



Parametric Design with Creo Elements/Direct Drafting Windows User Interface

Creo Elements/Direct Drafting 20.4.4.0

Copyright © 2023 PTC Inc. and/or Its Subsidiary Companies. All Rights Reserved.

Copyright for PTC software products is with PTC Inc. and its subsidiary companies (collectively “PTC”), and their respective licensors. This software is provided under written license or other agreement, contains valuable trade secrets and proprietary information, and is protected by the copyright laws of the United States and other countries. It may not be copied or distributed in any form or medium, disclosed to third parties, or used in any manner not provided for in the applicable agreement except with written prior approval from PTC. More information regarding third party copyrights and trademarks and a list of PTC’s registered copyrights, trademarks, and patents can be viewed here: <https://www.ptc.com/support/go/copyright-and-trademarks>

User and training guides and related documentation from PTC are also subject to the copyright laws of the United States and other countries and are provided under a license agreement that restricts copying, disclosure, and use of such documentation. PTC hereby grants to the licensed software user the right to make copies of product documentation and guides in printed form, but only for internal/personal use and in accordance with the license agreement under which the applicable software is licensed. Any copy made shall include the PTC copyright notice and any other proprietary notice provided by PTC. Note that training materials may not be copied without the express written consent of PTC. This documentation may not be disclosed, transferred, modified, or reduced to any form, including electronic media, or transmitted or made publicly available by any means without the prior written consent of PTC and no authorization is granted to make copies for such purposes.

UNITED STATES GOVERNMENT RIGHTS

PTC software products and software documentation are “commercial items” as that term is defined at 48 C.F.R. 2.101. Pursuant to Federal Acquisition Regulation (FAR) 12.212 (a)-(b) (Computer Software) (MAY 2014) for civilian agencies or the Defense Federal Acquisition Regulation Supplement (DFARS) at 227.7202-1(a) (Policy) and 227.7202-3 (a) (Rights in commercial computer software or commercial computer software documentation) (FEB 2014) for the Department of Defense, PTC software products and software documentation are provided to the U.S. Government under the PTC commercial license agreement. Use, duplication or disclosure by the U.S. Government is subject solely to the terms and conditions set forth in the applicable PTC software license agreement.

PTC Inc., 121 Seaport Blvd, Boston, MA 02210 USA

Contents

Preface	9
Introduction.....	11
Roadmap	12
What is Parametric Design?	12
Starting Parametric Design.....	13
Basic Working Methods.....	13
Learning to Use Parametric Design.....	15
Generating Variations	17
Roadmap	18
Load the Master Part.....	18
List the Current Constraints	19
Change Parameter Values.....	19
Solve for the Variation	20
Saving and Restoring the Parameter Value Table.....	21
Working with Parametric Design.....	23
Roadmap	24
Part Preparation	24
Assigning Constraints	28
Solve	35
Application	39
Constraint Types	45
Constraint Types	46
Angle	46
Collinear	48
Dimension.....	48
Distance	50
Fillet	53
Horizontal	54
Mirror.....	55
Parallel	56
Perpendicular.....	58
Point on	59
Reference Element.....	60
Reference Point.....	61
Same Distance.....	61
Same Size	62
Size	63
Slope.....	63

Symmetry Line	64
Tangent.....	65
Vertical	67
X-Distance and Y-Distance	67
Parameters.....	69
Creating Parameters.....	70
Assigning Parameters.....	71
Editing Parameter Definitions	72
Setting Parameter Values.....	76
Dimension Driven Modification	79
Roadmap.....	80
Overview.....	80
Viewing Parametric Dimensions.....	80
Display Attributes of Parametric Dimensions.....	81
Modifying Parametric Dimension Values.....	82
Swapping Parametric Dimensions.....	83
Design Intent Capture.....	85
Roadmap	86
Overview.....	86
Flexibility.....	86
Enabling, Disabling and Inquiring.....	86
Elements Affected	87
Constraint Creation.....	87
Operations on Existing Geometry.....	89
Advanced Topics and Tips	93
Roadmap.....	94
Geometry Cleaning.....	94
A Simple Sketch Input Implementation	95
Zone Gymnastics.....	98
Parametric Control of Splines	99
Rigid Bodies: Applications and Tips.....	101
Implementing Replication in Parametric Design	102
Allowing Rotation.....	103
Working with Multiple Parts.....	104
Using Modify With Parametric Design.....	104
Automatic Constraint Generation	106
A Short Description of the Solver.....	108
How Expressions are Evaluated	108
Invisible Geometry.....	109
Examples	111
Roadmap.....	112
Introduction.....	112
Example 1: Introductory Design Example	113
Example 2: Symmetry Lines	122
Example 3: Two Views	129

Example 4: Macros	133
Example 5: Rigid Bodies	139
Appendix A. Tutorial	143
Exercise 1	144
Exercise 2	147
Hints and Tips	150
Appendix B. Parametric Design Command Syntax	153
PD_AUTO_ANGLE_TOLERANCE function	155
PD_AUTO_SAME_DISTANCE_TOLERANCE function	155
PD_AUTO_SYMMETRY_COLOR function	155
PD_AUTO_SYMMETRY_LINE function	155
PD_AUTO_SYMMETRY_LINETYPE function	156
PD_AUTO_TANGENT_TOLERANCE function	156
PD_AUTO_ZERO_DISTANCE_TOLERANCE function	156
PD_CLEAN_DRAWING command	156
PD_DEFAULT_DIM_COLOR command	157
PD_DEFAULT_DIM_TEXT_SIZE command	157
PD_FIX command	157
PD_FREE command	161
PD_HIDE_DIMENSIONS command	163
PD_INFO_CONSTRAINT function	163
PD_INFO_ELEMENT function	163
PD_MAKE_DIMENSIONS command	163
PD_MODIFY_DIMENSION command	164
PD_NEW_C_LINE_COLOR function	164
PD_NEW_C_LINE_LINETYPE function	164
PD_NEW_C_LINE_VISIBILITY function	165
PD_NEW_POINT_COLOR function	165
PD_NEW_POINT_LINETYPE function	165
PD_NEW_POINT_VISIBILITY function	165
PD_PARAM_ADD command	165
PD_PARAM_FIX command	166
PD_PARAM_INQ function	166
PD_PARAM_REMOVE command	167
PD_PARAM_SAVE function	167
PD_PARAM_SHOW function	167
PD_PREVIEW_COLOR function	167
PD_RESOLVE command	168
PD_RESOLVE_MERGE_TOLERANCE function	168
PD_RIGID_ADD command	169
PD_RIGID_REMOVE command	169
PD_RIGID_REFRESH_TABLE function	169
PD_RIGID_SHOW function	169
PD_SHOW function	170
PD_SHOW_CLEAR function	171
PD_SHOW_COLOR function	172

PD_SHOW_LABEL_SIZE function.....	172
PD_SHOW_MOVE_TEXT function	172
PD_SHOW_USE_POSTFIX function	172
PD_USE_DIMENSION command.....	173
PD_ZONE_ADD command	173
PD_ZONE_REMOVE command.....	173
PD_ZONE_SHOW function.....	174
parameter name	174
Appendix C.Parametric Design Defaults.....	175
Setting Icon Defaults From the Screen Menu.	176
The PD_Defaults File.....	177
Index.....	181



Preface

This manual describes the Creo Elements/Direct Drafting Parametric Design & Drafting System (Parametric Design). Parametric Design adds full constraint-based and parametric modeling capabilities to Creo Elements/Direct Drafting without imposing limitations on your current working methods. Any existing Creo Elements/Direct Drafting drawing can be used with Parametric Design, and drawings created by the software are fully compatible with Creo Elements/Direct Drafting.

Parametric Design addresses the needs of the Design Engineer or Industrial Designer as well as the Draftsperson. For the Designer, the software provides a complete set of tools for assigning and evaluating sets of parameters and constraints on an existing part. For the Draftsperson, Parametric Design provides an efficient interface for generating variation parts from pre-constrained "master" parts.

Any Parametric Design user should be familiar with Creo Elements/Direct Drafting itself for the best results. If you are not familiar with Creo Elements/Direct Drafting, we recommend that you refer to the Creo Elements/Direct Drafting User's Guide manual before continuing with Parametric Design.

1

Introduction

Roadmap.....	12
What is Parametric Design?	12
Starting Parametric Design	13
Basic Working Methods	13
Learning to Use Parametric Design.....	15

Roadmap

Read this chapter first. It overviews the Parametric Design software, introduces several important terms, and directs you to the sections that tell you how to start Parametric Design and how to use it.

What is Parametric Design?

The Creo Elements/Direct Drafting Parametric Design & Drafting System software (Parametric Design) adds powerful parametric and constraint-based modeling capabilities to Creo Elements/Direct Drafting.

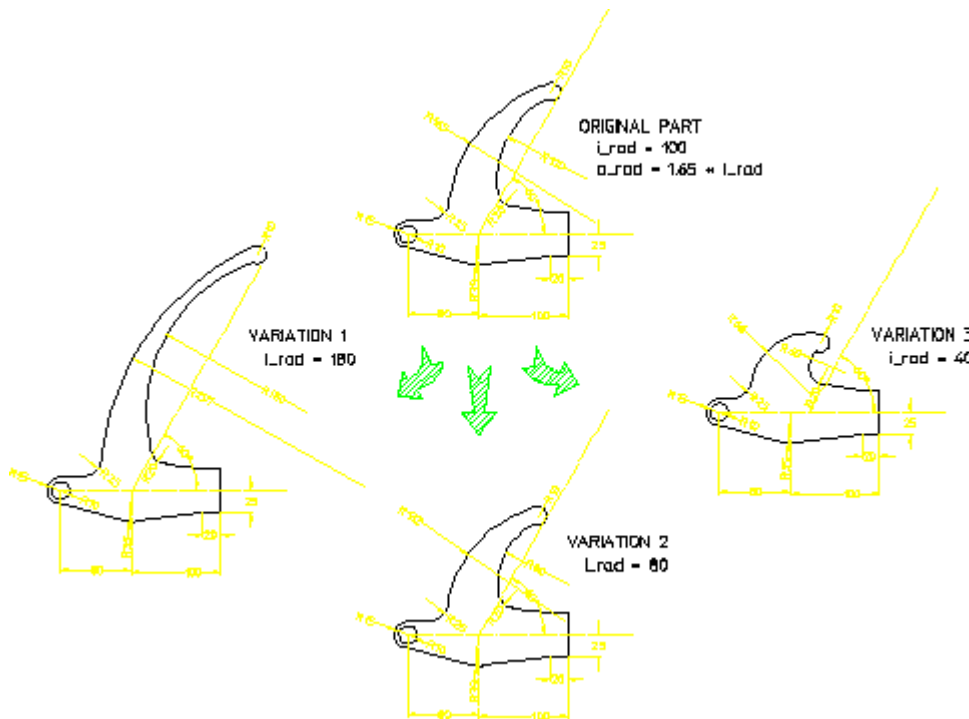
Parametric modeling allows you to attach "names" or parameters to drawing dimensions, such as radii or lengths, and then to change the drawing by changing the numerical values or expressions associated with these parameters.

Constraint-based modeling allows you to specify geometric properties and relationships. The software then modifies the part so that these constraints are satisfied. For example, you can say "these two elements should be tangent," or "this line should be horizontal," or "this dimension should have such-and-such a value."

Parametric Design combines both types of modeling into a module that allows you to quickly and easily perform modifications that are beyond the scope of the basic Creo Elements/Direct Drafting **Modify** toolset.

With Parametric Design tools you can use any existing part as a template for generating variations or a family of parts that otherwise would have to be drawn from scratch. [Figure 1. Master Part & Three Variations on page 13](#) below shows a "master" part and three variations that were generated with Parametric Design. Once constraints and parameters were defined for the master part with Parametric Design, each variation was generated by modifying only one dimension on the master.

Figure 1. Master Part & Three Variations



Parametric Design provides these capabilities without changing existing Creo Elements/Direct Drafting functionality or procedures. Loading the Parametric Design module does not change any existing Creo Elements/Direct Drafting menus or commands, so your drawing methods need not change to use it. You can use any existing Creo Elements/Direct Drafting part with Parametric Design, and parts generated by Parametric Design will be fully compatible with Creo Elements/Direct Drafting. The constraint and parametric information generated by the software is added directly to the master part (as info text), and is automatically stored/recovered along with the part.

Starting Parametric Design

In Creo Elements/Direct Drafting, click on **Parametric** to display the Parametric Design menu.

Basic Working Methods

For the Draftsperson

Parametric Design gives the Draftsperson a simple, efficient interface for generating variations from pre-constrained parts. When a pre-constrained part is loaded into Creo Elements/Direct Drafting, all parameters that can accept value

input appear in a table on this menu along with their current values. To generate variations of the part, the Draftsperson sets the appropriate values for any or all of these parameters and then issues a **Solve** command. Parametric Design then generates the variation, which can be saved, copied, or modified further.

For the Designer

The Designer or Engineer uses Parametric Design to assign, evaluate, and modify constraints and parameters on that part. Once the constraints and parameters are in place, **Solve** can be used to generate variations. The typical steps needed to constrain a part are as follows:

Preparation

Model inconsistencies can cause unexpected results in Parametric Design and should usually be resolved before assigning constraints to a part. The **Clean** command checks parts for inconsistencies such as extra points and duplicated or stacked geometry; and optionally adjusts or removes the offending elements.

Parametric Design operates on drawing elements within its zone. The zone may contain an entire part or selected elements of a part. The **Zone** commands select elements for the zone.

Assign Constraints

Before you can use Parametric Design to generate variations from a part, constraints must be assigned to the part. These constraints must be sufficient to tell the parametric solver exactly what to do with each element in the zone to generate the variation. If elements are not constrained or are incorrectly constrained, the solver will not be able to generate the variation.

Constraints may be assigned manually or automatically. Typically, the designer uses a combination of methods as follows:

1. Assign a full set of constraints that completely define the part's current configuration:
 - a. Manually assign a few reference constraints and perhaps symmetry line constraints to important elements in the part.
 - b. If large sections of the part's geometry can move together as a single unit, collect these sections into rigid bodies using the **Generate Rigids** command. This can greatly reduce the number of constraints needed on the part.

-
- c. Use **Complete** to automatically assign a reasonable set of constraints to the remaining elements.
 2. If you used **Solve** to generate variations, check the potential result(s) by means of **Preview**. You may need to modify certain constraints to better reflect your design intentions.
 3. Modify constraints as needed to generate the desired variation from the current configuration. You may add or delete constraints or change the values associated with existing constraints to come up with the variation you want.

Use the **Assign** action in the **Generate Constraints** dialog box to manually assign or modify constraints. Usually, you click the **Assign** radio button, select a constraint type (such as: **Angle**, **Dimension**, **Size**), and then select the drawing element(s) to receive the constraint. Iconic call-outs are used to label constraints as they are assigned.

Angle, distance, and dimensional constraints, among others, may be given parametric as well as numerical values when you assign them. A parametric value may be a text label, an Creo Elements/Direct Drafting expression, or a macro name which is resolved at solve time to generate a numeric value for the constraint. Parameters can perform math functions and can reference other parameters when they are resolved. You aren't required to use parametric constraints, but they are an especially powerful feature of the software.

Solve for the Variation

After you have constrained your part, use the **Solve** command to solve the set of constraints and generate the variation. Parametric Design will attempt to create new geometry that fits the constraints by adjusting sizes, locations, and relationships of elements in the part. If the current set of constraints is inconsistent or incomplete, the solver can selectively highlight the offending areas. A variation can be previewed before actually being generated, and it can be generated either as a copy of or as a replacement for the original geometry.

Learning to Use Parametric Design

The best way to learn Parametric Design is to work with the software, consulting this manual as needed for clarification. [Examples on page 111](#) provides several work-along examples that we recommend you try. To encourage you in this, we have included the parts used in the examples along with the Parametric Design software. You need only load the parts into Creo Elements/Direct Drafting to begin working with the examples.

In addition to the examples, the manual contains a command reference in [Parametric Design Command Syntax on page 153](#). The command reference is also available online through the Creo Elements/Direct Drafting help facilities.

2

Generating Variations

Roadmap.....	18
Load the Master Part	18
List the Current Constraints.....	19
Change Parameter Values	19
Solve for the Variation.....	20
Saving and Restoring the Parameter Value Table.....	21

Roadmap

Read this chapter to learn how to generate variations from parts that have already been constrained and parameterized with Parametric Design. To learn how to assign constraints and parameters, go on to [Working with Parametric Design on page 23](#) after you read this chapter.

Parametric Design provides an efficient interface for generating new parts (variations) from a master part that has already been processed through Parametric Design. Once you load a master part into Creo Elements/Direct Drafting, you need only supply a few parameter values and issue a **Solve** command to generate a particular variation. Again, the basic working method for generating a variation of a pre-processed master part is:

1. **Load** the master part.
2. Click **Current Constraints** in **Parametric**.
3. Change parameter values as needed in the Parameter Value Table.
4. **Solve** to generate the variation.

The following sections describe each step.

Load the Master Part

A master part can be any Creo Elements/Direct Drafting part that has Parametric Design constraint and parameter data associated with it. This data is generated by the Designer using Parametric Design and is automatically stored along with the part when it is saved to a file. Parametric Design data is loaded back into Creo Elements/Direct Drafting whenever you load the master part with the standard Creo Elements/Direct Drafting **Load** functions.

Parametric Design operates only on the current Creo Elements/Direct Drafting part. If you load a drawing that contains several parts, you must make sure that the part you wish to use as the master is the current Creo Elements/Direct Drafting part. Use the Creo Elements/Direct Drafting `Edit_part` command to select a particular part to be current.

To confirm that the part you've loaded has the necessary Parametric Design data associated with it to be a master part, check the parameter value table in the **Current Constraints** dialog box. If no entries appear in the table after you load a part or make it the current part, it cannot be used as a Parametric Design master from this menu.

List the Current Constraints

To create variations from a master part, display the **Current Constraints** dialog box. Then click **Advanced** to display the parameter value table. When you load a master part into Creo Elements/Direct Drafting, one or more entries appear in this table. Each entry contains a parameter name and a value. The parameter names correspond to elements in the master part that you can modify. The value entries represent the values that will be assigned to the corresponding drawing elements by the next **Solve** command.

Change Parameter Values

Use the parameter value table to specify how variations of the master part should be generated. When you change the value for a parameter listed in the table, you tell Parametric Design to modify all elements in the master part that are constrained to depend on this parameter. The constraints on the part determine exactly what modifications will occur. The changes are not actually made until you issue a **Solve** command.

To change the value for a parameter, simply select its entry from the table and enter the new value from the keyboard. If there are too many entries to view at once, use the scroll bar beside the table. The type of value you give (linear, angular, or point) depends on the type of the parameter. Usually, the parameter name will be descriptive enough to indicate what type of value is expected. It is easy to tell if a point value is expected, because point parameters take up two slots in the table. The upper slot contains the x coordinate of the point and the lower slot contains the y coordinate. Note that you still enter point values using the standard Creo Elements/Direct Drafting x,y format.

If you need more specific information about the correspondence between parameter entries in the table and the elements in the master part, click **Show** in **Current Constraints**. When you click **Show**, elements in the master that have parameters assigned to them are highlighted or marked with icons as described below.

If a parameter is associated with a dimension in the part, the parameter name is appended to the corresponding dimension text and both are highlighted (in CYAN by default). By examining the dimension, you can tell whether the parameter requires linear or angular input.

Parameter names may also be associated with several kinds of geometric constraints that have been assigned to the master part by a designer. If any of the parameters in the value table are associated with geometric constraints and click **Show**, these constraints are marked with icons as well as with the parameter name. Each type of geometric constraint has a unique icon to identify it. Icons and parameter labels are drawn in CYAN by default. Your current text font is used to

display the parameter names. Consult [Constraint Types on page 45](#) for complete information on geometric constraints, their icons, and the types of values to be supplied to each.

The **Show** highlighting is removed whenever you use **Solve** to generate a variation. Alternately, click the **Clear** radio button in **Generate Constraints** and then select **All Types** to remove the highlighting.

 **Note**

If the icons produced by Show are too large or too small, you can resize them using the Icon Setting dialog box.

Solve for the Variation

Once you've filled out the parameter value table, use the **Solve** command to create a variation with the desired values. **Solve** has three options:

- **Preview**
- **Keep**
- **No Keep**

Preview

By default, the solver works in **Preview** mode. The variation is computed, but is not actually created. Instead, an outline of the variation is superimposed over the current part. This gives you a good idea of what the new variation will look like without creating any new geometry or overwriting any existing geometry. If the preview indicates that the new variation is acceptable, you can actually create the new variation by re-solving with the **Keep** or **No Keep** options. The preview lines drawn by **Preview** are removed when you click **End**.

Keep

The **Keep** option generates the variation and inserts it into the current part as a modified copy of the original part. When you click **Keep**, the solver computes the variation and temporarily overlays it onto the current part so that you can see what it looks like. If the variation is not what you want, click **End** or **Undo** immediately to restore the original part. If the variation is OK, click a reference point for moving the new part, and then drag it to the desired location. After you've inserted as many copies as you want, click **End**. The overlaid variation is removed, and your original part is now restored.

Depending on how the master part was originally constrained, elements of the master such as construction geometry and dimensioning may not appear in copies made with **No Keep**. Also, the copies will not normally be affected by subsequent **Solve** commands. To change any of these behaviors, you will most likely need to modify the zone memberships and/or constraints in the master part. See [Working with Parametric Design on page 23](#) for details.

No Keep

Select the No Keep option if you want the solver to replace your current part with the variation. If the result of the solve is not what you expect, click **Undo** immediately to restore your original part. You can continue to modify the part by entering new values into the parameter table and re-solving.

Saving and Restoring the Parameter Value Table

The **Save** and **Input** buttons in the **Current Constraints** dialog box allow you to write out the current parameter value table to a file and later restore it. This feature lets you build a "library" of value tables that can be used to drive the master part. The advantage of this is that a series of standard parts based on a single master part can be stored as a single MI file (the constrained master part) and a collection of parameter value files. This arrangement takes up much less disk space than if an individual MI file were stored for each part in the series. To recreate any member of the series, you need only load the master part, **Input** the desired parameter value table, and **Solve**.

Note

When the parameter value table is written out, all Parametric Design parameters for the master part are written to the file. A number of these parameters may be expressions, which are used by the software, but do not appear in the table. Don't modify or remove these parameters from the output file!

Save the Parameter Value Table

To **Save** the current parameter value table to a file:

1. Click **Save** in **Current Constraints**.
2. Enter the name of the file to write out at the prompt.
3. Click **Save**.

Restore a Parameter Value Table

To **Input** a parameter value table:

1. If necessary, **Load** the master part that the value table file refers to. Use **EDIT_PART** to make sure that the master is the current part.
2. Click **Input** in **Current Constraints**.
3. Enter the name of the parameter value file to input.
4. To apply the values from this table to the part, click **Solve**.

Creating a Parameter Value Table by Hand

The file created when you **Save** a parameter value table is an ASCII macro file that contains one **PD_PARAM_FIX** command for each entry in the table, e.g., **PD_PARAM_FIX 'Width' 37.5**. It is relatively easy, therefore, to build your own parameter value tables outside of Creo Elements/Direct Drafting using an ordinary text editor. When writing a value table file by hand, keep the following points in mind:

- It may be very helpful later on if you include comment lines in your file that identify the master part this file applies to!
- Make sure that each **PD_PARAM_FIX** command in your file corresponds to a value parameter in the master part. If in doubt, **Save** a parameter value table with Parametric Design and use this file as a template for your own files.
- When Parametric Design saves a parameter value table, it includes several macros in the output file which ensure that the values are restored in the correct units. When you create this file by hand, you are responsible for ensuring that the units used in the file correspond to the current Creo Elements/Direct Drafting units when you restore it. One way to do this is to include the macros used by Parametric Design in your own files (use any system-created value table file as a template).

3

Working with Parametric Design

Roadmap.....	24
Part Preparation	24
Assigning Constraints	28
Solve.....	35
Application.....	39

Roadmap

Read this chapter to learn how to assign constraints to an Creo Elements/Direct Drafting part so that you or others can generate variations from it. If you only need to know how to generate variations from a pre-constrained part, skip this chapter and read [Generating Variations on page 17](#) instead.

This chapter is for the Designer or Engineer who wants to incorporate Parametric Design capabilities into his or her Creo Elements/Direct Drafting design environment. We begin by describing the Parametric Design tools that the Designer should be familiar with:

- Part preparation tools
- Constraint assignment tools
- Solving tools

The chapter concludes with several sections that cover the major applications for these tools: developing master parts for repeated use by yourself or others, and using Parametric Design as an extension of the Creo Elements/Direct Drafting **Modify** commands to do one-time adjustments of complex parts.

Part Preparation

Before you assign constraints to a part, two preparation steps should be considered. The first is to check the part for inconsistencies that can interfere with constraint generation. The second is to select or restrict the portions of the part to work with.

Clean, **Generate Rigids** and **Zone** handle preparation tasks.

Cleanup

Inconsistencies in a part (lines not quite parallel, small gaps between "tangent" objects, duplicate elements, etc.) are relatively common, but can cause problems for any program that must scan the part to extract information. To help eliminate drawing inconsistencies, Parametric Design provides several tools that can do limited "cleaning" of parts. The **Clean** command options are:

- **Points**
- **Duplicate**
- **Stacked**

These commands are functionally independent from the remainder of the Parametric Design software, and should be thought of as a "preprocess" for parts that are submitted for parametric design. The **Complete** and **Solve** commands have

additional part cleaning capabilities which are described in detail in [Advanced Topics and Tips on page 93](#). Each of the **Clean** options described below affects all geometry in the current part.

Points

Model points within a given distance of each other are merged, and the elements attached to the merged points are adjusted (moved, stretched, or rotated) as necessary to reflect the new positions of the points. For each set of points to be merged, **Points** chooses a location within the given tolerance and moves the points there. In general, there is no way to accurately predict or control which elements will be adjusted as a result of the merge. This should not be a limitation in most cases, however, because the tolerance used is likely to be small. **Points** is most useful for removing small gaps between elements. To use **Points**:

1. Click **Parametric, Clean** and **Points**.
2. Enter a distance tolerance (current length units).
3. Points within the given tolerance are marked with a *, and will be merged. Elements that will be modified as a result of the merge are highlighted.
4. Click **Confirm** to merge the marked points and modify the highlighted elements.

Duplicate

Duplicate scans the part for duplicate elements and deletes them. The command selects which element of a duplicate pair to delete. Elements are considered to be duplicates if their model points are identical. An exception is made for concentric circles (only): if the radii of concentric circles are within a given tolerance, they are treated as duplicates. Normally, **Duplicate** should be used after **Points** to clean up duplicate elements that may result from merging model points. To use **Duplicate**:

1. Click **Parametric, Clean** and **Duplicate**.
2. Enter a distance tolerance (current length units) for concentric circles.
3. Elements that will be deleted are highlighted.
4. Click **Confirm** to remove duplicate elements.

Stacked

Stacked finds stacked lines and arcs in the current part and calls **Split** to split them. To use **Stacked**:

1. Click **Parametric, Clean** and **Duplicate**. Elements to be split are highlighted and the points to be used for the splits are marked with a *.
2. Click **Confirm** to split elements.

Note

To clean up most common problems in a part, run the Clean commands in the following sequence:

Points	Merge disconnected points.
Duplicate	Remove existing duplicates and those caused by merging points.
Stacked	Split overlapping elements.
Duplicate	Remove duplicates caused by splitting.

Setting the Zone

The Parametric Design **Solve** and **Auto** commands operate only on the current Creo Elements/Direct Drafting part, and only on the portion of that part which is included in a zone that you define. The zone mechanism allows you to apply parametric design to sections of a part, leaving the remainder untouched. By including only those sections that require parametric design in the zone, you can often significantly reduce the number of constraints you need to assign as well as shorten **Solve** times.

The zone is implemented via Creo Elements/Direct Drafting info text. **Zone** commands are used to add or remove the info text PD_ZONE to geometry elements in a part. Parametric Design will only operate on elements that have the PD_ZONE info text. Sections of the part that do not have this info fall outside the zone and are never modified by **Solve**, remaining fixed in place. The **Zone** options are:

- **Display**
- **Add Single**
- **Add Multiple**
- **Remove**

When you start Creo Elements/Direct Drafting, the zone is empty. By default, elements you **Create** during an Creo Elements/Direct Drafting session are automatically added to the zone, but geometry you **Load** from unconstrained files is not included. **Zone** commands add, remove, and display elements in the zone.

Adding Elements to the Zone

To add elements to the zone:

-
1. Click **Parametric, Zone** and **Add Single** or **Add Multiple**.
 2. Select the geometry to be included in the zone. **Add Multiple** extends your selection to include all elements that share model points with the elements you select. The Creo Elements/Direct Drafting **Select** mechanism can be used with either option.
 3. Click **End** to conclude.

Elements are highlighted as you add them to the zone. Note that **Add Multiple** and **Add Single** do not add dimensioning or sub-parts in the master part to the zone. See [Zone Gymnastics on page 98](#) for additional information on adding to the zone.

