT also has to be correctly installed and configured (see release notification for appropriate version).

Plotting

18.1 Setting Plot Properties

☞ Click on the plot icon in the set action group. The following form for inputting plot properties is opened:

The dialog box is titled 'Set Plot Properties'. It contains three main sections: 'postscript', 'cgm', and 'picturefile'. The 'postscript' section has two checkboxes: 'use postscript font' and 'postscript logo'. The 'cgm' section has two checkboxes: 'use cgm font' and 'use PLOT hatch'. The 'picturefile' section has a 'mode:' label, a 'nominal linewidth:' text field with the value '0.18', and a list of options including 'options', 'multiple', and 'color_index'. Below these sections is a 'general' section with a label 'filled planes for b/w output:', a list of options including 'background' and 'foreground', and a checkbox for 'zoom linewidth and dashes'. At the bottom are 'ok' and 'cancel' buttons.

☞ Click the buttons in front of the desired property values subject to the plot output format or enter desired property values into available text fields.

☞ Click on **OK**. The desired values are set.



If you want to accept this setting for all models or for the complete system, save the configuration file and restart **EUKLID Design** (See „[Saving the Current Configuration](#)“ on page 17-7.).

18.2 Handling Texts

Texts can either be retrieved as a font or resolved as lines and arcs (action *set_plotprops*). Perfect matching with the screen display (text length, special characters, etc.) can only be achieved with resolved texts.

Texts which cannot be mapped to specified plot fonts are always resolved (mirrored texts or texts with oblique angle).

Texts in I-DEAS Picture File Format are always resolved.

Plotting

18.3 Handling Line Thickness

The line thickness is directly linked to the line styles used or to relevant standards relating to technical drawing in mechanical engineering. **EUKLID Design** outputs the absolute line thickness into the plotfile.

The table below shows the effect of the line thickness in plot output:

		PostScript (Absolute Line Widths)	CGM (PLOT) (Relative Line Widths)	picture-file (Color index)
Line Styles	Wide	0.50	4	9
	Medium	0.35	3	8
	Narrow	0.25	2	7
Texts, Resolved	Text Height < 2.4	0.18	1	6
	$2.4 \leq \text{Text Height} <$	0.25	2	7
	$3.4 \leq \text{Text Height} <$	0.35	3	8
	Text Height ≥ 4.9	0.50	4	9
Texts, not Resolved		Approx. 0.1 Text Height; Mapped to Scalable Fonts "Hel- vetica" or "Courier"	as for Resolved Texts; Mapped to Plot Fonts HELVETIC or COU- RIER	—

When plotting via CGM (PLOT), the actual line thickness is determined by the plot configuration (nominal line width, assignment of pens). To do this, the relative line thick-

ness (see table) is multiplied by the nominal line width (from PARAM.WSTAB), and the pen which conforms most closely with this result is used. It is therefore possible to assign various line groups (e.g. 0.5/0.7) by setting up plotter queues with different configurations (see documentation on PLOT).

There are two ways of representing the line width when plotting in I-DEAS Picture File Format:

1. Multiple lines drawn with the *multiple* and *nominal line width* options
Nominal line width should correspond to the width of the plotter pen
2. Line width assigned by means of the color index
Color index option (see table)

Plotting

18.4 Handling Line Patterns

CGM (PLOT):

The type decomposition of polylines can be set by means of the SMAUSKZ entries (line type output identifier) in PARAM.WSTAB:

2	Without memory	A new line type begins with each new polyline element.
3	With memory	A line type is plotted throughout, irrespective of the boundaries of polyline elements.

SM 2, ..., 5 (line type definition) should be set as the entries for ***EUKLID Design*** (default = DIN 15; possible variant: I-DEAS Master Series-CAD-2D).

The line mode memory is active beyond the element boundaries.

Example

If consecutive lines are plotted, the line type is continued, e.g. if the type is as follows:

—..—..

Line 1	—.
Line 2	..—..
etc.	

18.5 Handling Colors

Color information is provided via the CGM output, the color PostScript output, and the HP-GL output.

CGM

The red, green and blue values currently set in **EUKLID Design** are converted to percentages and entered in the CGM file (0 – 100%). The background color is set to white.

The influence of the colors on pen selection can be controlled by means of appropriate configuration of the "PLOT" product (optional). Take note of the priorities applying to pen selection (e.g. color interpretation ahead of width interpretation etc.) and the resulting interaction with the handling of line thicknesses.

Different handling of color information (such as optionally plotting in color or black and white) can be achieved by setting up appropriately configured plotter queues. The individual queues can be controlled by additional parameters in the shell script `#/d23bs/plot.cmd`.

Further details are given in the documentation on the "PLOT" product.

PostScript

The currently set colors are used for the PostScript output. The background is always set to white, however, and the standard colors (1 – 4) to black.

I-DEAS Picture File Format

The currently set colors are used for outputs in Picture File Format, providing the line width is represented by drawing multiple lines (option *multiple lines*). The background is always set to white and the standard colors (1 – 4) to black.

If this is not the case (option *color_index*), the set colors are ignored and the line width is controlled by means of the color index.

Plotting

18.6 Settings for HP-GL

If the plot is output in the HP-GL format, the `#/d23bs/hpgl.cfg` file is read.

Lines of this file starting with the letter *p* declare pens followed by the color number, the line thickness, and the press down point.

If a line starts with *i*, this letter is followed by a plotter-specific initialization string. For this entry, refer to your plotter documentation. This entry includes some *escape sequences* to be entered by special editor commands.

Example

```
p      1      0      0.18  2
p      2      0      0.25  2
p      3      0      0.30  2
p      4      0      0.50  2
p      5      0      0.70  2
p      6      0      1.00  2
                                pressure
                                line thickness
                                color number
pen number
p: start of pen declaration
ixxx
    xxx = initialization string (plotter-specific)
i: start of initialization
```

Line thicknesses not mapped to a pen are modeled by multiple lines drawn with the next smaller pen. This may, however, result in errors when larger line thicknesses are drawn and when the true pen width strongly deviates from the line thickness specified. In order to avoid this effect, you can map different line thicknesses to the same pen (by repeating the pen entry with another line thickness) so that these lines are all drawn by the same pen.

18.7 Plotting in Batch Mode

With the aid of AQL programs it is possible to automate plotting in such a way that, for example, one plot output with the appropriate format is triggered for each drawing frame that is found in a model (for an example see *#/examples.aql/plot.aql*).

The start call

design -batch -start <AQL name>

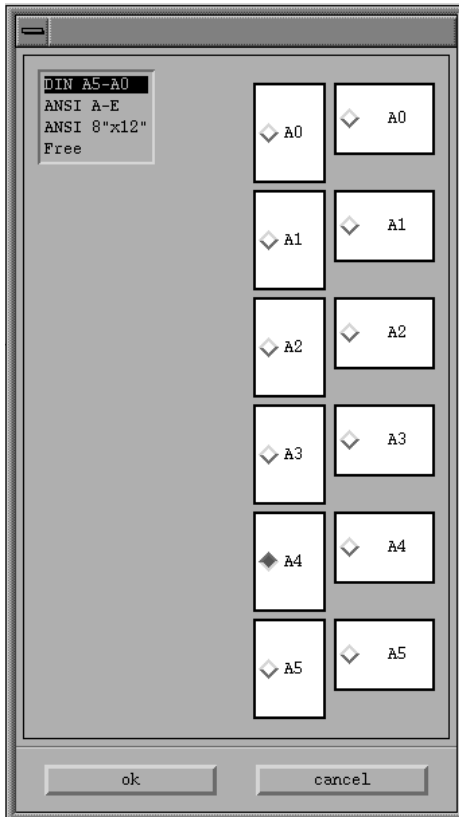
also allows corresponding programs to be executed in batch mode.

Plotting

18.8 Plot Output of Variable Extracts

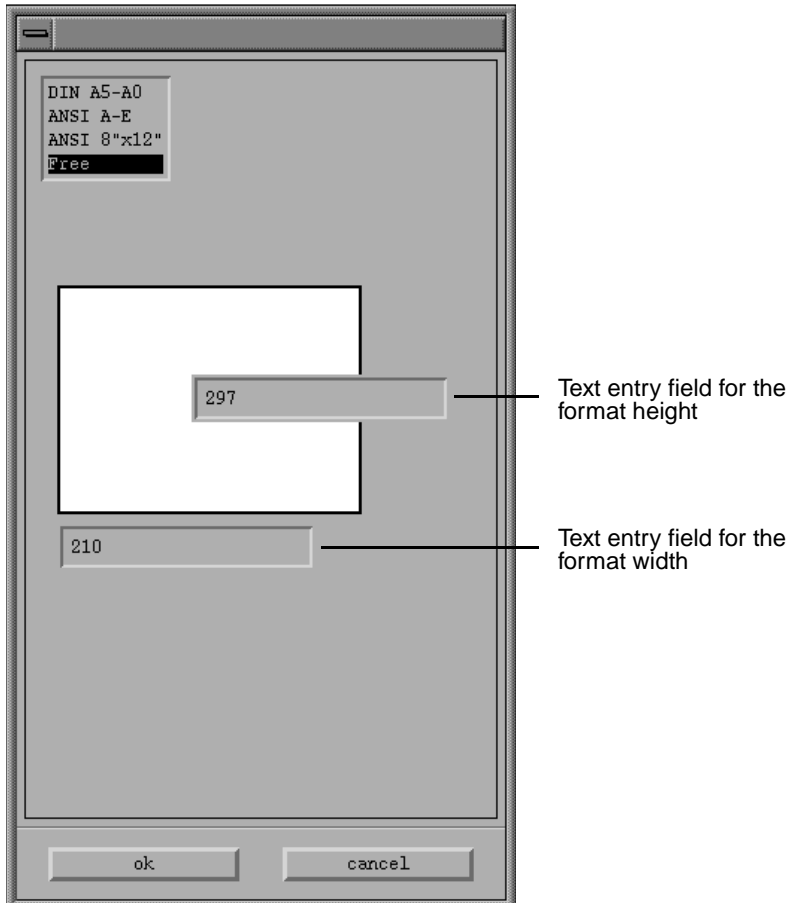
A rectangular window of any size may be defined and the contents output to a plot file or directly to the plotter. The selected rectangle area is scaled to the previously defined DIN or ANSI format or a customized format and then output accordingly.

To input the plot format, the plot format box is opened:



- ☞ Click on the desired format category in the list box.
- ☞ Click on the desired portrait or landscape field, unless you selected *free*.

If you select *Free* in the list window, the dialog box is modified:



- ☞ Enter height and width of the desired free format into the respective text entry fields.
- ☞ Click on **OK**. In the drawing area, a selection rectangle is controlled by the cursor in order to define the upper left corner of the section to be plotted.
- ☞ Click on the position in the drawing area which shall become the upper left corner of the rectangle. This position becomes the anchor point of the plot rectangle. The current cursor position indicates the lower right corner of the section to be plotted.

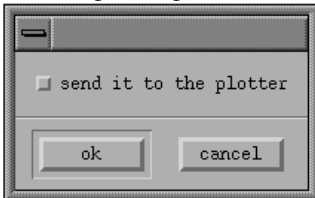
Plotting

- ☞ Click on the position in the drawing area which shall become the lower right corner of the rectangle. This completes the entry of the plot rectangular and the entry dialog box for the plot output format is opened:



- ☞ Click on the desired plot output format. The file box is opened to enable the name of the plot output file to be entered (see main chapter *Working with Models*, chapter *File Operations*)

- ☞ Input the name of the plot output file. The plot file will be created or overwritten. The following dialog box will be opened:



- ☞ Input whether the created plot file should be immediately output to the plotter or laserwriter:

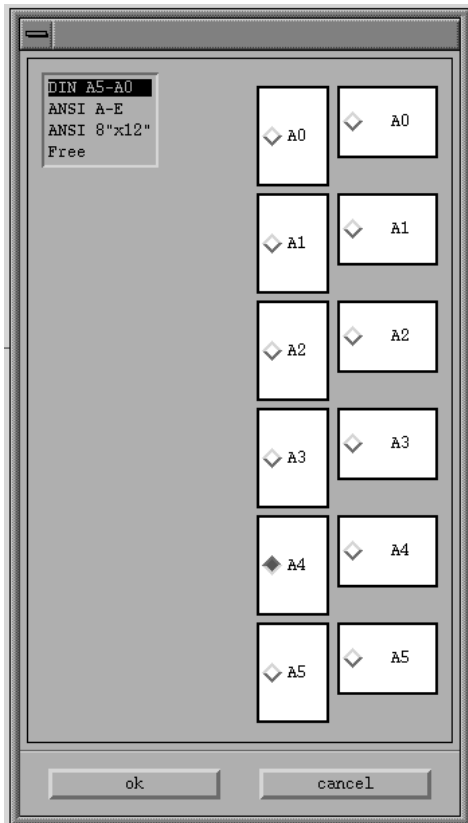
- By clicking the button *send it to the plotter*, you may input further optional parameters for output which are added as parameters to the command in the shell script.
The file `#/d23bs/plot.cmd` is called. As shell script, it includes all necessary commands for the output to the output devices to be customized for the specific user. The respective command is executed depending on the format selected.
- If you do not select *send it to the plotter*, no output is started. The plot file already created may be plotted at any future date at shell level.

- ☞ Click on *OK*.

18.9 Plot Output of Fixed Extracts

With this type of plot output, the extract is transferred 1:1 in a DIN, ANSI or customized format to the plotter. The extract is not scaled and the dimensions defined in creating the model remain intact.

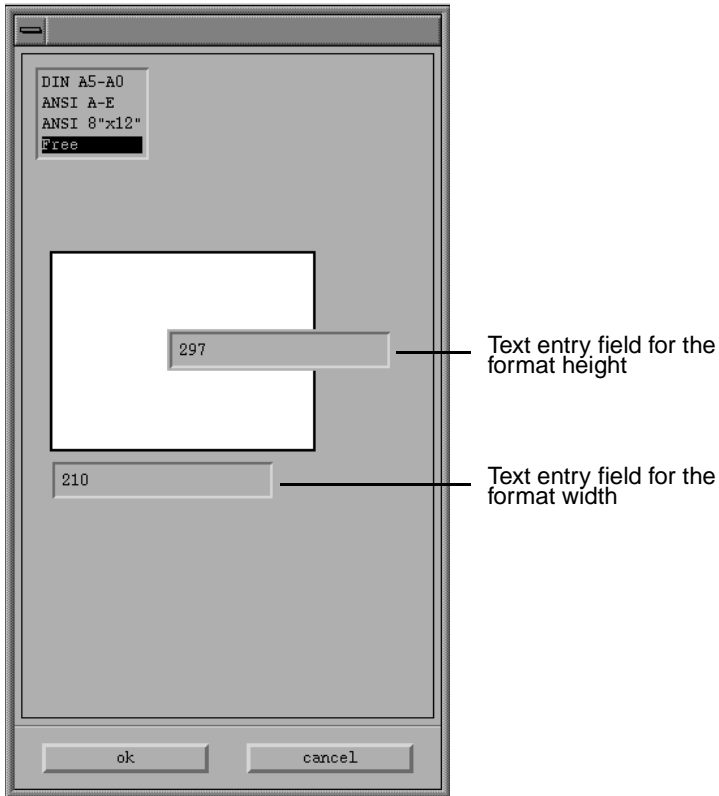
To input the plot format, the plot format box is opened:



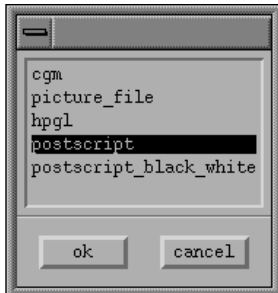
- ☞ Click on the desired format category in the list box.
- ☞ Click on the desired portrait or landscape field, unless you selected *Free*.

Plotting

If you select *Free* in the list window, the dialog box is modified:



- ☞ Enter height and width of the desired free format into the respective text entry fields.
- ☞ Click on *OK*. In the drawing area, the cursor is displayed as cross-hairs.
- ☞ Click on the position in the drawing area which shall become the upper left corner of the rectangle. The dialog box for the plot output format is opened:



- ☞ Click on the desired plot output format. The file box is opened to enable the name of the plot output file to be entered (see main chapter *Working with Models*, chapter *File Operations*)
- ☞ Input the name of the plot output file. The plot file will be created or overwritten. The following dialog box will be opened:



- ☞ Input whether the created plot file should be immediately output to the plotter or laserwriter:
 - By clicking the button *send it to the plotter*, you may input further optional parameters for output which are added as parameters to the command in the shell script.
The file `#/d23bs/plot.cmd` is called. As shell script, it includes all necessary commands for the output to the output devices to be customized for the specific user. The respective command is executed depending on the format selected.
 - If you do not select *send it to the plotter*, no output is started. The plot file already created may be plotted at any future date at shell level.
- ☞ Click on *OK*.

Plotting

19 Mixed-up 3D-2D design

19.1 SolidJoin

SolidJoin enables you to continue the 3D construction using the well-known relational 2D approach. Based on the 3D construction it reaches beyond the pure 2D documentation of 3D models. According to tasks and requirements you can do your constructional work in 2D or 3D. For example, you can create your parts in 3D and use 2D in order to add the drawings (conforming to DIN requirements, manufacturing oriented dimensioning). Further more you can create additional simple parts such as fixing brackets, sheet metal angles etc. using **EUKLID Design** in a very efficient way. You can also sketch your construction ideas using the proven methods.

SolidJoin imports PARASOLID models from 3D-CAD systems, e.g. from SolidWorks. Based on these 3D models you can continue your work in the 2D system independent of the hardware platform.

2D drawing views derived from 3D Parasolid models are represented by 2D objects which can be a basis for further relational 2D construction such as traditionally created **EUKLID Design** objects. Modifications of the 3D model will not only update the derived 2D drawing views automatically but also the 2D constructions which have been created based on the drawing views. This enables a mixed 3D-2D construction, almost uniquely in its way, according to tasks and requirements.

A very interesting benefit is that **EUKLID Design** can enter results of 3D volume calculations into formula variables and use them for its 2D construction via a material table. Using additional 3D part data an automatic bill-of-material generation can be performed including exact weight information.

Mixed-up 3D-2D design

19.1.1 Procedure

The following chapter explains the procedure in principle.

- **3D construction**

Please construct your part or assembly within your 3D volume modeler, e.g. SolidWorks.

- **Saving a 3D model**

Please save your model as a PARASOLID model.

- **Opening a 3D model in *EUKLID Design***

Please open the PARASOLID model with the input action in the 3D action group and locate it inside a 3D viewport. Each 3D part will be represented as UDO. For each part or assembly the ***EUKLID Design*** model stores the information as to which 3D model it originally belonged to. Should modifications to a part be necessary you can identify the original 3D model in order to perform the modification.

- **Creating drawing views and section views in *EUKLID Design***

Please generate the required 3D views and 3D sections from these views into a 2D viewport. They will be created as 2D geometry in relation to the corresponding part (as effect objects of the action).

- **2D construction on 3D parts in *EUKLID Design***

Please add just as usual further 2D relational geometry to the drawing views desired from 3D, e.g. further parts or manufacturing oriented drawings conforming to DIN requirements.

- **Modifications to the 3D model (Update)**

If you change something in an existing 3D model later you can save it again (into the same file) and update your ***EUKLID Design*** model — you can do this during an ***EUKLID Design*** session as well as while opening an ***EUKLID Design*** model. The add-on construction will remain complete, provided that the referenced objects continue to exist.

Note: In the case of a topological change the add-on construction might lose its relation (partially) and will not be updated automatically with the 3D drawing views. ***EUKLID Design*** would display a message on the screen.

- **Saving the *EUKLID Design* model**

Please save your *EUKLID Design* model as usual. A backup of the PARASOLID model will also be saved as an embedded model. You may recreate an external PARASOLID model from the internal embedded one.

19.1.2 Additional functionality

In this paragraph further important functions of SolidJoin are explained.

- **Locator**

The location and orientation of parts in the 3D space are saved in an special object. This displayed on screen as a three-dimensional coordinate system.

- **Sketch planes**

A sketch plane is an infinite two dimensional plane in 3D space. In the center of each sketch plane *EUKLID Design* automatically creates a 3D locator object defining the default axis directions x,y,z.

You might use sketch planes in order to create 2D construction geometry and to position 3D parts. 2D constructions in sketch planes may be the basis for positioning 3D parts. So you may wish to access on intersection points in order to locate a 3D part on it.

Creating a drawing view means creating another sketch plane automatically which will be displayed in the 2D viewport as coordinate system as well as in the 3D structure viewport.

After the program has been started with default settings you will see a 3D locator object with a sketch plane ("xy_plane") attached, which corresponds to the 2D drawing area with the coordinate system located in the origin.

Sketch planes are displayed in the 3D viewport as a grid (active) or as a little circle symbol (inactive), in the 3D structure viewport as structure element with green (active) or blue color (inactive).

Mixed-up 3D-2D design

Only **one** sketch plane can be active. Only constructions of the active x-y sketch plane ("xy_plane") will be displayed in the 2D viewport. 2D constructions of other sketch planes are displayed in the 3D viewport only if the according sketch plane is active.

- **3D viewing**

If your cursor is positioned in the 3D viewport the right mouse button offers you some special actions controlling the display, e.g. dynamic rotation and zooming.

- **Hidden lines**

In a 3D viewport you can display your 3D parts in wireframe or hidden line mode.

- **Structure viewport with 3D parts**

In a 3D structure viewport the logical structure of your parts can be displayed (in the same way as in 2D structure viewport). You can pick objects either in 3D structure viewport or in 3D viewport. Using the right mouse button, some special actions to control the display or modification of object properties are offered.

- **Assemblies**

You can read assemblies from your PARASOLID model. For each part you can derive 2D drawing views.