Showing Contents of the Zone

Click **Display** in **Zone** to show the current contents of the zone. **Display** redraws all elements in the zone with the current highlight color and linetype. Click **End** to remove the highlighting.

Removing Elements from the Zone

To remove items from the zone, click **Remove** in **Zone** and then select the elements you wish to remove. If these items have constraints attached to them, the constraints will not be removed, but the elements will not be modified by subsequent **Solve** commands. Note that **Remove** only removes from the zone those elements that you click directly and not elements that share model points with those you select. In other words, **Remove** is analogous to **Add Single** rather than **Add Multiple**.

Defining Rigid Bodies

A final preparation step that can often help simplify the design process is to identify areas of the part in which the geometry will not change in size or relative position, but which should be able to move (translate, rotate) as a unit. By defining these areas as rigid bodies, you can significantly reduce the number of constraints needed, and make the design process more predictable.

A rigid body is a named collection of geometry that acts as if it is sufficiently constrained to prohibit any relative change of size or position among the elements collected into it. The geometry collected into a rigid body can only be translated and/or rotated as a whole. To link a rigid body with other elements in a part, you can assign constraints between these elements and any element inside the rigid body.

Most Creo Elements/Direct Drafting geometry elements as well as hatching and subparts can be included in a rigid body. Like zones, rigid bodies are implemented via Creo Elements/Direct Drafting infos; entities with the info text PD_RIGID are linked into rigid bodies. For more in-depth information on rigid body applications see [Advanced Topics and Tips on page 93](#).

The **Generate Rigids** commands allow you to define and manipulate an unlimited number of rigid bodies inside a given part. The **Generate Rigids** options are:

- **Assign**
- **Show**
- **Clear**

Creating A Rigid Body

To create a new rigid body:

1. Click **Assign** in **Generate Rigids**.
2. Enter a name for the new rigid body. Be sure to select a name that is not already in the rigid body list in the **Generate Rigids** dialog box if you want to create a new rigid body.
3. Select the geometry to be included in the rigid body. When you make the first selection, the name of the new rigid body will be added to the table.
4. Enter the name of another rigid body or click **End** to conclude.

Elements are highlighted as you add them to the rigid body.

Showing and Clearing Rigid Bodies

To show or clear an existing rigid body:

1. In **Generate Rigids**, select the rigid body you want to show or clear. If there are too many entries in this table to view at once, use the scroll bar beside the table to bring the correct entry into view.
2. Click **Show** to highlight the rigid body or **Clear** to remove the rigid body from the list.
3. Click **End**.

Assigning Constraints

Constraints can be assigned manually or automatically to elements in the zone. The constraint assignment tools include:

- **Commands:**

Command	Function
Assign	Manually assign constraints to geometric elements.
Free	Remove constraints from geometric elements.
Show	Show existing constraints by displaying their icons.
Clear	Remove selected constraint icons (but not the constraints themselves) from the display.
Ban	Remove selected constraints and prevent them from being re-assigned by Complete .
Complete	Automatically generates a reasonable set of constraints to all remaining elements in the zone.

- Options: Options limit the actions of **Show**, **Clear**, **Free** and **Ban**. Available options are:

Option	Limiting Action
New	Act only on constraints added by the most recent Complete command. New must be used immediately after Complete . Any intervening command converts new constraints to "system" constraints. New only modifies the actions of Show and Clear .
User	Act only on constraints assigned manually by the user. If the user modifies a system-generated constraint, it converts to a "user" constraint.
System	Act only on constraints generated by Complete and unchanged by the user.
Used	Act only on constraints actually used by the solver during the most recent Complete or Solve command.
Unused	Act only on constraints that were not used by the solver during the most recent Solve command.
Banned	Act only on constraints that have been banned by the user; i.e., constraints that cannot be assigned by Complete .
Violated	Act only on constraints marked "violated" by the most recent Solve .

- Types: These blocks specify the available constraint types. You must always specify a type for **Assign**, **Show** and **Clear**. Types can also be used to limit the actions of **Free** and **Ban**. A generic type, **All Types**, refers to all the available constraint types. See [Constraint Types on page 45](#) for a complete description of each type.

The following sections describe each of the constraint assignment commands.

Note

The default size and positioning of constraint icons may not be ideal. You can resize or change the color of icons via Icon Setting and reposition them with Move Icon. To update the icon display after changing their size or position, you can use the Creo Elements/Direct Drafting Redraw function. The text labels that appear on some icons are drawn in the current Creo Elements/Direct Drafting text font.

Assign

Use **Assign** to manually assign constraints to elements. To assign a constraint:

1. In **Generate Constraints**, click the **Assign** radio button.
2. Select a constraint type from the **Types** list, e.g., **Angle**.
3. Select the appropriate geometry for the constraint type.
4. If applicable, enter a value or parameter name for the constraint.
5. Continue to select geometry, select a new constraint type, or click **End**.

When you successfully assign a constraint, one or more icons appear on the part to identify the type and identification number of that constraint. The leader line of the icon is drawn in the same linestyle as the element it references.

Note that when you use **Assign** to assign a constraint to a piece of geometry, the constraint is bound to that element and goes with it whenever it is moved. Thus, you can move elements in a part after assigning constraints without invalidating the constraints. On the other hand, if you delete a constrained element, all constraints on that element are removed along with it.

Some constraint types can be given numeric or parametric values when they are assigned. If you give a numeric value, the constraint will resolve to that value at solve time. If you give a parameter name or expression, the constraint resolves to the output of that parameter or expression at solve time. If you choose not to supply either a value or parameter, the constraint resolves to the corresponding value that is currently on the element at solve time. If a value or parameter name is assigned to a constraint, that data will appear along with the constraint's icon whenever it is displayed. See [Parameters on page 69](#) for a full discussion of how numeric values and parameters affect constraints.

Show and Clear

To assign and free constraints efficiently, it is important to know what constraints are currently assigned. **Show** and **Clear** work with the limit-options and constraint-type settings to selectively display and hide the icons and parameter labels of constraints currently on the part. Neither function affects the constraints themselves, just the display of their icons. Icons and parameter labels are displayed in the current show color (PD_PREVIEW_COLOR) and size. Parameter labels will use each dimension's text font for dimensional constraints and the current text font for geometric constraints.

Show and **Clear** operate identically. To use either:

1. In **Generate Constraints**, click the **Show** or **Clear** radio button.
2. Select an option to limit the function's effect, if desired.
3. Select the **Types** to show/clear.

Show options are additive; subsequent calls to **Show** do not erase the icons that are already on the screen, allowing you to build up the display. From time to time, as the display gets crowded, you can erase some or all of the icons with **Clear**.

Some **Show** examples:

Show all constraints:	Show, All Types
Show constraints set by Complete :	Show, System, All Types
Show reference elements set by the user:	Show, User, Refelem
Show angle and distance constraints:	Show, Angle, Show, Distance
Erase all constraint icons:	Clear, All Types

Free

Free removes constraints from elements. To remove constraints:

1. In **Generate Constraints**, click the **Free** radio button.
2. Use one of these methods to select constraints to free:
 - Click the icon of the constraint to be freed.
 - Select a constraint type from **Types** and then click the element itself.
 - Select a limiting option from **Act On** and then a constraint type to remove the selected constraint from all elements in the zone. For example, **Free**, **User** and **Dimension** removes all user-assigned dimensional constraints.
3. Click **End** or continue to select constraints for removal.

Whenever you free a constraint, the corresponding constraint icon is removed from the display. If you free a constraint by accident, an immediate **Undo** will restore it.

Freeing a constraint does not prevent it from being re-assigned to the master by **Complete**. If you want to prevent a constraint from being re-assigned by **Complete**, use **Ban** instead.

To remove all constraints generated by **Complete**, use **Free ▶ System ▶ All Types**.

To quickly remove all constraint and parametric data from the current part, use **Free, All Types**. This command requires a **Confirm**.

Complete

In order for **Solve** to be able to generate a variation of a part, every element in the part must be completely constrained. To manually assign a complete set of constraints to a part of even moderate complexity would be time consuming and prone to error. The **Complete** command provides an efficient alternative. **Complete** scans the geometry currently in the zone and generates a reasonable set of constraints that is sufficient to completely constrain the geometry in its current configuration. In many situations, only a few constraints need be set manually before running **Complete**, and a few more changed after **Complete** is run to generate the variation. The [Strategy for Assigning Constraints on page 33](#) section below explains how to best integrate the automatic and manual options.

Simply select **Complete** and Parametric Design generates constraints. A series of messages is displayed to inform you of the command's progress. While generating constraints, **Complete** may add a number of point elements to your part. These are needed to fix point-on or reference-point constraints, and should not be removed from the part. By default, these points are drawn as green ×s, but you can easily change this to make them nearly invisible. See [Parametric Design Defaults on page 175](#) for details.

Complete takes into account any constraints that you may have manually assigned to the part. If these constraints are inconsistent with one another, **Complete** may not be able to successfully generate a complete set of valid constraints. In this case, error messages and highlighting warn you about the problem and help you track down the cause. The error-trapping mechanism used by **Complete** is also used by **Solve**. See the [Solver Errors on page 36](#) section later in this Chapter for more information.

Ban

Ban works exactly like **Free** except that in addition to removing a constraint, **Ban** prevents that constraint from being re-assigned by subsequent **Complete** commands. Essentially, **Ban** forces a constraint to remain unset. To re-assign a banned constraint, you must assign it manually with **Assign**.

Strategy for Assigning Constraints

Complete and the manual constraint tools are designed to be used together to efficiently generate a workable set of constraints that lends itself to modification. The following strategy will produce the best results in most situations:

1. Before running **Complete**, manually assign sufficient constraints to:
 - "Anchor" the part. Do this by assigning **Refelem** or **Refpoint** constraints to one or more elements that you know should not change position in the variation. Anchoring helps to give **Complete** a frame of reference for generating new constraints.
 - Identify symmetry lines. Assign **Symmline** constraints to any elements that serve as symmetry lines in the part. **Complete** uses these lines to generate mirrored-element constraints. Parametric Design can also be configured to automatically recognize all lines with a particular color and linetype as symmetry lines. You enable this capability by modifying the `pd_def.mac` file. See [Parametric Design Defaults on page 175](#) for details.
 - If the part has several distinct components or views, make sure to anchor each view sufficiently to prevent them from interfering with each other in subsequent operations. You can do this either by assigning absolute constraints (**Refelem**, **Refpoint**) to each view, or by constraining the relative distance and/or orientation among all views with **Distance** or **Angle** constraints.
 - Collect suitable elements into rigid bodies whenever possible.
 - Manually constrain any elements whose position or orientation is critical in subsequent variations. For example, you should fix in place any elements that definitely should not move in the variation. Alternately, you could remove these elements from the zone before running **Complete**.
 - In general, the more constraints you can assign by hand, the better for **Complete**. This is especially true if the part is not fully dimensioned.

Note

Do not manually assign constraints designed to move elements away from their current location before running **Complete**. For example, don't assign a Vertical constraint to horizontal lines in the part before you use **Complete**. Such constraints make the extraction process much more difficult.

2. Run **Complete** to generate the remaining constraints needed to fully constrain the master part in its current configuration.

3. When **Complete** finishes, examine the constraints on the part to make sure that **Complete** hasn't made any poor decisions about assigning constraints. In particular:

- Check dimensions. Make sure that constraints were actually assigned to any dimensions that you might have to change. If not, add new constraints. Also, be aware that different types of dimensions can cause **Complete** to behave in different ways. For example, a horizontal dimension between two points is quite different from a parallel dimension between the same two points, even if both have the same value. The first case implies that the second point is on a vertical line a given distance away from the first point. The second case implies that the second point is on a circle of a given radius centered at the first point. **Complete** takes these distinctions into account when generating constraints. When in doubt about the type of linear dimension, redefine it.
- Check distance and size constraints. Distance constraints are usually assigned as a last resort to position disconnected geometry that is insufficiently dimensioned. The elements that receive distance constraints may not be the ones you would normally choose to receive them. Similarly, size constraints are normally only assigned to geometry that is connected to other elements, but is still underdimensioned. Again, a size constraint may not be an appropriate choice.
- Check angle and slope constraints. Are elements with these constraints supposed to be fixed at the given angle?
- Check the constraints on rigid bodies. Make sure that **Complete** hasn't over-constrained each rigid body with regard to the rest of the part.

The **Element** and **Icon** options in **Inquire** are especially useful for examining constraints generated by **Complete**. **Inquire Icon** identifies the element(s) associated with a given constraint by highlighting the element(s). You identify the constraint by clicking on its icon.

Inquire Element shows how the most recent **Complete** or **Solve** command determined the size and position of an element that you select. When you select **Inquire Element** and click an element, the icons for all the constraints that determined the size and position of that element are displayed. In addition, all geometry on which those constraints rely will have their geometry highlighted.

4. If you make changes to geometry or constraints at this stage, re-run **Complete**. Check the following points any time you re-run:

-
- If you removed any constraints, did you **Free** or **Ban** them? **Complete** is likely to re-assign the same constraints unless you used **Ban**.
 - If you added dimensioning to your part to eliminate the need for size or distance or angle constraints, did you **Free** the old system constraints? **Complete** will not automatically free existing system constraints and use the dimensioning to generate new constraints. If you **Free** the system constraints first, then the new dimensioning will be taken into account.
5. Once you are satisfied that the part is completely and correctly constrained in its current configuration, then start to modify those constraints that will be used to generate the variation(s) to ensure that the part responds as you intend.

Solve

All access to the Parametric Design solver is through the **Solve** options, which are available on all Parametric Design menus. They are:

- **Preview**
- **Keep**
- **No Keep**

When you select a **Solve** option, the solver finds all the geometry currently in the zone, and evaluates all constraints on that geometry. If all the constraints can be successfully evaluated, the solver generates the variation specified by the constraints. Depending on the option selected, the solver can either create geometry for a new variation or simply preview it. The sections below describe the **Solve** options.

Preview

By default, **Solve** works in **Preview** mode. The variation is computed, but is not actually created. Instead, an outline of the variation is superimposed over the current part. This gives you a good idea of what the new variation will look like without creating any new geometry or overwriting any existing geometry. If the preview indicates that the new variation is acceptable, you can actually create the new variation by re-solving with the **Keep** or **No Keep** options. The preview lines are drawn in the current PD_PREVIEW_COLOR color (MAGENTA by default). Preview lines are removed from the viewport by the next **Solve** command or when you click **End**.

Keep

The **Keep** option generates the variation and inserts it into the current part as a modified copy of the original part. When you select **Keep**, the solver computes the variation, temporarily overlays it onto the current part, and executes a **Move Multiple** command to allow you to copy the variation.

To use this option, select **Keep**. When the solver generates the variation and overlays it on the current part, evaluate the results. If the variation is not what you want, click **End** or **Undo** immediately to restore the original part. If the variation is OK, click or specify a reference point for moving the new part, and then drag it to the desired location. Repeat this step until you have inserted as many copies as you want. Finally, click **End**. The overlaid variation is removed, and your original part is restored.

Only elements in the zone (i.e., elements with the PD_ZONE info text) at solve time are copied by **Keep**. Dimensioning and hatching elements are never put into the zone by Parametric Design commands. Therefore, these elements are not normally included in the copies you make with **Keep**.

Copies made with **Keep** are not put into the current zone and do not have constraints assigned to them. The original part from which copies are generated is unaffected by the **Keep** option.

No Keep

Select the **No Keep** option if you want the solver to replace your current part with the variation. If the result of the solve is not what you expect, click **Undo** immediately to restore your original part. You can continue to modify the part after solving with **No Keep** by assigning new values to the constraints and re-solving.

Solver Errors

The solver or automatic constraint generator will fail without generating a complete solution if incomplete or inconsistent constraints exist on a part. Solver errors can result from any of the following conditions:

- The solver has successfully evaluated all the constraints it could find, but some elements cannot be resolved. More constraints are needed in order to completely constrain the part and generate a variation.
- A constraint has been violated while determining the new geometry. This is usually due to contradictory constraints on one or more elements in the part.

For example, if two lines are constrained with different **Slope** constraints and are also constrained to be parallel, one of these constraints cannot be satisfied.

- An expression has been violated while determining new geometry. As with a violated constraint, a violated expression occurs when the result of a parametric expression contradicts an existing constraint.

When the solver or automatic constraint generator fails due to one of the above problems, a descriptive message is written to the prompt line and the solver goes into preview mode to show the status of the solution at the time the error was encountered. Elements that were successfully solved are displayed in their new positions using the **Preview** color (MAGENTA by default). Elements that were still unsolved when the solver exited are displayed in their original position, color and linetype. This often gives a good indication of the elements that caused the failure.

The commands in the **Display** dialog box (accessible via **Settings**) usually provide the clearest indication of problem sources. After every solve or constraint-extraction command, one or more info strings are always written to each drawing element that was affected by the command. These info strings give the current solve status of each element. **Show All** and **Hide All** allow you to selectively display or hide elements based on their solve status. These status displays are very effective for pointing out problem areas, especially in complex drawings, where highlighting may not clearly identify problem elements.

The **Display** options are described below.

Display Option	Function
Show All	Display all elements in the model.
Hide All	Hide all elements in the model.
Solved	Display/hide elements that were completely determined by the last solve or constraint extraction command. These elements have the PD_STATUS SOLVED info text.
Unsolved	Display/hide elements that were completely undetermined by the last solve or constraint extraction command. Unsolved elements either had no constraints on them or the solver exited before reaching them. These elements have the PD_STATUS UNSOLVED info text.

Display Option	Function
Partially Determined	Display/hide elements that were partially determined by the last solve or constraint extraction command. Some, but not all degrees of freedom have been determined for these elements. These elements have the PD_STATUS PARTIAL info text.
Inconsistent	Display/hide elements whose constraints have caused an inconsistency in the most recent solve or constraint-extraction process. These elements have the PD_STATUS INCONSISTENT info text. Inconsistent elements always fall into one of the three previous categories as well.

Debugging Solver Errors

When you're confronted with a solver error, first look at the error message posted to the prompt line. It will usually tell you whether the problem occurred because of too few constraints or because of conflicting constraints.

If the error message is something like:

```
Not enough constraints to solve for all elements
```

your drawing does not have sufficient constraints for the solver to generate a variation.

To quickly correct this situation, select **Complete** to extract the missing constraints. If for some reason **Complete** is not applicable to your situation, use the **Display** functions to highlight the problem areas and manually assign additional constraints to the offending elements. Start by highlighting all partially solved elements (click **Hide All** and then **Show All** and **Partially Determined** in **Display**), and then use **Inquire Element** to see what constraints are there. You may also want to examine the constraints between the partially solved elements and any completely solved elements they border.

If the error is due to conflicting constraints, the error message may resemble one of the following:

```
Encountered violated constraints
```

```
Could not constrain without violating n constraints
```

These messages indicate a violated user or system constraint.

Use the **Display** functions along with **Inquire Element** and **Show Violated** to help identify the cause of a constraint violation. Whenever a constraint is violated and the reason for the violation is not clear, first concentrate on those entities that are tagged as completely solved. The solver preview mode shows how these entities are getting modified. Are they being modified as you expected? If so, next look at the solved and partially solved entities involved in the constraint that is being violated. Remove or re-assign the problem constraints to eliminate the conflict.

Another possibility is that the error message will resemble:

```
Inconsistently constrained elements found  
Unable to constrain all elements
```

These messages indicate that an internal constraint has been violated. Examples of an internal constraint violation include:

- The implicit point that resides at the intersection of two or more elements constrained so as to move off of one or more of those elements.
- The endpoints of an arc constrained so as not to be equidistant from the arc center.

In this case, you need to identify and re-constrain the elements that are causing the conflict. For example, to isolate all elements that caused an inconsistency in the last solve, use **Hide All** to hide the entire drawing, then use **Show All** and **Inconsistent** to display only the elements that caused the inconsistency.

If the solver problem is due to a violated expression, the displays described above will be used, plus the entry for the offending expression in the parameter definition table is highlighted. To correct this situation, examine the highlighted expression carefully. You may need to rewrite it to avoid the conflict. It is also possible that the expression conflicts with geometric constraints on the part that can be removed or re-assigned.

A final class of error messages indicates problems with rigid bodies. A message of the form:

```
Unable to keep rigid body 'name' rigid  
Unable to constrain while keeping rigid body 'name' rigid
```

indicates that competing constraints on the named rigid body are attempting to change its shape. This generally indicates that there are too many constraints assigned to the rigid body. Concentrate your debugging efforts on the named rigid body and eliminate all unneeded constraints.

Application

There are two basic uses that the Designer will have for Parametric Design. The first is to create master parts that will be used as templates for families of parts. The second is to perform modifications that are beyond the normal scope of the Creo Elements/Direct Drafting **Modify** tools. The sections below describe each application.

Designing Master Parts

A "master part" is any constrained part that is designed to be used as a template for generating variations. For this application, the constraint information is just as important as the geometry of the part, and should be considered an integral part of the part. In order for a master part to be used successfully, the constraints must be

robust enough to correctly generate the full range of required variations, and they should be set up to be accessible and easy to use, especially if the master is to be used by several people over an extended period of time.

To achieve these goals, the following procedure should be followed to prepare a master part:

1. Clean the input part
2. Constrain the part
3. Parameterize constraints
4. Evaluate and debug the part

Clean the Input Part

To consistently generate variations from a part, it is important that the original be free of geometric inconsistencies that could be carried over into the variations. You should always look out for inconsistent geometry in any part that you intend to use as a master part. If you find inconsistencies, either remove them by means of the **Clean** tools, or isolate problem areas from consideration by the solver by collecting these entities into rigid bodies.

Constrain the Part

Constrain your part following the procedures described in the [Strategy for Assigning Constraints on page 33](#) section above. After you assign the initial set of constraints, consider running **Solve No Keep** once. This serves two purposes. First, it quickly tells you whether your initial constraints are valid. Second, it provides an extra level of geometry cleaning by adjusting the sizes and positions of elements that fall within the **Complete** length and angle tolerances. [Advanced Topics and Tips on page 93](#) explains the **Complete** tolerance capabilities.

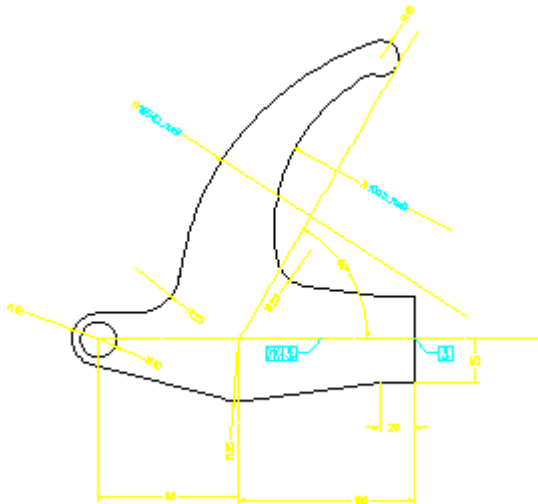
Parameterize Constraints

Two basic design goals for any master part should be robustness and ease of use. Parameters (described fully in [Parameters on page 69](#)) help you achieve both goals. By defining parametric values for some or all of the appropriate constraints on your part, you can reduce the number of constraints that the end user needs to modify directly in order to produce the variation. Other constraints can be parameterized to depend on the values supplied by the user. In complex situations, expressions and macros can be supplied to check or filter the user's input before acting on it.

If the constraints that must be modified by the user are themselves parameterized, the user need not reassign these constraints. Instead, they will appear in the parameter value table, which provides a much simpler interface for the occasional user.

For example, consider the following master part, which was used to generate the variations shown in [Figure 1. Master Part for Rocker Arm on page 41](#):

Figure 1. Master Part for Rocker Arm



For this part, we want to generate variations in which only the configuration of the large arm changes. In all variations, we know that the outer radius of the arm should depend on the inner radius. Rather than require the end user to calculate this relationship and manually **Assign** both radii for each variation, we use parameters to do most of the work.

After constraining the part normally, we assign a dimensional constraint to the inner radius and give it the parameter name `I_rad`. The outer radius also gets a dimensional constraint, named `O_rad`. [Figure 1. Master Part for Rocker Arm on page 41](#) shows the constraints we supplied for this master part. The remainder were generated by **Complete**.

Using the parameter definition table, we define the constraint represented by the parameter `O_rad` to be $(1.65 * I_rad)$. This defines the relationship between the two dimensions. As a result, the user only needs to give a value for `I_rad` to generate the variation. The `I_rad` parameter is available in the parameter value table, so the user can easily change it and then **Solve** for variations.

Another advantage of parameterization is that the parameter values can be stored and recovered for later use. The **Save** and **Input** functions in the **Current Constraints** dialog box allow you to write out the current parameter value table to a file and later restore it. This feature lets you build a "library" of value tables that can be used to drive the master part. The advantage of this is that a series of standard parts based on a single master part can be stored as a single MI file (the constrained master part) and a collection of parameter value files. This arrangement takes up much less disk space than if an individual MI file were stored for each part in the series, and is much easier to update if the geometry of

the master part needs to be changed. To recreate any member of the series, the user need only load the master part, **Input** the desired parameter value table, and **Solve**. [Generating Variations on page 17](#) describes this functionality.

Evaluate/Debug the Part

Once you have finished constraining a master part, you must test it to ensure that it generates variations as intended. The extent of the testing you do depends on the complexity of the part and the intended users. Obviously, you should never release a master part for use by others until you have thoroughly checked it to make sure that it operates as intended.

The easiest way to test a master part is simply to modify constraints and **Solve** for the variations you expect to be needed. **Preview** mode is especially useful for this type of checking because it never modifies the original part. If you need to examine the variation in more detail than **Preview** allows, you can use **Solve No Keep**, but make sure you write your original master part out to a file first so that you can recover it.

Here are some tips to keep in mind while you test:

- Try small changes to the part first, until you get a feel for how it is behaving. As you become more familiar with it, increase the size of the modifications until you reach the maximum expected modification.
- Try modifying a single constraint or parameter at a time. In complex parts, the interactions among different constraints are not always obvious until you separate them out this way.
- Try supplying nonsense values or values beyond the expected maximum. Do these break the part badly? If so, you may want to consider implementing macros or expressions that limit the values that the part will accept. These expressions can be incorporated directly into the constraint structure of the master part.

Advanced Modification

A second application for Parametric Design is to do modifications that cannot be easily done with the standard Creo Elements/Direct Drafting **Modify** tools.

The procedure for doing modifications is very similar to that for generating master parts, but since you are generally working for yourself on a part that you have designed yourself, the process can be less formal. Here are a few differences:

-
- Geometry cleaning may not be as important for this application, especially if you've designed the original part yourself and know that it is correct.
 - Constraints can be assigned more quickly. In many cases, it is sufficient to anchor a few elements with **Refelem** constraints, run **Complete**, and immediately modify for the variation you want.
 - When you have the modification you want, make sure to free all constraints from the variation and remove it from the zone. The zone and constraint data for a complex part can take up significant amounts of memory while the part is loaded, and will increase the size of the part's file when you store it.

4

Constraint Types






















Constraint Types	46
Angle	46
Collinear	48
Dimension	48
Distance	50
Fillet	53
Horizontal	54
Mirror	55
Parallel	56
Perpendicular	58
Point on	59
Reference Element.....	60
Reference Point	61
Same Distance.....	61
Same Size	62
Size.....	63
Slope.....	63
Symmetry Line	64
Tangent	65
Vertical	67
X-Distance and Y-Distance.....	67

This chapter describes the usage and function for each of the available constraint types.

Constraint Types

Each constraint type is identified by a unique icon as you assign it to the master. [Figure 3. Constraint Icons on page 46](#) shows these icons. You should take a few minutes to make sure you recognize these icons and the constraint types they represent.

Figure 3. Constraint Icons

	Angle		Ref Point
	Collinear		Ref Elem
	Dimension [*]		Same Dist
	Distance		Same Size
	Fillet		Size
	Horizontal		Slope
	Mirror		Symm Line
	Parallel		Tangent
	Perpendic		Vertical
	Point On		X-Distance
			Y-Distance

^{*}Dimension constraints are indicated by color change and a postfix or change to the main dimension text.

Angle

You assign **Angle** constraints to pairs of linear elements. At solve time, the relative angle between the elements in each angle-constrained pair will be adjusted as needed to match the angle specified by the constraint. If this angle cannot be achieved due to other constraints, the solve fails.

To assign an angle constraint:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Angle**.
2. Click **Apply**.
3. Click each of the two linear elements to be constrained.
4. Choose the angle for the constraint. You have three options:
 - Click **End** or select another angle pair. The constraint value will be set to the actual angle between the lines at solve time, thus maintaining this angle in the variation.
 - Enter a specific angle value. The elements will be adjusted to this angle.
 - Enter a parameter name or an Creo Elements/Direct Drafting expression that defines the angle (string input). The value returned in the parameter or the result of the expression is used to adjust the angle.

Each element in an angle-constrained pair is labeled with an angle icon, [Figure 3. Constraint Icons on page 46](#), and a reference number that is unique to the pair. If the constraint is set to a specific value or parameter name, that information is appended to the icon.

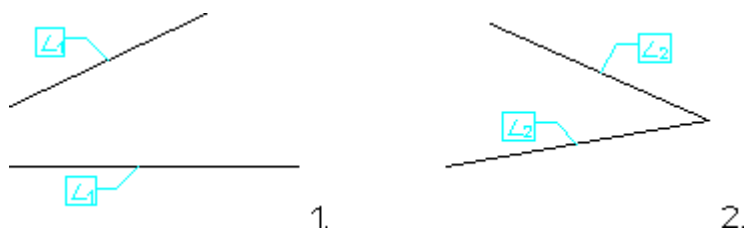
Application Notes

An **Angle** constraint will replace an existing **Parallel** or **Perpendic** constraint on a given pair of linear elements. This is because **Parallel** and **Perpendic** are simply special cases of the **Angle** constraint.

The angle between the elements in an **Angle** pair is given as the difference between the slopes of the two elements (see [Slope on page 63](#)). If the **Angle** constraint is parameterized, this angle can be accessed via expression by referencing the parameter name.

Complete will only extract angle constraints for line elements that share an endpoint or fillet. If you are assigning angle constraints manually, the members of the **Angle** pair need not be connected, [Figure 4. Angle-Constrained Lines on page 47](#).

Figure 4. Angle-Constrained Lines



Collinear

The **Collinear** constraint forces a pair of linear elements to become or remain collinear during the solve process. If other constraints on the part prevent the necessary adjustments to bring the pair together, the solve fails.

To assign **Collinear** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Collinear**.
2. Click **Apply**.
3. Click each of the two linear elements to be constrained.
4. Click **End** or go on to the next constraint.

Each linear element in a collinear pair is labeled with the collinear icon, [Figure 3. Constraint Icons on page 46](#), and a reference number that is unique to the pair.

Application Notes

Lines need not be connected in order to assign or extract a **Collinear** constraint. [Figure 5. Collinear Lines on page 48](#) shows a common application for **Collinear** constraints: "gluing" a line element to a construction line.

Figure 5. Collinear Lines



When adjusting a pair of lines to make them collinear, **Solve** tries to use the smallest adjustment possible to bring the pair into line.

Dimension

Dimension constraints set the dimensions of elements in the master part to specific values. **Solve** will attempt to adjust constrained dimensions to the values given by the constraint. If this proves impossible, the solve fails.

To assign **Dimension** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Dimension**.
2. Click **Apply**.

-
3. Click on the text of the dimension to be constrained.
 4. Choose a value for the constraint. You have three options:
 - Click **End** or select another dimension. The actual value of the constrained dimension at solve time will be assigned to the constraint. This dimension will not change in the variation.
 - Enter a specific value for the dimension. The dimension is adjusted to this value at solve time.
 - Enter a parameter name or Creo Elements/Direct Drafting expression that resolves to the dimension value (string input). The dimension is adjusted to the result of the expression.

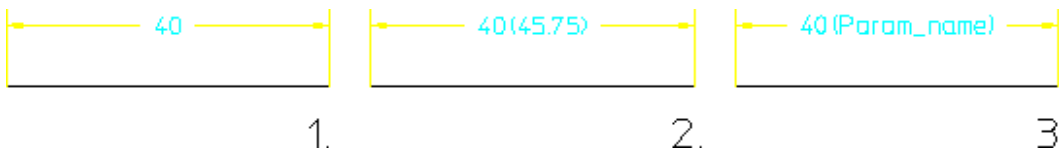
Dimension constraints are not identified by icons, but rather by a change to the dimension text and color. Depending on how you define the dimensional constraint, one of three displays will result at definition time and when you select **Show and Dimension** or **Show and All Types** in **Generate Constraints**:

- Dimensions that are constrained to keep their current value during the solve are redrawn in the current icon color (CYAN by default).
- Dimensions that are constrained to a given value are redrawn in the current icon color (CYAN) and the main dimension text is replaced by the value given to the constraint.
- Dimensions that are given a parametric value are redrawn in the current icon color (CYAN) and the main dimension text is replaced by the parameter name, which is drawn using the dimension's font.

Note

An alternate display style for dimensional constraints can be specified by calling `PD_SHOW_USE_POSTFIX YES`, either from the Creo Elements/Direct Drafting command line or via the `pd_def.mac` file. This option causes the parameter value to be written as a postfix to the main dimension text, which remains unchanged, [Figure 6. Dimension Constraints, Alternate Style on page 50](#). Both the dimension text and postfix are redrawn in the current icon color (CYAN), to indicate that a dimensional constraint has been applied. The Parametric Design postfixes do not interfere with Creo Elements/Direct Drafting dimension postfixes, if present.

Figure 6. Dimension Constraints, Alternate Style



You cannot assign a dimension constraint to symmetric Single or symmetric Datum Long dimensions.

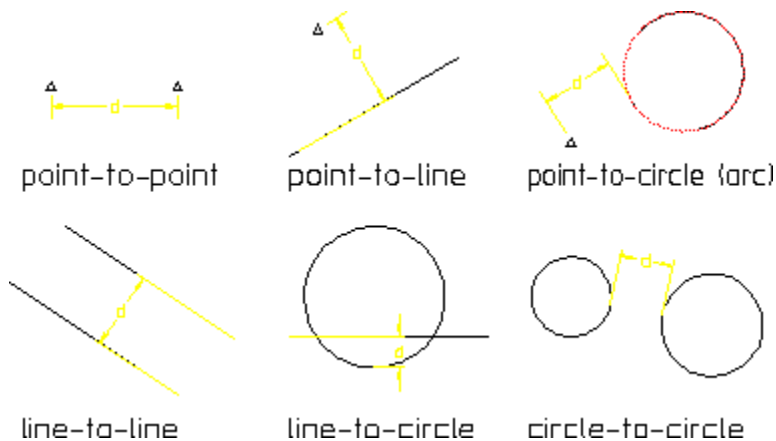
Distance

You can assign a **Distance** constraint to any pair of geometric elements except for spline-based entities (rational splines, bsplines, and ellipses). One or both elements in a distance-constrained pair may also be dimension text, as described below. At solve time, the relative distance between the extensions of the elements in the pair is adjusted to the distance specified by the constraint. If this adjustment cannot be made, the solve fails.

The extension of an element is its construction-geometry equivalent. For example, the extension of a line segment encompasses any point that would fall on a collinear construction line. The extension of an arc element encompasses any point that would fall on a construction circle that shares the arc's center and radius.

Elements in a distance-constrained pair need not be of the same type; e.g., the distance between a circle and a linear element can be constrained, as can the distance between a point element and an arc. [Figure 7. Distance Determination on page 51](#) illustrates how distance is defined for different combinations of elements.

Figure 7. Distance Determination



To assign a distance constraint:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Distance**.
2. Click **Apply**.
3. Click each of the two elements to be constrained.
4. Choose the distance for the constraint. You have three options:
 - Click **End** or select another distance pair. The constraint distance will be set to the actual distance between the elements at solve time.
 - Enter a specific distance value. The distance will be adjusted to this value at solve time.
 - Enter a parameter name or Creo Elements/Direct Drafting expression that defines the distance (string input). The distance will be adjusted to the result of this expression or parametric value at solve time.

Each element in a distance-constrained pair is labeled with the distance icon, [Figure 3. Constraint Icons on page 46](#), and a reference number that is unique to the pair. If the constraint is set to a specific value or parameter name, that information is appended to the icons.

Distance Constraints and Dimension Text

By default, when a dimensioned element moves or resizes as a result of a Parametric Design **Solve**, a new position for its dimension text is automatically determined. You can override this automatic placement by assigning **Distance** constraints directly to the text of the dimension(s) you want to control. This mechanism allows you to control the position of the center of the dimension text

relative to the dimensioned element and relative to other elements in your drawing. Note that only **Distance** constraints may be assigned to dimension text; no other constraint types are allowed.

Application of distance constraints to dimension text must be done manually. The automatic constraint generator will never attempt to add distance constraints to dimensions. Because of this, the preferred method for applying distance constraints to dimensions is to first completely constrain the part without any constraints on the dimension position and then constrain only those dimension positions which need to be controlled. To assign the constraint to dimension text, click on the centerpoint of the main text element. Two constraints are generally needed to fully constrain the dimension position; one for each degree of freedom. In most cases, if you add only a single distance constraint, subsequent solves will fail. Construction geometry can often be helpful for providing additional "distance markers" in this application.

Application Notes

Several powerful parametric applications are made possible by distance-constraining strategically located point elements with regard to other elements in a drawing. In both of the following examples, point elements are created specifically for use by Parametric Design. The point elements are necessary because distance constraints can only be assigned to geometric elements, not model points.

Figure 8. [Distance Constraints to Control Line Length on page 52](#) shows how distance constraints can be used to force a centerline to grow along with the rest of a part as it is resized. A point element is first assigned to each end of the centerline. Each point element is then distance-constrained to the nearest end of the part, (a). This distance will be maintained as the part is resized, (b).

Figure 8. Distance Constraints to Control Line Length

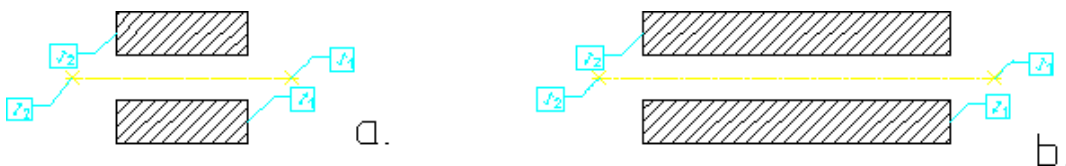
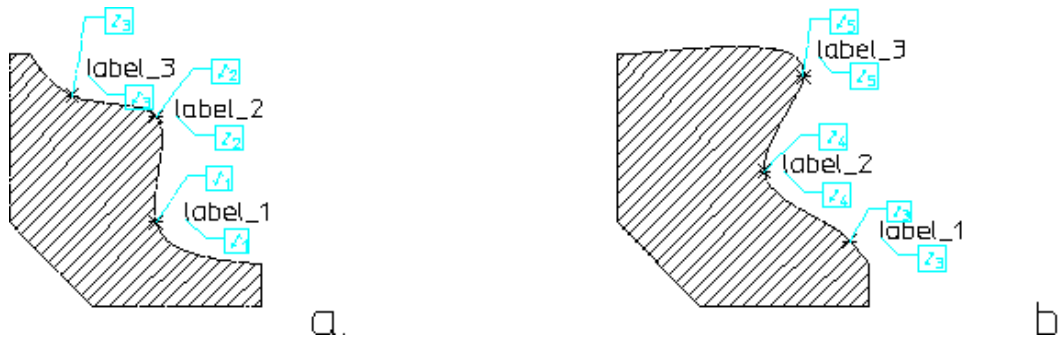


Figure 9. [Distance Constraints to Control Text Position on page 53](#) shows how distance constraints can be used to fix text labels to a given distance from a spline. The distance is maintained as the spline is reshaped. Point elements are assigned to the interpolation points of the spline and then distance-constrained to the reference points of the text elements, (a). This distance will now be maintained as the spline is changed, (b).

Figure 9. Distance Constraints to Control Text Position



Fillet

The **Fillet** constraint can be assigned to any arc that qualifies as a fillet in the model. To qualify as a fillet, an arc must have at least one line segment or arc element at each of its ends. When an arc is constrained to be a fillet, **Solve** will ensure that it remains tangent to both the elements at its endpoints. The fillet may change radius, length or orientation as a result of the solve.

To assign **Fillet** constraints:

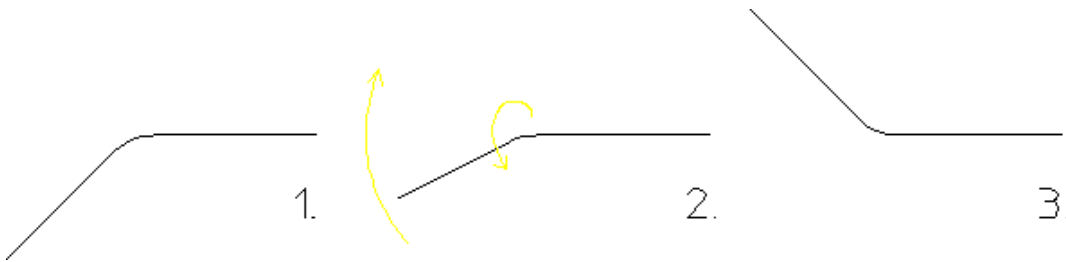
1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Fillet**.
2. Click **Apply**.
3. Click the arc to be constrained.
4. Click an element at each end of the arc to complete the fillet.
5. Click **End** or go on to the next constraint.

Fillet-constrained arcs are represented by the fillet icon, [Figure 3. Constraint Icons on page 46](#).

Application Notes

If you assign a **Fillet** constraint to an arc that has a line segment at each end, Parametric Design ensures that the angle subtended by the arc will never become greater than 180° by "flipping" the arc whenever this angle exceeds 180° in a variation, [Figure 10. Adjustment Made for Fillet Constraints on page 54](#). This adjustment is only made in the case of an arc with a line segment at each end.

Figure 10. Adjustment Made for Fillet Constraints



If you intend for the angle subtended by the arc to be greater than 180° in the generated part, do not use a fillet constraint; instead, use two **Tangent** constraints to glue the ends of the arc to the appropriate elements.

Parametric Design fillet constraints give you better control than the Creo Elements/Direct Drafting **Fillet Change** command for changing fillet radii. By applying other constraints along with the fillet constraints, you can control exactly how elements tangent to a fillet adjust to accommodate the new radius.

Horizontal

Line elements that are assigned a Horizontal constraint will be adjusted as necessary at solve time to become horizontal. If **Solve** cannot make this adjustment due to other constraints on the part, the solve fails. A horizontal constraint will not prevent a linear element from moving or changing length in the variation; it only adjusts the slope of that element to be zero.

To assign **Horizontal** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Horizontal**.
2. Click **Apply**.
3. Click the linear element to be constrained.
4. Click **End** or assign the next constraint.

The horizontal icon, [Figure 3. Constraint Icons on page 46](#), identifies horizontally constrained lines.

A given element may have only one of the **Horizontal**, **Vertical**, or **Slope** constraints assigned to it. If you attempt to assign a **Horizontal** constraint to an element that is already assigned a **Vertical** constraint, the **Horizontal** constraint will replace it.

The action taken when you assign a **Horizontal** constraint to an element that is already assigned a **Slope** constraint depends on whether or not the **Slope** constraint is parameterized. If it is not, then the **Horizontal** constraint simply replaces it. If the **Slope** constraint is parameterized, however, the **Horizontal** constraint is not

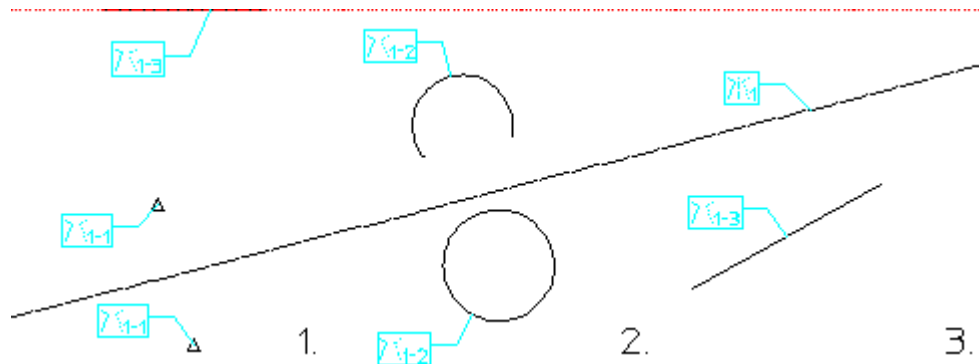
assigned. Instead, the value of the parameter is adjusted to 0° or 180° (effectively horizontal). This is done so as not to invalidate any expressions that may reference the existing slope parameter.

Mirror

When a pair of elements are constrained to be a **Mirror** pair, **Solve** will attempt to move and/or resize both elements as needed so that their extensions mirror each other across a specified line of symmetry. The extension of an element is defined as the element's construction-geometry equivalent. For example, the extension of an arc is the construction circle that shares the centerpoint and radius of the arc. The extension of a line element is a collinear construction line.

Both elements in the mirrored pair must be of the same type (e.g., both circles, both lines, or both points), but need not be the same size. Mirrored lines may be of different lengths, and an arc and a circle can mirror each other. Spline and ellipsoid elements are not eligible for mirror constraints, though some measure of control over these entities can be had by assigning point elements to their control/interpolation points and then fixing those point elements. [Figure 11. Mirrored Element Pairs on page 55](#) shows several examples of mirrored element-pairs.

Figure 11. Mirrored Element Pairs



To assign a mirror constraint:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Mirror**.
2. Click **Apply**.
3. Click a linear element to be the symmetry line for the mirrored pair. If this element does not already have a symmetry-line constraint assigned to it, one will be added.
4. Select a pair of elements to be mirrored across this line.
5. Click **End** or select another pair of elements to be mirrored about the original symmetry line.

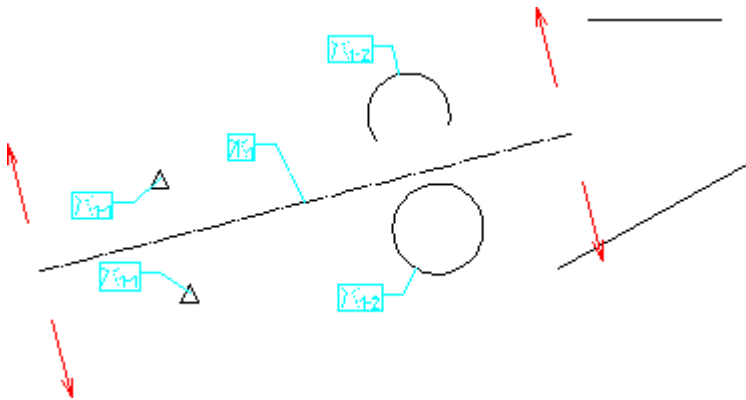
When you successfully assign a mirror constraint, both elements in the mirrored pair will be labeled with the mirror icon, and the line of symmetry will be labeled with a symmetry line icon, [Figure 3. Constraint Icons on page 46](#). The icon for each symmetry line in the model is given a unique number, and the icon for each member in the mirrored pair contains a number in the form s-p, where s is the number of the symmetry line that this pair is bound to and p is the unique number for this mirrored pair.

Application Notes

The symmetry line for a mirror constraint may be any linear element.

Complete only extracts mirrored pairs from areas that are orthogonal to defined symmetry lines. If your symmetry lines have infinite length (e.g., construction lines) this is no limitation, but if you choose a finite-length element as a symmetry line, items must lie completely within the orthogonal boundary in order to be selected for mirroring. The two line segments on the right of [Figure 12. Auto-selection of Mirrored Pairs on page 56](#), for example, would not be extracted because neither falls inside the boundary. If you are assigning mirrored pairs manually, this limitation does not apply.

Figure 12. Auto-selection of Mirrored Pairs



Parallel

The **Parallel** constraint is assigned to pairs of linear elements. A pair of parallel-constrained elements will be adjusted to be parallel at solve time. If other constraints on the part prevent the pair from being made parallel, the solve fails. The positions, lengths, and absolute slopes of parallel elements are not affected by this constraint and may or may not change during the solve, but the slopes of the lines will be the same.

To assign **Parallel** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Parallel**.
2. Click **Apply**.
3. Click each of the linear elements to be constrained.
4. Click **End** or assign the next constraint.

Each linear element in a parallel pair is labeled with the parallel icon, [Figure 3. Constraint Icons on page 46](#), and a reference number that is unique to the pair.

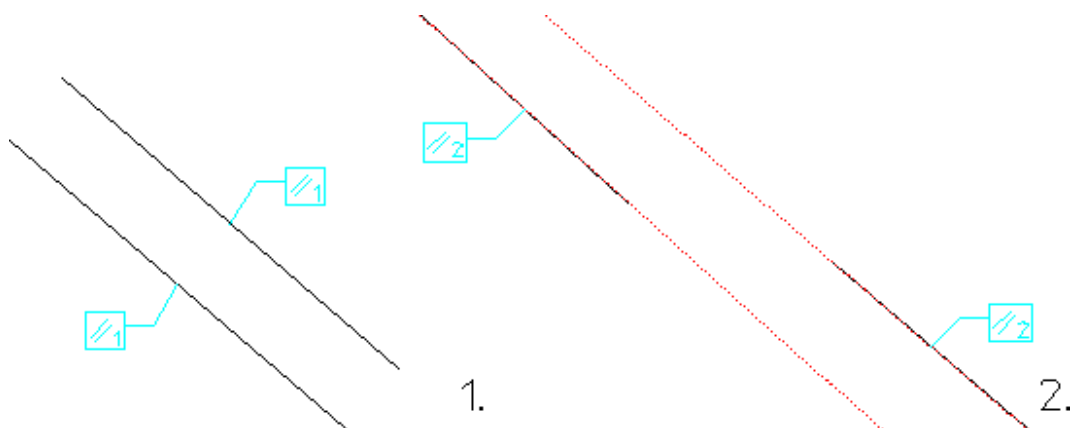
Application Notes

A given pair of linear elements may have only one of the **Angle**, **Parallel**, or **Perpendic** constraints assigned to it. This is because **Parallel** and **Perpendic** are simply special cases of the **Angle** constraint. If you attempt to assign a **Parallel** constraint to a pair of elements that is already assigned a **Perpendic** constraint, the **Parallel** constraint will replace it.

The action taken when you assign a **Parallel** constraint to a pair of elements that is already assigned an **Angle** constraint depends on whether or not the **Angle** constraint is parameterized. If it is not, then the **Parallel** constraint simply replaces it. If the **Angle** constraint is parameterized, however, the **Parallel** constraint is not assigned. Instead, the value of the parameter is adjusted to 0° or 180° (effectively parallel). This is done so as not to invalidate any expressions that may reference the existing angle parameter.

Parallel lines need not describe an orthogonal area in the part; [Figure 13. Parallel Lines on page 57](#) shows two sets of parallel lines.

Figure 13. Parallel Lines



Perpendicular

The **Perpendic** constraint is assigned to pairs of linear elements. A pair of perpendicular-constrained elements will be adjusted to be perpendicular at solve time. If other constraints on the part prevent this adjustment, **Solve** fails. The positions, lengths, and absolute slopes of perpendicular elements are not affected by this constraint and may change during the solve, but the slopes of the lines will be at right angles to one another.

To assign **Perpendic** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Perpendic**.
2. Click **Apply**.
3. Click each of the linear elements to be constrained.
4. Click **End** or assign the next constraint.

Each linear element in a perpendicular pair is labeled with the perpendicular icon, [Figure 3. Constraint Icons on page 46](#), and a reference number that is unique to the pair.

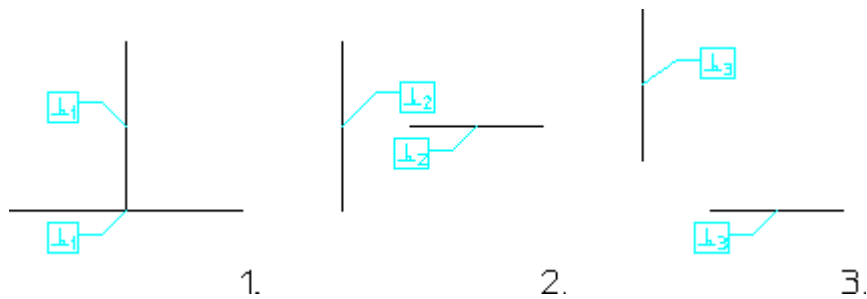
Application Notes

A given pair of linear elements may have only one of the **Angle**, **Parallel**, or **Perpendic** constraints assigned to it. This is because **Parallel** and **Perpendic** are simply special cases of the **Angle** constraint. If you attempt to assign a **Perpendic** constraint to a pair of elements that is already assigned a **Parallel** constraint, the **Perpendic** constraint will replace it.

The action taken when you assign a **Perpendic** constraint to an element pair that is already assigned an **Angle** constraint depends on whether or not the **Angle** constraint is parameterized. If it is not, then the **Perpendic** constraint simply replaces it. If the **Angle** constraint is parameterized, however, the **Perpendic** constraint is not assigned. Instead, the value of the parameter is adjusted to 90° or 270° (effectively perpendicular). This is done so as not to invalidate any expressions that may reference the existing angle parameter.

Lines need not be in contact with one another to be constrained perpendicular. [Figure 14. Perpendicular Lines on page 59](#) shows three different pairs of perpendicular lines.

Figure 14. Perpendicular Lines



Point on

Point on constrains a model point to remain on a given geometric element or its extension.

The point part of this constraint may be any visible model point. Line endpoints, circle centers, spline control points, and text reference points may all be used if visible. If necessary, tick the **Vertex** check box in the Creo Elements/Direct Drafting **Show** dialog box to make these elements visible.

The extension of an element is its construction-geometry equivalent. For example, the extension of a line segment encompasses any point that would fall on a collinear construction line. The extension of an arc element encompasses any point that would fall on a construction circle that shares the arc's center and radius. All Creo Elements/Direct Drafting geometric elements, including splines, B-splines, and ellipses, are eligible to receive **Point on** constraints.

The constrained point may slide anywhere along the extension of the element it is attached to, but will always remain in contact with the extension. If other constraints prevent this contact, **Solve** fails.

To assign **Point on** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Point on**.
2. Click **Apply**.
3. Click the point to be constrained. You can select any visible model point (Use the Creo Elements/Direct Drafting **Show** dialog box to display model points if needed). If no point element exists at that location, one will be created.
4. Click the geometry to be attached to the point.
5. Click **End** or assign the next constraint.

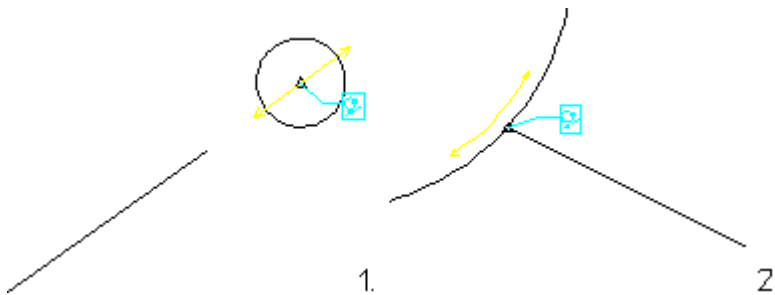
The **Point on** icon, [Figure 3. Constraint Icons on page 46](#), identifies this constraint.

Application Notes

Two common applications for **Point on** constraints are shown in [Figure 15. Point-On Applications on page 60](#). Circles can be constrained to stay on a linear element by fixing the centerpoint on the line. Conversely, elements can be constrained to remain attached to the perimeter of a circle, ellipse or spline.

Note that to assign a **Point on** constraint to the centerpoint of a circle or arc, you must first display centerpoints (with **Vertex** switched on in the **Creo Elements/Direct Drafting Show** dialog box) and then click the centerpoint directly.

Figure 15. Point-On Applications



Reference Element

Any element in a part (except ellipses) can be fixed in place with a **Refelem** constraint. Reference elements are not changed in any way by **Solve**, so they will appear in the same position in the variation as they do in the master part.

To assign **Refelem** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Refelem**.
2. Click **Apply**.
3. Click the element to be constrained.
4. Click **End** or assign the next constraint.

Reference elements are labeled with the anchor icon, [Figure 3. Constraint Icons on page 46](#).

Application Notes

You should almost always constrain one or two elements in the master part as reference elements (construction geometry is good for this) so that **Solve** has a frame of reference for generating the variation. Otherwise, the variation may not be placed where you expect.

Reference Point

Any visible model point can be constrained to be a reference point. **Solve** places reference points in the variation at the position specified by the constraint.

To assign a reference point:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Refpoint**.
2. Click **Apply**.
3. Click the point to be constrained. You may select any visible model point (Use the Creo Elements/Direct Drafting **Show** dialog box to display model points if needed) If a point element does not exist at the location you click, one will be created.
4. Input a position for the reference point. You have two options:
 - Click **End** or select another point. The reference point will be placed into the variation at its current position at solve time.
 - Enter a parameter name or Creo Elements/Direct Drafting expression that resolves to the point position (string input).

The reference point will be labeled with the position symbol, [Figure 3. Constraint Icons on page 46](#). If the reference point is set to a parameter name, that information is appended to its icon.

Application Notes

Reference points, like reference elements, can be used to "anchor" portions of the master part in place, so that they maintain their positions in the variation.

To assign a reference point constraint to a model point that is not usually visible, e.g., circle centers, spline control points, or text centers, you must first display them (with **Vertex** switched on in the Creo Elements/Direct Drafting **Show** dialog box) and then click them directly.

A reference point can also be used to supply data to macros or expressions when given a parametric value. If you supply a parameter label when assigning a reference point, the coordinate value of the point can be accessed in an expression by referencing the parameter name.