19.2 I-DEAS link

To link ***EUKLID Design*** and I-DEAS Master Series the following customization options and functions are available:

- Configuration of data transfer from I-DEAS Drafting Setup
- ***EUKLID Design*** as a 2D application from the I-DEAS Master Series
- Explicit data transfer from I-DEAS Drafting Setup
- Explicit data transfer to I-DEAS Master Modeler

Mixed-up 3D-2D design

19.2.1 Configuration of data transfer from I-DEAS Drafting Setup

Data transfer from I-DEAS Drafting Setup to **EUKLID Design** may be configured.

These settings affect

- automatic data transfer from Drafting Setup to **EUKLID Design** as well as
- direct loading of the *Solid Link Universal File* drawing layout file.

19.2.1.1 General

Configuration parameters may be edited by a user defined action (UDA). The configuration parameter values are stored with the model. The currently valid configuration parameter values are used for data transfer from I-DEAS to **EUKLID Design**.

In particular, the user can control the degree to which objects originating from I-DEAS Drafting Setup can be modified in **EUKLID Design** without having such changes revoked during the next transfer. It is thus possible to satisfy the demands of both 2D- and 3D-oriented users.

19.2.1.2 Configuration parameter values as user attributes

User attributes are defined for the individual configuration parameters and attached to the object *top.globals_list.first*, which exists only once for each model.

```
ref_object= top.globals_list.first
create_attr ( ref_object, "ID_DRAW" )
create_attr ( ref_object, "ID_DSPLINE" )
create_attr ( ref_object, "ID_CENTERLINE" )
create_attr ( ref_object, "ID_MEASURE" )
create_attr ( ref_object, "ID_LINestyle" )
create_attr ( ref_object, "ID_TEXT" )
create_attr ( ref_object, "ID_SYMBOL" )
create_attr ( ref_object, "ID_VIEW" )
create_attr ( ref_object, "ID_DELETE" )
```


19.2.1.3 Configuration parameter settings

- ID_DRAW
controls the representation of the **drawing frame**

Attribute name	Value	Meaning
ID_DRAW	0	Drawing frame is not produced
ID_DRAW	1	EUKLID Design drawing frame is produced
ID_DRAW	2	I-DEAS drawing frame is adopted Transferred only once, then all changes are maintained.
ID_DRAW	3	Current I-DEAS drawing frame is adopted
Default value: ID_DRAW 0		

Notes

- Deleting existing drawing frames
Drawing frames can generally be deleted by deleting the layer with the ending *_draw*. Thereafter, the setting of configuration parameter *ID_DRAW* is in effect.
- **EUKLID Design** drawing frame
If you use the **EUKLID Design** drawing frame (*ID_DRAW* 1), then a drawing frame is used which corresponds to the I-DEAS Drafting Setup drawing format. However, no objects are transferred.
- Toggling the drawing frame
In order to toggle from the I-DEAS drawing frame to the **EUKLID Design** drawing frame or vice versa, you must first unlock the layer of the **EUKLID Design** model ending in *_draw* and delete the entire layer with all objects contained therein.
In order to toggle from the **EUKLID Design** drawing frame to the I-DEAS Drafting Setup drawing frame, you must transfer the entire drawing layout during the ensuing transition.

Mixed-up 3D-2D design

- ID_DSPLINE

controls whether splines are transferred as **splines** or as **polylines**.

<i>Attribute name</i>	<i>Value</i>	<i>Definition</i>
ID_DSPLINE	0	Splines in spline representation
ID_DSPLINE	1	Splines as polygonal sequence
Default value: ID_DSPLINE 0		

Splines are transferred as contours in polygonized form. The contours consist of line segments.

- ID_CENTERLINE

controls whether **centerlines** are transferred with or without **overshoot**.

<i>Attribute name</i>	<i>Value</i>	<i>Definition</i>
ID_CENTERLINE	0	Centerlines without overshoot
ID_CENTERLINE	1	Centerlines with overshoot
Default value: ID_CENTERLINE 0		

- ID_MEASURE

controls **editability of dimensions**.

<i>Attribute name</i>	<i>Value</i>	<i>Definition</i>
ID_MEASURE	0	Always adopted by I-DEAS
ID_MEASURE	1	Edited positions of dimension and height of text dimension are maintained
ID_MEASURE	2	All changes from EUKLID Design are maintained.
Default value: ID_MEASURE 1		

- ID_LINestyle
controls editability of **line style** for geometric objects.

Attribute name	Value	Definition
ID_LINestyle	0	Line style is adopted by I-DEAS
ID_LINestyle	1	Changes made by user in EUKLID Design are maintained.
Default value: ID_LINestyle 0		

- ID_TEXT
controls **editability of text**.

Attribute name	Value	Definition
ID_TEXT	0	Always adopted from I-DEAS
ID_TEXT	1	Only text content is adopted from I-DEAS
ID_TEXT	2	All changes are maintained
Default value: ID_TEXT 0		

- ID_SYMBOL
controls **editability of symbols**.

Attribute name	Value	Definition
ID_SYMBOL	0	Always adopted from I-DEAS: Create all symbols anew.
ID_SYMBOL	1	Only symbol content is adopted from I-DEAS: Currently affects only symbol balloons.
ID_SYMBOL	2	All changes are maintained
Default value: ID_SYMBOL 0		

Mixed-up 3D-2D design

ID_SYMBOL 1 affects only on symbol balloons: the text content is adopted from I-DEAS Drafting Setup. For all other symbols such as geometric tolerances and objects such as hatching, the following applies:

- either: Delete and create again (*ID_SYMBOL 0*, *ID_SYMBOL 1*)
- or: All changes are maintained (*ID_SYMBOL 2*).

- **ID_VIEW**
controls **editability of position and scaling of views**

Attribute name	Value	Definition
ID_VIEW	0	Position and scaling factor are always adopted from I-DEAS
ID_VIEW	1	All changes are maintained
Default setting: ID_VIEW 1		

The position of views is defined by a coordinate system in **EUKLID Design**. You may change their position by editing the origin of the corresponding coordinate system.

You may use the program *ansipos.aql* in directory *<I_DEAS installation directory>/eng/design/examples/aql* to facilitate positioning of layoutviews in **EUKLID Design**.

- **ID_DELETE**
controls how deleted objects are treated.

Attribute name	Value	Definition
ID_DELETE	0	Makes deleted objects visible again.
ID_DELETE	1	EUKLID Design changes are maintained.
Default value: ID_DELETE 0		

This parameter affects transferred objects which were set to *deleted* by further manipulation in **EUKLID Design**.

Example

Chamfering and rounding

Lines originating in I-DEAS are chamfered or rounded and also set to *deleted*. However, they are not removed from the data structure, since other objects (the newly-created chamfer or rounding and nearby lines) are dependent on them.

Deleted I-DEAS objects are not created again if the *ID_DELETE* parameter has been set to 1. Otherwise, the objects set to deleted are always created again following data transfer from I-DEAS to **EUKLID Design**.

19.2.1.4 Input mask

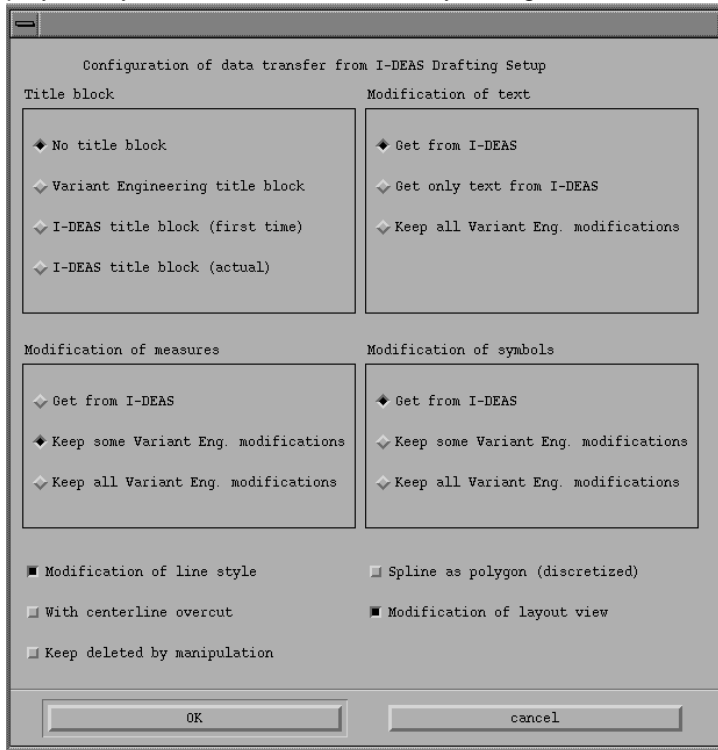
1. Use the following icons to call the action *ideas_config*:



2. Edit the configuration parameters as desired.
For example, in order to maintain the I-DEAS drawing frame during the first transition, click onto the point *I-DEAS title block (first time)* within the frame *Title block*.
This sets the value of the configuration parameter *ID_DRAW* to 2.
3. Confirm the action with *OK* if you wish to accept the edited parameters.

Mixed-up 3D-2D design

The next time this action is called, the current configuration parameters are displayed. If you do not wish to make any changes, abort the action with *cancel*.



Configuration of data transfer from I-DEAS Drafting Setup

Title block	Modification of text
<input checked="" type="radio"/> No title block	<input checked="" type="radio"/> Get from I-DEAS
<input type="radio"/> Variant Engineering title block	<input type="radio"/> Get only text from I-DEAS
<input type="radio"/> I-DEAS title block (first time)	<input type="radio"/> Keep all Variant Eng. modifications
<input type="radio"/> I-DEAS title block (actual)	

Modification of measures	Modification of symbols
<input type="radio"/> Get from I-DEAS	<input checked="" type="radio"/> Get from I-DEAS
<input checked="" type="radio"/> Keep some Variant Eng. modifications	<input type="radio"/> Keep some Variant Eng. modifications
<input type="radio"/> Keep all Variant Eng. modifications	<input type="radio"/> Keep all Variant Eng. modifications

☒ Modification of line style ☐ Spline as polygon (discretized)

☐ With centerline overcut ☒ Modification of layout view

☐ Keep deleted by manipulation

OK cancel

19.2.1.5 AQL program for editing configuration parameters

The configuration parameters may also be edited using an AQL program.

Example

```
ideas_config ({
  { "draw", application.v_int, 0 },
  { "dspline", application.v_int, 0 },
  { "center", application.v_int, 0 },
  { "measure", application.v_int, 1 },
  { "lstyle", application.v_int, 1 },
  { "text", application.v_int, 0 },
  { "symbol", application.v_int, 0 },
  { "view", application.v_int, 1 },
  { "keep_del", application.v_int, 0 } })
```

Definition

The expression { "draw", application.v_int, 0 }

- defines the user attribute *ID_DRAW* (if it does not already exist) for the reference object *ref_object= top.globals_list.first* and
- assigns the value 0 to the attribute.
ID_DRAW controls the representation of the drawing frame. The value 0 specifies that no drawing frame is created.

The following other expressions are handled in similar manner:

Action parameter	User attribute
draw	ID_DRAW
dspline	ID_DSPLINE
center	ID_CENTERLINE
measure	ID_MEASURE
lstyle	ID_LINESTYLE
text	ID_TEXT

Mixed-up 3D-2D design

<i>Action parameter</i>	<i>User attribute</i>
symbol	ID_SYMBOL
view	ID_VIEW
keep_del	ID_DELETE

19.2.2 *EUKLID Design* as a 2D component of I-DEAS Master Series

EUKLID Design allows associative transfer of solid models from the I-DEAS Master Series for further processing. Key dimensions of the I-DEAS solid can be changed in ***EUKLID Design***.

Installation note

EUKLID Design must be installed as a 2D component of I-DEAS Master Series.

19.2.2.1 Procedure for creating and editing a drawing

You can use ***EUKLID Design*** as follows to complete drawings of I-DEAS models:

1. Create a part or an assembly in I-DEAS Master Series.

Examples:

- Definition of a contour
 - Dragging and rotation of contours
 - Addition of features
 - Definition of key dimensions, geometric tolerances
 - Naming of parts
2. Produce the drawing layout in I-DEAS Drafting Setup:
 - Creation of several views, sections, details
 - Placement of information dimensions
 3. Start the transfer. Select the action *Variant Engineering...*:
 - ***EUKLID Design*** starts.
 - Drawing elements are automatically transferred to ***EUKLID Design***.
I-DEAS data management administers the ***EUKLID Design*** model file. A model file is created during the first transition; otherwise an existing one is updated.

4. Make the additional changes in **EUKLID Design**.
5. If necessary, use the action *ideas_modify* to edit the key dimensions:
 - Select the key dimensions in **EUKLID Design**.
 - Enter the new values in the temporarily activated I-DEAS Master Modeler.
 - **EUKLID Design** is automatically activated and the **EUKLID Design** model is automatically updated.
6. Exit **EUKLID Design** using *STOP*. I-DEAS is then automatically reactivated.

19.2.2.2 Transferring a model from I-DEAS Drafting Setup

The transition from I-DEAS Master Series to **EUKLID Design** is performed automatically when the action *Variant Engineering...* of the application I-DEAS Drafting Setup is called. This loads existing **EUKLID Design** models and brings them up to date.

Associativity

The drawing objects created in I-DEAS Drafting Setup are associative to the 3D model and are transferred to **EUKLID Design**. This brings with it the following implications:

- If the 3D model is edited, then the 2D drawing objects in Drafting Setup and their resulting 2D objects in **EUKLID Design** are changed.
- The objects produced in **EUKLID Design** are not linked, but instead are transferred independently of one another. This means for example that dimensions, hatching, geometric tolerances are separate objects and are not related to transferred geometric objects such as points, lines, etc.
- In **EUKLID Design** you can define new relations for transferred objects, e.g. *Line parallel to line* (extracted from 3D). When the 3D model is edited and then transferred to **EUKLID Design**, two possibilities arise:
 - The 3D object previously transferred remains unchanged or is modified: all relations in **EUKLID Design** dependent upon extracted objects remain valid. The changes are recreated. The model can be completely recalculated.
 - The 3D object previously transferred is deleted: all relations in **EUKLID Design** dependent upon extracted objects can no longer be definitively recalculated. In this case, the user must intervene; the system offers support by depositing problematic cases into a special layer with the ending *_error*.

Mixed-up 3D-2D design

Restriction

The associativity of objects from ***EUKLID Design*** to the I-DEAS 3D model is restricted. Editing the design of the 3D model can lead to deletion of "old" 2D drawing objects and replacement with "new" ones. Especially with models created by the I-DEAS Master Series sheet metal application, a change in geometry often leads to deletion and subsequent creation of geometrically identical objects.

Transferred objects

Transferred objects are marked with an identifier by I-DEAS. Geometrically identical objects can now be marked with a new identifier during recalculation of the 3D model, i.e. the objects with the old identifiers are deleted and new objects with new identifiers created in their place.

If relations are then defined for such objects, then the referring objects are moved into the `_error` layer. The user must then perform manual correction.

All information relevant to drawing completion are transferred to ***EUKLID Design***:

General drawing data

- Name of I-DEAS model
- Units
- Name of individual part or assembly
- Name of individual views and sections

Geometric objects

- Point
- Line
- Circular arc
- Circle
- Ellipse
- Spline

Dimension, geometric tolerances, text, hatching

- Dimensions
- Text
- Symbol balloons
- Geometric tolerances
- Hatching
- Section lines

Drawing frames

There are four possible alternatives (see chapter *Configuration of data transfer from I-DEAS Drafting Setup*):

- Do not produce a drawing frame
- Replace with **EUKLID Design** drawing frame
- 1:1 transfer of I-DEAS drawing frame as geometry and text objects: this occurs only the first time; subsequent modifications in **EUKLID Design** are retained.
- 1:1 transfer of the current I-DEAS drawing frame as geometry and text objects

Restrictions

In the following cases, exactly mapping of I-DEAS objects to **EUKLID Design** is not possible:

- Spline
Splines are represented internally in a different manner in **EUKLID Design** and I-DEAS. This leads to a different spline representation in the extracted **EUKLID Design** model. Especially when visible edges intersect with splines, gaps may be visible in the model.
Rational splines can be approximately represented in **EUKLID Design** only by non rational splines, if they cannot be converted to analytic objects.
Suggestion: transfer the splines as a polygonal sequence by setting the configuration parameter *ID_DSPLINE 1* (see chapter *Configuration of data transfer from I-DEAS Drafting Setup*).
- Symbol balloons
Contain no further bill of material information.

Mixed-up 3D-2D design

- Hatching

Hatching is transferred as a separate object without references to other geometric objects. The boundary elements are partially polygonized and made invisible. This means that the hatching can extend somewhat beyond the visible geometric object in some places.

Hatching in detail views

Detail views describe only a portion of a drawing. If the portion contains hatching, then this is covered by a hiding plane such that the hatching is visible only in the actual detail view.

- Moving a detail view

After a detail view is moved or the scaling factor is edited, hatching is not displayed correctly until transfer is repeated.

- Hatching extending beyond the drawing frame

If detail view hatching is to extend far beyond the drawing frame, it is sometimes not covered by the hiding plane.

Workaround:

Enlarge the hiding plane by editing the lower left and/or upper right corner of the external contour.

Default coordinates of the lower left corner:

$x = -\text{VAR_ID_global_dDrawHor}/2$

$y = -\text{VAR_ID_global_dDrawVer}/2$

upper right corner:

$x = \text{VAR_ID_global_dDrawHor} * 1.5$

$y = \text{VAR_ID_global_dDrawVer} * 1.5$

where:

$\text{VAR_ID_global_dDrawHor}$ = width of drawing

$\text{VAR_ID_global_dDrawVer}$ = height of drawing

- Text

Text and text blocks may be displayed differently in **EUKLID Design**:

Boxes: Text and text blocks are displayed without boxes in **EUKLID Design**.

Fonts: Different fonts are used.

Text blocks with arrows: Text blocks with arrows are split into two objects in **EUKLID Design**:

- *Symbol* object for arrow

- *Text* object for text

Texts are still manipulated in order to rotate them.

Text as balloon becomes a symbol balloon in **EUKLID Design**.

- Dimensioning
 - Dimensions are not displayed in different units (e.g. in mm and in inches) at the same time.
 - Dimension tolerances are displayed as absolute values. These become tolerances in **EUKLID Design**.
 - Dimetric dimensions are not always transferred. The names of non-transferred dimensions appear in the background window (pad).
- Symbols

In **EUKLID Design**, not all additional informations are transferred from I-DEAS. The specification (M) for maximum material condition cannot be represented by the *symbol/object* in **EUKLID Design**.



If errors occur during transfer, they are collected in a file which is announced following transfer. The following error message may occasionally appear but can be ignored:

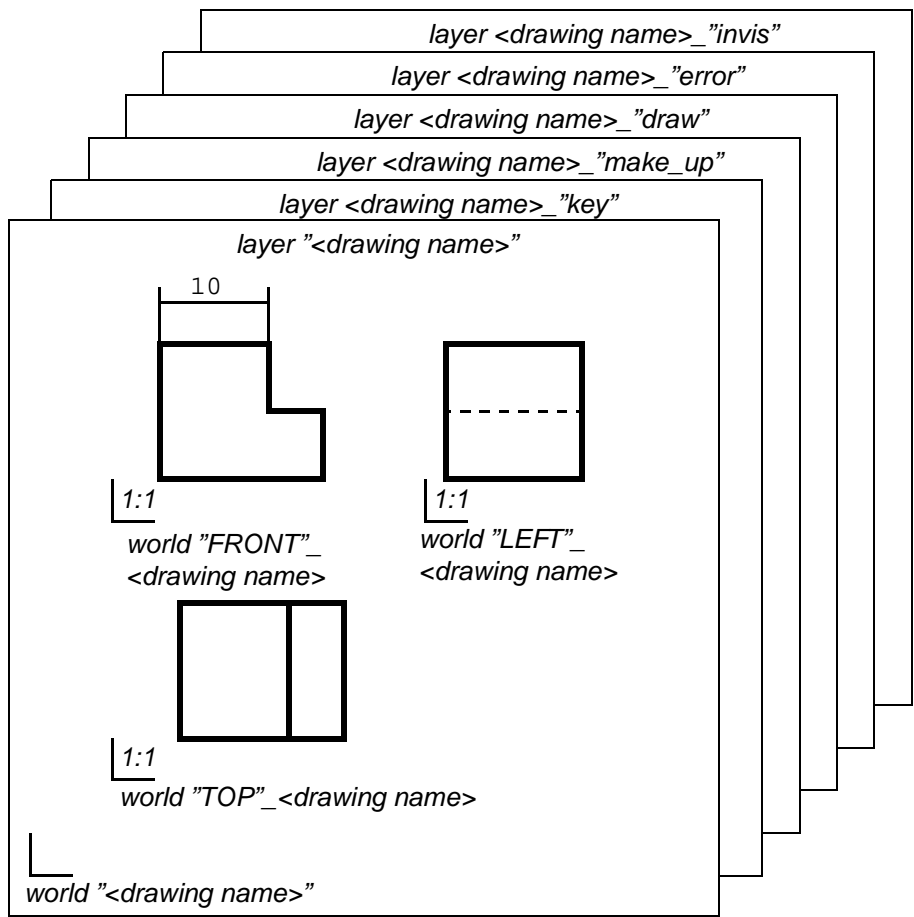
"Error%%Beginning and end point are identical%"

Mixed-up 3D-2D design

General model structure following transfer

The I-DEAS drawing layout is converted to an **EUKLID Design** data structure. The primary structural elements are layers and coordinate systems.

The following graphic illustrates the general model structure in **EUKLID Design** following transfer.



Layer

- Names of layers and coordinate systems
The name length is limited to the system-specific value for file name length (see *Open Architecture User Guide: Value = program.filename_size*). If the name exceeds this limit, it will be truncated to the maximum allowable length (27 characters).
- The name of the top layer is identical to the **EUKLID Design** model name.
- Below the top layer, a <drawing_name>layer is defined with the name of the I-DEAS layout. All geometric objects are generated in this layer. They are selectable after the transfer, but may not be edited (*locked* mode).
- Below the <drawing_name>, five layers are defined for all non-geometric objects (for dimension, geometric tolerances, bill of material, text, drawing frame) and for drawing frame objects:

Key dimensions	Layer with ending <i>_key</i>
All other objects except geometry and key dimensions	Layer with ending <i>_make_up</i>
Drawing frame objects	Layer with ending <i>_draw</i>
After editing in Master Solid	
non-newly calculated objects	Layer with ending <i>_error</i>
Invisible objects	Layer with ending <i>_invis</i>
- Transferred objects can be made invisible. All objects in the layers set to invisible, with the ending *_invis*, are invisible. If objects are to remain invisible following the drawing update, then you must move the objects into this layer.

Coordinate systems

- The names of coordinate systems are generated automatically and truncated if necessary.
- A separate coordinate system is defined for each view transferred (front view, top view, etc.).
- The coordinate systems of the individual view are all defined relative to the origin of the coordinate system <drawing_name>.
- All geometric objects within a view are defined relative to their coordinate system origin.

Mixed-up 3D-2D design

Editability

All extracted objects are located in locked layers. This prevents the objects from being inadvertently changed or deleted.

If transferred objects are to be modified and these changes are to be retained, then you must adjust the corresponding configuration parameters. The layer locking must be removed before the transferred objects can be edited.

19.2.2.3 ideas_modify

The **EUKLID Design** action *ideas_modify* allows you to edit the key dimensions of the 3D model.

Key dimensions (in the layer with the ending *_key*) are thus **bidirectional**, provided that **EUKLID Design** is installed as a 2D application of I-DEAS and is started for drawing completion using the action *Variant Engineering...* from I-DEAS Drafting Setup.

1. The action *ideas_modify* is called using the following icons:



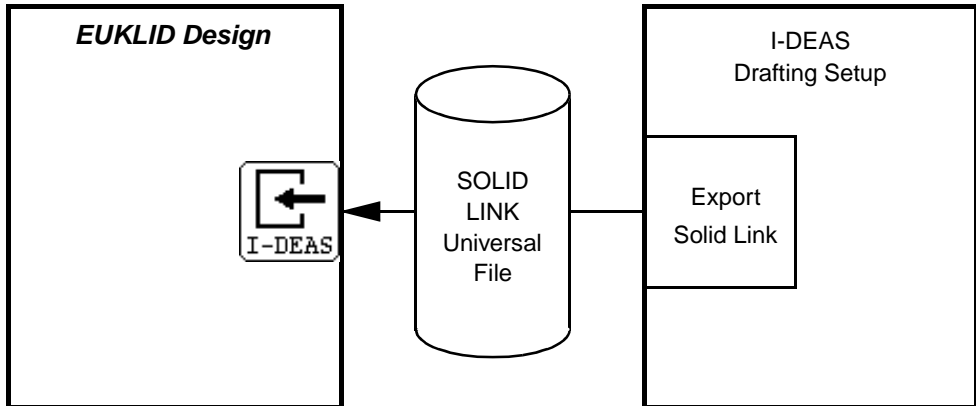
From **EUKLID Design** you may initiate a modification and recalculation of the 3D model:

2. Select the key dimension to be changed.
The dimensions are displayed as follows:
key dimensioncolor with color index 24 (default: green)
selected dimensionselection color (default: red)
3. Conclude the action.
The temporarily activated 3D process requires input of new values for key dimensions.
4. Enter the new values
The 3D model is recalculated and extracted in **EUKLID Design** and the **EUKLID Design** model is updated. **EUKLID Design** is then reactivated.

19.2.3 Explicit data transfer from I-DEAS to *EUKLID Design*

Data transfer from I-DEAS Drafting Setup to *EUKLID Design* is possible via a file.

Thus all information from the Master Series *Drafting Setup* application required for drawing completion can be transferred to *EUKLID Design* via an external file.

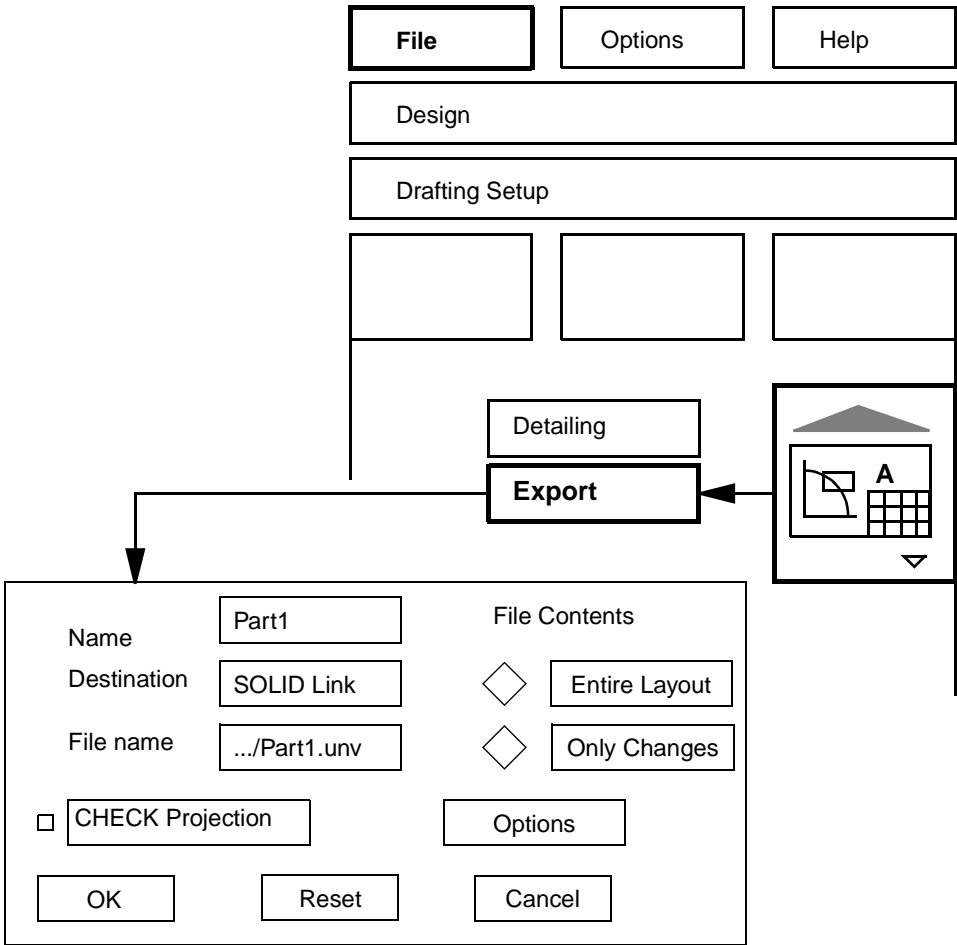


Mixed-up 3D-2D design

The following steps outline the procedure for data transfer:

I-DEAS

Generation of transfer file in Drafting Setup: SOLID Link Universal File



1. The action *Read transfer file of type I-DEAS Solid Link Universal File* can be called using the following icons:



EUKLID Design reads a file generated from the I-DEAS Master Series component *Drafting Setup* and produces an **EUKLID Design** model for further drawing completion.

2. Select the name of the transfer file from the file selection box. The transfer file must end in *unv* and be of the type *I-DEAS Solid Link Universal File*.

The drawing layout objects to be transferred with the transfer file are read.

During the initial transfer, original objects are generated. When transfer is repeated, the drawing layout is updated.

During transfer, the settings take effect which you made during configuration of data transfer from I-DEAS Drafting Setup (see chapter *Configuration of data transfer from I-DEAS Drafting Setup*).

The general composition of the resulting extracted **EUKLID Design** model corresponds to the drawing composition as described in the chapter **EUKLID Design as a 2D component of I-DEAS Master Series**.



Both file types *Solid Link Universal File* and *Design Universal File* use the ending *unv*, but each has a completely different composition and definition.

Mixed-up 3D-2D design

19.2.4 Explicit data transfer to I-DEAS Master Modeler

You may transfer data from ***EUKLID Design*** to I-DEAS Master Modeler via three different methods, each of which produces transfer files, but with different contents each time.

- Generation of a transfer file of type *Design Universal File–Data set wire frame*
- Generation of a transfer file of type *Design Universal File–Data set Profile*
- Generation of a transfer file of type *Program File*

Files of type I-DEAS *Design Universal File* are upwardly compatible and produce either a wire frame model or a profile in I-DEAS Master Modeler.

In contrast, the upwardly non-compatible I-DEAS *Program File* type may be used to produce geometric objects as if they had just been generated interactively.

19.2.4.1 Design Universal File (wire frame model)

ideas_outputunvw

Generates a transfer file of type *Design Universal File*- data set wire frame model—for I-DEAS Master Modeler.

File
name

Name of the transfer file via the file selection box.
Default value: *transferw.unv*.

Name

Name of wire frame model. If you do not specify a name, the object will be named according to the following scheme:
first wire frame model: WIREFRAME1
second wire frame model: WIREFRAME2
etc.

color

Color of elements in I-DEAS Master Modeler

Example

USE ACTUAL
0 - BLACK
1 - BLUE
2 - GRAY BLUE
3 - LIGHT BLUE
4 - CYAN
5 - DARK OLIVE
6 - DARL GREEN
7 - GREEN
8 - YELLOW
9 - ORANGE GOLD
10 - ORANGE
11 - RED
12 - MAGENTA
13 - LIGHT MAGENTA
14 - PINK
15 - WHITE

Default value *USE ACTUAL*: This gives all geometric objects the color *GREEN* (= Default setting from I-DEAS Master Modeler).

Mixed-up 3D-2D design



Selection group

Selectable elements are **points, lines, circles, circular arcs, contours, surfaces** and **groups**. Only the objects point, line, circle and circular arc are transferred, however. Only the calculated boundary objects (lines, circular arcs, circles) of contours and planes are transferred.

>From groups, the geometric objects contained therein are transferred. If this parameter is confirmed with *OK &Name*, then a group is generated which contains the selected objects. If confirmed only with "OK", a group is not generated.

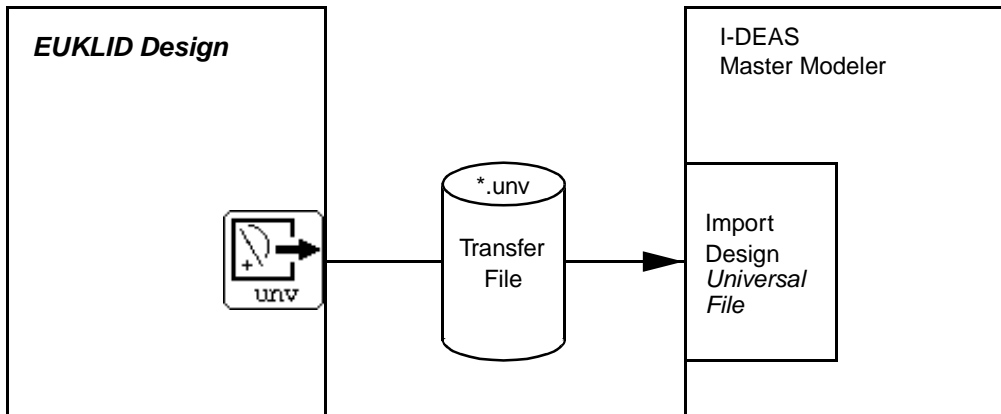
Data transfer

Data transfer from **EUKLID Design** to I-DEAS is performed via a transfer file of type *Design Universal File* with the suffix *.unv*. The contents of the transfer file consist of data sets of type *Wireframe*.

A *WIREFRAME* is a group or collection of geometric objects. The selected elements may intersect.

The following steps are required to perform a data transfer:

1. In **EUKLID Design**: Generate transfer file
2. In I-DEAS Master Modeler: Read transfer file



The transfer is performed according to the appropriate sequence selected. A sequence consists of:

- Name of wire frame model
- Color
- Group

Per sequence, the objects in an I-DEAS Master Modeler wire frame model (*WIREFRAME*) are mapped with corresponding name and color.

Mixed-up 3D-2D design

If objects belong to different coordinate systems, they are then coordinated with the appropriate different wire frame models. The name is then extended by one digit (example: wire --> wire_1, wire_2 for two coordinate systems).

In **EUKLID Design** a user attribute, *in_IDEAS_WIRE*, is attached to the groups- which were either selected or just generated (parameter group was confirmed with *OK & Name*)—along with date and time of transfer initiation. With repeated transfer to I-DEAS only date and time (attribute value) are updated.

The existence of the attribute in the groups can be determined using the following AQL expression: ... *valid(<element>.user_in_IDEAS_WIRE)*....

The attribute value (date and time of the most recent transfer call) is returned by the following AQL expression: [*<element>.user_in_IDEAS_WIRE*].

Application and recommendations for further use in I-DEAS

If the transfer file is read into I-DEAS Master Modeler with the command *Import - Design Universal File*, then wire frame models are generated.

The objects in a *WIREFRAME* may now be used to generate profiles.



- Different files with identical suffix
Files of type *Solid Link Universal File* and *Design Universal File* both have the same suffix, *unv*. However, each has a completely different structure and a different meaning.
- Action *ideas_outputunvp*
This action allows transfer of profiles (*planes*) to I-DEAS Master Modeler which may be used directly for generation of a swept or lifted solid.
- Objects in coordinate systems with non-unity scaling factors
If objects lie within coordinate systems having a scaling factor other than one, then they are displayed in graphically modified form, i.e. either larger or smaller.
During data transfer from **EUKLID Design** to an I-DEAS *Design Universal File*, the objects are transferred in their proper (actual) size. In order to obtain the proper visual impression, you should set the scaling factor to 1:1 before transferring to the I-DEAS *Design Universal File*.

19.2.4.2 Design Universal File (profile)

ideas_outputunvp

Generates a transfer file of type *Design Universal File- Data set profile* for I-DEAS Master Modeler.

File
name

Name of the transfer file via file selection box.
Default value: *transferp.unv*.

Name

Name of profile. If you have not specified a name, the objects are named according to the following scheme:
first wire frame:PROFILE1
second wire frame:PROFILE2
etc.

color

Color of elements in I-DEAS Master Modeler

Example

USE ACTUAL
0 - BLACK
1 - BLUE
2 - GRAY BLUE
3 - LIGHT BLUE
4 - CYAN
5 - DARK OLIVE
6 - DARL GREEN
7 - GREEN
8 - YELLOW
9 - ORANGE GOLD
10 - ORANGE
11 - RED
12 - MAGENTA
13 - LIGHT MAGENTA
14 - PINK
15 - WHITE

Default value *USE ACTUAL*: This gives all geometric objects the color *GREEN* (= Default setting from I-DEAS Master Modeler).

Mixed-up 3D-2D design



pl: Object to be transferred *plane*

The transfer is performed according to the appropriate sequence selected. A sequence consists of:

- ☐ Name of wire frame model
- ☐ Color
- ☐ Plane.

Data transfer

Data transfer from **EUKLID Design** to I-DEAS is performed via a transfer file of type *Design Universal File* with the ending *.unv*. The contents of the transfer file consist of data sets of type *Profile*.

This action allows transfer of planes as profiles (section, contour) to I-DEAS Master Modeler, which can be used directly to generate a swept or lifted solid.

You must perform the following steps:

- | | |
|------------------------------|---|
| 1. In EUKLID Design: | Generate transfer file |
| 2. In I-DEAS Master Modeler: | Read transfer file
(as with action <i>ideas_outputunvw</i>) |

Per sequence, one plane is mapped in a profile by I-DEAS Master Modeler with corresponding name and color.

In **EUKLID Design** a user attribute, *in_IDEAS_PROFIL*, is attached to the surfaces along with date and time of transfer initiation. With repeated transfer to I-DEAS only date and time (attribute value) are updated.

The existence of the attribute in the surfaces can be determined using the following AQL expression: ... *valid(<element>.user_in_IDEAS_PROFIL)*....

The attribute value (date and time of the most recent transfer call) is returned by the following AQL expression: *[<element>.user_in_IDEAS_PROFIL]*.

Application and recommendations for further use in I-DEAS

If the transfer file is read into I-DEAS Master Modeler with the command *Import - Design Universal File*, then wire frame models are generated with which swept and lifted solids can be generated.



- Different files with identical suffix
Files of type *Solid Link Universal File* and *Design Universal File* both have the same ending, *unv*. However, each has a completely different composition and a different definition.
- Intersection of geometric objects
In order to transfer wire frames (action *ideas_outputunvw*) the geometric objects may intersect, in order to transfer profiles (action *ideas_outputunvp*) they must not intersect.
- Objects in coordinate systems with non-unity scaling factors
If objects lie within coordinate systems having a scaling factor other than one, then they are displayed in graphically modified form, i.e. either larger or smaller.
During data transfer from **EUKLID Design** to an I-DEAS *Design Universal File*, the objects are transferred in their proper (actual) size. In order to obtain the proper visual impression, you should set the scaling factor to 1:1 before transferring to the I-DEAS *Design Universal File*.

Mixed-up 3D-2D design

19.2.4.3 Program File

ideas_outputprg

Generates a transfer file of type *Program File* for I-DEAS Master Modeler.



Name of transfer file via file selection box
Default value: *transferp.prg*.



Switch I-DEAS echo on and off
Each individual command of the *Program Files* in I-DEAS Master Modeler is output in the message box or not.



Color of elements in I-DEAS Master Modeler

Example

USE ACTUAL
0 - BLACK
1 - BLUE
2 - GRAY BLUE
3 - LIGHT BLUE
4 - CYAN
5 - DARK OLIVE
6 - DARL GREEN
7 - GREEN
8 - YELLOW
9 - ORANGE GOLD
10 - ORANGE
11 - RED
12 - MAGENTA
13 - LIGHT MAGENTA
14 - PINK
15 - WHITE

Default value *USE ACTUAL*: this gives all geometric objects the color set in I-DEAS Master Modeler as the current default value.



Selection group

Selectable elements are **points, lines, circles, circular arcs, contours, surfaces** and **groups**. Only the objects point, line, circle and circular arc are transferred, however. Only the calculated boundary objects (lines, circular arcs, circles) of contours and planes are transferred.

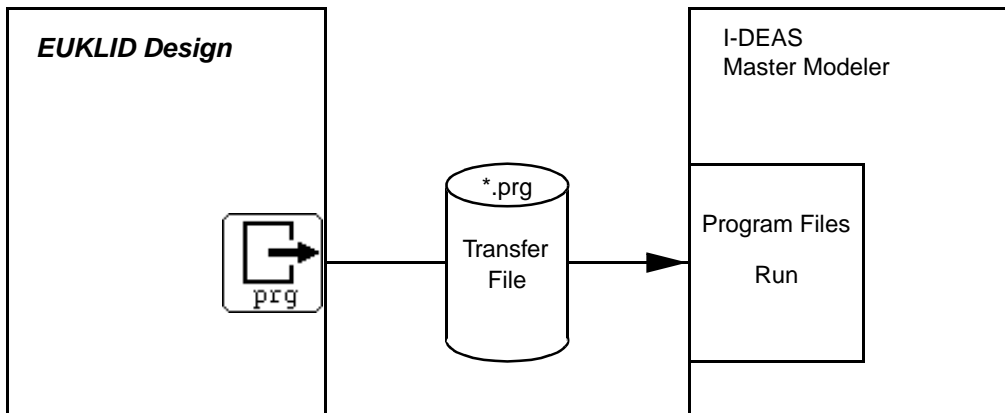
>From groups, the geometric objects contained therein are transferred. If this parameter is confirmed with *OK &Name*, then a group is generated which contains the selected objects. If confirmed only with "OK", a group is not generated.

Data transfer

Data transfer from **EUKLID Design** to I-DEAS occurs via a transfer file of type *Design Universal File* with the suffix *.prg*. The contents of the transfer file consist of generation commands for I-DEAS Master Modeler which are produced using the model data from **EUKLID Design**.

The following steps are required to perform a data transfer:

1. In **EUKLID Design**: Generate transfer file
2. In I-DEAS Master Modeler: Read transfer file



Mixed-up 3D-2D design

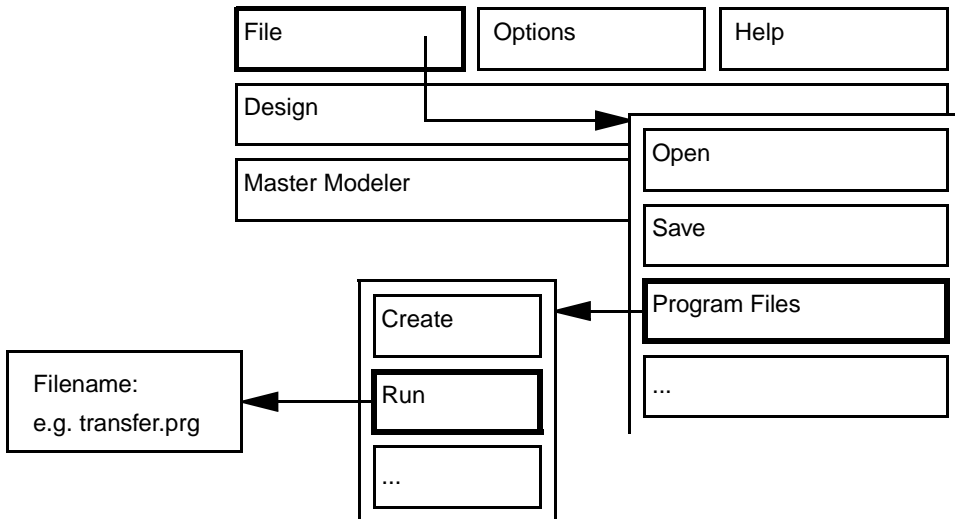
In **EUKLID Design** a user attribute, *in_IDEAS*, is attached to the groups - which have been either selected or just created (parameter *group* was confirmed with *OK & Name*)—along with date and time of transfer initiation.

With repeated transfer to I-DEAS only date and time (attribute value) are updated.

The existence of the attribute in the group can be determined using the following AQL expression: ... *valid (<element>.user_in_IDEAS)*....

The attribute value (date and time of the most recent transfer call) is returned by the following AQL expression: [*<element>.user_in_IDEAS*].