Same Distance

Samedist is assigned to two pairs of elements. All geometric elements except for spline-based entities (rational splines, bsplines, and ellipses) are eligible. You can assign constraints to the interpolation or control points of these entities, just not the spline itself. **Samedist** forces the distance between the members of the first

pair of elements to be the same as the distance between the members of the second pair. If other constraints prevent the two distances from being made equal, **Solve** fails.

To assign **Samedist** constraints:

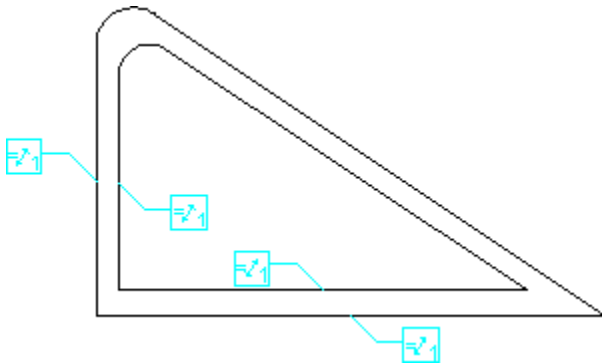
1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Samedist**.
2. Click **Apply**.
3. Click the two elements for the first pair.
4. Click the two elements for the second pair.
5. Click **End** or go on to the next constraint.

Each element in a same-distance set is labeled with the same-distance icon, [Figure 3. Constraint Icons on page 46](#), and a reference number that is unique to the set.

Application Notes

Samedist constraints are often useful for forcing two contours to become/remain equidistant, [Figure 16. Same Distance Application on page 62](#).

Figure 16. Same Distance Application



Same Size

The **Same Size** constraint is applied to pairs of geometric elements, which must be of the same geometric type. All elements except for spline-based entities (rational splines, bsplines, and ellipses) and construction lines are eligible for this constraint. The members of a same-size pair will be adjusted as needed by **Solve** to become the same size. Specifically, the radii of circular elements and the length of linear elements is adjusted. If the members of a same-size pair are not in fact the same size at solve time, **Solve** will look at other constraints on the members of the pair to determine which member should be changed to match the size of the other.

To assign **Samesize** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Samesize**.
2. Click **Apply**.
3. Click the two elements to be constrained.
4. Click **End** or assign the next constraint.

Each element in a same-size pair is labeled with the same-size icon, [Figure 3. Constraint Icons on page 46](#), and a reference number that is unique to the pair.

Size

You can assign **Size** constraints to lines and all circular elements. At solve time, these elements will be adjusted to the size specified by the constraint. For circular elements, the adjustment is made to the radius; for lines, the length is adjusted. If the adjustment cannot be achieved, **Solve** fails.

To assign a size constraint:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Size**.
2. Click the element to be constrained.
3. Choose the size for the constraint. You have three options:
 - Click **End** or select another element. The element will be held at its current size at solve time. This is the default.
 - Enter a specific size (radius for circular elements, length for lines).
 - Enter a parameter name or Creo Elements/Direct Drafting expression that defines the size (string input).

Each size-constrained element pair is labeled with the size icon, [Figure 3. Constraint Icons on page 46](#). If the constraint is set to a specific value or parameter name, that information is appended to the icon.

Slope

Slope constraints are assigned to linear elements. At solve time, the slope of these elements will be adjusted to match the angle specified by the constraint. If the adjustment cannot be made, **Solve** fails. Note that slope constraints do not fix the position or length of a line, which may change unless otherwise constrained.

To assign a slope constraint:

-
1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Slope**.
 2. Click **Apply**.
 3. Click the linear element to be constrained. The current slope of this element is displayed on the Creo Elements/Direct Drafting prompt line.
 4. Choose the angle for the constraint. You have three options:
 - Click **End** or select another element. The slope will be set to the actual slope of the line at solve time.
 - Enter a specific angle value for the slope.
 - Enter a parameter name or Creo Elements/Direct Drafting expression that defines the slope angle (string input).

Each slope-constrained element is labeled with the slope icon, [Figure 3. Constraint Icons on page 46](#). If the constraint is set to a specific value or parameter name, that information is appended to the icon.

Application Notes

A given element may have only one of the **Horizontal**, **Vertical** or **Slope** constraints assigned to it. If you attempt to assign a **Slope** constraint to an element that is already constrained **Horizontal** or **Vertical**, the **Slope** constraint will replace the existing constraint.

Parametric Design defines the slope of a linear element as the angle which is swept by rotating the vector (1,0) to the intrinsic vector of the element. This intrinsic vector is the line that runs from the first to the second endpoint of the element. The intrinsic vector of a construction line depends on how it was created. This method is used so that rotations greater than 90° can be specified.

As a result of the method used to measure slope, it is not possible to visually determine the slope of a linear element from the graphics display. For example, a vertical line may have a slope of 90° or 270°, depending on whether it was constructed from "top to bottom" or "bottom to top." You can, however, read the current slope from the prompt line in step 3 of the assignment procedure above.

Symmetry Line

The **Symmline** constraint identifies a line of symmetry that can be used by **Complete** to assign mirror constraints. If you want to assign mirror constraints by hand, you should use **Assign Mirror** instead. You assign **Symmline** constraints to linear elements, which are then labeled with the symmetry line icon, [Figure 3. Constraint Icons on page 46](#).

To assign **Symmline** constraints:

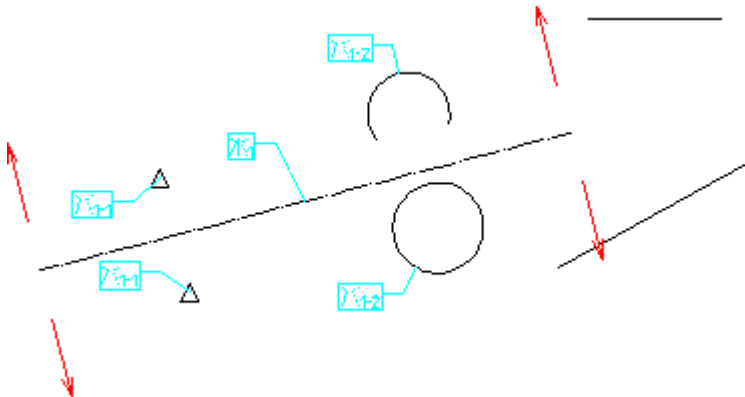
1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Symmline**.
2. Click **Apply**.
3. Click the linear element to be constrained.
4. Click **End** or assign the next constraint.

Application Notes

Parametric Design can automatically assign symmetry-line constraints to lines having a particular color and linestyle. You enable this behavior by calling the function `PD_AUTO_SYMMETRY YES` from the Creo Elements/Direct Drafting command line, or by including it in your `pd_def.mac` file. The functions `PD_AUTO_SYMMETRY_COLOR` and `PD_AUTO_SYMMETRY_LINETYPE` specify the color and linestyle. See [Parametric Design Defaults on page 175](#) for more information.

Complete only extracts mirrored pairs from areas that are orthogonal to defined symmetry lines. If your symmetry lines have infinite length (e.g., construction lines) this is no limitation, but if you choose a finite-length element as a symmetry line, items must lie completely within the orthogonal boundary in order to be selected for mirroring. The two line segments on the right of [Figure 17. Auto-selection of Mirrored Pairs on page 65](#), for example, would not be extracted because neither falls inside the boundary. If you are assigning mirrored pairs manually, this limitation does not apply.

Figure 17. Auto-selection of Mirrored Pairs

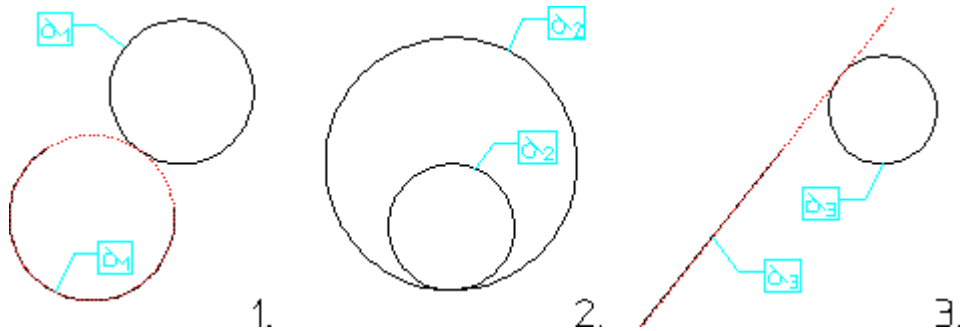


Tangent

Tangent constraints are applied to pairs of geometric elements, at least one of which must be circular (circle, construction circle, arc, or fillet). At solve time, the elements are adjusted to ensure that their extensions are tangent. See [Point on on](#)

page 45 for an explanation of extensions. If this adjustment cannot be achieved, **Solve** fails. Two elements are considered tangent if their extensions touch at one point only, Figure 18. Tangent Pairs on page 66.

Figure 18. Tangent Pairs



To assign **Tangent** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Tangent**.
2. Click **Apply**.
3. Click each of the two elements to be constrained. One must be circular.
4. Click **End** or go on to the next constraint.

Each element in a tangent pair is labeled with the tangent icon, Figure 3. Constraint Icons on page 46, and a reference number that is unique to the pair.

Application Notes

Complete will not extract a tangent constraint for a pair of elements unless the elements themselves (not their extensions) are tangent. **Solve**, however, recognizes linear and circular extensions when adjusting for tangency.

Solve attempts to maintain as much of the current configuration as possible when adjusting to satisfy tangency constraints. When adjusting a circular and a linear element to be tangent, **Solve** attempts to keep the circle on the same side of the line as it was originally. When adjusting two circles to be tangent, effort is taken to ensure that if one circle is inside the other, it remains inside, and that circles which are outside of each other stay that way.

Vertical

Linear elements that are assigned a **Vertical** constraint will be adjusted as necessary at solve time to become vertical. If **Solve** cannot make this adjustment due to other constraints on the element, **Solve** fails. A vertical constraint will not prevent a linear element from moving or changing length in the variation; it only adjusts the slope of that line to 90° or 270° .

To assign **Vertical** constraints:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Vertical**.
2. Click **Apply**.
3. Click the linear element to be constrained.
4. Click **End** or assign the next constraint.

The vertical icon, [Figure 3. Constraint Icons on page 46](#), identifies vertically constrained lines.

Application Notes

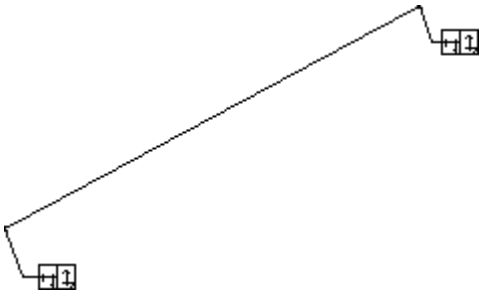
A given element may have only one of the **Horizontal**, **Vertical** or **Slope** constraints assigned to it. If you attempt to assign a **Vertical** constraint to an element that is already assigned a **Horizontal** constraint, the **Vertical** constraint will replace it.

The action taken when you assign a **Vertical** constraint to an element that is already assigned a **Slope** constraint depends on whether or not the **Slope** constraint is parameterized. If it is not, then the **Vertical** constraint simply replaces it. If the **Slope** constraint is parameterized, however, the **Vertical** constraint is not assigned. Instead, the value of the parameter is adjusted to 90° or 270° (effectively vertical). This is done so as not to invalidate any expressions that may reference the existing slope parameter.

X-Distance and Y-Distance

The X-Distance and Y-Distance constraints are special point-to-point distance constraints. At **Solve** time, the x-coordinates (and y-coordinates) are adjusted to match the distance specified by the constraint. In [Figure 19. X-Distance and Y-Distance Constraints on page 68](#), the two points will be constrained x- and y-distances.

Figure 19. X-Distance and Y-Distance Constraints



The X-Distance and Y-Distance constraints cannot be assigned manually. They are generated automatically during the Design Intent Capture process with LINE TWO_PTS and LINE POLYGON (see [Figure 24 on page 85](#) in [Design Intent Capture on page 85](#)).

5

Parameters

Creating Parameters	70
Assigning Parameters.....	71
Editing Parameter Definitions	72
Setting Parameter Values.....	76

A parameter is a named variable that you create with Parametric Design. You can create a parameter implicitly by giving a parameter name when you assign one of the several constraint types that support parameters; or you can create a parameter explicitly by adding a new entry to the parameter definition table, which you access by clicking **Advanced** in the **Current Constraints** dialog box. Regardless of how you create a parameter, it is given an entry in this table. By editing the entry, you can assign a value to the corresponding parameter.

The value you assign to a parameter may be either a discrete numeric value or an expression. An expression is any valid Creo Elements/Direct Drafting expression. This line can reference other Parametric Design parameters, Creo Elements/Direct Drafting macros, and any of the arithmetic and logical constructs supported by the Creo Elements/Direct Drafting macro language.

The most important feature of parameters is that they can be assigned to constraints. After you assign a parameter to a constraint, you can modify the constraint by changing the parameter's value. Conversely, you can access the current value of the constraint by inquiring the parameter. This capability makes parameters an especially powerful tool for defining relationships among elements in a part.

Creating Parameters

Creating Parameters Implicitly

Parametric Design implicitly creates a new parameter whenever you **Assign** one of the following constraint-types and supply an unused parameter name (string) when prompted to do so by the software:

- **Angle**
- **Dimension**
- **Distance**
- **Refpoint**
- **Size**
- **Slope**

The name you give must conform to the naming rules given in the [Assigning Parameters on page 71](#), section and must not be the same as an existing parameter name. If you give an existing parameter name, that parameter will simply be assigned to the constraint and no new parameter will be created.

When you create a new parameter with this procedure, Parametric Design implicitly creates a default definition for the parameter and then binds it to the constraint you're assigning.

By default, the new parameter is given the current value of the constraint. For example, if you assign a size constraint to a 10 cm line and give the label 'Len_3' at the prompt, Parametric Design will create a new parameter, Len_3, and give it a value of 100, assuming your current length units are mm. Note that parameters always use the units they were created with. Len_3, for example, will always be in mm even if you change your length units to inches.

Creating Parameters Explicitly

You can also create new parameters directly by adding entries to the parameter definition table. Parameters that you create this way are not automatically assigned to constraints on your part. For this reason, explicit creation is often used to create parameters that will be used only to supply constants, sub-expressions, or other values that have no relationship to the part.

To explicitly create a parameter, bring up the parameter definition table by clicking **Advanced** in the **Current Constraints** dialog box. The parameter definition table provides the following buttons for the creation of new parameters:

Command	Action
Length	Create a new length parameter
Angle	Create a new angle parameter

Command	Action
Point	Create a new point parameter
User	Create a new user (unitless) parameter

Each of these commands adds a new parameter to the table. These parameters are not associated with constraints, but you can name and define them like any other parameter. The new parameters can be referenced in other expressions and can reference other parameters in their own expressions. Once defined, you may use them for reference or assign them to constraints in your part. To create a new parameter:

1. Click the applicable button, for example: **Angle**.
2. Enter a name for the new parameter, e.g.: 'Angle_4'.

The name you give must be a suitable parameter label as described in [Assigning Parameters on page 71](#). An explicitly created parameter is given an initial value of 0 and a value type of "unknown." Its value remains unknown until you assign it a value or until you assign it to a drawing element.

Assigning Parameters

Only the following constraint-types may be parameterized:

- **Angle**
- **Dimension**
- **Distance**
- **Ref Point**
- **Size**
- **Slope**

To assign a parameter to any of these constraint-types, you simply **Assign** the constraint to the desired element(s), and enter the parameter name (enclosed in quotes) when prompted to do so.

Only those constraint types that support parameters will prompt you for them. If the named parameter does not exist, then Parametric Design creates a new parameter with this name, as explained above. If a parameter with this name does exist and matches the unit requirements of the constraint, it becomes associated with the constraint.

Parameter names are text strings. They must adhere to the naming rules for Creo Elements/Direct Drafting macros. Parameters cannot have the same name as an Creo Elements/Direct Drafting keyword, (e.g. rad or CONFIRM). To avoid such conflicts, parameter names should begin with an uppercase letter, and contain only lowercase letters, integers, and/or the underscore (`_`) character in the remaining positions. The names `Distance`, `Rad`, `Dim_3`, and `My_line` are examples of

valid parameter names. The software always displays parameter names with the first letter uppercase and remaining characters lowercase, regardless of how you enter them.

It is always a good idea to choose parameter names that are as descriptive as possible - especially if you are parameterizing a part that other people will use as a master part. A draftsman is much more likely to understand the purpose of the parameter Circle3_radius than the parameter Wwxyz.

Once you assign a parameter name to a constraint, the parameter remains assigned until you assign a different parameter name to this constraint, or until you **Free** the constraint.

Editing Parameter Definitions

The parameter definition table can be used to display and edit the definitions of all existing parameters on the current part. To bring up the table, click **Advanced** in the **Current Constraints** dialog box. The parameter definition table contains one entry for each parameter. Each entry has five fields, which define the parameter:

- **Name**
- **Kind**
- **Value**
- **New Value Type**
- **New Value or Expression**

Only the **New Value or Expression** field is directly modifiable. The first four fields just supply information. The following sections describe each field.

Name

The **Name** field gives the parameter name as you assigned it. This is simply the label that identifies the parameter.

Kind

The **Kind** field declares the units for this parameter. Units are determined by the type of constraint this parameter is assigned to. The following "kinds" of units are recognized:

Kind	Units
Angle	An angle in the angular units in force when the parameter was created (degrees, radians, etc.).
Length	A linear distance in the linear units in force when the parameter was created (mm, inches, miles, etc.).

Kind	Units
Point	An XY coordinate pair in the linear units in force when the parameter was created; e.g., 20,30.5.
user	Unitless. A user parameter is never associated with a geometric constraint, so the solver never has to convert this parameter into system units. Only parameters created with the User option can be user parameters.

Value

The **Value** field shows the value that is currently associated with this parameter. This value may change when you next **Solve** for a variation. The value for a parameter is always given in the same units that were in force when the parameter was first created.

Parameters that resolve to point values use two lines in the parameter table. The value on the first line is the X coordinate of the point, while the second line contains the Y coordinate.

New Value Type

The **New Value Type** field specifies the method that **Solve** will use to evaluate this parameter. **New Value Type** must be one of:

Value Type	Meaning
Value	Set the value for this parameter to the specific value in the New Value or Expression field. Assume that the given value is in the correct units. Other parameters can depend on this parameter via expressions. This value is available to any geometric constraints that reference this parameter to solve for the geometry.
Expression	Generate a value for this parameter by resolving the expression given in the New Value or Expression field. Assumes that the expression is a valid Creo Elements/Direct Drafting expression which resolves to a value of the correct type. If the expression contains references to other parameters then the expression is only evaluated after all referenced parameters have a value. Conversely, other parameters can depend on this parameter via expressions. The value produced by this expression is available to all geometric constraints that reference this parameter.

Value Type	Meaning
Geometry	<p>The value for this parameter is taken from geometry that is solved for using constraints that are not associated with this parameter. Other parameters can depend on this parameter via expressions.</p> <p>For example, take a part in which there are two circles whose radii are parameterized as Radius_1 and Radius_2. If the first circle is tangent to three fixed lines then the value of Radius_1 is completely determined by the "geometry" of the three lines. An expression for Radius_2 such as (2 * Radius_1) is then possible. This works because Solve recognizes that the expression for Radius_2 should not be evaluated until Radius_1 is known. Radius_1 can only be known after the three tangent line constraints are evaluated.</p>
Unknown	<p>This parameter has been named, but has no association to an existing constraint and has no value or expression assigned. This parameter type is not evaluated by the solver.</p>

New Value or Expression

Use this field to assign new values for parameters. The value or expression contained in this field at **Solve** time will be interpreted and assigned to the corresponding parameter. If you leave this field blank, then the parameter value is extracted from the geometry of the part.

To Assign A Specific Value To A Parameter:

1. Click the parameter's **New Value or Expression** field.
2. Type in a numeric value (in appropriate units) at the keyboard.

The number you give appears in the **New Value or Expression** field and the **New Value Type** field is set to "Value". This value will be assigned to this parameter by the next **Solve** command.

To Assign An Expression To A Parameter:

1. Click the parameter's **New Value or Expression** field.
2. Type in a valid Creo Elements/Direct Drafting expression. All expression input must be enclosed in quotes. For example:

(57 + Dist)	Assign the sum of 57 and 'Dist' to the parameter.
(IF (A < 8) 57 ELSE 45 END_IF)	Assign 57 if the value of 'A' is less

	than 8; otherwise assign 45.
(PNT_XY (X_OF 'PtA') 10)	Assign to a point parameter the x-value of 'PtA' and a y-value of 10.
My_macro	Evaluate 'My_macro' and assign the result.
My_macro2 Rad	Evaluate 'My_macro2', which needs a parameter that passes in the value of 'Rad'.

The expression you give appears in the **New Value or Expression** field and the **New Value Type** becomes "Expression." The next **Solve** command will cause this expression to be evaluated by Parametric Design and the result assigned to this parameter.

To Extract A Value From The Geometry:

1. Click the parameter's **New Value or Expression** field.
2. Type in an empty string ("") to clear the contents of the field.

The **New Value or Expression** field is now blank and the **New Value Type** becomes "Geometry." Subsequent **Solve** commands will cause Parametric Design to extract a value for this parameter from the constrained geometry.

Geometry-type constraints are often useful for determining the locations of elements in the part. Suppose for example that you have defined several expressions that need to know the location of a particular point in the part. An easy way to pass this data is with a parameterized **Refpoint** constraint:

- Assign the **Refpoint** constraint to the model point you're interested in. Give the parameter name `Pt1` to this constraint as you assign it.
- Use the parameter definition table to convert `Pt1` to a "Geometry" type parameter.
- Now, whenever you **Solve**, the new location of the model point will be extracted and assigned to `Pt1`. Your expressions can access the location by referencing `Pt1`.

Removing Parameters

Use the **Remove** command located in the upper right of the parameter definition table to remove parameters from the part. To **Remove** a parameter, click **Remove**, then click the **Name** field for the parameter you want to remove. Parametric Design will not allow you to remove parameters that are currently assigned to a constraint or that are referenced by another parameter via expression. To remove parameters that are assigned to constraints, first either **Free** the constraint that corresponds to the parameter or assign a new value to it, then **Remove** the

parameter entry. To remove a parameter that is referenced by other parameters, you must first remove the offending references by editing the expressions that contain them.

Showing Parameters

The **Show** button in the parameter definition table highlights all elements in the part that currently have parameters assigned to them. Highlighting is done by displaying the constraint icons of these elements along with the corresponding parameter names. Parameter labels will be displayed using the current text font. Highlighting is in CYAN by default.

The **Show** highlighting is removed whenever **Solve** is invoked. Alternatively, you can select **Clear** and **All Types** in **Generate Constraints** to remove the highlighting.

Setting Parameter Values

A common strategy in Parametric Design is to define your parameters so that the entire constraint structure of a part depends on several "Value"-type parameters. This strategy has several advantages:

- It is often possible to generate new variations by simply changing a few key value-type parameters.
- In situations where a Designer or Engineer is responsible for designing constraints for a master part and a Draftsperson is responsible for generating variations, this strategy "isolates" most of the constraint structure from the Draftsperson, who only has to modify the values of a few parameters to generate variations.

Parametric Design facilitates this strategy by giving easy access to all the currently set "Value" parameters in the parameter value table in **Current Constraints**. Rather than display the entire parameter definition table whenever you want to change a "Value" parameter, you can display the **Current Constraints** dialog box, which contains only "Value" parameters.

Each entry in the parameter value table contains the name of a currently defined "Value" parameter along with the value that will be assigned to that parameter by the next **Solve** command. This value corresponds to the **New Value or Expression** field of the parameter definition table. Only those parameters with a type of "Value" appear on this menu. If there are too many entries to view at once, use the scroll bar beside the table. To change the value of a parameter, simply select its entry and enter the new value. A full description of the parameter value table can be found in [Generating Variations on page 17](#).

 **Note**

While you can enter an expression rather than a value into the parameter value table, it is not recommended that you do so, because this changes the type of the parameter from "Value" to "Expression," and thus removes its entry from the parameter value table (but not from the parameter definition table).

6

Dimension Driven Modification

Roadmap.....	80
Overview	80
Viewing Parametric Dimensions	80
Display Attributes of Parametric Dimensions	81
Modifying Parametric Dimension Values	82
Swapping Parametric Dimensions	83

Roadmap

Read this chapter to learn how to modify geometry by modifying parametric dimensions. To learn how to assign constraints explicitly read [Working with Parametric Design on page 23](#) instead. To learn how to create, assign, and modify parameters read [Parameters on page 69](#).

Overview

Dimension Driven Modification simplifies access to some of the power of parametric modification. Geometry can be modified by simply selecting dimensions and entering new values.

Constraints associated with a number or a parameter (angle, dimension, distance, size, slope) can be considered parametric dimensions. Regardless of how they were created, all parametric dimensions can be modified via **Parametric, Modify, Dimension**.

Viewing Parametric Dimensions

Display

To display parametric dimensions associated with selected geometry, click **Parametric ► Modify ► Dimension ► Show**.

Displaying parametric dimensions may require the automatic creation of temporary geometry (which is not visible). For example, dimensioning the radius of a construction circle is not allowed, so a real circle will be created temporarily for the display of parametric dimensions. Any temporary geometry is deleted by the Parametric Design module as soon as parametric dimensions are removed from the display.

Parametric dimensions can be displayed only when a part is fully and properly constrained. If the part is underconstrained when you click **Show** and then **Dimensions**, you will be asked to confirm the running of **Complete**. **Complete** can be executed reliably when the part is mostly constrained — if the geometry was created with Design Intent Capture enabled, for example. But if the part does not contain any constraints **Complete** should not be run until some initial constraints have been assigned. Refer to [Working with Parametric Design on page 23](#) for information on constraint assignment.

Note

Keep in mind parametric dimensions are only associated with constraints that contain a number or a parameter. Constraints such as parallel, horizontal, same size, reference element, etc. will not have parametric dimensions associated with them. A combination of constraint editing and dimension creation can be used to construct any desired parametric dimensions.

Hide

To hide parametric dimensions associated with selected geometry, click **Parametric ► Modify ► Dimension ► Hide**.

Especially with large parts, the ability to selectively display and hide parametric dimensions can reduce screen clutter. A designer can quickly focus on the parametric dimension of interest.

Display Attributes of Parametric Dimensions

Color

All parametric dimensions are displayed with the same color. To set the color value use PD_DEFAULT_DIM_COLOR. For example, to specify green as the color for parametric dimensions use:

```
PD_DEFAULT_DIM_COLOR 0,1,0 [Enter]  
PD_DEFAULT_DIM_COLOR 0,1,0 [Enter]
```

All other attributes are controlled by the current settings of the dimension module.

Location

Parametric dimensions — unlike other dimensions — are displayed in default locations. They can be moved for easier viewing with **Parametric ► Modify ► Dimension ► Move**.

 **Note**

If you have the Dimension Advisor Module enabled, the parametric dimensions are placed according to the Strategy option (Automatic in the Options tab in Dimension Settings).

Modifying Parametric Dimension Values

Once parametric dimensions are displayed, their values can be modified. Changing the value of a parametric dimension will result in a corresponding change to the geometry of a drawing. The drawing can be updated after every modification, or the drawing update can be deferred until many parametric dimensions have been modified.

Immediate Modification

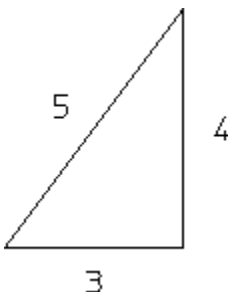
Immediate specifies that all geometry should be updated as soon as the value of a parametric dimension is modified. This method can be useful with small drawings — where the speed of updating the geometry and redrawing the screen is not a concern. It can also be useful for experimenting with several different values for one parametric dimension.

Deferred Modification

Deferred specifies that geometry should not be modified until the user makes an explicit request. This method will improve performance, because the drawing will be updated only once — even if the values of several parametric dimensions have been modified.

Deferred is actually necessary when a set of parametric dimension modifications do not have valid intermediate geometry. For example, consider a triangle with sides of lengths 3, 4, and 5:

Figure 20. Triangle Requiring Deferred Modification



It would not be possible to change the side lengths to 30, 40, and 50 using **Immediate**, because no valid triangle can be created with a combination of old and new lengths.

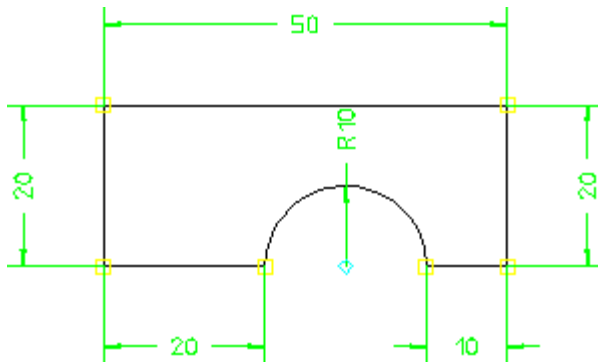
 **Note**

Often a combination of Immediate and Deferred is useful when several parametric dimensions need modification.