The file produced, *Program File* must now be called explicitly in I-DEAS Master Series and thus executed. This is possible using the following menus in I-DEAS Master Mod-
eler menus:



Application and recommendations for further use in I-DEAS

If the transfer file is run in I-DEAS *Design* with the command *Program File-Run*, then the geometric objects are generated as if they had been generated interactively. They therefore do not lie within their own wire frame model.

Profiles for swept or lifted solids may now be generated from the individual transferred geometric objects in I-DEAS Master Modeler.

If you would like to define key dimensions for the contours of these solids, then it is recommended that you first lift the solid with the contour, then edit the solid in *wire frame mode*. The key dimensions, parallelism etc. may be defined in this condition.



- The *program files* are not upwardly compatible. It is **not** guaranteed that they will also run in later versions. The *program files* produced by **EUKLID Design** may be run only under I-DEAS Master Modeler, not e.g. under I-DEAS Drafting Setup.

- Objects in coordinate systems with non-unity scaling factors
If objects lie within coordinate systems having a scaling factor other than one, then they are displayed in graphically modified form, i.e. either larger or smaller.

During data transfer from **EUKLID Design** to I-DEAS *Design*, the objects are transferred in their proper (actual) size. In order to obtain the proper visual impression, you should set the scaling factor to 1:1 before transferring to the I-DEAS *Design Universal File*.

Mixed-up 3D-2D design

20 Data transfer

The following data transfers are possible:

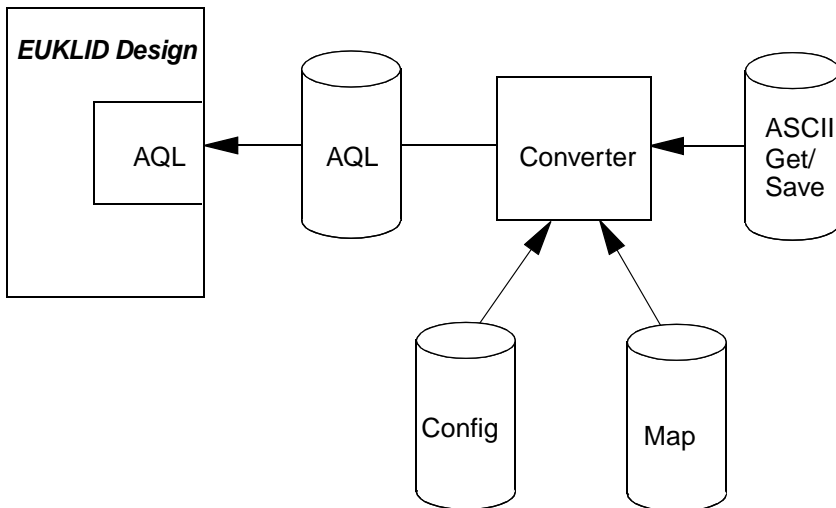
- Conversion of Draft data
- Data transfer in IGES format
- Data transfer in DXF format (V12)
- Data transfer in DWG / DXF format
- Linking to external programs

Data transfer

20.1 Converting Draft Data

Model data can be transferred from Draft to **EUKLID Design** using a converter. The files must be available in ASCII-GET/SAVE. Only one Draft drawing may be contained in each such file.

The converter is an independent program. It is called from a UDA in **EUKLID Design**, reads a Draft file (in ASCII-GET/SAVE format) and generates an AQL program from it. This AQL program is then executed automatically and generates the **EUKLID Design** model.



20.1.1 General Element Characteristics

In Draft, characteristics such as pen, pen type, color, and font are viewed as interpretation types (integer values). In the mapping file, these are encoded into the individual characteristics according to element.

The interpretation type in Draft is an index to a table. The following characteristics are determined by it:

- Color

There is a fixed number of colors in Draft. Since line color can be specified as an RGB value in **EUKLID Design**, exact conversion is possible. The values chosen for the conversion can be set in the mapping file. No color command is generated in AQL for the value 0.
- Pen (pen width)

The line width values 0.0, 0.25, 0.35, 0.5, 0.7 and 1.0 are coded in the table as digits 1 through 6.
- Pen type (style)

The pen type determines characteristics such as dashing, end symbols and line color.
 "Solid," "dashed," "dotted," "dashdotted" and "dashdotdotted" are available for encoding as digits 1 through 5 in the line table (*linetable*).
- End symbols

In Draft, the line type may also specify special end symbols such as arrows or circles for the beginning and end of lines. (Here, line-type elements such as lines, circular and elliptical arcs count as lines—in contrast to planes, for example.) No line end symbols are implemented In **EUKLID Design**. Lines are therefore generated without end symbols. Special end symbols for dimension representation are converted by generation of a dimension that contains these end symbols as far as **EUKLID Design** provides for this.
 For the case where line generation with end symbols is not covered by dimension generation, separate UDA's are generated for the beginning and endsymbols in order to place an appropriate end symbol as a UDO. The line itself is the victim parameter of this UDA.
- Font

Texts have the font characteristic instead of pen type and end symbols.

Data transfer

Layers

Elements located in a layer in Draft for which no entry was made in the mapping file are generated in the main layer of ***EUKLID Design***.

If no entries have been made in the mapping file for the arrangement of layers (*layerable*), then the entire layer handling is deactivated.

Empty layers are not generated in ***EUKLID Design***, i.e. at least one element must be transferred per layer.

User attributes

Invisible attributes of a Draft element are attached as user attributes to the object generated in ***EUKLID Design***. For visible attributes, a text is generated, the string value of which depends upon the attribute value.


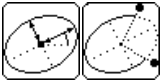







if the object with the attribute is deleted, this text is retained because the relation still exists.




Element names

Draft names are attached to the generated objects as ***EUKLID Design*** names.

20.1.2 Elements transferred

<i>Draft</i>	<i>EUKLID Design</i>	<i>Remarks</i>
Ellipse		
Elliptical arc		
Plane, hatching		In <i>EUKLID Design</i> , the order of boundary element nodes determines the surface definition.
	<p>The associativity to the original boundary elements remains unchanged in <i>EUKLID Design</i> as well, if the list of boundary elements was still complete in Draft.</p> <p>Restriction: Self intersecting planes are not supported by <i>EUKLID Design</i>.</p>	
Group		The name is adopted
Circle		
Circular arc		
Line		






Data transfer

<i>Draft</i>	<i>EUKLID Design</i>	<i>Remarks</i>
Series of lines (linear interpolation curve, polygon)		Group of individual lines, interpolation points as normal points
Point		Relative to the coordinate system acting as parameter when UDA is called
Spline		Not yet implemented
Symbol	Internal UDO's	
Text		Draft fonts are also available in <i>EUKLID Design</i> . Conversion via entries in mapping file. A default font is fixed for unknown fonts.






Dimensioning

The dimensions in Draft are not associative but rather joined to the dimensioned object only via the end points of the dimension boundary lines. At the endpoints of these lines, an object appropriate for the dimension type is searched for in order to dimension it. If this cannot be done, then another representation of the dimension is generated as a line illustration.

The individual portions of a dimension in Draft are not protected from changes. It is possible, for example, to delete dimension line halves or dimension lines. However, if the dimension is not longer complete, it can no longer be converted into a dimensioning element by **EUKLID Design**. Instead, the image of the dimension is recomposed as a group consisting of lines and texts.

<i>Draft</i>	<i>EUKLID Design</i>	<i>Remarks</i>
Point-Point		
Point-Line		For the line, a point is generated where the dimension line or the dimension auxiliary line ends in Draft.
Line-Line		For the line, points are generated where the dimension lines or the dimension auxiliary line ends in Draft.
Diameter, projected diameter		If the end points of the dimension line or dimension auxiliary lines both lie on the same circle and this is clearly identifiable in <i>EUKLID Design</i>
		Otherwise, between the end points of the dimension line or the dimension auxiliary lines.

Data transfer

<i>Draft</i>	<i>EUKLID Design</i>	<i>Remarks</i>
Radius		If the circle can be clearly identified in <i>EUKLID Design</i> from the end points of the dimension line
	Line drawing	otherwise
Arc length		If it is possible to identify the circle uniquely in <i>EUKLID Design</i> via the endpoint of the dimension line
	Line drawing	otherwise
Angle between two lines		For the line, points are generated where the dimension lines or the dimension auxiliary line ends in Draft.
Angle between three points		
Coordinate dimensioning		
Geometric tolerances	Groups of lines, circles and text	No separate data structure element exists in Draft.

Variant techniques

Only the geometry is assumed from the variable portions of the draft that were generated using variant techniques.

The internal attributes of the variant system may be recognized by a special character preceding their names and are suppressed during conversion.

20.1.3 Configuring the converter

The converter characteristics can be controlled via a configuration file and an mapping file. Frequently used parameters are located in the configuration file and can be set using the configuration editor: it can be called in the UDA, in the dialog box for parameter *Name of configuration file*. This configuration editor reads the configuration file and writes it back again if you make changes and exit the dialog box with *OK*.

20.1.4 Mapping tables

Lengthy mappings are stored in their own mapping file. This is divided into individual sections by key words. Characters following a '#' are ignored as comments until end of line. The ends of the individual sections are marked with a line containing only the word *end*.

Key words are case-insensitive.

Within the sections, Draft values are converted to **EUKLID Design** values:

<Draft value> <**EUKLID Design** value(s)>

The structure of lines in the mapping file varies from section to section:

Lines

The line table uses the following structure:

```
linetable
<Interpretation type> <Color> <Pen> <Pen type> <Beginning sym-
bol><Ending symbol>
...
end
```

The beginning and end symbols are numbers used in the AQL program generated when the AQL function *make_symbol* is called. They determine the selection of the end symbols to be generated.

Data transfer

Colors

The color table uses the following structure:

```
colortable
<Color number>  <AQL color value>
...
end
```

<Color number> Color range: 1 to 9.

<AQL color Color name, color index or RGB value.

value> The color value may not contain any spaces, since otherwise only the characters up to the first space are considered. The expression is used directly and without checking in the AQL program generated.

Text and fonts

Fonts are defined by:

```
font <Font number> <Font name>
map <Draft character code>  <EUKLID Design charactercode>
...
end
```

The font number is any desired number under which the font description is stored and under which it can be referred to in the section *texttable* (see below).

The text table is structured as follows:

```
texttable
<Interpretation type>  <Font number>  <Color>   <Pen>    <Pentype>
...
end
```

The font number is a reference to an entry in the font table defined in the section *font* lines.

The font with number 0 is then used even if no font definition exists for the number specified in the text table.

Layers

Conversion of Draft layers to **EUKLID Design** layers can be specified.

```
layertable
<Draft layer>  <EUKLID Design layer>
<Start Draft layer>-<End Draft layer> <EUKLID Design layer>
end
```

Example:

```
layertable
3                "lines"
4                "dimensions"
17-255          "various"
end
```

The specification of a range is possible only in this section. The range boundaries must be separated by a hyphen. No spaces may appear before or after the hyphen. The layer must be specified as a string within quotation marks. If several of the specified ranges overlap, the last one specified has priority.

Example of a mapping file

```
font 0 Draft_ptext
map 123 30 # degree sign
map 124 31 # plusminus
map 125 24 # quad
map 126 29 # diameter
end
```

```
font 1 Draft_ptext
map 123 30 # degree sign
map 124 31 # plusminus
map 125 24 # quad
map 126 29 # diameter
end
```

Data transfer

```
layertable
3      "lines"
4      "dimensioning"
17-255 "various"
end
```

```
linetable
1 3 4 1 0 0
2 1 4 1 0 0
3 2 4 1 0 0
4 7 4 1 0 0
...
100 7 2 1 0 -6
101 5 2 1 0 -6
102 6 2 1 0 -6
...
270 1 2 4 0 0
271 8 1 1 0 0
end
```

```
texttable
1 1 3 2 1
2 1 1 2 1
3 1 2 2 1
4 1 7 2 1
...
100 14 7 4 1
101 14 5 4 1
102 14 6 4 1
...
257 18 5 6 1
258 18 6 6 1
end
```

Special options for test purposes

In the configuration file, options may be switched on for test purposes to obtain additional output during conversion. These options should normally remain switched off in order not to inflate the data structure in **EUKLID Design** unnecessarily.

`debugAttributes <on/off>`

If this option is set to *on*, then two additional attributes are attached to the **EUKLID Design** objects or actions generated:

<code>address</code>	the original address of the Draft data structure element for which the object was generated
<code>lineNumber</code>	the line in the ASCII-Get/Save file in which the Draft data structure element begins

Example (Excerpt):

```
e1_1 = point_relative(ori,130,130)
create_attrib(e1_1, "address", 352)
create_attrib(e1_1, "lineNumber", 37)
e2_1 = circle_centerradius(z,{"solid",0.5}, ax, e1_1, 80)
create_attrib(e2_1, "address", 371)
create_attrib(e2_1, "lineNumber", 49)
e3_1 = point_relative(ori,60,200)
create_attrib(e3_1, "address", 396)
create_attrib(e3_1, "lineNumber", 61)
e4_1 = point_relative(ori,200,200)
create_attrib(e4_1, "address", 415)
create_attrib(e4_1, "lineNumber", 73)
```

Data transfer

`debugNames <on/off>`

If this option is set to *on*, then all objects and actions generated in ***EUKLID Design*** will get names. The name used is that of the AQL variable with which the respective Draft element was correlated.

Example (Excerpt):

```
e1_1 = point_relative(ori,130,130)
name(e1_1, "e1_1")
e2_1 = circle_centerradius(z,{"solid",0.5}, ax, e1_1, 80)
name(e2_1, "e2_1")
e3_1 = point_relative(ori,60,200)
name(e3_1, "e3_1")
e4_1 = point_relative(ori,200,200)
name(e4_1, "e4_1")
```

This allows the generating positions in the AQL file to be found again using objects or actions in ***EUKLID Design***. This works in the other direction as well by marking elements through an AQL program, e.g.:

```
show_highlighted (e3_1)
```

20.2 Data Transfer via IGES, DWG and DXF

EUKLID Design allows data transfer with other CAD systems, in both directions, on a graphic-logic basis. Data transfer is performed either via an IGES file, DWG file or via a DXF file:

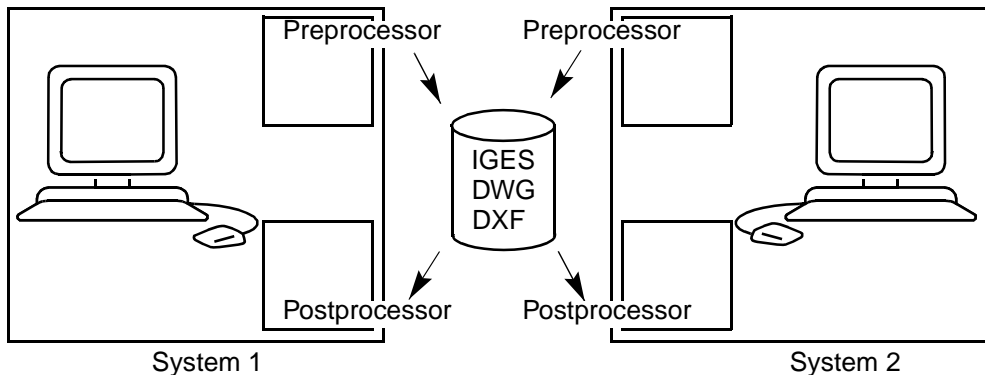
- Either **EUKLID Design** evaluates, converts and writes the selected data structures into the IGES, DWG or DXF file (**Preprocessor**), or
- a pre-existing IGES or DXF file is read and an **EUKLID Design** data structure is determined and built up from it (**Postprocessor**).

These functions are called using an action within **EUKLID Design**.

The advantage of this arrangement is that only two conversion programs, respectively, are required for data transfer between different CAD systems.

EUKLID Design is able to exchange data with any CAD system containing an IGES or DXF processor. Such processors are available for most CAD systems on the market.

For informations on the input and output actions please refer to the Online Help.



Data transfer

20.2.1 Data transfer via IGES

IGES (Initial Graphics Exchange Specification, *Versions 1.0 to 4.0*) is a transfer format for exchanging CAD data.

The entities specified in IGES cover a wide range of application, including entities for 2D and 3D mechanical design, for dimensioning and illustration, for electrical diagrams and for 3D solids ("CSG" and "BREP").

IGES itself, however, does not contain a specification for implementation, i.e. IGES does not define which entities a system must support for a given application. Therefore, every IGES processor gives support preference to the entities of the system for which it was created. This can lead to considerable loss of information during transformation between different systems.

This experience led to the definition of the VDA/VDMA IGES Subset (VDAIS), with the goal of providing the most complete possible data transfer of geometry dimensions and text. This subset is limited on purpose to the IGES entities relevant to the exchange of CAD data between the auto industry, its suppliers and machine tool manufacturers. This specification is defined by VDMA/VDA Uniform Page 66 319.

In addition to the limitation of element scope, there are also standards specifying how geometry not available in a system is to be illustrated in substitute elements. Several formal limitations are also given in order to provide a consistent transfer format.

VDAIS performance levels

The version to which this handbook applies includes the following performance levels of IGES transformation:

- performance level **G1/B1** 2D Basic geometry/Dimensioning
- performance level **G1/B2** G1B1 & Views (Viewport)
- performance level **PLUS**
- performance level **VDAIS** The performance level is taken from the IGES file (valid only in the postprocessor)

The performance level PLUS goes beyond the capabilities of VDAIS. It allows transfer of surfaces, associative dimensioning as well as **EUKLID Design**-specific handling of element groups (UDOs).

PLUS is meant to handle the IGES entities equivalent to **EUKLID Design** objects like associative dimensioning, surfaces, hatching and certain UDOs. PLUS especially helps to avoid information loss during data exchange.

Makeup of an IGES file

IGES files are sequential ASCII files with a fixed format per data set (80 characters).

An IGES file is divided into five separate sections, each of which serves a different purpose:

- The **Start section** denotes the processor version. The identifier S is located at Position 73 of the data set.
- The **Global section** contains information describing the preprocessor, which allows the postprocessor to correctly interpret the IGES data.
- Such information includes for example the time of file creation, author, product identification etc. The section identifier G appears in position 73.
- The **Directory entry section** includes a directory of all entities contained in the IGES file, as well as a fixed number of independent data (width, color, name,...) per entity. The identifier D is always located at position 73.
- The **Parameter data section** contains the element-dependent data (coordinates, number of points in a sequence, etc.) for all entities. Composition and content of the individual data sets depend upon the respective entity type and are therefore not structured uniformly like the directory entry section. The section identifier in position 73 is always P.
- The **Terminate section** marks the end of an IGES file. It tells the number of data sets per section. The section identifier in position 73 is T.

Data transfer

20.2.2 Data transfer via DXF (V12)

DXF (Drawing eXchange Format) was originally developed for AutoCAD and is used by many CAD systems to exchange CAD model data with other systems.

Makeup of a DXF file

DXF files are ASCII files.

A DXF file is divided into four separate sections, each of which serves a different purpose:

- The **Header section** contains general information on the DXF file.
- The **Tables section** contains the following tables:
 - Linetype table (LTYPE)
 - Layer table (LAYER)
 - Text style table (STYLE)
 - View table (VIEW)
 - User coordinate table (UCS)
 - Viewport configuration table (VPOR)
 - Dimension style table (DIMSTYLE)
 - Application identification table (APPID)
- The **Blocks section** contains the block definition entities (description of block definitions such as user-defined objects, etc.)
- The **Entities section** contains all entities to be displayed, including references to defined blocks (Blocks section).



DXF files containing a newer format than V12 (V13, V14) may be read in but all unknown (new) sections and entities are ignored.

20.2.3 Data conversion

Data conversion is performed by a separate program. This requires distinction among the following processes:

- The **preprocessor** generates an IGES, DWG or DXF file.
- The **postprocessor** processes an existing IGES or DXF file.

Reading a configuration file

Several variables are read from a configuration file to determine the following:

- where the processor is to create and search for the IGES or DXF file structures;
- where any temporary files needed should be created and found;
- where the reference table are located.

Creating a data structure tree

The processor creates an internal data structure tree. It allows processing of complex elements in hierarchical form and syntax checking of the IGES, DWG or DXF file to be processed.

First the processor creates its internal data structure tree to write subsequently the IGES or DXF file.

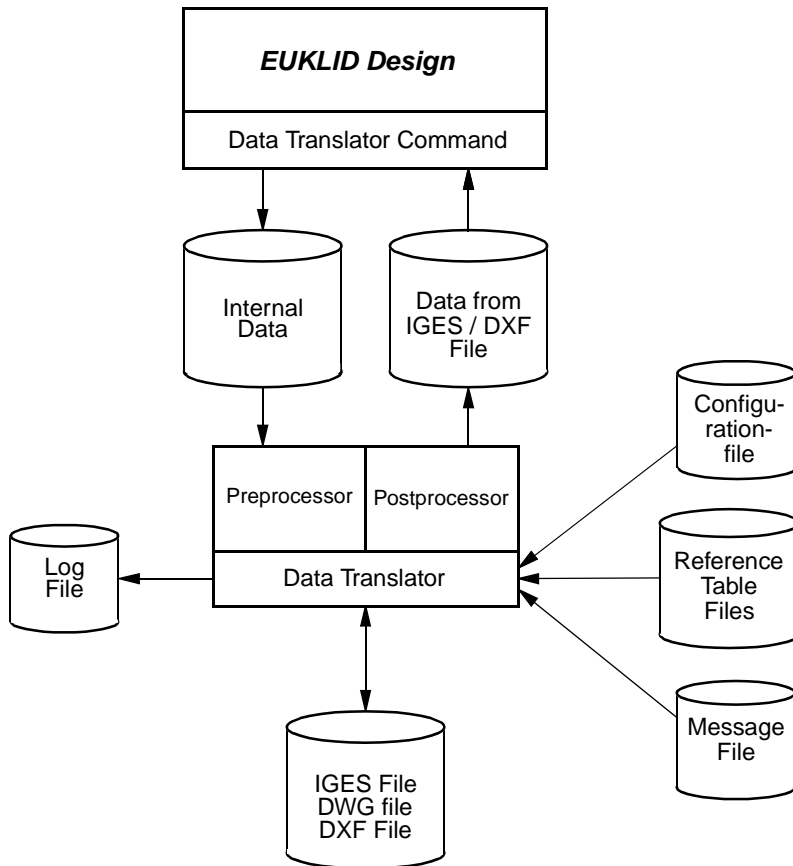
The postprocessor reads the IGES or DXF file and creates its internal data structure tree. Afterwards it generates an AQL program which creates **EUKLID Design** data structure.