Swapping Parametric Dimensions

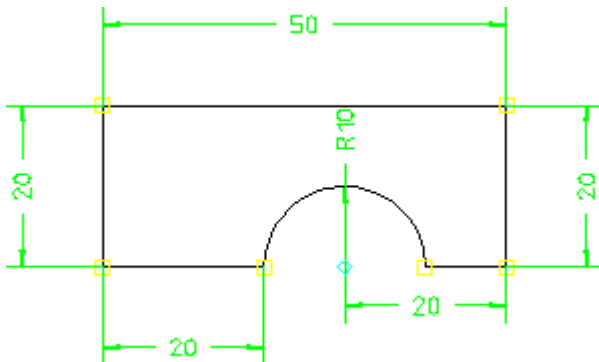
The parametric dimensions generated by **Complete** many not be appropriate in all cases. In these cases it is possible to create a traditional dimension and then indicate that it should become a parametric dimension. For example, consider this drawing and the parametric dimensions that were generated with **Complete**:

Figure 21. Parametric Dimensions Generated by "Complete"



If the center of the arc should be located from the right side, tools in the **Dimension** menu can be used to create an appropriate dimension. Then you click **Swap**, and the newly-created dimension is selected to replace the parametric dimension generated by **Complete**. The resulting parametric dimensions are:

Figure 22. Parametric Dimensions After Swapping



Restrictions

In order for an exchange to occur both dimensions must control the same geometry. For instance, in the example above the height of the rectangle could not be exchanged for the distance from the center of the arc to the right edge.

Design Intent Capture

Roadmap.....	86
Overview	86
Flexibility	86
Enabling, Disabling and Inquiring.....	86
Elements Affected	87
Constraint Creation	87
Operations on Existing Geometry	89

Roadmap

Read this chapter to learn how to create constraints as geometry is created. To learn how to assign constraints and parameters explicitly, read [Working with Parametric Design on page 23](#) instead.

Overview

Design Intent Capture generates and assigns parametric constraints as geometry is constructed. Constraint types depend on the method used to construct geometry. For example, when creating one line perpendicular to another line a perpendicular constraint will be created and associated with the two lines.

Design Intent Capture improves the speed, accuracy, and quality of parametric constraint definition by combining the separate steps of geometry creation and constraint assignment into one action.

Without Design Intent Capture constraints must be assigned after geometry has been created. This forces the designer to remember the relationships between geometric elements and record them by assigning constraints. Even worse, if the geometry was created by someone else the designer must try to reason backward and guess at the appropriate constraints. These problems are avoided by using Design Intent Capture.

The quality of constraints in the final parametric drawing is usually much better when Design Intent Capture is used compared with relying on **Complete** to generate all constraints. Quite possibly a drawing will be fully constrained when Design Intent Capture is used. But even if some additional constraints are required, **Complete** will have a much better starting point.

Flexibility

Although Design Intent Capture generates constraints, the designer is free to modify or remove any of them. The constraints are not history-based, so they may be modified regardless of the order in which they were created. Any constraint can be altered or removed without negating all of the work that occurred after the constraint was generated.

In other words, Design Intent Capture is a method of generating constraints — as are **Complete** and direct assignment. All of the Parametric Design module's constraint editing capabilities can be used on them. Refer to [Working with Parametric Design on page 23](#) for more information on editing constraints.

Enabling, Disabling and Inquiring

To enable Design Intent Capture type:

```
UA_DESIGN_INTENT ON [Enter]
```

```
UA_DESIGN_INTENT ON [Enter]
```

To disable Design Intent Capture type:

```
UA_DESIGN_INTENT OFF [Enter]
```

```
UA_DESIGN_INTENT OFF [Enter]
```

To inquire the on/off status of Design Intent Capture use the arithmetic function `UA_GET_DESIGN_INTENT`. A value of 0 is returned if Design Intent Capture is disabled, and a value of 1 is returned if Design Intent Capture is enabled. For example, to display the current on/off status of Design Intent Capture type:

```
DISPLAY (UA_GET_DESIGN_INTENT) [Enter]
```

```
DISPLAY (UA_GET_DESIGN_INTENT) [Enter]
```

Elements Affected

The following methods of geometry creation generate parametric constraints when Design Intent Capture is enabled:

- point
- line
- circle
- arc
- fillet
- chamfer
- construction line
- construction circle
- text
- equidistance
- overdraw

Constraint Creation

The Parametric Design module considers two guidelines when determining the appropriate constraints to generate with Design Intent Capture:

- construction technique
- CATCH information

Construction Technique

Different constraints will be generated depending on the method of geometry creation. LINE PARALLEL, LINE PERPENDICULAR, LINE HORIZONTAL, and LINE TAN2 all convey different intentions, and all will result in different constraints.

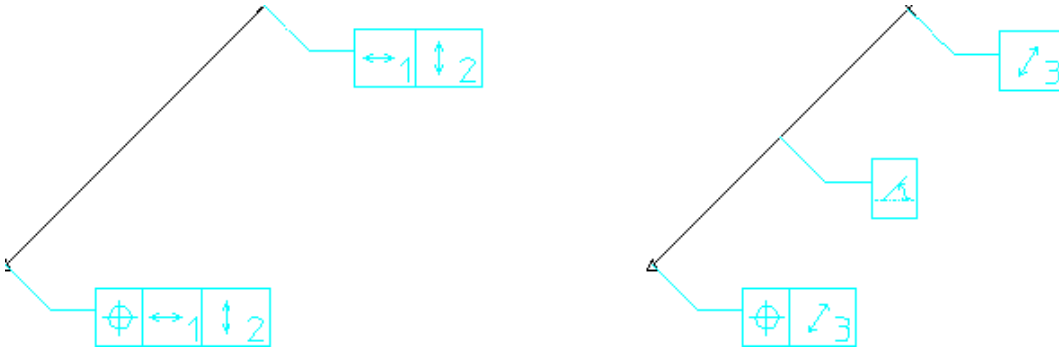
Even if two drawings look identical, they probably will have different constraints if different construction techniques were used to create them. For example, consider these two lines. Each is connected to a fixed point at one end. But one was created with LINE TWO_PTS and the other was created with LINE PT_ANG_DIST. Both drawings look the same:

Figure 23. Lines Displayed Without Constraints



But the constraints are different:

Figure 24. Lines Displayed With Constraints



CATCH Information

Design Intent Capture also considers CATCH information when determining which constraints to generate. With CoPilot enabled the ability to catch to tangencies, etc. is much easier.

For example, assume CoPilot is enabled and LINE TWO_PTS is invoked. If the endpoints of the line are picked so that they are approximately tangent to two arcs, then two tangent constraints will be created.

Note

Because different constraints are generated based on construction technique and CATCH information only advanced users should enable Design Intent Capture when macro files are being created or input.

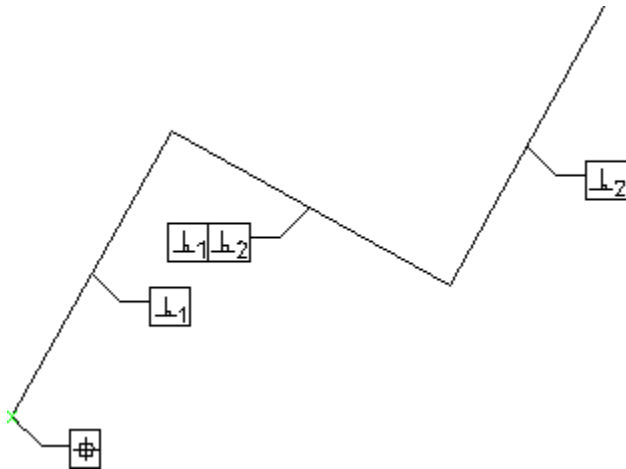
Operations on Existing Geometry

In addition to operations which create geometry, Design Intent Capture applies to operations that modify geometry. Constraints will be created, deleted, modified, or transferred appropriately for different modification operations.

Delete

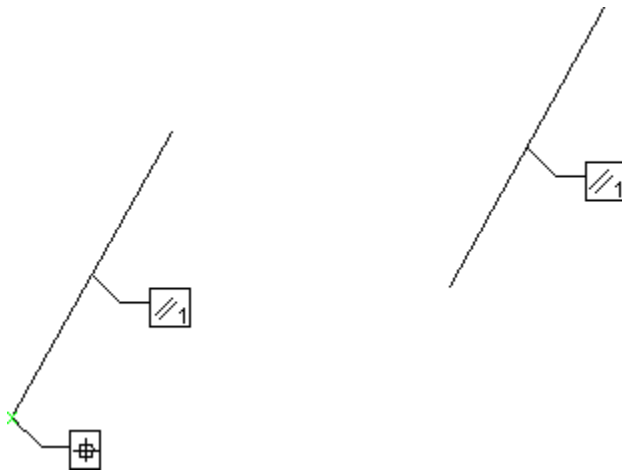
When geometry is deleted all relational parametric constraints will be transferred to other geometry whenever possible. For example, consider three perpendicular lines:

Figure 25. Perpendicular Lines with Constraints



If the middle line is deleted the perpendicular constraints associated with it will also be deleted. But a new constraint will be generated so that the parallelism between the remaining lines is kept:

Figure 26. One Perpendicular Line Deleted



Split

When elements are split constraints will be assigned to maintain consistency between the pieces. Each portion of a split line will be given a collinear constraint. Split circles and arcs will be assigned same size constraints.

In addition to the basic constraints for split lines and arcs, other constraints may need to be modified or assigned. For example, if a line with a size constraint is split, a distance constraint will be created and associated with the endpoints of the original line.

Merge

When elements are merged constraints will be combined whenever possible. For example, when line segments are merged the collinear constraint associated with them will be removed (if it exists). Relational constraints between the segments and other elements will be associated with the new, merged line.

Modify Keep, Stretch Keep, Mirror Keep

Geometry copied with MODIFY KEEP or STRETCH KEEP will also have all associated constraints copied. Note that no constraints will exist between the original and the new copies of the geometry.

Mirrored elements will be assigned constraints so that both portions of the geometry continue to have the mirror property. In other words, once geometry is mirrored with Design Intent Capture both portions will remain mirror images of each other — even as other modifications are made.

Modify NoKeep, Stretch NoKeep, Change_fillet

Design Intent Capture will attempt to modify constraint values without changing any constraint types. For example, the radius of a fillet or the distance between two elements may be changed, but a perpendicular constraint will not be changed to a collinear constraint. If Design Intent Capture is not able to calculate a new value for a constraint, the constraint will be eliminated.

8

Advanced Topics and Tips

Roadmap.....	94
Geometry Cleaning	94
A Simple Sketch Input Implementation.....	95
Zone Gymnastics	98
Parametric Control of Splines.....	99
Rigid Bodies: Applications and Tips	101
Implementing Replication in Parametric Design	102
Allowing Rotation	103
Working with Multiple Parts	104
Using Modify With Parametric Design	104
Automatic Constraint Generation.....	106
A Short Description of the Solver	108
How Expressions are Evaluated	108
Invisible Geometry.....	109

Roadmap

The information in this section is mostly of interest to advanced Parametric Design users. If you are new to Parametric Design, skip this section for now and come back to it later.

Parametric Design has a number of useful features and applications that may not be immediately obvious to the casual user. This chapter describes several of these and provides some in-depth information on the workings of **Complete** and **Solve**.

Geometry Cleaning

Complete and **Solve** both have the ability to recognize and compensate for minor geometric inconsistencies in a part. Common problems such as lines which are slightly out of parallel, elements with zero length or radius, and small gaps between "tangent" elements can be recognized and corrected.

Note

The "cleanup" capabilities used by **Complete** and **Solve** are separate from the **Clean** command. See [Working with Parametric Design on page 23](#) for more information on the **Clean** options.

Parametric Design handles most geometric inconsistencies by incorporating tolerances into several of the rules used by **Complete** to extract constraints from a part. If an element does not precisely fit the requirements for a given constraint, but is within the tolerance, the constraint is assigned anyway. Once **Complete** is done, you need only **Solve** with the **No Keep** option to adjust these elements into the precise configuration given by the constraint.

For example, **Complete** can extract a **Horizontal** constraint for any line element with a slope that is $0 \pm$ the current angular tolerance (given by `PD_AUTO_ANGLE_TOLERANCE`). If the tolerance is 0.5, then lines with slopes of 0 ± 0.5 can be assigned **Horizontal** constraints. When **Solve** is run, these lines will be adjusted as needed to become truly horizontal.

The following tolerances are used by **Complete**:

`PD_AUTO_ZERO_DISTANCE_TOLERANCE` Specifies the absolute tolerance used by **Complete** to compare a distance to zero. This tolerance is used to detect coincident points, collinear lines and **Point on** constraints. Must be greater than or equal to 0. The initial setting is 0.000001 mm.

`PD_AUTO_SAME_DISTANCE_TOLERANCE` Specifies the absolute tolerance used by **Complete** to compare two non-zero distances. Used in several areas; to check whether two circular elements have the same radius, for example. Must be greater than or equal to zero. The initial value is 0.000001 mm.

PD_AUTO_ANGLE_TOLERANCE Specifies the absolute tolerance used to compare angles. This tolerance is applied, for example, when checking whether two lines are parallel or a single line is horizontal. Must be greater than or equal to zero. The initial value is 0.000001 radians.

PD_AUTO_TANGENT_TOLERANCE Specifies the absolute distance tolerance used by **Complete** to check whether two elements are tangent. Applied only if one of the extracted elements is circular. The tolerance is used to compare the radius of the circular element to the distance from the circular element center point to the other element. The number supplied must be greater than or equal to zero. The initial setting is 0.000001 mm.

By default, these tolerances are set very small to ensure that Parametric Design only compensates for true anomalies in the part and does not remove or "adjust away" small details that are important to you. Setting a tolerance to 0 effectively eliminates its effect on **Complete**. Depending on the type of parts you normally work with, you may want to experiment with different values for each of these tolerances until you arrive at a set of tolerances that provides maximum cleaning without removing important details. Tolerances may be redefined on the Creo Elements/Direct Drafting command line, and the default values can be changed by editing the `pd_def.mac` file.

An additional tolerance, **PD_RESOLVE_MERGE_TOLERANCE**, is used directly by **Solve** to merge coincident points and remove very small ("zero length") elements from variations. Points that are within this tolerance are merged, thus removing duplicate points and eliminating elements that are smaller than the tolerance. The default, which is defined in `pd_def.mac`, is 0.000001 mm.

Caution

It is usually a good idea to disable tolerance-based merging in motion-study or kinematics applications where several entities will be moved around in close proximity to one another. In these applications, it is often the case that two entities which are intended to remain separate will come into close enough contact that a point on each will fall within the merge tolerance. If merging is on, the two entities will be effectively fused together, preventing further independent motion. To disable merging, set the tolerance to -1.

A Simple Sketch Input Implementation

The preceding discussion of geometry cleaning has assumed that the tolerance capabilities of the solver will be used to correct for small deficiencies in the input geometry. Suppose however, that you expect substantial inconsistencies in a part and that you don't care if you have to adjust it substantially to clean it up. A

"sketch input" application is a common situation where these assumptions would be true. In these situations, you can use Parametric Design with relatively large tolerances to "square up" a quickly-drawn part.

The following macro implements a simple "sketch input" application using Parametric Design:

```
DEFINE SquareUp                                {A simple drawing straightener}
{Accept linear and angular tolerances as arguments}
  PARAMETER LinearTolerance
  PARAMETER AngularTolerance

{Set adjustment tolerances to the supplied values }
  PD_AUTO_ZERO_DISTANCE_TOLERANCE  LinearTolerance
  PD_AUTO_SAME_DISTANCE_TOLERANCE  LinearTolerance
  PD_AUTO_ANGLE_TOLERANCE          AngularTolerance
  PD_AUTO_TANGENT_TOLERANCE        LinearTolerance
  PD_RESOLVE_MERGE_TOLERANCE       LinearTolerance

{If AUTO must create new points, make sure we can delete them later}
  ADD_CURRENT_INFO "DELETE" END

{Load the entire current part into the zone}
  PD_ZONE_ADD ALL                          {ZONE Add Mult ALL}

{Generate constraints; tolerances are applied}
  PD_RESOLVE GENERATE                      {AUTO}

{Resolve conditionally.  If the preview is OK, accept the change}
  PD_RESOLVE PREVIEW                      {SOLVE Preview}
  WINDOW FIT
  READ 'Please enter CONFIRM (c) or UNDO (u)' DEFAULT 'c' Accept_flag
  IF ((Accept_flag = 'c') OR (Accept_flag = 'C'))
    PD_RESOLVE REPLACE                      {SOLVE NoKeep}
  ELSE
    END
  END_IF

{If AUTO had to generate ref. points to constrain, remove them now}
  DELETE SELECT INFOS "DELETE" ALL END

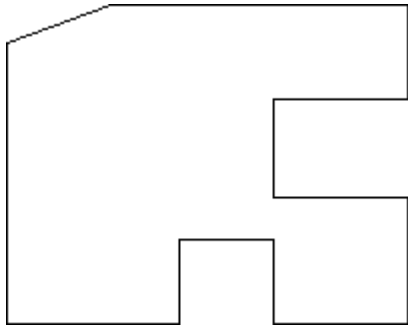
{Cleanup: remove constraints, zone data, DELETE info. reset tolerances}
  PD_FREE NOBAN ALLTYPES CONFIRM
  PD_ZONE_REMOVE ALL
  CHANGE_CURRENT_INFO "DELETE" "" END
  PD_AUTO_ZERO_DISTANCE_TOLERANCE  1E-06
  PD_AUTO_SAME_DISTANCE_TOLERANCE  1E-06
  PD_AUTO_ANGLE_TOLERANCE          1E-06
  PD_AUTO_TANGENT_TOLERANCE        1E-06
  PD_RESOLVE_MERGE_TOLERANCE       1E-06
```



```
{Display what I've made!}  
  REDRAW  
END_DEFINE {of SquareUp}
```

To get a feel for how this macro works, copy the text above into a file and then Input the file to Creo Elements/Direct Drafting. Note that this chapter is supplied in HTML format on the Creo Elements/Direct Electronic Manuals CD-ROM. Now quickly create the part below. Use **Polygon** in **Geometry** with the Creo Elements/Direct Drafting catch-mode set to **OFF**. Try to reproduce the figure as shown, but don't worry about inconsistencies in your part.

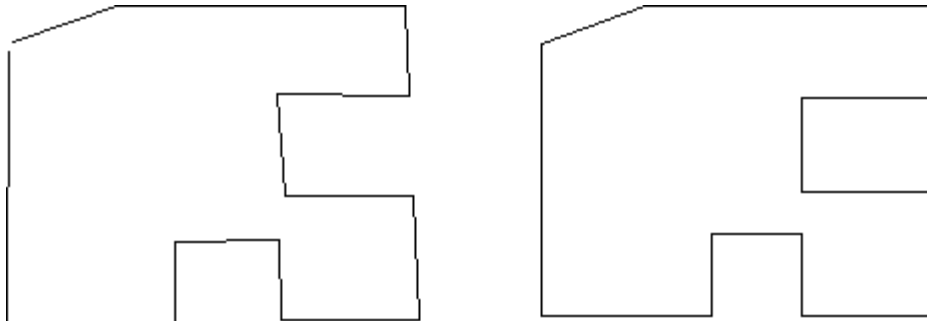
Figure 27. Sample Polygon



Now type in: SquareUp 2 5 [Enter]

If all went well, Parametric Design has cleaned your "sketch." Specifically, gaps of 2mm or less should now be closed. Lines that were within +/- 5 degrees of horizontal or vertical should now be "squared up." Also, lines that were not quite collinear (+/- 2mm) should now be snapped into place, [Figure 28. A Cleaned Sketch on page 97](#).

Figure 28. A Cleaned Sketch



If the first pass at this didn't work for you (remember, SquareUp is a very simple macro that will handle many, but certainly not all cases!), reject the results and experiment with different values for the length and angle tolerances.

As you experiment with SquareUp, it may be helpful to try it with severely flawed sketches and to give "unreasonable" values for the linear and angular tolerances to learn the strengths and weaknesses of this technique.

Zone Gymnastics

Parametric Design's zone mechanism is implemented using Creo Elements/Direct Drafting infos. The info text PD_ZONE is used to specify which drawing elements are in the Parametric Design zone. Any drawing element in the current part that has the info text PD_ZONE attached to it is included in the zone. By treating the zone as simply a collection of element infos, several powerful options for working with the zone become available. The following sections contain a few ideas.

Automatic Inclusion

As shipped, the pd_def.mac file contains the string:

```
ADD_CURRENT_INFO "PD_ZONE" END
```

This causes all new geometry created during your Creo Elements/Direct Drafting session to be automatically included in the zone (i.e., the PD_ZONE info is added to all new geometry). You can turn off the auto-inclusion feature at startup by removing this line from pd_def.mac. To turn off auto-inclusion from within Creo Elements/Direct Drafting, you could type in:

```
CHANGE_CURRENT_INFO "PD_ZONE" ""
```

Note that these functions will not affect geometry input from an MI file. If the contents of an MI file were in the zone when they were saved to the file, they will be put into the zone when the file is restored. If the contents were not in the zone when the file was created, they are not placed into the zone when the file is restored.

Alternate Add/Remove Schemes

You can use **Info** commands instead of the screen menu to add or remove elements from the zone. This can be useful when using Parametric Design from within a macro. Typical commands might be:

```
ADD_ELEM_INFO "PD_ZONE"           Add selected elements to the zone
CHANGE_ELEM_INFO "PD_ZONE" ""      Remove selected elements from the zone
```

Pre-Define Multiple Zones

In an evolving, complex part, you may find that several different areas are frequently being modified with Parametric Design. Each time you want to work on a different area of the part, you need to redefine the contents of the zone. If your part is truly complex, this may be a time-consuming task. If you want to make sure that the same elements are always included in the zone each time you work in a particular area, you'll have to be very careful when you assign elements to the zone. This task may become even more difficult and error prone if several people are working on the same part.

One solution to this problem is to use infos to pre-define several "quasi-zones" in a part and then to activate each one as needed. Suppose, for example, you have a part with three different areas that must regularly be modified with Parametric Design. Try the following strategy:

- First, clear all zone infos from the part:
`CHANGE_GLOBAL_INFO "PD_ZONE" ""`
- Next, assign to all the elements in the first area an arbitrary info. Let's call it "AREA_1": `ADD_ELEM_INFO "AREA_1"` (select elements)
- Repeat this step for the second and third area, giving each its own info: `ADD_ELEM_INFO "AREA_2"` (select elements) `ADD_ELEM_INFO "AREA_3"` (select elements)

Now that each group of elements has a unique info associated with it, you can easily "activate" each group for Parametric Design by swapping its info text for the PD_ZONE info text and "deactivate" it by restoring the old info:

- To load the elements in "AREA_1" into the zone: `CHANGE_GLOBAL_INFO "AREA_1" "PD_ZONE"`
- To remove "AREA_1" elements from the zone and load "AREA_3" elements into the zone: `CHANGE_GLOBAL_INFO "PD_ZONE" "AREA_1"`
`CHANGE_GLOBAL_INFO "AREA_3" "PD_ZONE"`
- To remove "AREA_3" elements from the zone: `CHANGE_GLOBAL_INFO "PD_ZONE" "AREA_3"`

Parametric Control of Splines

Creo Elements/Direct Drafting currently supports two types of spline entities: rational cubic splines (C-splines), which are created with the SPLINE function, and B-splines, which are created with the BSPLINE function. Earlier versions of Creo Elements/Direct Drafting only supported C-splines. Parametric Design can be used to parametrically control the shape of any spline, but the procedure for constraining C-splines and the newer B-splines differs somewhat. The following two sections give some tips and techniques for controlling each type of spline.

C-Splines

To parametrically control the shape of a C-spline, you first place point elements on the data points used to create the spline and then assign the appropriate constraints to these points to actually control the shape of the spline. Any constraint that can be assigned to a point can be used.

To find the location of the C-spline's data points, turn on vertices with `SHOW VERTEX ON`. Before assigning point elements to the spline data points, you should also make sure that splitting is turned off.

Note

By default, Creo Elements/Direct Drafting converts all rational splines in a drawing into B-splines when you load it. The conversion will invalidate any constraints that you have assigned to the data points of these rational splines. Refelem constraints which are assigned directly to spline elements are not invalidated, and will be carried over from the rational spline to the new B-spline.

To prevent spline conversion and the subsequent invalidation of constraints on rational splines, you must enter `SPLINE_CONVERSION OFF` prior to loading any master part that contains constrained rational splines. Alternatively, you could include this line in the Creo Elements/Direct Drafting `pd_def` file to permanently disable spline conversion.

B-Splines

Unlike C-splines, which are controlled via their data points, the shape of a B-spline can only be controlled by constraining its control points. To parametrically set the shape of a B-spline, you must first display its control polygon, either with the `CNTRL POLY` command on the screen menu, or by entering `SHOW_CPOLY ON` and selecting the B-spline. With the control points displayed, use same method as for C-splines to constrain them: place point elements on each control point and then constrain those point elements. At solve time, as the control points move, the shape of the spline will also change.

If you assign constraints only to the data points of a B-spline and it appears that they are causing the spline to change shape, it's likely that splitting was on when you placed the point elements on the data points and you are now manipulating multiple B-splines!

Rigid Bodies: Applications and Tips

Rigid bodies are useful for a wide range of applications. The list below presents a few ideas.

- Motion studies of linkage assemblies becomes much simpler. Make each link a rigid body. Then use constraints to describe the motion between the links. Your model will not be cluttered with constraints for the links, so you can concentrate more on the interactions between links.

A feature of rigid bodies that is especially useful for motion studies is the fact that a single element can legally belong to more than one rigid body. A single point element included in two rigid bodies acts much like a hinge between them, while a single construction line works much like a slider. An arc or line segment shared by two rigid bodies serves to "weld" them together.

- Parametrics can now control (to some degree) subparts and/or shared subparts. Collect each subpart into a rigid body along with some geometry from the current part. Construction geometry is often a good choice for this. You can control the position and orientation of each subpart through constraints on the construction geometry.
- Complicated parts often have large amounts of geometry which you want to move as a group. Because the automatic constraint generator has a poor notion of locality, it is usually simpler and more predictable to identify these groups of entities manually.
- Control hatch angle. Suppose you want a hatching to remain at a given angle to some line. Create a construction line collinear to this line (and constrain it to be collinear), then make a rigid body out of the construction line and the hatching. Use the construction line instead of the original line in the rigid body so that you don't prevent the original line from changing size by including it in the rigid body.
- Control the location and orientation of symbol subparts.
- Provide a form of model cleaning without modifying geometry. Stacked elements, unclosed polygons, etc. can wreak havoc on parametric applications. The usual approach to solving these problems is to use the **Clean** commands to merge points and remove stacked elements. In cases where it is not possible or desirable to clean the original geometry, you may be able to use rigid bodies to isolate problem geometry from the solver. Simply collect problem areas into rigid bodies so that the solver does not need to examine individual elements too closely. Of course, this approach may limit what you can accomplish with parametrics, as the contents of the rigid bodies can only be moved and rotated, not resized.

Here are a few potential trouble spots that you should keep in mind while working with rigid bodies:

- Be careful when creating a rigid body that contains only subparts. While these are perfectly valid rigid bodies, remember that you cannot assign constraints to subparts. As a result you would not be able to control the position or rotation of the rigid body via constraints. It will remain in its original position in all subsequent variations.
- Similarly, be careful when creating rigid bodies that contain just a subpart and a single point element from the current part. Such a rigid body can be translated via constraints, but its rotation cannot be changed.
- A subpart may be assigned to more than one rigid body. At solve time, however, it will always remain associated with one of the rigid bodies to which it was assigned. All other rigid body associations will be ignored.
- Finally, consider the case of a rigid body that contains only a subpart and a construction line. In this case it is possible to control the rotation of the rigid body and translations that are perpendicular to the axis of the construction line, but not translations along the axis of the construction line. The subpart will retain its original position along the axis of the construction line regardless of subsequent rotation and off-axis translation.

Implementing Replication in Parametric Design

A common requirement when creating a family of parts is to have a particular geometric feature replicated a variable number of times, depending on the overall size or configuration of the surrounding geometry in each variation. A circular plate, for example, might be expected to have 8 holes if its radius is 4 inches, and 12 holes if the radius is 6 inches.

While Parametric Design provides no functionality to perform such modifications directly, the following procedure often does the trick:

1. Define the zone to be all of the geometry in the part except for the portion that is to be replicated. Constrain the geometry in the zone as if you were setting up a master part.
2. Create some additional parameterized constraints that can be used to query the result of a variation. These can be **Refpoint**, **Dimension**, **Distance**, **Angle**, **Size**, or **Slope** constraints. Assign these constraints so that they return the values needed to determine the number of replications needed.

-
3. Write a macro to query the necessary parameters, compute the number of replications from this data, and create those replications in the correct location.
 4. Finally, write a macro which first invokes PD_RESOLVE to solve for the geometry in the zone, and then calls the macro defined in the previous step to take care of the replications.

A number of alternative strategies can be derived from this procedure. For example, if both the number of replications and the size of the replicated geometry needs to vary, consider using two different zones, one for the surrounding geometry and one for a single instance of the replications. In this case, the final macro might have the following logic:

1. Run PD_RESOLVE for the surrounding geometry.
2. Use PD_PARAM_INQ to read out the results.
3. Compute values for the input parameters for the instance of the replication geometry from the result you've just obtained.
4. Change the current zone to the replication geometry.
5. Use PD_PARAM_FIX calls to load the input parameters needed for the replication geometry.
6. Run PD_RESOLVE for the replication geometry.
7. Use **Modify** commands to move, rotate and copy the replication geometry as needed.

Allowing Rotation

It is often desirable for variations of a part to be able to rotate about a given point as a condition for their creation. The simplest way to allow a part to rotate is to define the reference constraints on the master part to be a reference point and a line slope. Usually it is best if the reference point is on the line whose slope is being constrained. After applying the reference constraints, constrain the rest of the part without using slope constraints (i.e., no **Slope**, **Horizontal**, or **Vertical** constraints. If you use **Complete** to generate constraints, make sure to check for any system-generated slope constraints, and change these to angle-type constraints.

Now by simply changing the value of the constrained slope the entire part will rotate about the reference point.

Working with Multiple Parts

Parametric Design constraints and parameters are all local to a given Creo Elements/Direct Drafting part. Thus, subparts can be stored independently from the rest of a drawing with their complete parametric definition, and parameterized subparts can be loaded into a drawing without corrupting the constraint and parameter data for other parts already in the drawing.

There are two situations, however, in which the parametric data from more than one part can come into contact. The interactions that occur are worth noting:

You can explicitly set up interrelationships between parts by writing macros like the one given in the "Macros" example in [Examples on page 111](#). This example shows how a parameter in one part can have its value depend on the resulting geometry in another part. Since a parameter can refer to an angle, slope, distance, size, or dimension, and since any number of points can be parameterized for output query, this method provides a very general solution to the problem.

A second (often less desirable) form of interaction can occur as a result of the Creo Elements/Direct Drafting **Smash** and **Gather** commands. **Smash** and **Gather** have the following behavior:

- If you **Gather** without copying a subset of the constrained elements in a parameterized part, all of the constraint and parameter data for that part is lost.
- If you **Gather** all of the constrained elements in a part or **Smash** one or more subparts, then the constraint and parameter data will be kept. Note, however, that if the same parameter name exists in two or more of the parts being merged, the results can be unpredictable. Generally, if there is a conflict between parameter names, the parameter assigned to the part that is highest in the parts tree is maintained. If both parts are at the same level, one or the other will be selected, but it is impossible to predict which.

Using Modify With Parametric Design

The Creo Elements/Direct Drafting **Modify** command can generally be used along with Parametric Design to modify geometry. **Modify** can be used successfully between the time that a part is constrained and the time that **Solve** is issued, but modifications during this time will also affect the results of **Solve**. These effects may be surprising if you are not expecting them, but can be very useful once understood.

- If a parameterizable constraint (i.e., **Angle**, **Size**, **Distance**, **Slope**, or **Dimension** constraint) is assigned without giving an expression or parameter name, then the value of the constraint is determined at solve time. Thus, any modifications you make to such elements will be recognized by **Solve**. For example, if you

assign a valueless **Size** constraint to a 7 inch line, and then **Stretch** that line to 12 inches prior to solving, **Solve** will keep the line at 12 inches.

- **Collinear, Parallel, Perpendic**, and **Tangent** constraints all have orientation characteristics which can be affected by **Modify**.
 - The angle between two **Parallel** or **Collinear** lines can be 0 or 180 (see the explanation of **Slope** constraints in [Constraint Types on page 45](#)). **Solve** will use whatever angle is closest to the actual positions of the lines at solve time.
 - Similarly, the angle between two **Perpendic** lines can be 90 or 270 degrees. The preferred angle is the angle that it is closest to at solve time.
 - If a line and circle are constrained **Tangent**, **Solve** attempts to find a location for the circle center point on the same side of the line that it started out on at solve time.
 - If two non-concentric circles are constrained **Tangent**, there are three possible solutions. Either the first circle will be inside the second, the second circle will be inside the first or both circles will be tangent externally. **Solve** attempts to maintain the particular relationship that is in place at solve time.

In all of the above cases, **Modify** can be used to change the location of the constrained elements and thus change **Solve**'s preferred solution, even though the constraints haven't changed.

- Geometry constrained as reference elements can be modified at any time before solving. **Solve** will always leave such elements in their current position at solve time.

Modifying Constraint Icons

Parametric Design constraint icons are simply text elements whose contents and location are managed by the software. While there is nothing to prevent you from modifying the location, text attributes, or contents of the icons with Creo Elements/Direct Drafting, doing so has several implications that you should be aware of:

- Each constraint icon string is derived from the constraint and from the applicable icon viewing commands (e.g., **Show** and **Clear**). Modifying the contents of the string, or deleting the string altogether, doesn't alter the underlying constraint in any way.
- Copying an icon string or placing the leader line on a different element doesn't copy the constraints to that element. In fact, a copy of an icon string is not recognized by Parametric Design and becomes just another text element that can be added to the zone and constrained!

- Any change in the attributes of an icon string, such as slant, will be forgotten when the entire icon string is cleared from the screen. The next time the icon string is constructed it will use the default settings for new text elements.
- The constraint icon characters are part of the hp_symbols font. These characters are inserted into locations 65-96 of hp_symbols when Parametric Design is enabled. You should not attempt to use these locations either before or after Parametric Design is enabled. Unpredictable results can occur.

Automatic Constraint Generation

To extract constraints from an existing part, **Complete** uses the Parametric Design solver to apply a set of extraction rules to the geometry in the zone. Understanding how these rules work will help you to anticipate what kinds of constraints are generated by **Complete**.

When you select **Complete**, the solver first scans the geometry for existing constraints. If these fully constrain the part, **Complete** is done. If the existing set of constraints does not fully constrain the part, the solver then starts to evaluate the under-constrained members of the part and assign new constraints to them according to the rules listed below. Each time a new constraint is extracted, it is added to the list of known constraints and the part is checked again to see whether or not it is now fully constrained. As soon as it is, the process stops. Otherwise, the solver continues to go down the list of rules, adding constraints and checking until the part is fully constrained. The solver will not go on to a new rule until all possible constraints have been extracted according to the current rule. A rule will not be applied to elements that have the **Ban** flag set for the constraint type assigned by the rule. The rules and their basic functions are (in order of application):

Rule	Action
Special Symmetry lines	If PD_AUTO_SYMMETRY is on, find all lines that have the auto-symmetry linetype and color and assign them Symmline constraints.
Collinear Lines	Find collinear lines and assign them Collinear constraints.
Coincident Points	Find coincident points and constrain them together.
Points On Geometry	Find suitable elements and assign them Point on constraints.
Mirrored Elements	If symmetry lines are defined, find mirrored elements and assign them Mirror constraints.
Dimensions	Constrain any free dimensions to their current value.
Fillets	Find arcs that qualify as fillets and assign them Fillet constraints.
Parallel Lines	Find parallel line elements and assign them Parallel

Rule	Action
	constraints.
Perpendicular Lines	Find perpendicular elements and assign them Perpendicular constraints.
Tangent Elements	Find tangent linear or circular elements and assign them Tangent constraints.
Horizontal Lines	Assign Horizontal constraints to horizontal elements.
Vertical Lines	Assign Vertical constraints to vertical elements.
Mirrored Points	If symmetry lines are defined, find mirrored points and assign them Mirror constraints.
Same Radius	Find circles with the same radius and assign SameSize constraints.
Free Angle	Find elements with free slopes and assign Angle constraints.
Free Slope	Find elements with free slopes and assign Slope constraints.
Free Radii	Find circles with free radii and assign Size constraints.
Free Length	Find lines with free lengths and assign Size constraints.
Free Elements	Find any remaining free elements and constrain them with Distance and/or Refelem constraints.
Free Parameter	Look through the parameter table. For all parameters that do not have a set value, assign the current value.

When extracting constraints from the part, the solver must determine the location and connectivity of each element in order to decide if that element is a suitable candidate for the constraint type currently being assigned. The user can specify several tolerances for the solver to use when determining an element's location and connectivity. These tolerances, described in detail in [Geometry Cleaning on page 94](#) above, allow the solver to handle minor inconsistencies that may occur in the part.

Finally, after all the geometry has been examined and all new constraints have been extracted, **Complete** goes back through the part looking for redundant or unused constraints. These are removed. For example, if a line element is found that is constrained horizontal and is also constrained collinear with another line that is constrained horizontal, the horizontal constraint can be freed from the first line.

A Short Description of the Solver

The Parametric Design solver is a constraint solution kernel. The input to this kernel is a list of the geometric elements in the zone and the set of topological and metric constraints assigned to the geometry. Using the geometry list and constraints, the kernel can solve for the (new) location of the geometry.

The kernel is really just a smart sequencer of solution steps. Each step determines, or partly determines, some geometric entity (point, line, etc.) This is done by applying one (or more if necessary) constraints and some adjacent fixed entities to make a change in some other entity. When a change is made to an entity, constraints associated with that entity are scanned to see if one or more might be applicable.

Once the solution sequence has been determined, an expanded set of algebraic closed-form algorithms is invoked, rather than an iterative numerical technique. These algorithms employ simple geometrical constructions, such as curve/curve intersectors. The kernel is designed for speed and robustness, and to minimize calculation error.

One of the limitations in the kernel is that coupled or circular constraints are not resolved, since they usually imply a simultaneous system of equations, which the kernel does not attempt to solve. In these situations, the kernel calls the automatic constraint generator, which proposes an alternative set of constraints.

How Expressions are Evaluated

User expressions for parameters are evaluated when all of the independent parameters are known. When it is time to evaluate an expression, **Solve** constructs the following macro:

```
DEFINE Pdeval
  PARAMETER <independent parameter name 1>
  :
  PARAMETER <independent parameter name N>
  <user expression>
END_DEFINE
```

The dependent parameter is then assigned to the value of

```
Pdeval <value of independent parameter 1> ...
```

For example, if the expression for parameter A is (B+10) and at some point in the solution process B becomes 6, then Pdeval will be:

```
DEFINE Pdeval
  PARAMETER B
  (B+10)
END_DEFINE
```

And A is assigned to be the result of

```
Pdeval 6
```

A syntax error can occur during the evaluation. When this happens, a message is displayed indicating which dependent parameter has the faulty expression and **Solve** is terminated. Since the offending expression is the last expression to be evaluated, you can view it by running `EDIT_MACRO Pdeval` immediately after the failure occurs. This procedure may point out problems that might otherwise escape attention, since there is very little syntax checking done when the expression is defined.

For example, if the above sample expression were `(rad+10)` then the above macro would be:

```
DEFINE Pdeval  
  (rad+10)  
END_DEFINE
```

which fails because `rad` is a keyword and keywords are not recognized as parameters, even if there was a parameter called `Rad`.

Invisible Geometry

As constraints are created the Parametric Design module may need to create points and construction lines to serve as reference geometry. For example, when Design Intent Capture is enabled and `LINE PARALLEL` is invoked construction lines are created between the original and new lines, so that the two lines will remain both parallel and the same length.

Invisible Geometry is managed completely by the Parametric Design module. Whenever operations such as `DELETE`, `MODIFY`, or `STRETCH` are performed on real geometry any necessary changes are made to invisible geometry.

Viewing

Invisible geometry is displayed whenever an associated constraint is displayed. Typically constraints are displayed via the **Display** dialog box.

Viewing invisible geometry can be helpful in understanding the relationship between constraints and geometry.

Attributes

The only two types of invisible geometry that are created are points and construction lines.

PD_NEW_POINT_VISIBILITY and PD_NEW_C_LINE_VISIBILITY determine whether or not invisible geometry created by the Parametric Design module should be displayed at all times or only when constraints are displayed. Allowable qualifiers are VISIBLE and INVISIBLE.

Other functions that set display attributes of invisible geometry are:

- PD_NEW_POINT_COLOR
- PD_NEW_C_LINE_COLOR
- PD_NEW_POINT_LINETYPE
- PD_NEW_C_LINE_LINETYPE

9

Examples

Roadmap.....	112
Introduction.....	112
Example 1: Introductory Design Example	113
Example 2: Symmetry Lines.....	122
Example 3: Two Views.....	129
Example 4: Macros.....	133
Example 5: Rigid Bodies	139

Roadmap

Work through the examples in this chapter to familiarize yourself with the Parametric Design software. If you have not yet done so, you should read [Introduction on page 11](#) before continuing. If you are not familiar with Creo Elements/Direct Drafting, we also recommend that you read through *Creo Elements/Direct Drafting User's Guide* before continuing.

Introduction

This chapter contains several examples that will help you become familiar with Parametric Design tools and techniques.

All the examples assume that you are at a workstation, have Creo Elements/Direct Drafting up and running, and that Parametric Design is installed.

The first example introduces all the Parametric Design tools, proceeding step-by-step through an extended session with a simple drawing. We strongly recommend that you spend a little time going through this example before going on to the others. Subsequent examples present particular techniques or applications that you are likely to encounter as you use the software. These examples assume that you are comfortable with the Parametric Design interface.

The Creo Elements/Direct Drafting parts used in the examples are included online with the software. They are installed as uncompressed MI-format files in the `pd_demos\` directory. Several of the examples depend on information in these files, so we encourage you to load each file as you work through the corresponding example.

The examples were chosen to be simple, and yet introduce you to the breadth and depth of this product. After going through these examples and carefully considering the questions at the end of each one, you should have developed a feel for nature of parametric design.

It would be a mistake to hope that you will be an expert Parametric Design user after just using these examples. In the real world, geometry is often more complicated. Models are underdimensioned, have unwanted symmetries, suffer from geometric inaccuracies, or have other "features" that are difficult for the automatic constraint generator to interpret correctly.

There are two rules of thumb that apply when using Parametric Design:

- If it isn't clear to you, at least in general, how a part should change in response to a modified dimension, then AUTO will likely have trouble as well.
- The better you understand the interplay between the various constraint types, and the rules governing each constraint type, the more effectively you'll be able to use Parametric Design.

One very good way to become expert in the science of constraining a part, is to practice using manual constraints on parts of increasing complexity. Use the automatic constraint generator to see how it would solve a particular problem, but make sure you understand the constraints that are generated. Once you fully understand how the various constraint types can be used and you get used to using the debugging tools, then begin to rely more on the automatic constraint generator.

Example 1: Introductory Design Example

This example introduces the tools and working methods that the Designer or Engineer must use to constrain parts and generate variations with the Parametric Design module. During this first example, you will:

- Place the part in the zone.
- Fully constrain the part with manual and automatic methods.
- Examine the constraints.
- Modify the constraints needed for a variation.
- Solve for variations.
- Parametrize constraints to generate more useful variations.
- Add new geometry to a constrained part.

You should come away from this demo with a clear understanding of the tools and basic working techniques for constraining parts — information that you can apply in subsequent examples, which focus in more depth on particular applications of Parametric Design.

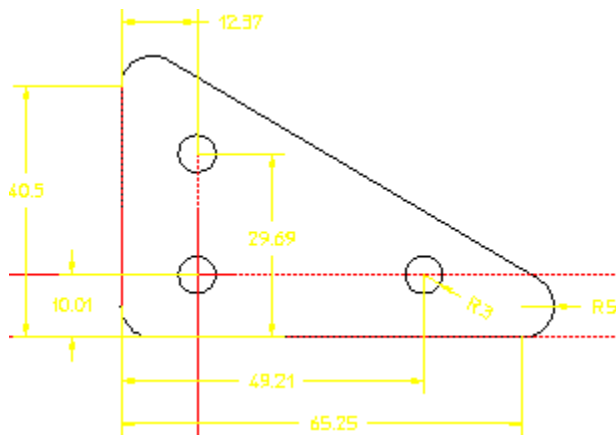
As with all of the examples in this section, the part for this example is supplied on line so that you can focus on parametric design instead of on drawing — after all, the main goal of Parametric Design is to reduce the time you spend drawing!

Load the Part

The part shown in this example is supplied with Creo Elements/Direct Drafting. To load this part:

1. Pull down the **File** menu and click **Open** to display the Creo Elements/Direct Drafting File Browser.
2. In the Creo Elements/Direct Drafting File Browser, switch to the `pd_demos` directory.
3. Select the `demopart01.mi` file and click **Open**.

Figure 29. Demopart01



The important features of this part for parametric design are as follows:

- It is an ordinary Creo Elements/Direct Drafting part.
- It is fully dimensioned. The length and height are specified, as is the radius of one corner. The location and radius of the holes is also given. Generally, a part with full dimensioning works best for Parametric Design.
- It is "clean." There are no duplicate or stacked elements. Such inconsistencies can lead to unexpected results. Parametric Design has several commands that find and fix inconsistencies in a part, but you won't have to use them right away!
- Construction lines have been used to mark the "origin" of the part as well as the centerlines of the holes. Construction geometry, if available, is often very useful to the Parametric Design solver. When possible, you should include construction geometry to mark important features of a part: major axes, lines of symmetry, centerlines of circles, etc.

Include the Part in the Zone

Parametric Design only modifies elements included in the zone. To include the entire part in the zone:

1. In the **Parametric** menu, click **Zone**.
2. Click **Add Multiple** in **Zone**.
3. Enter **all** and press **[Enter]**. All elements are highlighted as they are added to the zone.
4. Click **End**.

The entire part is now included in the zone.

Fully Constrain the Part

Before you can use the Parametric Design solver to generate variations from an existing part, each element in the zone must have one or more constraints assigned to it. Enough constraints must be assigned to tell the solver exactly what to do with each element in the part. If elements in the zone are not fully constrained or are incorrectly constrained, the solver will not be able to generate the variation.

The best way to assign a full set of constraints is to first constrain all the elements in the part in their current configuration. After the part is fully constrained in its current configuration, you can modify constraints as needed to generate a variation.

Both manual and automatic options are available for assigning constraints. Typically, the manual options are used to assign a few important constraints like reference elements, and then **Solve** will generate the remaining constraints needed to fully constrain the part.

Manually Fix Reference Elements

It is always a good idea to "anchor" a part by manually constraining one or more elements from moving. This helps to give the software a frame of reference for automatically assigning constraints later on. For this example, fix the lowest horizontal and leftmost vertical construction lines in place with **Refelem** constraints to anchor the part.

1. In the **Parametric** menu, click **Generate Constraints**.
2. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Refelem**.
3. Click **Apply**.
4. Click on the lowest horizontal construction line (collinear with the bottom edge of the plate).
5. Click on the leftmost vertical construction line (collinear with the left edge of the plate).
6. Click **End**.

Notice the reference-element icons. The leader lines of these icons are dotted so that you know they are assigned to construction geometry and not to the edges of the triangular plate. Each constraint type has its own unique icon. See [Figure 3. Constraint Icons on page 46](#) for a complete list of the constraint-type icons.

Modify Constraints

To generate a variation that is different from the original, you must change one or more constraints from their current values to the values you want to see in the variation.

-
1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Dimension**.
 2. Click **Apply**.
 3. Click the dimension text that gives the height of the triangle (40.5).
Type in: 55 [Enter]
 4. Click the dimension text that gives the length of the triangle (65.25).
Type in: 55 [Enter]
 5. Click **End**.

Notice how the dimension text is updated.

Automatically Generate the Remaining Constraints

Rather than assign the remaining constraints by hand, which would take considerable time, use the **Complete** option provided in the **Generate Constraints** dialog box. This option evaluates the part and assigns sufficient constraints to fully constrain it in its current configuration.

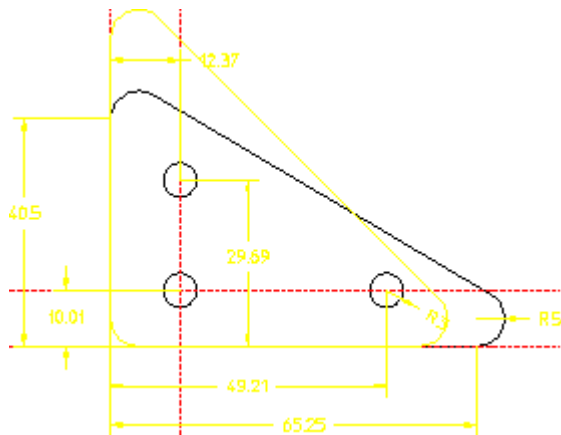
1. In the **Generate Constraints** dialog box, click the **Complete** radio button.
2. Click **Apply**.
3. Click **End**.

Solve

Now use **Solve** to generate the variation.

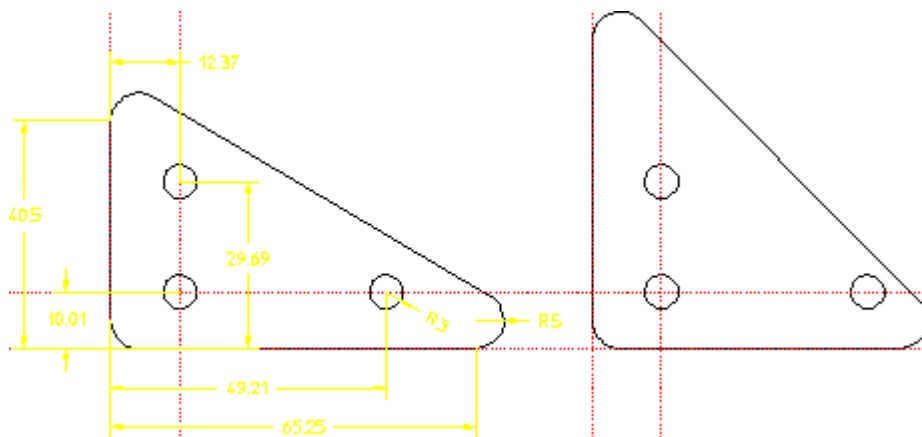
1. Click **Preview** in **Solve**. The magenta outline shows how the variation will be created without disrupting the current part, [Figure 30. Preview Mode on page 117](#).
2. Click **End**.
3. Click **No Keep** in **Solve**. Your original part is now replaced with the variation. Notice that the dimensions you assigned to be 55 earlier are now indeed 55. The hole geometry has not moved.
4. Click **Undo** to restore your original part.

Figure 30. Preview Mode



The third **Solve** option, **Keep**, allows you to keep both the original and the variation [Figure 31. Part and Kept Variation on page 117](#). If you try **Keep** now, make sure you delete all the variations (but not the master part) when you have finished.

Figure 31. Part and Kept Variation



Parameters

At this point, you've created a variation of your original part, but it's probably not exactly what you want. One obvious flaw is that the holes in the plate are not constrained to adjust along with the perimeter. Although you could move them by giving new values to their dimensional constraints (just like you did for the length and height of the plate), this is not efficient, because you would have to calculate and re-enter these values each time you generated a variation with different dimensions. Parameters provide a much better method for accomplishing this. Parameters allow you to use expressions to define relationships among constraints.

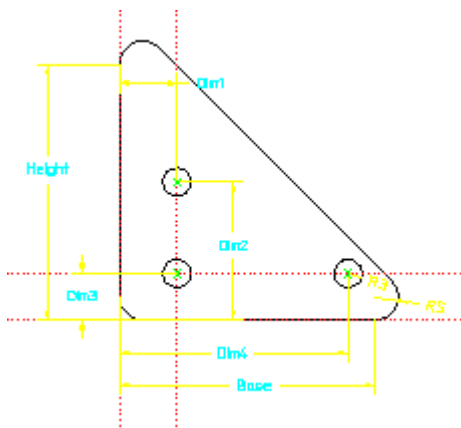
Assign Parameter Names

First, assign parameter names to the appropriate dimensional constraints.

1. Unless you have already done so, click **No Keep** in **Solve** and then **End** to replace the original part with the variation you created in the last step.
2. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Dimension**.
3. Click on the dimension text that gives the length of the triangle (55).
Type in: 'Base' [Enter]
4. Click on the dimension text that gives the height of the triangle (55).
Type in: 'Height' [Enter]
5. Click on the "12.37" dimension on the top hole.
Type in: 'Dim1' [Enter]
6. Click on the "29.69" dimension on the top hole.
Type in: 'Dim2' [Enter]
7. Click on the "10.01" dimension on the leftmost hole.
Type in: 'Dim3' [Enter]
8. Click on the "49.21" dimension on the rightmost hole.
Type in: 'Dim4' [Enter]
9. Click **End**.

The dimension text should now look like [Figure 32. Parameter Names Assigned on page 118](#).

Figure 32. Parameter Names Assigned



With the names assigned, you now define the relationships among the parameters in the parameter definition table. For this example, we'll set up the parameters so that all parameterized dimensions are functions of the Base parameter. The parameter definition table should resemble [Figure 33. Parameter Values Defined on page 119](#) when you are finished.

Define Parameter Values

1. Click **Current Constraints** in **Parametric**.
2. In **Current Constraints**, select the list entry for Height.
3. Click on the parameter value entry field to the left of the **Apply** button.
Type in: $(.75 * \text{Base})$ and click **Apply**. Note that the value for the Height constraint is now defined to be .75 of the value of Base.
4. Click the Dim1 entry.
Type in: Dim3 and click **Apply**.
5. Click the Dim2 entry.
Type in: $(\text{Height} - \text{Dim3})$ and click **Apply**.
6. Click the Dim3 entry.
Type in: $(.2 * \text{Height})$ and click **Apply**.
7. Click the Dim4 entry.
Type in: $(\text{Base} - \text{Dim1})$ and click **Apply**.
8. Click **Advanced** to view the result of these changes in the parameter table.

Figure 33. Parameter Values Defined

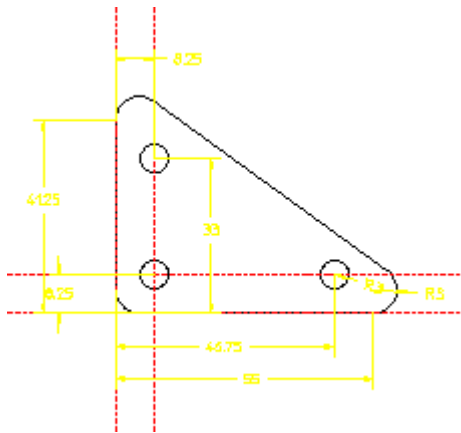
Parameters				LENGTH	ANGLE	USER	POINT	REMOVE	SHOW	DIFF	
Name	Kind	Value	New Value Type	New Value or Expression							
Base	Length	55	Value	55							
Dim1	Length	10.924	Expression	Dim3							
Dim2	Length	10.924	Expression	Height - Dim3							
Dim3	Length	10.924	Expression	$(.2 * \text{Height})$							
Dim4	Length	44.076	Expression	Base - Dim1							
Height	Length	37	Expression	$(.75 * \text{Base})$							

Solve for Variations

With the parameters defined, you can now **Solve** for variations:

- Click **No Keep** in **Solve**. All the parameterized dimensions are now adjusted according to the expressions you entered in the parameter definition table, [Figure 34. Variation of Demopart01 on page 120](#). The height is now .75 of the base, the "Dim1" and "Dim3" dimensions are equal, and both are .2 of the base dimension.

Figure 34. Variation of Demopart01



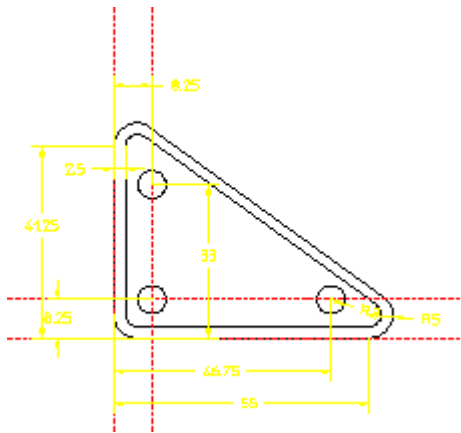
- Since all the linear dimensions are related to the base dimension, you only need to modify this dimension to make the entire part adjust. Try assigning several different values to the base dimension, solving after each change. You can change this dimension one of three ways:
 1. In **Generate Constraints**, click the **Assign** radio button, select **Dimension** from the **Type** list, click **Apply**, click the dimension label, and enter the new value.
 2. Display the **Current Constraints** dialog box, select the dimension to be changed, enter a new value and click **Apply**.
 3. In the **Current Constraints** dialog box, click **Advanced** to display the parameter table. Then click on the value to be changed in the **New Value or Expression** column and enter the new value.

Adding New Geometry

It is often useful as you work with a constrained part to be able to add or delete geometry. Many parametric design systems do not allow this, but Parametric Design handles this situation easily. In the final section of this example, we'll add a new feature to our triangular plate to illustrate how new geometry is integrated into a constrained part.