Auxiliary files (temporary files) which may be created are located in the directory which is defined via the variable IGES_TEMP or DXF_TEMP (See „[IGES configuration file](#)“ on page 20-22. or „[The DXF configuration file](#)“ on page 20-32). They will be deleted at the end of the processor running.

Data transfer

Statistics and Log file

Information regarding type of translation and any errors is stored during processing in a log file.



20.2.4 Parameter files

The **Data Translator** requires a number of parameters which are stored in parameter files for easier handling.

The following files are described in detail on the pages that follow:

- Configuration file
- Reference table files:

<i>File</i>	<i>Meaning</i>
startsection.text	Will be copied into the Start section
linienart.zuor	IGES: Reference of line type
farbwerte.zuor	IGES: Color converting
masspfeil.zuor	IGES: Reference of dimension arrows
zeichensatz.zuor	IGES: Character
iges.zuor	global IGES setting
color.zur	DXF: Color converting
dxfont.zur	DXFIN: ASCII character set table
dx.zur	global DXF settings
dwgfont.zur	DWGOUT: ASCII character set table
Dwg_Para.iga	DWG: Parameter file for DWG/DXF

- Reference tables in AQL program(s):
dxftable

Data transfer

20.2.4.1 The configuration file

The processor works with a series of variables that are used to control data conversion into and from IGES and DXF. These variables are defined in the configuration file so that the files need be present only once in the system and can be located anywhere. Furthermore, you need only change the corresponding line in the configuration file when handling your own data.

Standard names of configuration files:

iges_in.iga	– configuration file for IGES import
iges_out.iga	– configuration file for IGES export
dxf_in.iga	– configuration file for DXF import
dxf_out.iga	– configuration file for DXF export
dwg_out.iga	– configuration file for DWG export

The configuration file consists of any desired number of variables. Each variable consists of a **key word** and a **Value**, which are separated from one another by any desired number of spaces.



Key words may not be used as values.

IGES configuration file

Standard values of key words are underlined. Instead of the keyword “true” it also is possible to use “ja” or “yes”, instead of “false” it also is possible to use “nein” or “no”.

Key words that determine data directories

IGES_PFAD	<Directory name> Here the processor searches for IGES files or stores the created IGES files.
IGES_TEMP	<Directory name> Temporary files are stored in the directory indicated here.
IGES_ZUOR	<Directory name> The reference tables are sought in the directory indicated here. If this directory cannot be accessed, then the standard directory name <installation path>/design_iges/zuor is used. <ul style="list-style-type: none">○ Simplified access to central reference tables: The # character as the first character of the path in IGES_ZUOR is interpreted as "<installation path>/iges". Thus the entry IGES_ZUOR #/zuor specifies presets for IGES.
IGES_PROT	<Directory name> The log files are stored in the directory indicated here.
IGES_UBBL	<Directory name> (postprocessor only) UDOs are stored or sought here.
IGES_AQLP	<Directory name> (postprocessor only) Here the AQL programs created by the postprocessor are stored.

If you do not define these directories, then the processor searches or stores in the current directory.

Data transfer

Variables for controlling the translation process

The following variables control the translation process. They may only be set to one of the values indicated.

IGES_STUF G1B1/G1B2/PLUS
This variable defines the IGES performance levels according to VDAIS.
G1B1: Geometry and simple dimensioning
G1B2: like G1B1 + handling of views
PLUS: Special non-VDAIS entities used by Siemens-Nixdorf and SDRC.

IGES_LANG deutsch/english/espanol/russian
Language for messages and the log file.

IGES_DRAW true/false (used only by the preprocessor)
If “true,” then graphics transfer is given priority.
If “false,” then geometry transfer is given priority.
For complete transfer, you should always use “IGES_DRAW true” to export drawings. If you are more interested in real geometry (e.g. NC systems), then set “IGES_DRAW false” since the display can differ considerably from the geometry. This occurs when coordinate systems with different scales or offset coordinate systems are used.



Once exported from a drawing using “IGES_DRAW true,” the original geometry cannot be recreated when read in by **EUKLID Design**, but it can, however, be plotted in the correct way.

When exported from a drawing using “IGES_DRAW false,” the original depiction can be reconstructed by creating coordinate systems subsequently and using the function “Separate”. The display will differ without these subsequent handling steps.

IGES_GRAF	<u>true/false</u>	
Preprocessor:	Sets the type of graphics export. Supplements or replaces the variable IGES_DRAW ! <u>false</u> <i>true</i>	The variable IGES_DRAW is binding. Graphic objects (symbols) can be exported as well and IGES_DRAW is set to true
Postprocessor:	Conversion of General Label and General Symbol. <u>false</u> <i>true</i>	Create as SYMBOL if possible Convert to groups. The subentities of the entity are created as text, lines etc. and formed into a group.
IGES_SPLI	<u>true/false</u> Type of spline conversion <i>true.</i> <i>false</i> (Prerequisite: IGES_DRAW true) Splines within contours are always polygonized.	splines are created as splines splines are polygonized
IGES_KONT	<u>true/false</u> (preprocessor only) If “true,” then contour sequences are output sorted, independent of selection order during creation of the contour.	
IGES_GRP	<u>true/false</u> If “true,” then the entire object group is selected when any object in the group is selected. If the entry is “false,” then no objects of type Group are selected.	(Used only by the preprocessor)

Data transfer

IGES_LAYR	<i>true/false</i> Preprocessor : If “true,” then the entire layer is selected and exported as a group when any object in the layer is selected. Postprocessor: “true” produces layers with the entries in the level field of the DE section of the IGES file, e.g.: iges_level_1. The corresponding layer is set <i>active</i> before an object is created. “false” produces lengths with these entries (default). These lengths are used as Z values whenever possible if no entity is available for controlling visibility. IGES files produced by the Data Translator should not be read in with “IGES_LAYR <i>true</i> ”, since a separate layer is produced for each Z value in that case.
IGES_MASS	<i>true/false</i> (preprocessor) <i>true/false</i> (postprocessor) The standard value for the postprocessor depends upon the current performance level selected. With the performance level PLUS, the standard value is “true,” otherwise it is “false.” Type of dimension conversion <i>true</i> Dimensions are produced from dimension entities whenever possible. <i>false</i> Dimension entities are converted to groups. The subentities of the entity are created as text, lines etc. and formed into a group. If the entry is missing “ <i>false</i> ” is used with the performance level G1B1/G1B2 and “ <i>true</i> ” is used with the performance level PLUS.

IGES_USER	<u><i>vorh[anden]/neu/auf[lösen]</i></u>	
	Controlling UDO handling	
	Preprocessor:	
	<u><i>vorh[anden]</i></u>	Create empty subfigures (Entity 308). The other system can then use existing UDOs.
	<u><i>auf[lösen]</i></u> <i>(neu)</i>	Create subfigures. not possible, since UDOs of completely different character can exist and therefore no generally valid definition can be created.
IGES_VIEW	Postprocessor:	
	<u><i>auf[lösen]</i></u>	UDOs are converted to groups (without creating UDOs)
	<u><i>true</i></u> / <i>border</i> / <i>false</i>	(only postprocessor)
	Type of view handling	
	<i>false</i>	No handling of views (Entity 410) or drawings (Entity 404).
	<u><i>true</i></u>	Views (Entity 410) within a drawing (Entity 404) are evaluated.
	If the system of origin is a 3D system, problems may, however, occur; please set IGES_USER to <i>auf[lösen]</i> if necessary.	
	<i>border</i>	The view border is visualized with lines.

Data transfer

IGES_VINR	<u>false</u> /Nummer	(postprocessor only)
	View number for display (postprocessor only)	
	A specific view can be selected.	
	<u>false</u>	No limitation
	Nummer	Number of view in drawing
	Only the entities in the selected view are created. The view number is located in the xx parameter of the entity view (410) in the parameter section of the IGES file.	
IGES_NAME	<u>false</u> /true	(postprocessor only)
	Information added as attribute to an object	
	<u>false</u>	no names used
	true	Following attributes will be created:
	IGES_ENTITY	Entity number
	IGES_FORM	Form number
	IGES_NAME	Name
	IGES_ZEILE_DE	Line number DE-Section
	IGES_ZEILE_PD	Line number PD-Section
IGES_DATP	<u>false</u> /true/ <u>all</u>	
	Record all file names in a background window.	
	<u>false</u>	No recording
	true	Record the log file name
	<u>all</u>	Record all file names.
IGES_UNIT	<u>false</u> /inch/mm	
	Explicit definition of dimension units (global section of I-DEAS)	
	<u>false</u>	No explicit definition of dimension units
	inch	Dimension unit is inch !
	mm	Dimension unit is mm !
IGES_STAT	<u>false</u> /true	
	Additional output of statistics into background window.	
	<u>false</u>	No output
	<u>true</u>	Output into background window

IGES_MESS	<u>norm[al]</u> / <u>meld[ung]</u> / <u>warn[ung]</u> / <u>err[or]</u> / <u>no</u> Definition of message level <u>norm[al]</u> Standard message level <u>meld[ung]</u> Output errors, warnings and messages. <u>warn[ung]</u> Output errors and warnings. <u>fehl[er]</u> Output errors only.
IGES_TIME	<u>false</u> / <u>true</u> Time log for processor sequence <u>false</u> no time logging <u>true</u> output of starting, ending and running time
IGES_SPEC	<u>auto</u> / <u>false</u> (only postprocessor) Automatic recognition of special settings for several sending systems <u>auto</u> automatic special settings <u>false</u> no special settings
IGES_WORL	<u>false</u> / <u>true</u> (only preprocessor) Choose reference coordinate system
IGES_WORL	<u>false</u> / <u>true</u> (only postprocessor) New coordinate system for views (410) <u>false</u> no new coordinate system will be created. <u>true</u> if the view is scaled a new coordinate system with this scaling factor will be created.
IGES_POIN	<u>pos</u> / <u>laenge</u> (only postprocessor) Creating points: Use position or length
IGES_CLIP	<u>true</u> / <u>false</u> (only preprocessor) clip before output (only G1B1)

Data transfer

Variables for calling an AQL program

You may call an AQL program before and after each IGES conversion.

The parameters of this AQL program are

- the name of the IGES file,
- the name of the AQL program (= postprocessor) or the name of the transfer file (=preprocessor) and
- the name of the configuration file.

IGES_PREP <Name>
This AQL program will be called before every conversion.

IGES_PSTP <Name>
This AQL program will be called following every conversion.

Variables of the names in the start section (preprocessor only)

The following names are entered in the start section of the IGES file:

IGES_FIRM <Name>
Source company

IGES_ASPR <Name>
Responsible person

IGES_TEL <Name>
Telephone number

IGES_ADRS <Name>
Address (Enter as one line without line feeds)

IGES_PROJ <Name>
Project

IGES_PROK <Name>
Project reference identifier

IGES_VARI	<Name> Variant
IGES_VERT	<Name> Confidentiality
IGES_GDAT	<Name> Validity date
IGES_AUTH	<Name> Name of author
IGES_EFA	<Name> Receiving company
IGES_ENA	<Name> Receiver name and department

Names may contain blanks, but must not contain several lines.

If the following names cannot be found in either the start section file or in the configuration file, then the process is interrupted, since these parameters must be specified according to VDAIS:

- Source company
- Responsible person
- Telephone
- Address
- Project
- Project identifier
- Receiving company
- Receiver name/department

Data transfer

The DXF configuration file

Standard values of key words are underlined. Instead of the keyword “true” it is also possible to use “ja” or “yes”, instead of “false” also “nein” or “no”.

Variables for setting file directories

Normally, you should set the following file directories (one for each) in every configuration file:

DXF_PFAD	Directory name The processor searches here for DXF files.
DXF_TEMP	Directory name Temporary files are deposited into this directory.
DXF_ZUOR	Directory name In this directory, reference tables are stored. If this directory is inaccessible, the standard directory <installation path>/iges/zuor_dxf is used. Simplified access to central reference directories: The # character as the first sign in the path in DXF_ZUOR is interpreted as “<installation path>/design_iges”. Thus the entry DXF_ZUOR #/zuor_dxf specifies DXF defaults.
DXF_PROT	Directory name (postprocessor only) Log files are stored in this directory.
DXF_UBBL	Directory name (postprocessor only) UDOs are stored into or read from this directory.
DXF_AQLP	Verzeichnisname (postprocessor only) The AQL programs produced by the postprocessor are stored here.

If you do not define the directories mentioned here, the processor searches and stores in the current directory.

Variables for controlling the translation process

The following variables control the translation process. You may set them only to one of the values listed.

DXF_LANG	<i>deutsch/english/espanol/russian</i> Language for messages and the log file.
DXF_SPLI	<i>true/false</i> (postprocessor only) Type of spline conversion <i>true.</i> splines are created as splines <i>false</i> splines are polygonized Splines inside contours are always polygonized .
DXF_MASS	<i>true/false</i> (preprocessor) <i>true/false</i> (postprocessor) Type of dimension conversion <i>true</i> Dimensions are created, whenever possible, from dimension entities. <i>false</i> Dimension entities are converted to groups. The subentities of the dimension entity are created as text, lines etc. and made into a group.
DXF_USER	<i>diss[olve]</i> (postprocessor only) Controlling handling of UDOs (block handling) <i>diss[olve]</i> Convert block entities to groups (do not create UDOs).

Data transfer

DXF_VIEW	<i>false/true</i>	(postprocessor only)
	Type of view handling	
	<i>false</i>	No handling of viewports.
	<i>true</i>	Viewports are evaluated. If the source system is 3D, problems may occur. If necessary set DXF_USER to <i>diss</i> .
DXF_VINR	<i>false/Nummer</i>	(postprocessor only)
	Number of viewport to be displayed	
	A specific view can be selected	
	<i>false</i>	without restriction
DXF_NAME	<i>Nummer</i>	Viewport ID
	Only the objects in the designated viewport are created.	
	<i>false/true</i>	(postprocessor only)
	Information added as attribute to an object	
DXF_DATP	<i>false</i>	no names used
	<i>true</i>	Following attributes will be created:
	DXF_NAME	Name
	DXF_ZEILE	Line number
DXF_UNIT	<i>false/true/all</i>	(postprocessor only)
	Record all file names in the background window.	
	<i>false</i>	Do not maintain log
	<i>true</i>	Record the log file name
DXF_STAT	<i>all</i>	Record all file names
	<i>false/inch/mm</i>	(postprocessor only)
	Explicit definition of dimension unit.	
	<i>false</i>	Do not define dimension unit explicitly
DXF_STAT	<i>inch</i>	Dimension unit is inch!
	<i>mm</i>	Dimension unit is mm!
	<i>false/true</i>	(postprocessor only)
	Additional output of statistics into background window	
DXF_STAT	<i>false</i>	No output
	<i>true</i>	Output into background window

DXF_MESS	<i><u>norm[al]</u>/meld[ung]/warn[ung]/fehl[er]</i> (postprocessor only) Definition of message level <i><u>norm[al]</u></i> Standard message level <i><u>meld[ung]</u></i> Issue error messages, warnings and status messages <i><u>warn[ung]</u></i> Issue error messages and warnings <i><u>fehl[er]</u></i> Issue only error messages
DXF_TIME	<i><u>false</u>/true</i> (postprocessor only) Processor time log <i><u>false</u></i> Do not maintain time log <i><u>true</u></i> Output starting, ending and running times
DXF_WORL	<i><u>false</u>/true</i> (preprocessor only) Choose reference coordinate system
DXF_POIN	<i><u>pos</u>/laenge</i> (postprocessor only) Creating points: Use position or length
DXF_CLIP	<i><u>true</u>/false</i> (preprocessor only) clip before output (only G1B1)
DXF_PREC	<i><u>default</u>/number</i> (preprocessor only) Precision (number of digits)
DXF_LNAM	<i><u>false</u>/true</i> (preprocessor only) Use original layer name

Data transfer

Variables for calling an AQL program

You may call an AQL program before and after each DXF conversion.

The parameters of this AQL program are

- the name of the DXF file,
- the name of the AQL program (postprocessor only) and
- the name of the configuration file.

DXF_PREP	Name
	This AQL program is called before every conversion.

DXF_PSTP	Name
	This AQL program is called after every conversion.

The DWG configuration file

Standard values of key words are underlined. Instead of the keyword “true” it is also possible to use “ja” or “yes”, instead of “false” also “nein” or “no”.

Variables for setting file directories

Normally, you should set the following file directories (one for each) in every configuration file:

DWG_PFAD	Directory name The processor searches here for DWG files.
DWG_ZUOR	Directory name In this directory, reference tables are stored. If this directory is inaccessible, the standard directory <installation path>/iges/zuor_dxf is used.
DXF_PROT	Directory name Log files are stored in this directory.
DWG_AQLP	Directory name The AQL programs produced by the postprocessor are stored here.
DWG_PARA	Directory name The directory of the parameter file (Dwg_Para.iga) for controlling the DWG output.

If you do not define the directories mentioned here, the processor searches and stores in the current directory.

Data transfer

Variables for controlling the translation process

The following variables control the translation process. You may set them only to one of the values listed.

DWG_LANG	<i>deutsch/<u>english</u>/espanol/russian</i> Language for messages and the log file.
DWG_DATP	<i>false/<u>true</u>/full</i> (postprocessor only) Record all file names in the background window. <i>false</i> Do not maintain log <i><u>true</u></i> Record the log file name <i>full</i> Record all file names
DWG_STAT	<i><u>false</u>/true</i> (postprocessor only) Additional output of statistics into background window <i><u>false</u></i> No output <i>true</i> Output into background window
DWG_MESS	<i><u>norm</u>[al]/meld[ung]/warn[ung]/fehl[er]</i> (postprocessor only) Definition of message level <i><u>norm</u>[al]</i> Standard message level <i><u>meld</u>[ung]</i> Issue error messages, warnings and status messages <i><u>warn</u>[ung]</i> Issue error messages and warnings <i><u>fehl</u>[er]</i> Issue only error messages
DWG_CLIP	<i><u>true</u>/false</i> clip before output

Variables for calling an AQL program

You may call an AQL program before and after each DWG conversion.

The parameters of this AQL program are

- the name of the DWG file,
- the name of the AQL program (postprocessor only) and
- the name of the configuration file.

DWG_PREP	Name
	This AQL program is called before every conversion.

DWG_PSTP	Name
	This AQL program is called after every conversion.

Data transfer

20.2.4.2 Reference tables

In order to make object properties which are not contained in IGES or DXF more flexible, the processor reads the reference tables from files during initialization. The processor matches each object with its corresponding features that are described in the reference tables.

IGES reference tables

With IGES and VDAIS, the numbers of line modes, colors and typefaces are restricted.

The processor searches for the files in the directory specified in the configuration file under the entry IGES_ZUOR. If the processor cannot read the data, then the process is aborted because the contents must be available for the processor to run correctly.

Makeup of reference tables

The format of this files is as follows:

The IGES value is located in the first column, followed by a comma (",") and the matching ***EUKLID Design*** value.

```
/***** /
0,0                               No color, System-Farbe (schwarz)
1,0                               Black, Schwarz
1,6
1,8
1,10
1,12
1,14
2,50                               Red, Rot
```

└── ***EUKLID Design***-value
└── comma
└── IGES-value

Characters that follow the **EUKLID Design** value may serve as comments, asince they are not interpreted.

You can manage your own table files by changing the entry under IGES_ZUOR in the configuration file.

IGES reference tables (in the directory #/zuor):

- startsection.text Sample for start section of IGES file
- linienart.zuor Conversion of line modes
- farbwerte.zuor Conversion of colors
- masspfeil.zuor Conversion of dimension arrows
- zeichensatz.zuor Conversion of typefaces
- iges.zuor General conversion parameters

The “startsection.text” file

During initialization, a text file is read and interpreted. This file contains system-wide texts which are entered into the transfer header (start section of the IGES file).

You will find the complete description of the format in this file.

The “linienart.zuor” file

IGES features five line modes:

3. solid
4. dashed
5. phantom
6. centerline
7. dotted

Since **EUKLID Design** supports more line modes they must be mapped to those of IGES.

The exact file format is described within the file itself.

Data transfer

The “farbwerte.zuor” file

In **EUKLID Design** up to 64 colors (in IGES nine colors) are available. Just as with line weights, the colors must be matched.

The IGES value 0 plays a special role; in IGES it is called no color, which means that the destination system for an object of this color should use the system color.

In addition, IGES also features color definition (entity 314).

The “masspfeil.zuor” file

EUKLID Design offers a number of arrowhead shapes for dimensioning arrows. If the preprocessor produces VDAIS files, then you may set the IGES values of this file to 1, 4, 9 and 10 only.

The “zeichensatz.zuor” file

The **EUKLID Design** character bank allows 256 different characters. The file *zeichensatz.zuor* contains an entry for every ASCII value of a character. This entry determines within which IGES typeface the exact definition of the character may be found.

The typeface numbers 0 and 1 contain a special significance: if a character of typeface 0 is found, then it is processed as a blank. A character of typeface 1 is processed as a standard ASCII character if the file *iges_font1.zuor* doesn't exist.

The range of values for standard ASCII text lies between 30 and 127. In case of doubt, a space is always used. For other font numbers, the processor branches to a further reference file and searches there for an entry for that character.

This further file is named as follows:

`iges_fontxxx.zuor`

The characters “xxx” stand for the number of the corresponding IGES font.

If such a severe error occurs during processing that the desired character set cannot be loaded, you may be able to prevent the non-defined character from appearing as a blank space by copying one of the included font files with the naming conventions.