First, let's modify the triangular plate by adding a small "lip" to the outside edge. Use the Creo Elements/Direct Drafting **Equidist** command to create a contour that is 2.5 mm inside the edge of the plate. Dimension the width of this new lip as shown in [Figure 35. New Geometry on Demopart01 on page 121](#).

Figure 35. New Geometry on Demopart01



To incorporate this new feature into the Parametric Design constraint system, you only need to click the **Complete** radio button in **Current Constraints** again. Clicking this button generates the additional constraints needed for the new geometry. Subsequent **Solve** commands will recognize the new feature:

1. After adding the new geometry, click the **Complete** radio button in **Current Constraints**.
2. In **Current Constraints**, click the **Show** radio button, then select **New** from the **Act On** list and **All Types** from the **Type** list. Then click **Apply** to see how the new geometry was constrained.
3. In some cases, the addition of new geometry adds information to the part that causes existing constraints to become unnecessary. **Complete** will tell you if it finds unused constraints. If applicable, display these constraints with **Show, Unused, All Types** and **Free** them if you wish.
4. Change the value of the base dimension to 48.
5. Use **No Keep** in **Solve** to generate the variation. Note that the new lip adjusts along with the rest of the part.

To Consider

Here are a couple of additional tasks that you can try with this drawing. Both will give you some practice with manual constraint assignment.

- Through the center of each hole, add a vertical and horizontal line that extends a short distance beyond the circle (like a cross hair). Use manual constraints to force each line to move with the hole and to continue to extend the correct distance beyond the circle when the hole changes size. Hint: Apply **Point on** and point-to-circle **Distance** constraints.
- Go back to the original geometry, but first remove the both fillets on the left vertical line of the model. Go through exercise again, including the creation of

the equidistant inner contour. This time, **Complete** will not cause the inner contour to remain equidistant. Why? Can you add manual constraints to fix the problem? Hint: Use **Samedist** constraints.

Example 2: Symmetry Lines

In this example, you're given a drawing of a plate with a large slotted hole and several smaller boltholes. Your goal is to generate variations of this plate in which the slotted hole has different sizes and configurations. Each variation should be generated by simply changing the dimensions for the radius of the hole and the width and depth of the slots.

To accomplish this task, you'll assign several Symm Line constraints, which help to maintain the configuration of hole as it changes size. You'll also use parameters to control how the variation is generated.

As you work through this example, pay particular attention to the way in which symmetry lines are used by Parametric Design, and how mirror constraints are generated. You should come away from this example with a clear idea of how mirroring works.

Load the Drawing

1. In the Creo Elements/Direct Drafting File Browser, switch to the `pd_demos` directory.
2. Select the `demopart02.mi` file and click **Open**.
3. In the **Parametric** menu, click **Zone**.
4. Click **Add Multiple** in **Zone**.
5. Enter `all` and press **[Enter]**.
6. Click **End**.

-
1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Symmline**.
 2. Click **Apply**.
 3. Click on each of the four construction lines that run through the center of the part.
 4. Click **End**.

Automatically Generate the Remaining Constraints

Having manually assigned the initial constraints, use **Complete** to generate the remaining constraints needed to fully constrain the part.

1. In the **Generate Constraints** dialog box, click the **Complete** radio button.
2. Click **Apply**.
3. Click **End**.

Examine Constraints

Following a **Complete** command, you should always check the part to make sure that a reasonable set of constraints has been assigned. Check the following:

1. Are all important dimensions constrained?
Click the **Show** radio button in **Generate Constraints**, select **Used** from the **Act On** list and **Dimensions** from the **Type** list, then click **Apply**. Notice that the dimension for the top of the plate has not been constrained. This is not needed in this case because other constraints fix this line.
2. Are there any **Distance** or **Size** constraints? **Complete** usually only assigns these constraints as a "last resort" if it can find no other way to constrain an element. A distance-constrained element may move along with other elements when it not intended to. Click the **Show** radio button and select **Size** and **Distance** to display any instances of these constraints.

An alternate way to check the constraints on your part is with **Element** and **Icon** in **Inquire**. For this example, you can use **Element** to see how **Complete** used the symmetry lines you defined to constrain components of the center circle:

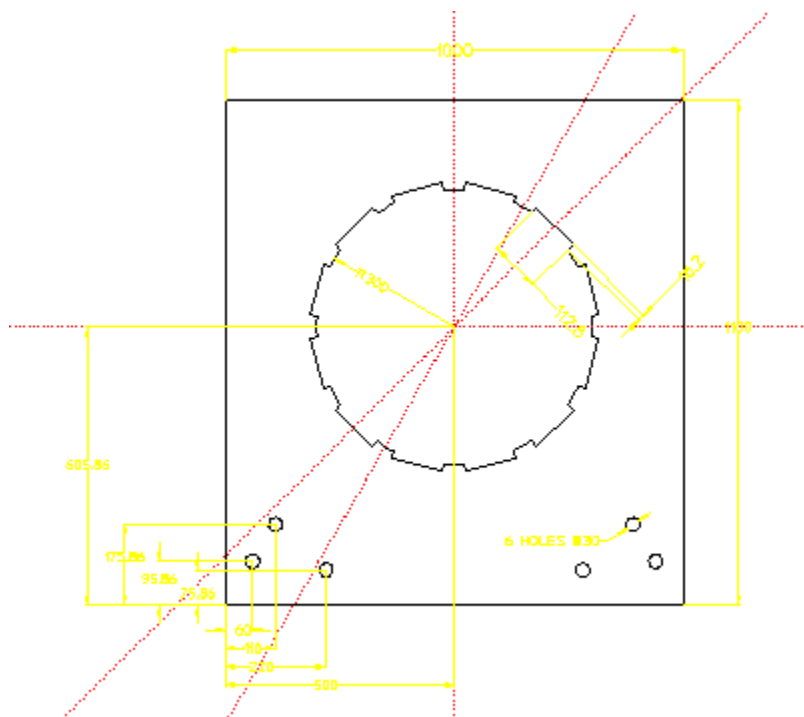
1. In **Parametric**, click **Inquire** and then **Element**.
2. Click on each line in the dimensioned slot of the part. Notice how mirror constraints were extracted for each element in the slot. Try clicking on other components around the center circle.
3. In **Generate Constraints**, press **Clear** and select **All Types**. Then click **End** when you have finished with **Inquire Element**.

Generate a Variation

The part is now fully constrained. To generate the first variation:

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Dimension**.
 2. Click **Apply**.
 3. Click the dimension for the radius of the large hole (currently 373.32).
Type in: 250 [Enter]
 4. Click the dimension for the width of the slot (currently 28.48).
Type in: 20.75 [Enter]
 5. Click the dimension for length of the slot (currently 31.41).
Type in: 58 [Enter]
 6. Click **End**.
 7. Click **Preview** in **Solve**. The magenta outline shows how the variation will be drawn without changing the part. Alternatively, you can replace the master part with the new geometry by selecting **No Keep**.
 8. Click **End** to remove the preview, or **Undo** if you solved with **No Keep**.
- Repeat this procedure several times, substituting new values of your choosing.

Figure 38. Demopart02 Variation



Parameterize

Suppose you are asked to generate several variations of this part in which the slots change size proportionally to the radius of the large hole. You could accomplish this using the methods shown above, but you would have to recalculate the dimensions of the slots each time you re-sized the hole. Your task is much easier if you parameterize the three dimensions in question. For example:

Assign Parameter Names

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Dimension**.
2. Click the dimension for the radius of the large hole.
Type in: 'HoleRad' [Enter]
3. Click the dimension for length of the slot.
Type in: 'SlotLen' [Enter]
4. Click the dimension for the width of the slot.
Type in: 'SlotWid' [Enter]
5. Click **End**.

Define Parameter Values

1. Click **Current Constraints** in **Parametric**. In **Current Constraints**, there should be three table entries; one for each parameter name you assigned.
2. Select the list entry for HoleRad.
3. Click on the parameter value entry field to the left of the **Apply** button.
Type in: 375 [Enter]
4. Click the SlotLen entry.
Type in: $(0.084 * \text{HoleRad})$ and click **Apply**.
5. Click on the SlotWid entry.
Type in: $(0.076 * \text{HoleRad})$ and click **Apply**.

Notice the "New Value Type" field for each of the three parameter definitions. While HoleRad is defined directly by a value (375), SlotWid and SlotLen are defined by evaluating expressions.

The parameters now define a variation very similar to the original part. If you wish, you can **Solve** now (use **No Keep**) to restore this configuration.

Generate Variations

Because HoleRad is defined as a "value" parameter, and both SlotWid and SlotLen can be resolved from it, you need only change the value of HoleRad to generate variations. Use the parameter value table to do this:

1. Click **Current Constraints** to bring up the parameter value table. Only parameters defined as "Value" parameters appear in this table.
2. Click on the entry for HoleRad in this table and type in:
425 and click **Apply**.
3. Click **No Keep** in **Solve** to create the variation.
4. Click **End** to accept the variation.

As you can see, the dimensions of the slots now change with the hole radius. Input several different values for HoleRad now.

Other Things to Try

- You may wish to try some other parameter definitions for this part. How would you define parameters so that the width of the slots controls the radius of the hole?
- Another interesting feature of this example is the way in which the small boltholes are constrained. If you assign a new diameter to one hole, the rest will resize when you solve. Why? Examine the constraints on these holes to find out.
- Start again at the beginning of this example. After using **Complete**, try modifying the position of the uppermost pair of boltholes (the pair whose y-position is specified by the dimension 175.86). Modify this to be 240. The x-position dimension (110) moves to be above the dimension 60. Use **Distance** constraints on the dimension texts to keep them in the same relative positions.
- Normally either the top horizontal dimension of 1000 or the bottom horizontal dimension of 500 will be extracted by the automatic constraint generator. What if you wanted to be able to control both dimensions independently? This would imply that the two vertical sides of the part need not remain vertical. How would you constrain the part to allow this to happen?

This is a very simple example of a very common situation. The default part contains obvious constraints that are unintended in general. Even in this simple example, the number of unintended constraints that the automatic constraint generator could extract is large (side lines are parallel, each side line is perpendicular to the top and bottom lines, each side line is parallel and perpendicular to some of the lines in the internal cutout and to some of the construction lines, etc).

There are three general strategies for this situation. Try each one.

- Use **Complete** and then **Ban** those unintended constraints. Continue this process until the goal has been reached.
- Manually constrain the side elements so that **Complete** doesn't need to add any constraints.
- Use **Modify Stretch** to change the default part into one that is more general.

Which is easiest?

Example 3: Two Views

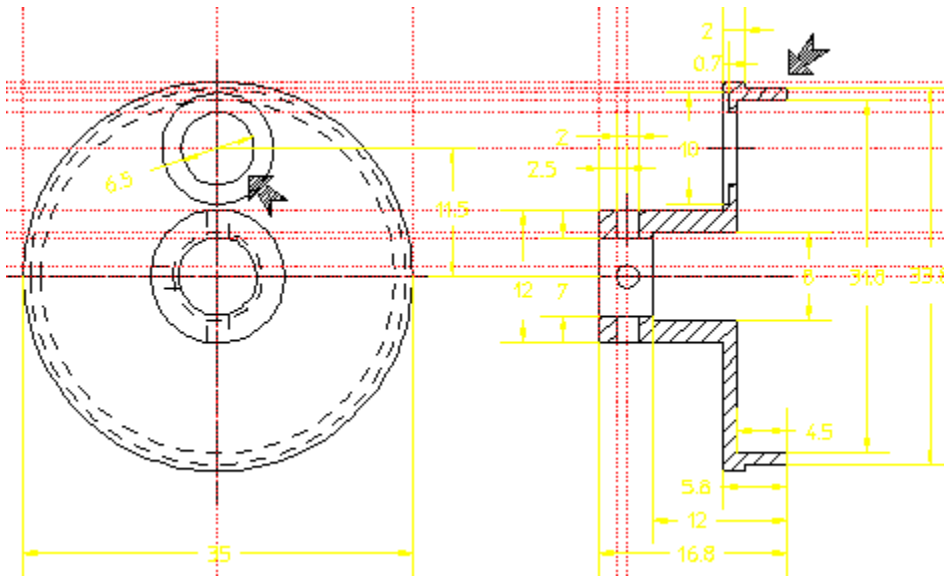
In this example, you're given a drawing that contains a front and side view of the same part. You need to change the size of several features in the part, but don't want to have to redraw both views to reflect the changes. The goal, then, is to use Parametric Design to carry the changes you make to one view over to the second view.

The key to accomplishing this task is to use construction lines to transfer constraints placed on one section of a part to another section.

Load the Drawing

1. In the Creo Elements/Direct Drafting File Browser, switch to the `pd_demos` directory.
2. Select the `demopart03.mi` file and click **Open**.
3. In the **Parametric** menu, click **Zone**.
4. Click **Add Multiple** in **Zone**.
5. Enter `all` and press **[Enter]**.
6. Click **End**.

Figure 39. Demopart03



For this example, you will resize the hole and the ring indicated by the arrows in [Figure 39. Demopart03 on page 130](#). Modifications to the large hole in the front face should carry over to the side view. Changing the width of the ring in the side view should change the radii of the hidden-line circles in the front view.

The main mechanism used to carry dimension information from one section of the part to the other is the construction line. When building this part, we included many construction lines whose only purpose was to link elements for Parametric Design. Look, for example, at the small hole in the side view. In most situations, it would be considered excessive to place horizontal and vertical construction lines on the perimeter as well as the center of this hole. For Parametric Design, however, these lines are critical: The vertical lines "link" the diameter of this hole with the two vertical holes in the side view, while the horizontal construction lines give us a link to the front view.

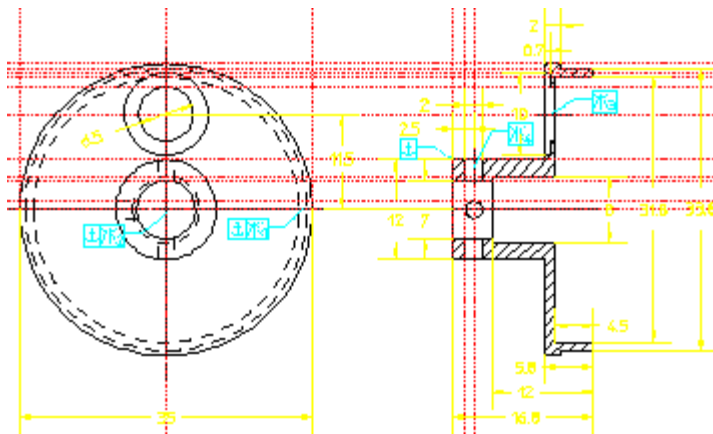
Assign Reference Elements

With a two-view part like this, it is important to anchor both views so that they don't move to overlap each other during the solve. For this part, assign **Ref Elem** constraints to the two centerlines on the front face and to the vertical construction line on the side view that's closest to the front view, [Figure 40. Manually Assigned Constraints on page 131](#):

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Refelem**.
2. Click **Apply**.

3. Click on the two centerlines and one construction line as shown in the figure below.
4. Click **End**.

Figure 40. Manually Assigned Constraints



Assign Symmetry Lines

In addition to the reference elements, the configuration of this part suggests that it would be useful to assign symmetry line constraints to the centerlines of the elements that we're going to be resizing.

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Symmline**.
2. Click **Apply**.
3. Click on each of the four centerlines that run through the holes in the two views. [Figure 40. Manually Assigned Constraints on page 131](#) shows the placement of these constraints.
4. Click **End**.

Automatically Generate the Remaining Constraints

Having manually assigned the important constraints for this example, use **Complete** to generate the remaining constraints needed to fully constrain the part.

Examine Constraints

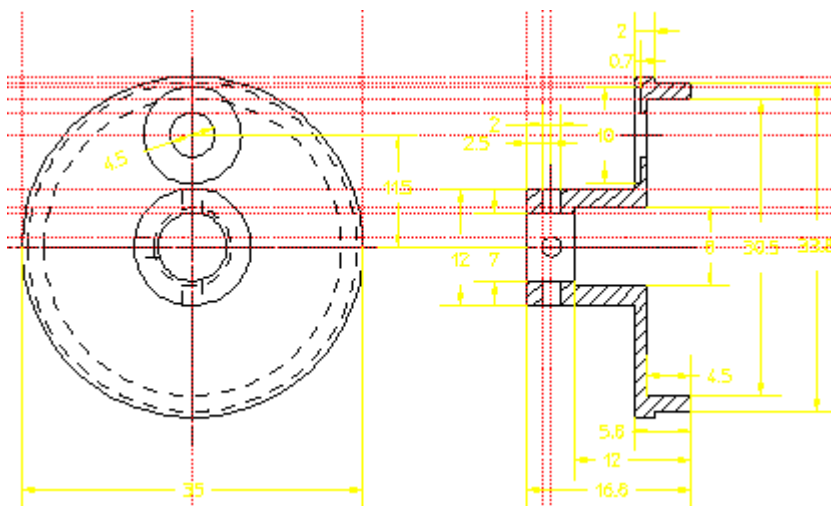
When **Complete** finishes, examine the part carefully with both **Show** and **Inquire Element** to see how **Complete** used the construction geometry to link up the elements we want to resize. **Inquire Element** is especially useful for showing how individual construction lines are associated with other geometric elements.

Generate The Variation

When you feel you understand the connections between the two views of the part, modify the constraints on the hole and the ring and **SOLVE** to see how Parametric Design makes use of these connections.

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Dimension**.
2. Click **Apply**.
3. Click the dimension for the diameter of the large hole in the front view (This dimension is currently 6.5).
Type in: 4.5 [Enter]
4. Click the dimension for the inner diameter of the ring in the side view. (This dimension is currently 31.8.)
Type in: 30.5 [Enter]
5. Click **End**.
6. Click **NoKeep** in **Solve**.

Figure 41. Demopart03 Variation



To Consider

- What other dimensions can be carried over from view to view in this part with the existing construction lines? Try resizing some other dimensions and see how the solver handles them.
- Could you delete some of the construction geometry in this part and still accomplish the example? How?
- Suppose that you want to increase the distance between the two views of this part. This is trivial to do with Parametric Design, but there is one hitch: How do you get the centerline that runs through both views to "stretch" as the views move apart? Hint: try adding a point element to each end of the center line and then, for each point, fix the distance between it and an element in the nearest view.

Example 4: Macros

Your task is to design a series of fan housings. The housings are identical except that the air intakes and exhaust ports of each housing must vary to accommodate fans of different power. It is important that the area of the air intakes equal the area of the exhaust ports to maintain proper air flow through the housing. Unfortunately, the intakes are wedge-shaped while the exhaust ports are slots. This makes the area calculations more complicated.

To accomplish this task, you'll use an Creo Elements/Direct Drafting macro that:

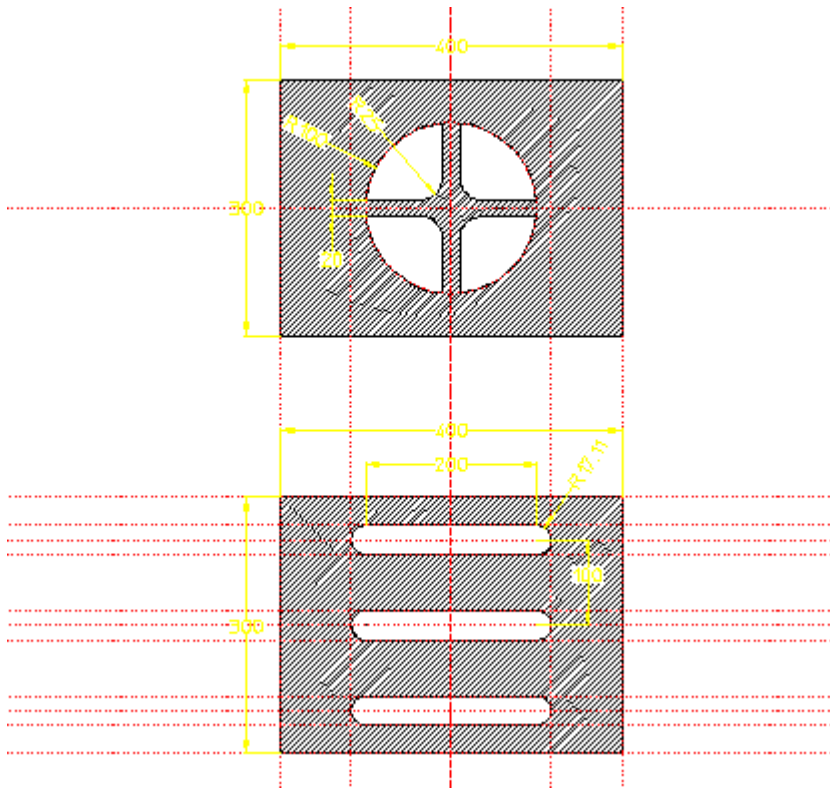
1. Calls Parametric Design to design the air intakes.
2. Calculates the area of the intakes.
3. Transforms the area value into the exhaust port size.
4. Calls Parametric Design to design the exhaust ports.

Most of the setup for this example has been done for you. The part, `demopart04.mi`, has already been constrained and parameterized, and the macros you need are already defined in the file `demopart04.mac`. As you work through this example, focus on the contents of the macro and how they relate to what you see on your screen.

Load the Drawing

1. In the Creo Elements/Direct Drafting File Browser, switch to the `pd_demos` directory.
2. Select the `demopart04.mi` file and click **Open**.

Figure 42. Demopart04



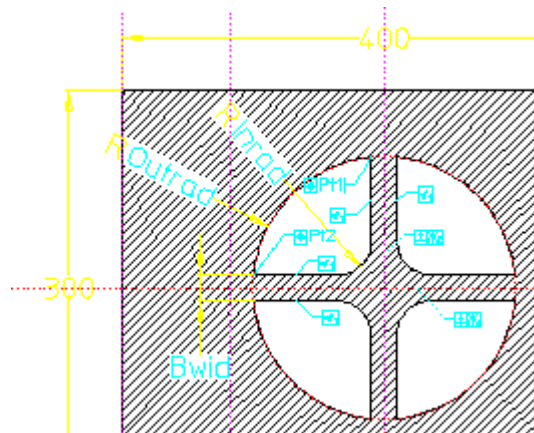
Examine the Drawing

Take a few minutes now to look at the features of this drawing.

- Notice the part structure. The front face is the top part, while the back face is a sub-part named "slot-grid." Parametric Design only works on the current part, so our macro must be able to switch from part to part.
- Both parts of the drawing are already in the zone.
- Examine the initial constraints on the top part, [Figure 43. Intake Constraints on page 135](#):
 1. Make the top part current. Type: `EDIT_PART TOP [Enter]`
 2. Click the **Show** radio button in **Generate Constraints**, select **User** from the **Act On** list and **All Types** from the **Type** list, then click **Apply**.

The **Dimension**, **Distance**, and **Symmline** constraints ensure that all four intake ports retain their configuration in variations. The **Refpoint** constraints Pt1 and Pt2 were created only to provide information to our macro. This is explained later.

Figure 43. Intake Constraints



- Examine the constraints on the "slot_grid" part now. (Hint: make "slot_grid" the current part in order to examine it). Notice that the slot radius has been parameterized as "Newrad." How did we ensure that all three exhaust ports will remain the same size?

When you have finished looking at the constraints, make sure that top part is current and that your main window shows both parts.

Examine the Parameters.

Bring up the parameter definition table to examine the parameters on the part:

1. Click **Advanced** in **Current Constraints**.

Assuming that the top part is active, there should be five parameters defined:

Bwid	Gives the width of the bars between the air intakes. Set to 20 .
Inrad	Gives the inner radius of each air intake. Set to 25 .
Outrad	Gives the outer radius of each air intake. Set to 100 .
Pt1	A reference point. Pt1 is a geometry parameter, so its value will always correspond to the top corner of an air intake, no matter how this part is resized.
Pt2	A reference point. The value of Pt2 will always correspond to the bottom corner of Pt1's air intake.

To prove that the parameters work as we say they do, try them:

- a. Change the value for "Out_rad" to 120. Click on the "Outrad" entry in the table and enter **120** from the keyboard.
- b. Click **Preview** in **Solve**.
- c. Notice how the hole configuration changes.

- d. Restore the parameters to their original values when you've finished experimenting.

Take a look at the parameters on the exhaust ports now.

- e. Make "slot_grid" the current part.
- f. Preview a few variations of this part by inserting new values for the "Newrad" parameter and solving in **Preview** mode.
- g. Set "Newrad" to **5** and solve in **No Keep** mode.
- h. Make the top part current.

The configuration of the parameters should give you an idea of the strategy we'll use to accomplish our task. We can easily vary the size and configuration of the four air intakes in the top part by changing the parameter values for "Bwid," "Inrad," and "Outrad." On the "slot_grid" part, we can vary the width of all three exhaust ports by entering a new value for the "Newrad" parameter. All that needs to be done now is to translate the area defined by "Bwid," "Inrad," and "Outrad" into a radius that can be fed to "Newrad." This is exactly what our macro, `Do_it`, does.

Try the Do_it Macro

Before looking closely at `Do_it`, let's try it. `Do_it` is defined in the file `demopart04.mac` in the `pd_demos\` directory. To load `Do_it` into your system, type:

```
input `<Creo Elements/Direct Drafting installation directory>\pd_demos\demopart04.mac`  
where
```

```
<Creo Elements/Direct Drafting installation directory>  
is the directory where you installed Creo Elements/Direct Drafting.
```

Now, just type in: `Do_it` [Enter]

What happened? If all went well, the exhaust ports returned to their original size. Now try this:

1. Display the parameter value table via **Current Constraints**. Change "Bwid" to 10, "Inrad" to 35, and "Outrad" to 75.
2. Type in: `Do_it` [Enter]

It looks like your task is now very easy! All you need to do in order to generate your fan housings is to enter the parameters for the intake ports in the parameter table and type in `Do_it` to compute the necessary width for the exhaust slots. Let's look at `Do_it` now to see how it's done.

Examining Do_it

The definition for Do_it is listed below:

```
DEFINE Do_it                                {Generate holes in demopart04.mi}
  EDIT_PART TOP                             {Setup: make sure we are at top part}
  SPOTLIGHT OFF
  PD_RESOLVE REPLACE                        {Solve for front holes}
  AREA_PROPERTY 1 Find_midp DEL_OLD '\temp\area'
  END                                        {Get the area properties of 1 front hole}
                                          {and write to a temporary file.      }
  EDIT_PART 'slot_grid'                    {Goto the second part, slot_grid}
  PD_PARAM_FIX 'Newrad' New_radius          {Find radius needed to give equivalent}
                                          {area. Fix part radius to this value.}
  PD_RESOLVE REPLACE                        {Solve for back slots}
  EDIT_PART TOP                             {Clean up: go back to top part}
  DISPLAY ('Variations Generated. Done.')
END_DEFINE                                  {of Do_it}
```

Do_it calls the macros Find_midp and New_radius, both of which are listed here:

```
DEFINE Find_midp {find a point inside a front-face hole}
  LOCAL A                                  {Declare local variable A}
  LOCAL B                                  {Declare local variable B}
  PD_PARAM_INQ 'Pt1'                       {Find Pt1 location from parameter table}
  LET A (PNT_XY (INQ 3) (INQ 4))           {assign Pt1 location to A}
  PD_PARAM_INQ 'Pt2'                       {Find Pt2 location from parameter table}
  LET B (PNT_XY (INQ 3) (INQ 4))           {assign Pt2 location to B}
  ((A+B)/2)                                {Return midpoint of A and B}
END_DEFINE                                  {of Find_midp}

DEFINE New_radius                           {Get area of front hole,}
                                          {return radius of back slot.}
  LOCAL L                                  {Declare local variable L}
  LOCAL A                                  {Declare local variable A}
  LET A 'junk'                             {Set A to some junk value!}
  LET L 200                                 {Set L to length of a back slot}
  OPEN_INFILE 1 '\temp\area' {open file of area properties written above}
  WHILE ((SUBSTR A 1 3)<>'A =') {scan area_spec for the area (A=) field}
  READ_FILE 1 A                             {and load the area field into A}
  LET A (TRIM A)
  END_WHILE
  LET A (SUBSTR A 4 (LEN A))                {set A to be just area value (area of 1 hole)}
  LET A ((VAL A)*4)                        {total area of front holes is 4A}
  (((SQRT(L*L + (PI*A)/3))-L)/PI)          {Given area of holes A and length of slots L}
                                          {return hole radius for equivalent area.  }
END_DEFINE                                  {of New_radius}
```

The algorithm used by Do_it is as follows:

-
1. Solve the variation for the intake ports.
 2. Use `AREA_PROPERTIES` to find the area properties of an intake port and write this data to a file. Call the macro `Find_midp` to generate the point which identifies this port.
 3. Change the current part to the exhaust ports.
 4. Call the macro `New_radius` to calculate the needed radius from the data in the file written above. Assign this radius to "Newrad."
 5. Solve the variation for the exhaust slots.