The processor reads the font files for every character, but rather does not summon the relevant parts until a character first appears from a character set that is not yet loaded.

In order to give you a firm grasp on the concept and handling of this file, the following example (preprocessor) shall explain how it works:

Example

The **EUKLID Design** drawing contains the special character for degrees “°”. This character arrives in the processor with ASCII value 159. In the file *zeichensatz.zuor*, this is recorded as: “159,1001”. Character set 1001 is therefore summoned.

The processor now summons the corresponding entry from the file *iges_font1001.zuor*: “111,159”. The character is therefore issued the ASCII value 111 in the IGES file, which corresponds to an “o”. In the IGES specification, the character “o” in character set “1001” stands for a degree symbol “°”, so that the desired result is achieved.

The postprocessor uses the same reference files, but in reverse direction.

Data transfer

The “iges.zuor” file

General or global values used by the processor are located in this file. All values have default settings which are always used whenever nonvalid values (or none at all) are specified by the user.

The individual meanings:

MINM_RESO_WERT	Minimum resolution: entered into the global section by the postprocessor.
ALLG_SYST_EPSL	General system tolerance: with performance level PLUS, this is taken from the global section by the postprocessor.
LEER_FEHL_MELD	Default error message. Used when the error message file contains incorrect format information and prevents the IGES processor from finding the message.

Ranges and default values

<i>Name</i>	<i>Range</i>	<i>Default value</i>
MINM_RESO_WERT	0.1 - 0.000001	0.001
ALLG_SYST_EPSL	0.1 - 0.000001	0.0001
LEER_FEHL_MELD	<string>	”Error message cannot be found”

DXF reference tables

The makeup of these DXF files is analogous to that of IGES reference tables.

These files exist:

- color.zur Color conversion
- dxf_font.zur Typeface conversion
- dxf.zur General conversion parameters (preprocessor)
- dxftable General conversion parameters (postprocessor)

The “color.zur” file

In **EUKLID Design**, up to 64 colors are available. Just as with IGES, you must match the colors to one another.

The “dxf_font.zur” file

The character set of **EUKLID Design** allows for 256 different characters. The file *dxf_font.zuor* specifies corresponding **EUKLID Design** values for every ASCII value of characters entered in the DXF file.

The “dxf.zur” file

This file contains general or global values used by the processor. All values have defaults which are used whenever nonvalid values (or none at all) are specified by the user.

The individual meanings:

- | | |
|----------------|---|
| ALLG_SYST_EPSL | General system tolerance |
| LEER_FEHL_MELD | Default error message used whenever the error message file contains incorrect format information and prevents the DXF processor from summoning the message. |

Data transfer

Ranges and defaults

<i>Name</i>	<i>Range</i>	<i>Default</i>
MINM_RESO_WERT	0.1 - 0.000001	0.001
ALLG_SYST_EPSL	0.1 - 0.000001	0.0001
LEER_FEHL_MELD	<string>	"Error message cannot be found"

The “dxftable” file

The preset values for the standard layers, line modes and DXF are stored in the AQL program *dxftable* . User-specific adaptations can be made directly in this AQL program.

DWG reference tables

The makeup of these DWG files is analogous to that of IGES reference tables.

These files exist:

- dwg_font.zur Typeface conversion
 - Dwg_Para.iga General conversion parameters
- For an exact description of the parameters please refer to the online help. Please follow the hyperlink *Further information on DWG format*.

Data transfer

20.2.5 Limits of data conversion

20.2.5.1 Limits of IGES data conversion

Preprocessor

When performance level G1/B1 is used, many relationships and structures of ***EUKLID Design*** are lost. The expanded performance level PLUS should reduce the loss of information when coupling to surface-oriented and associative systems.

Performance level G1/B1

With performance level G1/B1, the following restrictions must be made:

- The ***EUKLID Design*** relations are lost, since the IGES entities have no relations nor creation history.
- Important objects (e.g. coordinate systems and vectors) cannot be exported.
- User elements are exported only as groups (i.e. separated).
- The relations between dimensions and their geometric reference objects are lost.
- The surface properties of ***EUKLID Design*** are lost:
- Hatching is depicted as groups of lines.
- With VDAIS G1/B1, hatching cannot be depicted with the relation to the surface.

Performance level PLUS

This performance level provides functionality which goes beyond that of VDAIS G1/B1. It treats the objects “Surface” (with holes), “User element” and “Dimension” specially, so that associative relations are maintained after transformation.

However, information loss cannot be completely prevented. It can be reduced, though, through appropriate selection of variables in the configuration file.

Postprocessor

Not all entities of the VDAIS performance levels can be depicted directly in corresponding objects by **EUKLID Design**. In such cases, VDAIS provides for substitute entities into which the VDAIS entities are transformed. If no corresponding object for the substitute entity exists in **EUKLID Design**, then a substitute entity of the substitute entity can be used for translation.

Use of substitute entities always means loss of information.

The **EUKLID Design** postprocessor places the following restrictions on the performance levels G1/B1 and PLUS:

Performance level G1/B1

Parabolic and hyperbolic arcs are translated into splines.

Performance level PLUS

Mirrored "SUBFIGURE INSTANCES" (Entity 408) are separated and the objects are mirrored.

Dimension text in non-scaled drawing portions may not be correctly reproduced.



The function of the scale for depicting very large or very small geometries in predetermined drawing areas is often realized by a different method than that of **EUKLID Design**. In order to use the scaling functions of **EUKLID Design**, the AQL procedure *iges_masstab.aql* was developed.

Since CAD systems often use only one single global coordinate system, all non-scaled details are created using the model data calculated to that scale. This means, for example, that a 10 mm long line scaled 1:1 runs from point 0,0 to point 10,0; for 10:1 scaling, from 0,0 to 100,0. If this 100 mm long line is dimensioned, then the distance between points is calculated to scale in model coordinates and entered as a dimension value.

When non-scaled drawings are transferred from such CAD systems, a model is produced in **EUKLID Design** with non-scaled dimensions since it interprets the scale differently.

Data transfer

A model drawn not to scale has the same model coordinates in **EUKLID Design** as the corresponding graphic depiction scaled 1:1. The “size” of the model depiction, however, is controlled via an additional coordinate system so that in that case, the screen coordinates are transformed when the scale is changed.

This yields the advantage that the scale can be changed very easily during operation, depending upon the available drawing surface.

The procedure *iges_masstab.aql* has the following features:

- Selection of affected objects via pick position or group name.
- Automatic determination of scale value from the corresponding distance dimension or via manual entry.
- Write the scaling factor into the corresponding dimension measure.

Procedure:

- Read the file into **EUKLID Design** (the variable IGES_MASS may not be set to *false*, since no dimension objects are created in this case).
- Call the AQL procedure *iges_masstab.aql* in **EUKLID Design**. (Accessible when correctly installed under *#/design_/iges/iges_masstab.aql*.)

20.2.5.2 Limitations of data conversion via DWG and DXF

The same limitations apply to DWG and DXF as those of IGES (especially those regarding the PLUS performance level).

20.2.6 Subdirectory and files in the directory “design_iges”

The directory *design_iges* contains the following subdirectories and files:

Subdirectories

<i>Subdirectory</i>	<i>Meaning</i>
mess	message files, IGES entity description
zuor	Reference files for “normal” IGES
zuor_ext	Reference files with extended IGES
zuor_dxf	Reference files for DXF

Files

<i>File</i>	<i>Meaning</i>
design_iges	The IGES/DXF processor (UNIX only)
iges_in.iga	Example of a configuration file
iges_out.iga	Example of a configuration file
dxf_in.iga	Example of a configuration file
mess/iges_<>.meld	message file (language-dependent)
mess/entity_tab	Table of designations of IGES entities
zuor/iges.zuor	System setting file for IGES
zuor_dxf/dxf.zur	System setting file for DXF

Data transfer

20.2.7 Error messages

If the **Data Translator** cannot access a message file it only issues the message number. You can look for it in the following file:

mess/iges_english.meld

The characters &t and &i, which appear in this file, are replaced by the following character on the screen:

- &t text appears at this location
- &i an *integer* value appears at this location

20.3 Interface to External Applications

This action can be used to link external applications to the **EUKLID Design**. Partner companies e.g. offer PPC interface modules.

The interface between **EUKLID Design** and a production planning and control system (PPC) avoids the need to reinput the production data generated during the design stage to the PPC system, and is particularly relevant to bills of material and various types of master data. You can pass the data directly from the CAD session to the PPC data resources.

PPC data are also available to you, e.g. prices of bought out items and on hand stocks. Online access from the PPC data resources from **EUKLID Design** is possible at all times.

Click on the *extern prog* icon in the first menu column of the respective menu.

Click on the *call* icon in the fourth menu column. The PPC system will be started in a process window and the **EUKLID Design** process window is retrieved to background.

If the **EUKLID Design** transfer is selected during running of the PPC program, the **EUKLID Design** process window is retrieved to foreground; the external process remains in background.

To conclude the PPC program, click on the *stop* icon in the fourth menu column.

21 Data Safety

21.1 Procedure in Case of Error

It is not possible to avoid errors (or their external causes) one hundred percent. A distinction is made between three classes of errors, according to their cause:

User errors	Handling errors: These are intercepted by EUKLID Design . An appropriate error message is output.
Program errors	This category includes actual software errors within EUKLID Design , which would cause the program to behave inappropriately or to terminate abnormally. This kind of error is intercepted by either EUKLID Design itself or the operating system.
System errors	These are errors in the operating system or faults in the hardware itself, and may lead to critical malfunction or system failure, according to their severity.



- A log file can be used to repeat the complete session up to a certain point and to restart a previous session at an arbitrary point (see section *Logging Sessions*).
- A deliberate provocation of error adds extra functions for the user. It is possible to check the design logic, since **EUKLID Design** is a system dealing with the facts of a design, for example to establish the limits of particular design. An error will inform the designer if a change to key data or parameter values causes the design to become unfeasible. An error will thus serve as an additional source of information for the user.

Data Safety

21.1.1 User Errors

The user prompting facilities of the **EUKLID Design** user interface essentially limit the errors a user can make to logical ones. An exception is the restricted input options in the text area.

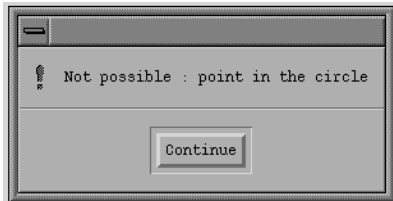
A logical user error occurs in the following cases:

- A calculation does not make sense mathematically.
Example: point of intersection of two parallel lines.
- A function is called, which is not possible at the time.
Example: deletion of an object type not included in the model.
- A function is called, which is not permissible.
Example: accessing a locked file.

Example

The following example illustrates how **EUKLID Design** reacts to user errors, and how the user acknowledges them.

A tangent is to be created from a circle to a point. If the point lies within the circle, or is caused to be moved there by a change to its coordinates, a dialog box containing the appropriate error message in the user's default language is opened in the drawing area.



The objects causing the error are highlighted if possible.

☞ To acknowledge error message, click on.

The system reacts differently according to the context in which the user error occurs:

- in *Create* mode
- in *Edit* mode
- during iteration

User Error in *Create* Mode

The creation sequence is aborted and the object involved not created.

Example

An attempt is made to create a point of intersection between two parallel lines; the point is not created.

User Error in *Edit* Mode

The object involved remains in the data structure but is no longer displayed (invalid). Inputting one incorrect parameter value may prevent several objects from being updated properly. In this case, a number of error messages may be displayed one after the other, each of which must be acknowledged. When the parameter value is corrected, the object concerned and all other interdependencies are reset to active.

Example

The center of a circle is the point of intersection between two lines; if the lines are changed to parallels, neither the point of intersection of the two lines nor the center of the circle can be recreated.

User Error during Iteration

As in *Edit* mode, the objects involved are retained in the data structure. It is advisable here to acknowledge the error messages individually (although there be a large number, depending on the relations), and to reset the value to its original value by selecting the *Reset* icon in the third menu column.

Example

A dimension which cannot be applied has been input explicitly. A popup menu may be used to simplify this procedure if desired.



Input of invalid parameters in *Edit* mode, without subsequent correction, may cause invalid objects to be created. Models saved with an uncorrected error still contain this error (i.e. invalid objects) when subsequently reloaded. Delete objects from the model which have been rendered inactive by errors

Data Safety

by starting **EUKLID Design** with option *-cleanup* and rewrite the model subsequently.

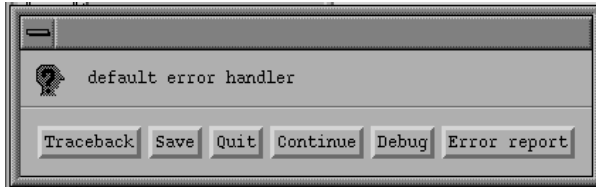
design -cleanup <modelname>

Invalid objects, however, are removed, even if they were inserted voluntarily. Thus you should execute this function carefully, not as default start option.

21.1.2 Program Errors

In spite of the effort and care with which the program and user interface have been generated, it is not possible to avoid program errors absolutely.

- If the sole effect of the error is the output of an incorrect result by the program, it is usually possible to find a way round the error and carry out the desired function.
- If the error causes an abnormal termination, the following dialog box is opened:



Here you can choose the following options:

Traceback	for development purposes only
Save	the model file is saved in a CRASH file
Quit	EUKLID Design is quit without saving the model
Continue	EUKLID Design is continued, e.g. after AQL program termination by accidentally pressing a key
Debug	for development purposes only
Error report	not used

EUKLID Design attempts to save the model data in the event of an abnormal termination. If this is successful, the system creates a so-called CRASH file. This has the name *<modelname>.crash*. This file can usually be loaded as a model (the option *-cleanup* may be useful here; see section *User Errors*). You can then resume normal working. If no CRASH file is created, or if the CRASH file cannot be loaded without error, the log file is available, if written.

Data Safety

21.1.3 System Errors

As already stated, no complex program is error-free. This naturally applies also to operating systems. Hardware faults, power failures and powering down of one computer in a network, can all have severe consequences. **EUKLID Design** keeps the damage caused to a minimum by keeping a constant log file and affording the option of reproducing the session (see section *Log File*)

21.1.4 Error Reports

Although **EUKLID Design**, by means of its special reaction characteristics, is able in most cases of program error to maintain normal working (see section *Program Errors*), every program error should still be reported. The quality of the software, and with it the range of applications, are to a large degree dependent on your collaboration in this context, as is also problem-free, user-friendly system operation.

If an error in **EUKLID Design** can be definitively identified as a program error and is reproducible, inform the service engineer in the *CAD/CAM Strässle* regional office. The following documentation should be supplied to assist in error diagnosis:

<i>Documentation</i>	<i>Remarks</i>
Error report	Brief, but precise description of the error
Model file <i><model name>.mod</i>	Entire directory in which work was done
CRASH file <i><model name>.crash</i>	If an abnormal termination of EUKLID Design occurred
LOG file <i>name.aql</i>	Log file of the session during which the error occurred
Configuration directory <i>design_config</i>	
UDOS, UDAs and Tables	If included in the model, the entire directory
Hardcopy	As evidence in the case of incorrect graphic display when the error cannot be reproduced
AQL Programs	If included

The documentation listed above may be written to a data medium.

21.2 Logging Sessions

You can save all inputs of an ***EUKLID Design*** session in an AQL file. It is thus possible to repeat the entire session up to a desired point and a session previously interrupted can be resumed from any point.

☞ Start ***EUKLID Design*** using the option *-protocol*:

design – protocol <file name> [<model name> ...]

The log file is written in the form of an AQL program and is named *<file name>.aql*



It is a good idea to keep the log files to a manageable size by finishing and resuming the session at certain strategic points in time. This particular approach saves a good deal of trouble in the case of errors, avoiding having to run through the whole day's work.

You can start a log file just like other AQL programs.

22 Appendix

22.1 System Data Conventions

Every program defines with the internal representation of data the principles of processing accuracy and the limitations. The limits of ***EUKLID Design*** will not be reached while the day-to-day practice.

22.1.1 File Names

Restrictions of the operating system cause file names to be limited to a **maximum of 31 characters**, excluding file name supplements (e.g. the suffix ".mod").

They can consist of at most 31 characters. You can use all small and capital letters, numbers and the underline character "_". The first character should be a letter. Blanks, special characters and Umlauts are not allowed.

22.1.2 Real Numbers

Real numbers are represented in memory in **double precision**. The 64 bits are utilized as follows:

1 to 52	Mantissa without leading 1
52 to 63	Exponent + 1023
64	Sign

Appendix



The resolution of a number is inversely proportional to its length. The number "1.34567" is thus more accurate than the number "134567". For a practical CAD application this implies that constructions close to the zero point ($x=0$, $y=0$) are more accurate than those far away from it.

22.1.3 Lengths and Angles

Lengths and angles are represented **using real numbers** (see above). In order to prevent sequence errors and overflow errors during internal calculations, only **values up to "2.1*10⁹"** may be entered.

22.1.4 Character Strings

Character strings may have unlimited number of characters.

23 Glossary

Attribute

The **number and type of bill of material data** which is to be stored in the bill of material balloon may be specified by attributes.

Directory section

Short form of "directory entry section". The abbreviation used in the IGES specification is DE section. This section contains a two-line entry for each component, describing the component type, the attribute and the location of the geometry data in the parameter section.

Element

The term "element" is used in this manual to express **part element of a set**, as understood in design drafting (but not data processing). An element may also be represented by a object.

Examples:

- Constructional element of an engineering design:
A component which is to serve a particular function, as for example a screw, which is used to join two other components
- Parameter within a geometric construction procedure:
These are components used to calculate a resulting element, for example two lines used to plot a point of intersection.
- User element:
This is a user-defined part of an overall design, which may be assigned standard geometry, function and mathematical characteristics.

Entity

Entities are data structure components in accordance with the IGES standard, which are subdivided into three categories:

- Geometry components
- Text and dimensioning components
- Structure components

Glossary

Enumeration value

The enumeration value is a special case of the terminal type. If the quantity of **input options for a given value is limited and predefined**, these options are displayed numbered in column 4 of the icon menu. dialog boxes may also be used. The desired value is selected with the mouse. This is not a case of implicit construction, but merely a selection from a limited range of input options. The user is thus offered a more convenient input facility, since it is not necessary to transfer the hand from the mouse to the keyboard.

Global section

This section contains general format specifications for graphics to be transferred, e.g. scaling, unit of measurement, line thickness factor.

Highlighting

This is a traditional facility of graphics user interfaces. Highlighting is used to make a part of the **display more conspicuous**. Icons, objects and fields are given a different-colored background, or, in the case of black and white screens, a gray background.

Icon

An icon is a **pictogram displayed on the screen**, and is an important aspect of the graphics user interface. The appearance of the icons and the way they are arranged on the screen are essential to the user prompting facility. The icons are treated in the same way as **push buttons**.

Functions of the icon:

- To inform the user on status and available functions
- To start actions an icon is selected with the mouse
- To improve the clarity of an existing data structure

Identify

This term refers to "**notify**", "**make known**" or "**select**". If a model contains many objects of the same type, and only one is to be used as a parameter in a creation sequence, then that object is to be identified.

IGES file

File where CAD structures are stored in accordance with the IGES standard.

Model

The term "model" refers to the **sum of all data input relevant to a given drawing** (i.e. objects and their relations to each other). A model therefore contains not only the visible aspects of the drawing but also a range of data not displayed, but essential for the correct action of the system and only available under **EUKLID Design**. When stored as data, within the files or working memory of a computer, it is referred to as a data structure.

Model files

Files in which **EUKLID Design** stores the data of models permanently.

Object

A object is the **smallest unit of the data structure addressable** via the user interface. All geometry elements (point, line, circle etc.) and features in general use (e.g. length, angle etc.), are represented by objects. A object always contains also a rule as to how it is to be defined (the action), how it is to be displayed, how and whether it may be identified, and so on. By the same token, there are different object types. In opposite to an element a object never exists in its own right, but is always **related to its environment**.



The term "object" is used in **EUKLID Design** and refers to what other CAD systems call an element to assist an understanding of the principles of the system. What is concerned here is not just constructional elements in the usual sense, but **objects**, which are not always the same as geometric elements. An example is the character string of a text, which is stored in separate (implicit) object "string". It would be missing the point to refer to this as an element.

Orientation and direction

Every geometry object has direction, or orientation. This may be seen to mean the way the object is **"pointing" within the drawing area**. An arc, for example, is drawn counterclockwise, if there is no user input to the contrary.

The orientation of a object is significant, since it is the basis for a number of mathematical calculations carried out by the system; these may have far-reaching consequences for the drawing itself. If, for example, a line L2 is to be drawn perpendicular to another line L1, the orientation of line L1 is significant. The orientation alone determines in which direction (angle of orientation L1 + 90°) line L2 is drawn. A similar situation i.e. is met with in the case of distance parameters for a parallel contour.

Glossary

Parameter section

Short form of "parameter data section". The abbreviation used in the IGES specification is PD section.

This section contains the actual geometry data for each component, e.g. coordinates, number of points in point sets.

Postprocessor

Processor which reads an IGES file and converts it to a SIGGRAPH picture.

Preprocessor

Processor converting the data structure of a SIGGRAPH picture to an IGES file.

Rubberbanding

This traditional term of graphics user interfaces allows the user's subjective idea of a parameter to be represented on the screen without committing it to definitive input. Lines, circles, rectangles etc. are displayed as if **stretched (hence the analogy to a rubber band) between a fixed reference point on the screen and the cursor**, which reflects the movements of the mouse.

Select

This term has a similar meaning to "identify", in the sense of **notify or make known**. The difference is that the object selected is added to a possible existing set of objects, and is thus part of a selected set. It is then possible to carry out operations on this selected set, i.e. all objects within it, for example "delete", "copy" etc.

Start section

This section contains organization and administration data and general notes.

Terminate section

An entry in this section gives information on the number of lines occurring in the five sections.

Value

EUKLID Design employs a relational data structure, in which objects are always related to each other. Data from which **no further logical data relationships can be found**, i.e. where only pure figures and character strings exist, is said to be of value, since no further relations exist deeper in the structure.

If a value occurs as a parameter, it can only be furnished with data by input of a numeric value or a character string via the keyboard.

Index

#	
Installation directory.....	5-42
#/aql.aql	
directory.....	16-30
#/design_config	
directory.....	16-22
#/symbols	
directory.....	16-24
+	
Enter in dialog box.....	5-42
Standard library	17-28
.	
Working directory.....	5-42
..	
Parent directory	5-42
.bom	
file	16-22
~	
Home directory	5-42

A

Abort definition (UDA)	
Menu command.....	13-26
Abort definition (UDO)	
Menu command.....	13-11
Accept	
Menu command.....	4-31 , 4-33
Action	
circle_tglineline	10-4
Concepting	13-25

Defining	13-26
Edit definitions	13-45
Editing	8-14
fase_fase	10-3
fase_round	10-4
line_fase	10-3
Modifying for object	8-15
Modifying in fourth menu column	8-17
Selecting.....	7-3
trim_line2lines	10-2
trim_lineonline	10-2
Action group	3-12
Appending UDA icon	13-56 , 13-58
Appending UDOT icon ..	13-56 , 13-58
Editing	17-20
Inserting.....	17-17
Removing	17-21
Selecting.....	7-3
Action group box	13-56 , 13-59
Action group dialog box	17-19 , 17-20
Action group menu	
Editing	17-17
Hiding	17-17
Inserting.....	17-15
Removing	17-17
Action parameter	
Characteristics.....	13-36
Action type	
of transformed objects	8-23
UDA.....	5-3
Add model	
Menu command.....	5-50
Adjust action	8-34
Alternative parameter.....	4-14

Create mode	4-16
Edit mode	4-16
Angle	
Object type	7-7
representation in memory	22-2
Append (UDA)	
Menu command.....	13-36
Append (UDOT)	
Menu command.....	13-16 , 13-17
Append types>Enums	
Menu command.....	13-38
Append types>Objects	
Menu command.....	13-37
Append types>Values	
Menu command.....	13-38
Application concept	5-1
Applications	
complex	2-28
AQL	2-27
UDA attribute	13-32
AQL program	
abort	3-42
Appending to UDA.....	13-33
Argument	
Formula	3-32 , 13-104
Arithmetical operations.....	13-98
Assembly	
input of 3D models.....	19-4
Assembly checks	2-32
Assembly drawing	5-11
Attribute	
Allocating for UDOT.....	13-19
Allocation for UDA	13-32
Attributes	
Menu command.....	13-41
Automatic line style	14-11

B

Backup version	
of a model file	5-40
Open.....	5-48
Balloon	16-1
actions	16-3
copy example	16-19
display	16-4
Balloon position	
Object type	7-8
Base object	
Menu command.....	13-8
Basic objects	7-6
basic/aql/edit.aql	
file	13-32
Bill of material	
change in models	16-9
configuration	16-22
configuration for file output	16-30
configuration for model	
incorporation	16-24
creating header.....	16-25
creating line	16-26
creation example	16-27
data.....	16-1
data configuration.....	16-22
defining attributes	16-22
example	16-16
extraction example	16-20 , 16-21
generating.....	16-1
output.....	16-7
standard.....	16-1
storing data in model	16-3
transfer to model.....	16-7
using drawing frame text	
element	16-25
Bill of material balloon	
generation example	16-17

bomheader.uda	
user-defined action	16-24
bomheader.udo	
user-defined object	16-24
bomline.uda	
user-defined action	16-24
bomline.udo	
user-defined object	16-24
Bottom up structure	5-24
Bottomup structuring	5-13
Break line	14-10
Browse-Mode	3-19

C

Calculation		Close definition (UDOT)	
Closed contours.....	13-114	Menu command.....	13-14 , 13-15 , 13-23 , 13-87 , 13-89
Dialog boxes.....	13-114	Close model	
Calculations		Menu command.....	5-52
Application concept	5-8	Color	
Input.....	3-32	Fill color	14-13
Chamfer	10-3	Layer	14-13
between 2 lines.....	10-3	Model color in window frame	14-3
between 2 lines created shorter ..	10-3	Object colors	14-12
Change		Rectangle functions.....	14-13
Structure element	14-24	Standard color	14-13
Change comment.....	5-39	Structure element	14-13 , 14-25
Change visualization	14-22	Column name	
Changing		Object type	7-7
UDO in model	13-8	Combine (UDOT)	
Character string		Menu command.....	13-19
length.....	22-2	Command	
Circle		discontinuing	2-22
Object type	7-14	Command digression	2-22
circle_tgline		Command language.....	2-27
action	10-4	Comment	
Close definition (UDA)		Change comment	5-39
Menu command.....	13-29 , 13-45	Commercial standard parts	5-14
		Component	
		Standardization	13-117
		Configuration	
		bill of material	16-22
		bill of material data	16-22
		bill of material in file	16-30
		bill of material in models	16-24
		File system	17-25
		Loading.....	17-6
		Menu columns	17-14
		Online Documentation	3-37
		Online Help.....	3-36
		Saving	17-7
		System	17-1
		Configuration box.....	13-64

Configuring the User Interface	17-11
Construction	
explicit.....	7-1
Implicit	7-5
Construction modification	
Simplification	13-95
Constructional relationship	
changing	2-4
Contour	8-32
Object type	7-15
sketching	15-27
tracing the contour	15-39
Convert in layer	
Menu command.....	13-24
Convert to user	
Menu command.....	5-34
Coord	
Object type	7-7
Coordinate	
Object type	7-7
Coordinate system	
Application	7-26
Axial scale	7-23
Object type	7-10 , 7-23
Relationships between coordinate	
systems	7-25
Scaling factor.....	7-23
Copy vector	
between 2 coordinate systems	7-22
Object type	7-22
Create	
Menu command.....	5-20
Create action	
Editing parameter	8-7
Editing property	8-7
Create mode	2-18
Action groups.....	3-12
Parameter list	4-11

Creation	
explicit.....	7-1
Objects	7-1
UDO in model	13-7
Creation of contours and planes	
contour type (contour)	15-34
orientation.....	15-40
parameters	15-32
type of contour (contour)	15-32
Cursor	3-24

D

Data	
Extraction from tables.....	13-139
Standardization.....	13-122
Data input philosophy.....	2-18
Data structure.....	2-5
Examining.....	8-2
Data transfer	
from I-DEAS Drafting Setup	19-23
in DWG format.....	20-15
in DXF format	20-15
in IGES format.....	20-15
to I-DEAS Master Modeler.....	19-26
Default	
Line style	14-10
Delete	
Menu command.....	4-34 , 5-37
Structure element	14-24
Delete (UDA)	
Menu command.....	13-36
Delete (UDOT)	
Menu command.....	13-16 , 13-17
Delete all	
Menu command.....	4-32
Delete mode.....	2-18 , 9-3
deleted	
object attribute	9-1

Deletion	
action-specific.....	9-1
object.....	9-1
object-specific.....	9-1
undo delete.....	9-11
undoing.....	9-10 , 9-11
undoing a previous deletion.....	9-13
Deselect	
Menu command.....	4-33
Deselect all	
Menu command.....	4-31
Design	
Examining data structure.....	8-2
implicit.....	2-20
Retracing	8-2
Task distribution	5-9
Design of variants, defining the	
necessary variables.....	6-12
Design phase	5-10
Design proposal	5-7
Design techniques.....	7-1
Design variants	2-30
Design variations.....	13-117
Desktop	3-8
Working area	14-1
Destination coordinate	
system	11-10 , 11-13 , 11-15
Detail design	2-31
Detailing	
Piece parts.....	5-10
Dialog box	2-21
Directory	17-28
File names	5-40
for allocating of UDA parameter	
attributes	13-42
for allocating UDA parameter	
attributes	13-41
for area calculations	13-114
for entering property names	13-18
for entering the UDA name and	
the original model	13-27
for entering the UDOT name and	
the original model	13-12
for entering UDA parameter	
names	13-39
for input of bill of material	
attributes	16-5
for input of plot output format	18-12
for inputting the plot output	
format.....	18-14
for moving sublayers	5-26
for selecting the UDA parameter	
type	13-37 , 13-38
for setting balloon display	16-4
for setting layers active.....	5-20
Layer	5-25
Layer status	5-22
Line style	14-10
Object group.....	17-22
plot format	18-13
Selection.....	4-36
Updating location table.....	17-26
Dimension	
modification by geometry change	2-24
Types.....	7-19
Z value.....	14-15
Dimensioning	12-1 , 12-2
angle dimensions	12-12
automatic projection switching	12-4
chain dimensions.....	12-14
clipping witness lines	12-24
coordinate tables	12-25
creating individual dimension	12-6
decimal and min/sec.....	12-19
decimal degrees	12-19
definition of the reference point .	12-15
delete.....	12-14
diameter dimensions	12-11

dimension text position	12-19	text following dimension value... 12-19
dimension type	12-19	text position
distance dimensions	12-6	tolerance
distance dimensions position.....	12-8	witness line correction
dynamic positioning	12-22	Dimensions
Entering specific value.....	8-31	definitive
factor for desired reference		Directory
system.....	12-19	#/aql.aql
fixed projection	12-3	#/design_config
font properties.....	12-19	#/symbols
gon.....	12-19	Home directory
half-section dimensioning	12-18	Installation directory.....
length of circular arcs	12-12	Parent directory
line limit.....	12-19	Standard library
Modifying existing models	8-31	Working directory.....
normal dimensioning	12-18	Directory dialog box
number of decimal places.....	12-18	Directory name
opening angle	12-12	Filter.....
parameters	12-17	Input.....
position of circles	12-7	Distance ratio
position of diameter dimensions	12-11	Object type
position of dimensions	12-20	Documentation
position of radius dimensions	12-10	Help menu
positioning of measures.....	12-3	Drawing
prefix	12-19	Scrolling with keyboard.....
projection	12-3	Drawing area.....
properties.....	12-17	Drawing frames
radian.....	12-19	Drawing view of 3D models.....
radius dimensions.....	12-10	drop
referenced dimensions	12-14	UDA attribute
repetitive dimensions	12-19	Duplication
selector	12-7	vector.....
setting dimensions.....	12-18	DWG format
special characters in front of the		data transfer
dimension number	12-19	DXF format
symmetry dimensioning.....	12-18	data transfer

E

E(xternal)

Layer state..... 5-23

Edit action group

Menu command..... 13-56 , 13-63

Edit AQL file

UDA attribute 13-32

Edit definition (UDA)

Menu command..... 13-45

Edit definition (UDOT)

Menu command. 13-23 , 13-87 , 13-89

Edit icon (action group)

Menu command..... 13-59

Edit icon (menu)

Menu command..... 13-61

Edit last object

Menu command..... 8-29

Edit menu

Menu command. 13-58 , 13-63 , 17-17

Edit mode 2-18

deleting object 9-7

Object groups 3-12

Edit object group

Menu command..... 17-20 , 17-23

Editing

Action..... 8-14

List parameter..... 4-11

Object 8-6

Object in current mode 8-28

Parameter..... 8-6

Parameter in complex design

geometry 8-9

Parameter of create action 8-7

Parameter of manipulator action ... 8-8

Property 8-6

Property in complex design

geometry 8-9

Property of create action 8-7

Property of manipulator action 8-8

Editing techniques

Adjusting a sketched

profile 8-34 , 12-24

Editing via iteration 8-30

Effect objects

Determining for UDA 13-31

Ellipse

Object type 7-14

Enlarge window..... 6-22

Error reports 21-7

Errors 21-1

Input 2-22

Escape key 3-31

Expression

Formula 13-98

Extension of file names 5-43

F

F5-F9

Function keys 14-9

fase_fase

action..... 10-3

fase_round

action..... 10-4

Features

special 2-24

File

.bom 16-22

Backup version 5-40

Change comment 5-39

Filter 5-42

History 5-39

Open models 5-45

plot.cmd 18-12 , 18-15

Preview graphics 5-39

File box 5-40

File history..... 5-39

File name	
Extensions	5-43
Filter	5-42
length	22-1
Memory	17-25
Tooltips	14-1
File name Memory	17-25
File operations	5-39
File preview	5-39
File search rule	17-25
File system	
Configuration	17-25
File-Menu	3-17
Fill color	14-13
Fill style	14-16
Fillet	10-4
between 2 lines	10-4
between 2 lines created shorter ..	10-4
Filter	
Directory names	5-42
Enter in dialog box	5-42
File names	5-42
Model names	5-41
Finish a session	3-41
Flag	
S - selectable	5-22
V - visible	5-23
Formula	
Input	13-98
Formula processing	13-97
Kinematic simulation	13-112
Formula string	13-98
Modification	13-108
Formulae	
Linking	13-109
Freeze	
Layer state	5-23
Function keys	17-13
F5 - F9	14-9
View with F5-F9	14-9
Functional contexts	2-33
Functions	
Formula	13-98
G	
Geometric window	
Window type	14-2
Geometry	
modification by definitive	
dimension	2-24
Geometry objects	7-14
Go to	
Menu option	3-11
graphical	
UDOT instance type	13-19
Grid	14-9
Group	4-35
Changing	4-39
Creating	4-36
Deleting	4-39
Displaying	4-39
Object type	7-29
Using	5-4
Group mode	4-31
Menu command	4-33
H	
Help	
Documentation	3-37
Help menu	3-37
Menu command	4-29
On context	3-37
Online Help	3-36
Tooltips	3-38
Help-Menu	3-22, 3-37

Help-Text	
see Tooltips	
Hidden line	
Automatic line style.....	14-17
Window specific.....	14-7
Hidden lines	
Automatic line style.....	14-11
Plane border objects.....	14-16
Suppression.....	14-14
Hidden lines in 3D viewports.....	19-4
Hide menu	
Menu command.....	17-17
Hide object group	
Menu command.....	17-24
Hiding	
Layer.....	14-6
Hiding plane	14-15
Highlighting	2-20
History	
File history	5-39
Hold	
Layer status	14-7
Hold-Mode.....	3-19
Home directory	
Enter in dialog box.....	5-42
Hyperlink	3-36
I	
Icon	
Close window	14-3
Copying for UDOT/UDA	13-48
Creating for UDOT/UDA.....	13-46
Determining for UDA para- meters	13-39
Determining for UDOT	13-16
Determining for UDOT property.	13-17
Determining for user-defined action	13-30
Editing for UDOT/UDA	13-48
Iconized model window	14-1
Maximize window	14-3
Minimize window	14-3
Icon (parameter)	
Menu command.....	13-39 , 13-46
Icon (property)	
Menu command.....	13-17 , 13-46
Icon (UDA)	
Menu command.....	13-30 , 13-46
Icon (UDOT)	
Menu command.....	13-16 , 13-46
Icon editor	13-46
Actions.....	13-48
Copying an area	13-51
Copying an icon into the working area.....	13-53
Creating text	13-53
Cutting out an area	13-50
Drawing circles	13-50
Drawing lines	13-49
Drawing modes	13-48
Drawing points.....	13-49
Drawing rectangles.....	13-49
Filling contours	13-50
Inserting a cutout.....	13-51
Setting a font	13-52
Setting an icon editor.....	13-53
I-DEAS link.....	19-5
Identification	4-21
Identification number.....	17-25
UDOT	13-2
IGES format	
data transfer	20-15
implicit	
UDOT instance type	13-19
Implicit construction	7-5

Index	
calculated	13-138
Table-implicit	13-143
Init definition>From model (UDA)	
Menu command.....	13-27
Init definition>From model (UDOT)	
Menu command.....	13-12
Init definition>From UDOT	
Menu command.....	13-13 , 13-28
Init definition>New (UDA)	
Menu command.....	13-27
Init definition>New (UDOT)	
Menu command.....	13-12
init_uda	
UDOT instance type	13-19
Input	
alphanumeric	3-29
Calculations	3-32
in dialog boxes.....	4-3
in text area	4-3
Line style	14-10
Numbers	3-32
Object parameter	4-3
Octal method	3-30
Parameter	7-4
Property	7-4
Special characters	3-33
Umlauts	3-34
Z value	14-16
Input facility	4-1
Insert menu	
Menu command.....	13-61 , 17-15
Insert object group	
Menu command.....	17-22
Installation directoy	
Enter in dialog box.....	5-42
Instance creation	
UDA	13-2
UDOT	13-2
Invisible	
Layer.....	14-6
Is a relation	
Menu command.....	5-35 , 13-22
Using	5-4
Iteration	8-30
K	
Key assignments	
Create.....	17-14
Key combinations	
Calling the menus.....	3-35
Keyboard.....	3-29
Keyboard input	
alphanumeric characters	3-29
Numbers	3-32
Keypad tool	3-30
Keys	
Special functions	3-34
Kinematic simulation	2-32
Example.....	13-112
L	
LAVIS structogram	14-18
Using	5-4 , 5-16
Layer	5-18
Adding UDO definition	5-35
Assigning name	5-20
Changing	5-22
Changing name	5-23
Color.....	14-13
Converting to userdefined object. 5-34	
Creating	5-20
Declaring external layer	5-27
Deleting	5-36
External	5-27
From converted UDO	13-24

Menu	
Appending UDA icon	13-61
Appending UDOT icon.....	13-61
Calling via key combinations	3-35
Creating action group menu	17-15
Editing action group menu	17-17
Hiding action group menu.....	17-17
Removing action group menu....	17-17
Selecting.....	7-3
Switch	3-11
Menu area	
Scrolling.....	3-35
Structure	4-1
Menu bar.....	3-16
File-Menu.....	3-17
Help menu	3-37
Help-Menu	3-22
Layer-Menu	3-20
Mode-Menu	3-18
Options-Menu	3-22
Uda-Menu.....	3-21
Udo-Menu.....	3-21
Undo-Menu.....	3-17
Window-Menu.....	3-18
Menu box	13-58 , 13-61
Menu column.....	2-19
first.....	2-19 , 4-2
fourth	2-20 , 4-2
second	2-19 , 4-2
third.....	2-20 , 4-2
Menu columns	
Configuring	17-14
Menu command	
Abort definition.....	13-11
Abort definition (UDA).....	13-26
Accept.....	4-31 , 4-33
Add model	5-50
Adit action group.....	13-56
Append (UDA)	13-36
Append (UDOT).....	13-16 , 13-17
Append types>Enums	13-38
Append types>Objects	13-37
Append types>Values	13-38
Attributes	13-41
Base object.....	13-8
Close definition (UDA) ...	13-29 , 13-45
Close definition (UDOT).....	13-14 , 13-15 , 13-23 , 13-87 , 13-89
Close model.....	5-52
Combine (UDOT).....	13-19
Convert in layer	13-24
Convert to user	5-34
Create.....	5-20
Delete	4-34 , 5-37
Delete (UDA)	13-36
Delete (UDOT).....	13-16 , 13-17
Delete all.....	4-32
Deselect.....	4-33
Deselect all	4-31
Edit action group.....	13-63 , 17-20
Edit AQL file.....	13-34
Edit definition (UDA)	13-45
Edit definition (UDOT) ..	13-23 , 13-87 , 13-89
Edit icon (action group).....	13-59
Edit icon (menu)	13-61
Edit last object	8-29
Edit menu	13-58 , 13-63 , 17-17
Edit object group.....	17-23
Group mode.....	4-33
Help	4-29
Hide menu	17-17
Hide object group	17-24
Icon (parameter)	13-39 , 13-46
Icon (property)	13-17 , 13-46
Icon (UDA).....	13-30 , 13-46
Icon (UDOT)	13-16 , 13-46

Init definition>From model (UDA)	13-27	Select by name.....	4-18 , 4-31
Init definition>From model (UDOT)	13-12	Select in user.....	4-26
Init definition>From UDOT	13-13 , 13-28	Select indirect.....	4-28
Init definition>New (UDA)	13-27	Set active.....	5-21
Init definition>New (UDOT)	13-12	Set library	17-28
Insert menu.....	13-61 , 17-15	Set optional (UDA)	13-40
Insert object group.....	17-22	Set optional (UDOT).....	13-18
Is a.....	5-35 , 13-22	Set parameter.....	13-36
Layer.....	5-25	Set property	13-16 , 13-17
Load (UDA).....	13-44	Set result object.....	13-31
Load (UDOT)	13-21	Set result object (UDA).....	13-31
Load external	5-32	Show ancestors.....	4-29 , 4-34
Move (UDA).....	13-40	Show environment.....	4-34
Move (UDOT)	13-18	Show sons	4-29 , 4-34
Move objects	5-24	Single mode	4-32
Move sublayer	5-26	Status	5-22 , 5-27
Move to layer	4-32 , 4-34	Subobjects.....	13-8
Name	4-34	Top object.....	13-8
Name (UDA)	13-39	undo	9-11
Name (UDOT)	13-18	undo delete.....	9-11
New model.....	5-44	Update location table.....	17-26
Next	4-33	Viewing.....	4-32 , 4-34
Previous.....	4-34	Viewing>Quick edit.....	8-28
Quick edit.....	8-28	Menu extension area	3-15
Redefine	8-15	Menu-Icon	3-15
Remove action group	17-21	Message box.....	3-15
Remove menu	17-17	Model	
Remove object group	17-23	Backup version	5-40
Reopen model	5-49	Change.....	14-7
Save configuration.....	13-64 , 13-65	Closing	5-52
Save definition as	13-54	Derivation of ordering list.....	5-11
Save external.....	5-31	Derivation of parts list.....	5-11
Save model.....	5-53	Inserting model.....	5-50
Save model as.....	5-56	Moving objects to another layer ..	5-24
Select.....	4-29	New model	5-44
Select by attribute.....	4-25 , 4-32	Open backup version	5-48
		Opening	5-45
		Reopen.....	5-49
		Save	5-53

Saving as	5-56
Set active	14-4
Setting active	5-21
Symbolic visualization	5-16
Tooltips on icon	14-1
Model Inquiry.....	2-23
AQL	2-27
Model names	
Filter.....	5-41
Model selection box.....	13-13 , 13-54 , 13-87
Model window	14-1
Arrange.....	14-5
Change size.....	14-4
Icon	14-1
Open	14-3
Mode-Menu	3-18
Modification	
Formula string.....	13-108
Modify	
Parameter list	4-11
Mouse	3-23
Move (UDA)	
Menu command.....	13-40
Move (UDOT)	
Menu command.....	13-18
Move objects	
Menu command.....	5-24
Move sublayer	
Menu command.....	5-26
Move to layer	
Menu command.....	4-32 , 4-34
Multiple elevation design.....	2-28

N

Name	
Changing	4-20
Creating	4-17
Deleting	4-20

Displaying	4-18
Layer.....	5-23
Menu command.....	4-34
Name (UDA)	
Menu command.....	13-39
Name (UDOT)	
Menu command.....	13-18
name by action	
UDA attribute.....	13-32
Name selection box.....	4-18
New model	5-44
Next	
Menu command.....	4-33
Nilprim	
Object type	7-8
no protocol	
UDA attribute.....	13-32
normal	
UDOT instance type	13-19
Number	
Object type	7-8
Numbers	
Input.....	3-32

O

Object	
Action types of transformed	
objects.....	8-23
Basic objects	7-6
Characteristics.....	13-36
Color.....	14-12
copying	11-2
copying by defining a mirroring	
axis.....	11-6
copying by defining copy vector ..	11-4
Creating	7-1
deleted attribute.....	9-1
deleting	9-1

deleting different object types	9-5	Removing	17-23
deleting in delete mode	9-3	Object name	4-23
deleting in edit mode	9-7 , 9-9	Changing	4-20
deleting same object types	9-4	Creating	4-17
Direction	7-14	Deleting	4-20
duplicating	11-9	Displaying	4-18
Editing	8-6	Object parameter	4-2
Editing action	8-14	Input	4-3
Editing parameter	8-6	Redefining	8-17
Editing property	8-6	Object selection	
Edting in current mode	8-28	Single mode	4-33
Explicit creation	7-1	Object sets	
Geometry	7-14	deleting	9-4
Identification	4-21	Object type	7-6
invalid	21-3	Adopting definition for UDO	13-22
manipulating	10-1	Angle	7-7
Moving to another layer	5-24	Circle	7-14
Orientation	7-14	Column name	7-7
Quick edit of object closest to		Conception	13-10
cursor position	8-28	Contour	7-15
Quick edit of object last created ..	8-29	Coord	7-7
Redefining	8-15	Coordinate system	7-23
Redefining object set	8-22	Copy vector	7-22
Relationship between individual		Defining	13-11
objects	8-31	deleting in edit mode	9-8
Selecting	4-21	Designing	13-14
Separating	8-25	Ellipse	7-14
Snap to grid point	14-9	Group	7-29
Standard color	14-13	Inquiring	8-4
Table	13-118	Length	7-7
updating	8-1	Line	7-14
Object group	3-12 , 4-35	Loading	13-21
Changing	4-39	Measure	7-19
Creating	4-36 , 17-22	Number	7-8
Deleting	4-39	Plane	7-17
dialog box	17-22	Point	7-14
Displaying	4-39	Posmeas	7-8
Editing	17-23	Posttext	7-8 , 7-9
Hiding	17-24	Prop	7-9

Rawval.....	7-9
Spline.....	7-14
String.....	7-10
Symbol.....	7-20
Table.....	7-30
Table index.....	7-30
Text.....	7-18
User.....	7-31
Variable.....	7-27 , 13-91
View.....	7-10
Visualization methods.....	14-8
Wcoord.....	7-10
Weldfork.....	7-10
Octal input method.....	3-30
On context	
Help menu.....	3-37
Online Documentation.....	3-37
Online Help.....	3-36
Online Short Help	
see Tooltips	
Operating modes.....	2-18
Create mode.....	2-18
Delete mode.....	2-18
Edit mode.....	2-18
Temp mode.....	2-18
Operators.....	3-32
Formula.....	13-98
Optional parameter.....	4-8
Reset.....	4-9
Set.....	4-9
Optional property.....	4-8
Options	
Ruler on/off.....	3-15
Options-Menu.....	3-22
Ordering list	
Derivation from model.....	5-11
Output	
bill of material.....	16-7

Overlapping	
Window arrangement.....	14-5

P

Parameter.....	4-5 , 4-6
Allocation of attributes for UDA.....	13-41
Appending parameters used by AQL to UDA.....	13-37
Appending to UDA.....	13-35
Changing the sequence for UDA.....	13-40
data assignment.....	2-20
Designation for UDA.....	13-39
Determining parameter icons	
for UDA.....	13-39
Editing.....	8-6
Furnishing.....	7-4
Inquiry.....	8-5
Mandatory parameter.....	4-7
Modification by variable.....	13-95
Optional setting for UDA.....	13-40
Supply by the table.....	13-139
UDA.....	5-3
Parameter (in complex design geometry)	
Editing.....	8-9
Parameter (of create action)	
Editing.....	8-7
Parameter (of manipulator action)	
Editing.....	8-8
Parameter input.....	4-5
Parameter list.....	4-10
Create mode.....	4-11
Extending.....	4-13
Shortening.....	4-13
Parameter type	
Alternative parameter.....	4-14
Optional parameter.....	4-8
Parameter list.....	4-10
Parameter types.....	4-7

Parent directory		Previous	
Enter in dialog box.....	5-42	Menu command.....	4-34
Partial projects	5-10	Priority	
Parts list		Visualization	14-15
Derivation from model	5-11	Product	
Parts structure.....	5-13	Specifying.....	5-6
Path names	13-54	Structuring	5-13
Piece parts		Product development process	
Detailing.....	5-10	Structuring	5-6
Plane	7-17	Product model	
Hiding	14-15	Structure visualization	14-20
Plot format.....	18-10 , 18-13	Visualization	5-16
Plot format box.....	18-10 , 18-13	Visualize	14-18
Plot output file	18-12 , 18-15	Product structure	
Plot output format.....	18-12 , 18-15	Structure window.....	14-2
Plotting	18-1	Production data	
batch mode.....	18-9	Taking from the model.....	5-12
color.....	18-7	Production planning	5-12
fixed extracts	18-13	Program error	
line thickness	18-4	report	21-7
text.....	18-3	Program errors	21-5
variable extracts	18-10	Program finish.....	3-41
Point		Program interruption	3-42
Object type	7-14	Program start	3-2
Popup menu		Programming language.....	2-27
second column (selection).....	4-28	Project planning	5-9
third column (selection)	4-29	Prompt, list parameter.....	4-11
Window frame.....	14-3	Properties	
Posball		Optional property	4-8
Object type	7-8	Property	4-5
Posmeas		Appending for UDOT	13-16
Object type	7-8	Color	14-12
post action		Designation for UDOT	13-18
UDA attribute	13-32	Determining icon for UDOT	13-17
Posttext		Editing	8-6
Object type	7-9	Furnishing.....	7-4
pre action		Line style	14-10
UDA attribute	13-32	Mandatory property	4-7
Preview graphics.....	5-39	Optional setting for UDOT	13-18

UDOT	13-11
with UDOs	5-3
Z value	14-15
Property (in complex design geometry)	
Editing	8-9
Property (of create action)	
Editing	8-7
Property (of manipulator action)	
Editing	8-8
Property types	4-7

Q

Query	
Parameter list	4-11
Quick edit	
Menu command	8-28
of object closest to cursor	
position	8-28
of object last created	8-29
Quick edit mode	3-19 , 8-28
Discontinuing	8-29

R

Rawval	
Object type	7-9
Real number	
Object type	7-8
Real numbers	
representation in memory	22-1
Rectangle functions	
Color	14-13
Line style	14-10
Z value	14-15
Redefine	
Menu command	8-15
Redefinition	11-9
Object	8-15

Object parameter	8-17
to simple actions	8-22
Relation	
between coordinate systems	7-25
Remove action group	
Menu command	17-21
Remove menu	
Menu command	17-17
Remove object group	
Menu command	17-23
Reopen model	
Menu command	5-49
Required parameter	
duplication vector	11-13
Result object	
Determining for UDA	13-31
Return key	3-31
Right mouse button	
Popup menu of window frame	14-3
Roughness	
Object type	7-9
Ruler	3-13

S

S(electable)	
Layer state	5-22
Save configuration	
Menu command	13-64 , 13-65
Save definition	
Menu command	13-54
Save definition as	
Menu command	13-54
Save external	
Menu command	5-31
Save model	
Menu command	5-53
Save model as	
Menu command	5-56

Saving	
Model.....	5-53
Model as.....	5-56
UDA.....	13-54
UDA configuration	13-64
UDOT	13-54
UDOT configuration.....	13-64
Screen area.....	3-7
Scrollbars	3-29
Scrolling	
Menu area	3-35
Search operations.....	13-117
Section views of 3D models	19-2
Select	
Menu command.....	4-29
Select by attribute	
Menu command.....	4-25 , 4-32
Select by name	
Menu command.....	4-18 , 4-31
Select in graph. inst	
Menu command.....	4-32
Select in user	
Menu command.....	4-26
Select indirect	
Menu command.....	4-28
Selectable	
Layer.....	14-7
selectable	
UDOT instance type	13-19
Selection	4-21
Action.....	7-3
Action group	7-3
by attribute.....	4-25
by distance	4-22
by name.....	4-23
Group mode.....	4-31
in rectangle.....	4-22
in user.....	4-26
Indirect.....	4-28
Menu	7-3
Single mode	4-33
Selection box	4-36
Selection mechanisms	4-21
Selection modes	4-31
Separating objects	8-25
Session	
logging.....	21-8
Set active	
Menu command.....	5-21
Set library	
Menu command.....	17-28
Set optional (UDA)	
Menu command.....	13-40
Set optional (UDOT)	
Menu command.....	13-18
Set parameter	
Menu command.....	13-36
Set property	
Menu command.....	13-16 , 13-17
Set result object	
Menu command.....	13-31
Set result object (UDA)	
Menu command.....	13-31
Short Help	
see Tooltips	
Show ancestors	
Menu command.....	4-29 , 4-34
Show environment	
Menu command.....	4-34
Show sons	
Menu command.....	4-29 , 4-34
Single mode	
Menu command.....	4-32
Single object	
deleting in delete mode	9-3
Sketch	
Snap to grid point	14-9
Z value.....	14-16

Sketch plane of a 3D model	19-3	Standard production methods	13-117
Sketcher	8-30 , 15-2	Standardization	2-25 , 13-1
cross section shading	15-23	Application concept	5-14
designing with	15-13	Data	13-122
Sketching		of components	13-117
application features	15-3	Start options	3-3
capture angle	15-25	Starting the program	3-2
capture range	15-3	Status	
contour	15-25 , 15-27	Menu command	5-22 , 5-27
contours and planes	15-2	STOP-Icon	3-13
notes on the application	15-29	String	
planes	15-25	length	22-2
priority of objects in the search		Object type	7-10
range	15-5	Structure	
properties	15-29	Substructure	14-22
symmetric profile	15-31	Window type	14-18
virtual points	15-5	Structure element	14-22
x range	15-3	Change	14-24
y range	15-4	Color	14-13
Snag tests	2-32	Creation	14-21
SolidJoin	19-1	Delete	14-24
Sons selection	4-24	Graphical display	14-25
Special characters	3-33	Set active	14-22
Specifying		Structure model	
Product	5-6	Creating and processing	14-20
Spline		Structure viewport with 3D parts	19-4
Object type	7-14	Structure window	14-18
Standard		Window type	14-2
Window arrangement	14-5	Structuring	
Standard calculations	13-2 , 13-117	Product development process	5-6
Standard color	14-13	Top Down/Bottom Up	5-13
Standard components		Sublayer	
Application of tables	13-146	Visibility	14-7
Standard functionality	13-2	Subobjects	
Standard geometry	13-2	Menu command	13-8
Standard library		Sub-project	
Enter in dialog box	5-42	In complete project model	5-11
Setting	17-28	Substructure	14-22
Standard manufacturing methods	5-15		

suppress properties	
UDA attribute	13-32
Suppression technique.....	14-14
Switch the menu.....	3-11
Symbol	
Object type	7-20
Z value.....	14-15
Symbols	12-1 , 12-27
attributes of symbol balloons	12-32
conical tapers	12-28
general symbols	12-1 , 12-27 , 12-28
geometric tolerances	12-28
note and name lines	12-28
path and type of the weld	12-34
reference object.....	12-28
section line characteristics	
.....	12-28 , 12-29
surface characteristics.....	12-28
symbol balloons	12-27 , 12-31
welding symbols	12-27 , 12-33
System basics	2-1
System configuration file	17-6
System data	
conventions	22-1
System errors.....	21-6
System handling.....	4-1
System philosophy	2-1
T	
TAB key.....	3-31
Table	13-117
Application to standard	
components	13-146
Applications	13-117
Creating	13-124
Creating an index	13-136
Data extraction	13-139
Database table	13-117 , 13-128
Design phase	13-119
Designing tables.....	13-122
File table.....	13-117
File table (example).....	13-134
File tables	13-127
Index.....	13-118
Internal table.....	13-117 , 13-127
Internal table (example).....	13-134
Matrix.....	13-122
Nested table	13-117
Nesting of tables.....	13-142
Object type	7-30
Preparatory phase.....	13-118
Structure.....	13-122
Subtable	13-117 , 13-129
Using	5-5
Table editor	13-117
Table index	7-30
Tables	
Removing dependencies via	
separation	8-27
Tables function.....	2-30
Temp-Mode.....	2-18 , 3-18
Text	
Object type	7-18
Z value.....	14-15
Text box	3-15
Text editing	3-31
Text input	
editing.....	3-31
Tiled horizontally	
Window arrangement	14-5
Tiled vertically	
Window arrangement	14-5
Title bar	3-16
tmp	
UDA attribute.....	13-32
Tolerance	
Object type	7-9

Tooltips.....	3-38
for icons	3-38
for objects	3-39
Iconized model window	14-1
Model name	14-1
Top object	
Menu command.....	13-8
TopDown structuring	5-13
Transformation techniques.....	11-1
Transformations	13-117
trim_line2lines	
action	10-2
trim_lineonline	
action	10-2
trimming	
line up to a line	10-2
lines	10-2
two lines up to an intersection	
point	10-2

U

UDA.....	13-2 , 13-4
Action types	5-3
Allocation of attributes	13-32
Appending AQL programs	13-33
Appending icon to the user	
interface	13-55
Appending icons to new action	
groups	13-58
Appending icons to new menus.	13-61
Appending parameters used	
by AQL	13-37
Changing the sequence of	
parameters.....	13-40
Characteristics.....	13-82
Concepting	13-25
Creating an icon	13-46
Creating manipulated objects	13-90
Defining	13-26
Definition example	13-73
Designation.....	13-27
Designation of parameters	13-39
Designing.....	13-29
Determining effect objects.....	13-31
Determining parameter icon	13-39
Determining result object.....	13-31
Determining the original model..	13-27
Edit definitions	13-45
Editing an icon	13-48
Extending a manipulated UDO ..	13-89
Extending the resulting UDO	13-87
Loading.....	13-44
Model-internal.....	13-54
Moving icon	13-63
Optional setting of parameters	13-40 ,
13-41	
Parameter.....	5-3
Relationship to a manipulated	
parameter of an UDO.....	13-88
Relationship to an UDO para-	
meter.....	13-84
Relationship to an UDO result...	13-85
Saving.....	13-43 , 13-54
Saving the configuration	13-64
Using	5-3
UDA definition box	13-29
Uda-Menu	3-21
UDO	
Access the individual object	13-24
Adopting UDOT definition.....	13-22
Changing in model.....	13-8
Converting into layer.....	13-24
Creating in model	13-7
Creating with UDA	13-7
Placing.....	13-5
Properties	5-3
Saving the configuration	13-64

Using	5-3	Moving icon	13-63
Using in model	13-9	Nesting	13-3 , 13-67
Udo-Menu	3-21	Optional setting of properties	13-18
UDOT	13-2	Properties	13-11
Accessing via base object	13-8	Saving	13-54
Adopting definition for UDOT	13-22	Structure	13-10
Allocation of attribute	13-19	UDOT definition box	13-14
Appending icon to the user		Undo	
interface	13-55	deletion	9-11
Appending icons to existing		menu command	9-11
action groups	13-56	Undo delete	
Appending icons to new action		menu command	9-11
groups	13-58	Undo-Menu	3-17
Appending icons to new menus.	13-61	Update facility	8-1
Appending property	13-16	Update location table	
Change the name	13-15	Menu command	17-26
Changing the sequence of		Update of a 3D model	19-2
properties	13-18	User	
Class affiliation	13-10	Object type	7-31
Classify, to	13-12	User action	
Combining the entry of value		Concepting	13-25
properties	13-19	Defining	13-26
Conception	13-10	Designation	13-27
Creating an absolute instance	13-7	Designing	13-29
Creating an icon	13-46	Determining the original model..	13-27
Creating an instance	13-7	User errors	21-2
Defining	13-11	User icon	
Defining the view	13-15	Changing the sequence	13-18
Definition example	13-73	User interface	2-18
Designate, to	13-12	Appending UDA icon	13-55
Designation of properties	13-18	Appending UDOT icon	13-55
Designing	13-14	desktop	3-8
Determining icon	13-16	Drawing area	3-8
Determining property icon	13-17	Introduction	3-1
Edit definition	13-23	Keypad tool	3-30
Editing an icon	13-48	Layer box	3-15
Identification number	13-2	Main menu	3-8
Loading	13-21	Menu bar	3-16
Model-internal	13-54	Menu extension area	3-15

Menu-Icon.....	3-15	Changing the sequence of parameters.....	13-40
Message box	3-15	Characteristics.....	13-82
Ruler	3-13	Creating an icon	13-46
Screen areas	3-7	Creating manipulated objects	13-90
Text box.....	3-15	Definition example	13-73
Title bar.....	3-16	Designation of parameters	13-39
User object type		Determining effect objects.....	13-31
Adopting definition for UDO.....	13-22	Determining icons.....	13-30
Allocation of attribute	13-19	Determining parameter icons	13-39
Appending property	13-16	Determining result object.....	13-31
Combining the entry of value properties	13-19	Edit definition.....	13-45
Conception	13-10	Extending a manipulated UDO ..	13-89
Defining	13-11	Extending the resulting UDO	13-87
Defining view	13-15	Loading.....	13-44
Designation of properties.....	13-18	Model-internal.....	13-54
Designing.....	13-14	Moving icon	13-63
Determining icons.....	13-16	Optional setting of parameters	13-40 , 13-41
Determining property icon	13-17	Parameter.....	5-3
Loading.....	13-21	Relationship to a manipulated parameter of an UDO.....	13-88
Nesting of user object types	13-67	Relationship to an UDO parameter.....	13-84
Optional setting of property	13-18	Relationship to an UDO result ...	13-85
User-defined action	13-2 , 13-4	Saving.....	13-54
Action types	5-3	Saving the configuration	13-64
Allocation of attributes	13-32	Using	5-3
Appending AQL program.....	13-33	Using in model.....	13-9
Appending icon to the user interface	13-55	User-defined object	
Appending icons to existing action groups	13-56	Access the individual object	13-24
Appending icons to new action groups	13-58	Adopting UDOT definition.....	13-22
Appending icons to new menus.	13-61	bomheader.udo	16-24
Appending parameters	13-35	bomline.udo	16-24
Appending parameters used by AQL	13-37	Changing in model.....	13-8
bomheader.uda	16-24	Converting into layer.....	13-24
bomline.uda	16-24	Creating an icon	13-46
		Creating in model	13-7
		Properties	5-3

Using	5-3	Viewing	
User-defined object type	13-2	Menu command.....	4-32 , 4-34
Appending icon to the user interface	13-55	Viewing a 3D viewport	19-4
Appending icons to existing object groups	13-56	Viewing>Quick edit	
Appending icons to new action groups	13-58	Menu command.....	8-28
Appending icons to new menus	13-61	Viewport	
Change the name	13-15	Color of model in window frame ..	14-3
Classify, to	13-12	Viewport type	17-12
Definition example	13-73	Visible	
Designate, to	13-12	Automatic line style	14-11 , 14-17
Edit definition	13-23	Layer	14-7
Model-internal.....	13-54	Sublayer	14-7
Moving icon	13-63	Visualization	
Saving.....	13-54	Deleting a structure element	14-24
Saving the configuration	13-64	Editing a structure element.....	14-24
		Layer	14-18
		Planes	14-16
		Priority of objects.....	14-15
		Visualization methods	14-2 , 14-8
		Visualization of layers	
		Modifying	14-22
		Visualization techniques	14-1
V		W	
V(isible)		Wcoord	
Layer state.....	5-23	Object type	7-10
Value property		Weldfork	
Combining the entry for UDOTs	13-19	Object type	7-10
Variable	13-91	Window	
Area calculation	13-114	Arrange.....	14-5
Creating formula string	13-107	Change size	14-4
for parameter update	13-95	Close	14-3
Object type	7-27	Maximize	14-3
Variant construction	13-95	Minimize	14-3
Vertical menu bar	3-8	Open.....	14-3
View		Overlapping	14-5
Define with F5-F9	14-9	Set active.....	14-4
Defining for UDOT	13-15	Standard.....	14-5
Function keys F5-F9.....	14-9		
Object type	7-10		
View object.....	14-3		

Tiled horizontally.....	14-5	Desktop	14-1
Tiled vertically.....	14-5	Working directory	
Visualization methods.....	14-8	Enter in dialog box.....	5-42
Window frame		Working window popup menu	4-21
Functionalty	14-2	Working with models.....	5-1
Popup menu	14-3	Works standard parts	5-14
Window type.....	14-7		
Geometric window	14-2	Z	
Structure	14-18	Z value	14-15, 15-33
Structure window	14-2	Input.....	14-16
Window-Menu	3-18	Rectangle functions	14-15
Windows.....	14-1	Sketch.....	14-16
Working area			