This procedure is straightforward, but does use some interesting features of both Creo Elements/Direct Drafting and Parametric Design commands:

- Although Parametric Design only operates on the current part, a macro like `Do_it` can easily manage interchanges among multiple parts.
- The macro `Find_midp` uses the reference-point constraints `Pt1` and `Pt2` to determine a point that is guaranteed to be inside an intake port. Since `Pt1` and `Pt2` are constrained to always be on the corners of this port, the point between them will always be inside the port. Passing this point to `AREA_PROPERTIES` makes sure that we always get the properties for an intake port. Note how `PD_PARAM_INQ` is used to grab the locations of these points from the parameter table.
- New-radius simply reads the area data from a file and uses it along with hard-coded data about the slot length to generate the appropriate radius. The output is input to Parametric Design via `PD_FIX_PARAM`. Of course a much more complex macro could be used to supply this input!

To Consider

- Currently, `Do_it` counts on the slot length being 200. How would you rewrite `Do_it` to inquire the slot length directly from the part? (Hint: You probably need to add a new parameter to the part.)
- `Do_it` is not foolproof. You can cause problems by setting "`B_wid`" to 0. How would you fix this flaw? How would you implement general error or range checking in `Do_it`?
- Could you implement `Do_it` so that slot size drives intake size? So that either dimension could drive the other?
- `Do_it` is linked very tightly to this particular drawing. How would you generalize `Do_it`?

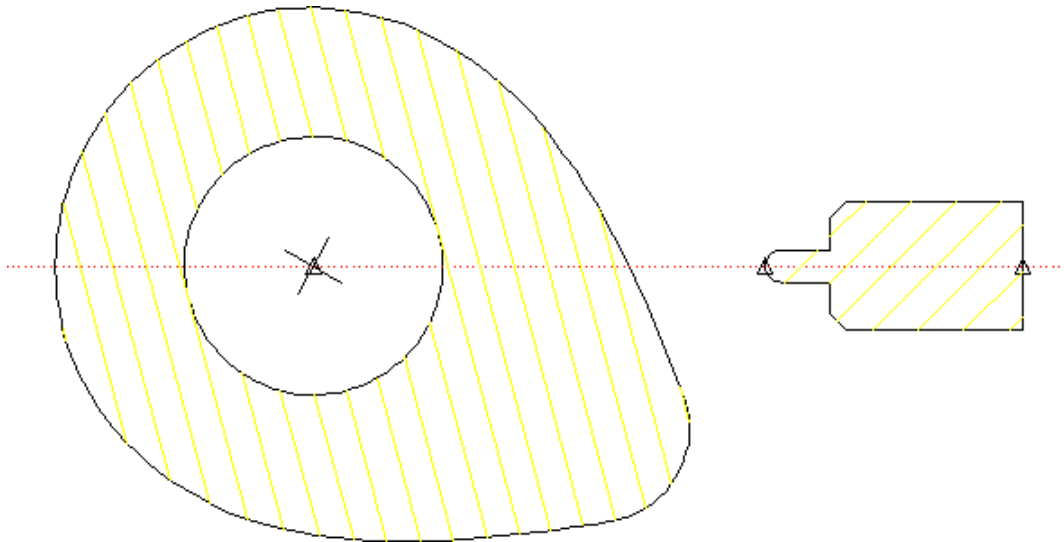
Example 5: Rigid Bodies

This example illustrates how rigid bodies can be used to reduce the number of constraints in a drawing while allowing a complex range of motion. You're given a drawing of a cam and follower. By collecting each into a rigid body, you can set up a simple motion study using just six constraints.

Load the Drawing

1. In the Creo Elements/Direct Drafting File Browser, switch to the `pd_demos` directory.
2. Select the `demopart05.mi` file and click **Open**.
3. In the **Parametric** menu, click **Zone**.
4. Click **Add Multiple** in **Zone**.
5. Enter **all** and press **[Enter]**.
6. Click **End**.

Figure 44. Demopart05



The goal of this example is to force the follower to stay in contact with the cam profile as the cam rotates. By setting up both the cam and the follower as rigid bodies, we dramatically reduce the number of constraints needed to implement this action, and can concentrate more on interaction between the two pieces.

The `demopart05.mi` model is fairly simple. The only elements in the drawing that are needed specifically for parametrics are the construction line and the three point elements, which are used to facilitate placement of several **Point on** constraints.

Create Rigid Bodies

The cam and the follower each need to be collected into a rigid body. The construction line is the only part of the model that is not in a rigid body. Each rigid body must be named before elements can be collected into it:

1. Click **Generate Rigids** in **Parametric**.
2. In **Generate Rigids**, type in 'cam' and click **Apply**.
3. Gather all of the cam geometry into the cam rigid body by boxing it.
4. Click **End**.
5. Repeat the procedure to create a rigid body called follower that contains all of the follower geometry.
6. Click **Show** in **Generate Rigids** and select each of the table entries you just created to confirm the contents of each rigid body. The construction line should not be included in either. Close **Generate Rigids** when finished.

Assign Constraints

Only six constraints are needed to completely constrain this drawing. Assign each one manually:

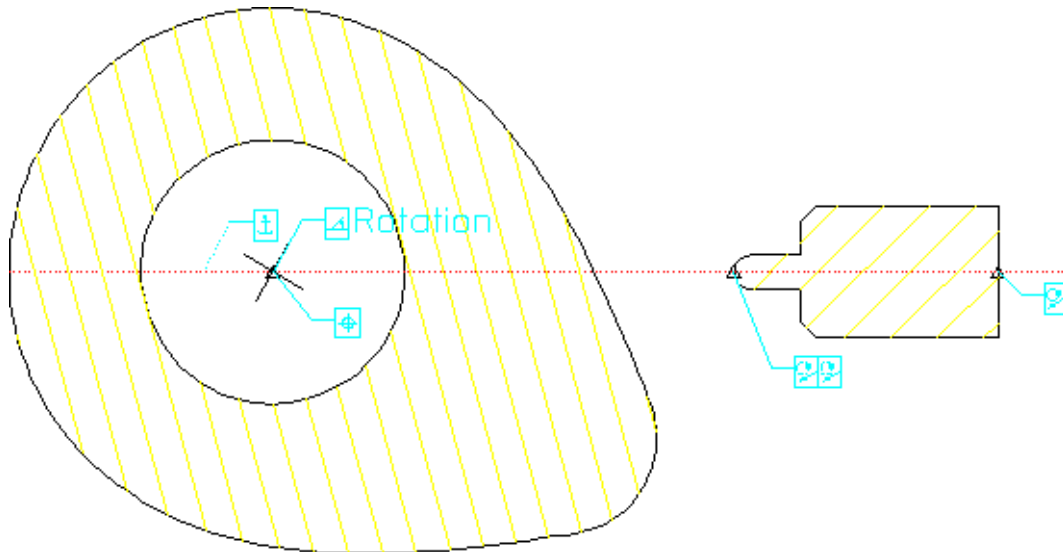
1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Refelem**.
2. Click **Apply**.
3. Click the construction line to fix it in place.
4. Use the same general method to assign a **Refpoint** constraint to the point element at the centerpoint of the cam. This fixes the cam in place on the construction line while allowing it to rotate.
5. Use the same general method to assign a **Slope** constraint to one of the small line elements that cross at the centerpoint of the cam. Name this constraint **Rotation**, but don't assign it a specific value yet.

Note that this constraint and the previous one are all that's needed to completely constrain the "cam" rigid body: The reference point fixes translation while the slope constraint fixes rotation.

6. Constrain the follower: **Assign** a **Point on** constraint between the leftmost point element on the follower and the construction line. Fix the rightmost point element to the construction line with a second **Point on** constraint. These two constraints fix the rotational degree of freedom for the follower, but its motion along the axis of the construction line still needs to be fixed.

7. **Assign** a final **Point on** constraint between the leftmost point element on the follower and B-spline profile of the cam. This fixes the translation of the cam.
8. In **Parametric**, click **Inquire** and **Icon** to confirm that you've assigned each of the constraints as described above. The constraints should resemble [Figure 45. Demopart05 Constraints on page 141](#). As you examine the constraints, make sure you understand how each one constrains the motion of the rigid body to which it's applied.

Figure 45. Demopart05 Constraints



Solve

Before running the solver on this drawing, it is important to turn off the point-merging functionality of the **Solve** command. Remember that by default the solver will merge any two points that fall within the PD_RESOLVE_MERGE_TOLERANCE distance of one another. Normally this is desirable, as it helps to keep the drawing clean. In this case however, it is possible for the endpoint of the follower profile to come into contact with the endpoint of the cam profile as the cam rotates. Were this to occur, these two points would be merged, and further parametric manipulation of this drawing would be impossible. To keep this from happening, type in: `pd_resolve_merge_tolerance -1` at the command line.

Now, click **Solve** and **No Keep** to generate the initial variation.

To confirm that the follower will indeed maintain contact with the cam as it rotates, display the **Current Constraints** dialog box and enter several different values for the Rotation parameter, solving after each one.

To Consider

- Rotating the cam by "hand" (entering a new slope and re-solving) is fairly tedious. It should be simple to write a macro to do this.
- Try this example with `pd_resolve_merge_tolerance` set to the default value (1E-6). Rotate the cam until the follower comes to rest at the very top of the cam profile. What happens with the very next rotation?
- Try to replicate this example without using rigid bodies. How many constraints are needed? How would you force the hatching on the cam to rotate along with the elements?

A

Tutorial

Exercise 1.....	144
Create the basic design	144
Assign Constraints	144
Modify the design	145
Swap the dimensions	145
Now copy the design	146
Exercise 2.....	147
Create the basic design	147
Identify the dimensions to change	148
Change the tooth dimensions	148
Modify the design	149
Hints and Tips	150

The exercises presented in this tutorial familiarize you with:

- Dimension-Driven Modification
- Design-Intent Capture

The drawings and macros that are needed for this tutorial can be found in the following subdirectory (or in the subdirectory where you installed Creo Elements/Direct Drafting):

`$MEDIR\pd_demos`

This tutorial is intended for students who are already familiar with the Creo Elements/Direct Drafting user interface for doing such things as creating geometry, windowing, and dimensioning. Instructions for the mouse input device are given first.

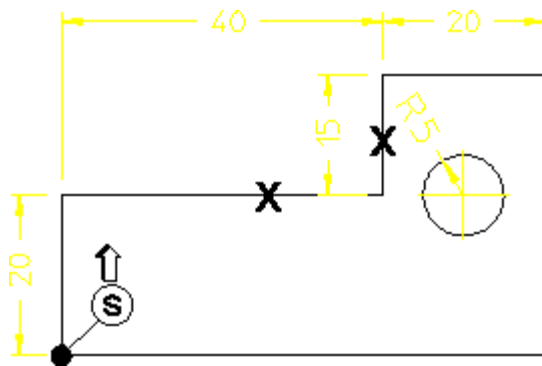
Exercise 1

In this exercise, you draw a plate and then modify its shape by changing the driving dimensions.

Make sure the **Copilot** and **Parametric Design** functionalities are enabled in the **System** tabs (accessible via **Setup**).

Create the basic design ...

1. Click **Geometry** ▶ **Line** ▶ **Polygon** and draw the shape shown in [Figure 46 on page](#) , starting at point S in the direction shown.
2. Click **Geometry** ▶ **Circle** ▶ **Center & Point or Radius** and draw the hole (the offset is from points X at values 10 0).
3. Click **End** to complete the command.
4. Dimension the drawing as indicated in [Figure 46 on page](#) .



Assign Constraints ...

1. In the **Generate Constraints** dialog box, click the **Assign** radio button. Then pull down the **Type** selection list and select **Refpoint**.
2. Click **Apply**.
3. Click point S (see [Figure 46 on page](#)) to mark it as the reference point.
4. Assign the **Dimension** constraint to the dimensions shown in [Figure 46 on page](#) .

Modify the design ...

To modify the design, click **Parametric** ► **Modify** ► **Dimension** ► **Show** and enclose the design in a box. Creo Elements/Direct Drafting displays the design's dimension in green to indicate that these are the driving dimensions. It is the driving dimensions that are used to modify the design according to the design-capture rules. These dimensions reflect the way in which you created the design.

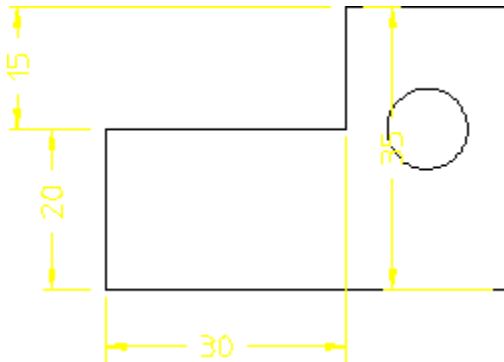
Note

Remember to use the windowing functions for fitting, panning, and zooming!

To modify the design:

1. Click **Parametric** ► **Modify** ► **Dimension** ► **Deferred** .
2. Click the dimension **40** (it changes to yellow).
3. Enter **30** on the command line.
4. Click **End**.

The design changes according to the modified dimension.



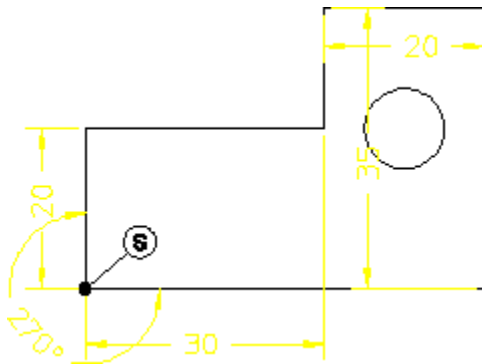
Swap the dimensions ...

1. Click **Parametric** ► **Modify** ► **Dimension** ► **Immediate**.
2. Click the dimension **20** on the left of the drawing.
3. Enter **25** on the command line.

The shape changes. But because this is not the design you want, you need to redimension it.

4. Click **Undo**.
5. Click **Redraw** in **View**.
6. Click **Dimension**, **Angular**, **Direct**.

7. Click the intersecting lines at **S**.
8. Click **Redraw in View**.
9. Click **Parametric, Modify, Dimension, Show**, and enclose the entire design in a box.
10. Click **Parametric, Modify, Dimension, Swap**, and click the yellow angle.
11. Click the dimension **15** (this dimension disappears because the angle becomes the driving dimension).
12. Click **Redraw in View**.

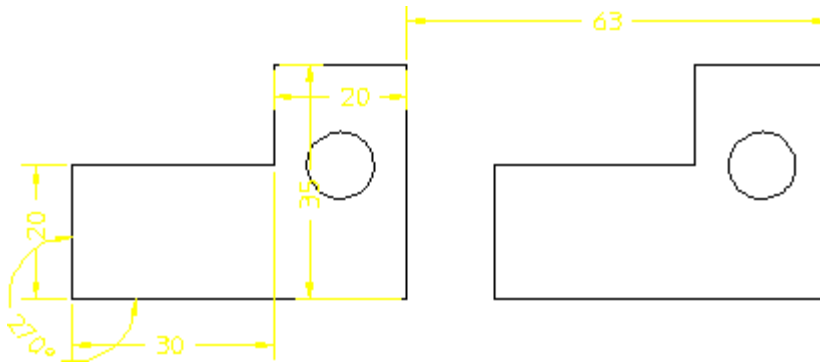


Now copy the design ...

The next step is to repeat the above but this time copy the design to show that the copy does not contain driving dimensions. The design changes and the position of the hole remains relative to the defining offset.

1. Click **Parametric ► Modify ► Dimension ► Deferred**.
2. Click the dimension **20** on the left of the drawing and enter **25**.
3. Click **End**.
4. In the **Modify** menu, tick the **Keep** checkbox.
5. In **Modify**, click **Move**.
6. Enclose the design in a box and position the copied drawing.
7. Click **Parametric ► Modify ► Dimension ► Show** and enclose both drawings in a box.
8. Click **Parametric ► Modify ► Dimension ► Deferred**.
9. Click the dimension **25** and enter **20** on the command line.
10. Click **End**.

Both the original and the copy change shape — Creo Elements/Direct Drafting remembers that you want both designs to have the same shape.



To clear the display, click **Delete** ► **All** ► **Confirm**.

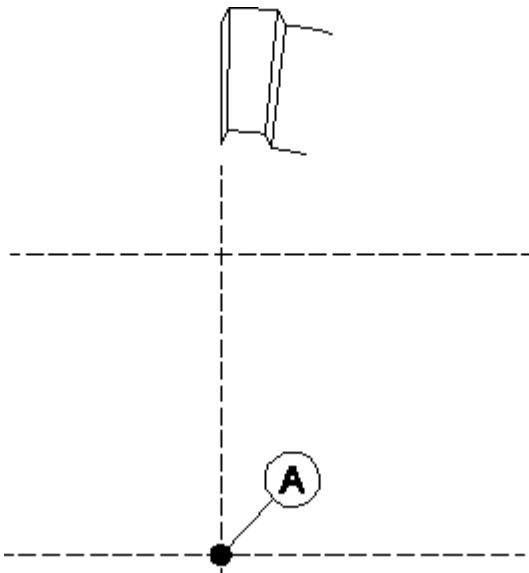
Now go on to Exercise 2.

Exercise 2

In this exercise, you copy and modify a cogwheel to demonstrate how to create families of parts.

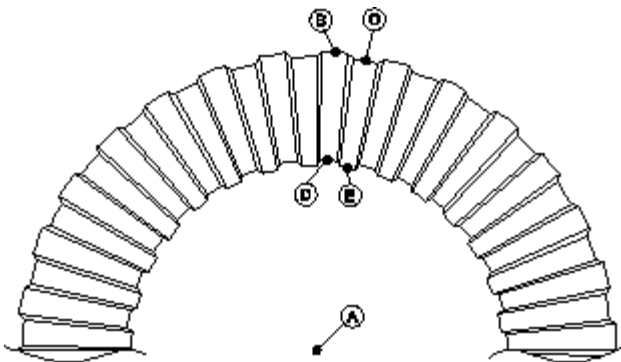
Create the basic design ...

1. Use the Creo Elements/Direct Drafting File Browser to load the file `cogwheel` from the `pd_demos` directory.
Creo Elements/Direct Drafting displays a single tooth together with some construction geometry.
2. In the **Modify** menu, tick the **Keep** and the **Repeat** checkboxes.
3. In **Modify**, click **Rotate, Center**.
4. Enter the repeat factor **29**.
5. Click point **A** to mark the center of rotation.
6. Enter **12** to define the rotation angle.
7. Enclose the single tooth in a box.
8. Click **Fit**.



Identify the dimensions to change ...

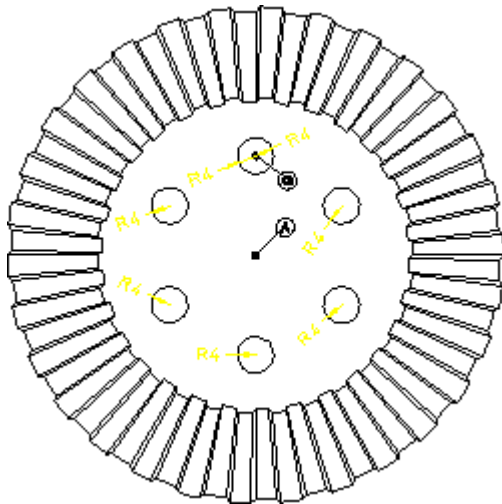
1. Click **Parametric** ▶ **Modify** ▶ **Dimension** ▶ **Show**.
2. Click point **B** on arc of tooth.
3. Repeat for points C, D, and E.
4. Click **Fit**.
5. Click **Parametric** ▶ **Modify** ▶ **Dimension** ▶ **Deferred**.
6. Click the dimensions in turn and move them to unclutter the drawing.



Change the tooth dimensions ...

1. Click **Parametric** ▶ **Modify** ▶ **Dimension** ▶ **Deferred** and modify the radii:
 - a. Click R40 and enter **55**
 - b. Click R39 and enter **54**

- c. Click R31 and enter **35**
- d. Click R30 and enter **34**
2. Click **End**. Creo Elements/Direct Drafting updates the cogwheel accordingly.
3. Click **Fit**.
4. Click **Geometry** ▶ **Circle** ▶ **Center & Point or Radius**.
5. Draw a circle with radius 4 mm at point G.
6. In the **Modify** menu, tick the **Keep** and the **Repeat** checkboxes.
7. In **Modify**, click **Rotate** ▶ **Center**.
8. Enter the repeat factor **5**.
9. Click point **A** to mark the center of rotation.
10. Enter **60** to define the rotation angle.
11. Enclose the 4 mm circle in a box.
12. Click **Dimension** ▶ **Circular** ▶ **Radius** ▶ **Centerline** and dimension the radii of the six circles.
13. Click **Parametric** ▶ **Modify** ▶ **Dimension** ▶ **Show** and click the original circle (not one of the copies).



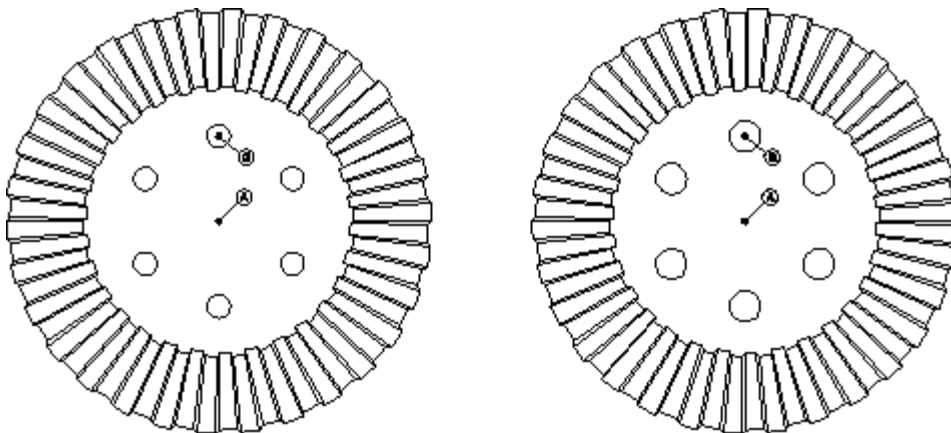
Modify the design ...

The final step is to modify the design by changing parametric values.

1. Click **Parametric** ▶ **Modify** ▶ **Dimension** ▶ **Immediate**.
2. Click R4 (green dimension) and enter **3**. Creo Elements/Direct Drafting changes the radii of all six circles — the yellow (annotation) dimensions are also updated.

3. Click **Delete** and delete the six yellow dimensions.
4. In **Generate Constraints**, click the **Show** radio button, select **Size** from the **Type** list, and click **Apply**.
The parametrics module has used the actions performed during the creation of the design to generate constraints for dynamic modification of the design.
5. In **Generate Constraints**, click the **Assign** radio button, select **Size** from the **Type** list, and click **Apply**.
6. Click the original circle at **G**.
7. Enter 'Borehole_radius' on the command line.
8. Click **End**.
9. Click **Advanced** in **Current Constraints**.
10. Click the value **3** in the column New Value or Expression.
11. Enter **4** and click **Preview** in **Solve**.
12. Click **Keep** in **Solve** and click the design.
13. Move the copy of the design to another location (you may need to **Pan**).
14. Click the destination point for the copy.
15. Click **End**.
16. Click **Fit**.

Note the smooth transition from simple design creation using the green driving dimensions to parametric design.



Hints and Tips

You have had a look now at the latest technology available in the Mechanical Engineering CAD environment today.

Here are some hints and tips when creating parametric drawings:

-
- To make sure your design intent drawing is always stored right away, switch Parametric Design and Copilot on every time you create geometry. This avoids having to give any constraints afterwards.
 - Note, that when you modify geometry using **Solve** in Parametrics, you can decide whether you want to keep the original geometry and let the modified be a copy or if you want to modify the original geometry without a copy. This is done by selecting either the **Keep** or the **No Keep** option.
 - When you specify a parameter, you can either specify a figure or enter an expression that is evaluated when the design is solved. This is useful, for example, when maintaining relations between width and height of designs.
 - In case you want to do specific modification to more complex designs, consider forming rigid bodies of geometry you want to move and translate as a whole. This can facilitate the modification of large designs.
 - How you create the design is important. For example, try creating [Figure 46 on page](#) using horizontal and vertical geometry instead of using the **Polygon** command. You would get very different results.

B

Parametric Design Command Syntax

PD_AUTO_ANGLE_TOLERANCE function.....	155
PD_AUTO_SAME_DISTANCE_TOLERANCE function.....	155
PD_AUTO_SYMMETRY_COLOR function	155
PD_AUTO_SYMMETRY_LINE function.....	155
PD_AUTO_SYMMETRY_LINETYPE function	156
PD_AUTO_TANGENT_TOLERANCE function	156
PD_AUTO_ZERO_DISTANCE_TOLERANCE function	156
PD_CLEAN_DRAWING command	156
PD_DEFAULT_DIM_COLOR command.....	157
PD_DEFAULT_DIM_TEXT_SIZE command.....	157
PD_FIX command	157
PD_FREE command	161
PD_HIDE_DIMENSIONS command	163
PD_INFO_CONSTRAINT function.....	163
PD_INFO_ELEMENT function.....	163
PD_MAKE_DIMENSIONS command.....	163
PD_MODIFY_DIMENSION command	164
PD_NEW_C_LINE_COLOR function	164
PD_NEW_C_LINE_LINETYPE function.....	164
PD_NEW_C_LINE_VISIBILITY function	165
PD_NEW_POINT_COLOR function.....	165
PD_NEW_POINT_LINETYPE function	165
PD_NEW_POINT_VISIBILITY function.....	165
PD_PARAM_ADD command.....	165
PD_PARAM_FIX command	166
PD_PARAM_INQ function.....	166
PD_PARAM_REMOVE command	167
PD_PARAM_SAVE function	167
PD_PARAM_SHOW function	167
PD_PREVIEW_COLOR function	167
PD_RESOLVE command.....	168

PD_RESOLVE_MERGE_TOLERANCE function	168
PD_RIGID_ADD command	169
PD_RIGID_REMOVE command	169
PD_RIGID_REFRESH_TABLE function	169
PD_RIGID_SHOW function	169
PD_SHOW function	170
PD_SHOW_CLEAR function	171
PD_SHOW_COLOR function	172
PD_SHOW_LABEL_SIZE function	172
PD_SHOW_MOVE_TEXT function	172
PD_SHOW_USE_POSTFIX function	172
PD_USE_DIMENSION command	173
PD_ZONE_ADD command	173
PD_ZONE_REMOVE command	173
PD_ZONE_SHOW function	174
parameter name	174

This appendix contains the command syntax for the Parametric Design module. Parametric Design is implemented in the Creo Elements/Direct Drafting macro programming language, and the descriptions here follow Creo Elements/Direct Drafting conventions. Macros are listed in alphabetical order.

PD_AUTO_ANGLE_TOLERANCE function

```
--> (PD_AUTO_ANGLE_TOLERANCE) ---- |number| --->
```

This function specifies the absolute tolerance used during PD_RESOLVE AUTOMATIC constraint extraction to compare angles, such as checking whether two lines are parallel, or a single line is horizontal. The number supplied must be greater than or equal to zero. The initial value is 0.000001 radians.

PD_AUTO_SAME_DISTANCE_TOLERANCE function

```
--> (PD_AUTO_SAME_DISTANCE_TOLERANCE) ---- |number| --->
```

This function specifies the absolute tolerance used during PD_RESOLVE AUTOMATIC constraint extraction to compare two non-zero distances, such as checking whether two circular elements have the same radius. The number supplied must be greater than or equal to zero. The initial value is 0.000001 mm.

PD_AUTO_SYMMETRY_COLOR function

```
--> (PD_AUTO_SYMMETRY_COLOR) ---+--- |color| ---+----->
                                     |           |
                                     +---- (ANY) ----+
```

This function specifies the color of lines which will be treated as symmetry lines during automatic constraint generation. If ANY is specified then color will not be used to exclude lines from being automatically constrained as symmetry lines.

PD_AUTO_SYMMETRY_COLOR is used in combination with PD_AUTO_SYMMETRY_LINE and PD_AUTO_SYMMETRY_LINETYPE. The initial setting is ANY.

PD_AUTO_SYMMETRY_LINE function

```
--> (PD_AUTO_SYMMETRY_LINE) ---+---- (NO) ---+----->
                                     |           |
                                     +--- (YES) ---+
```

This function specifies whether the automatic constraint generator should scan the drawing for symmetry lines. If so, then all lines which have the color specified by PD_AUTO_SYMMETRY_COLOR and linetype specified by PD_AUTO_SYMMETRY_LINETYPE will qualify. The initial setting is NO.

PD_AUTO_SYMMETRY_LINETYPE function

```
-->(PD_AUTO_SYMMETRY_LINETYPE) --+---|linetype|-----+---->
                                     |
                                     +----- (ANY) -----+
```

This function specifies the linetype of lines which will be treated as symmetry lines during automatic constraint generation. If ANY is specified then linetype will not be used to exclude lines from being automatically constrained as symmetry lines.

PD_AUTO_SYMMETRY_LINETYPE is used in combination with PD_AUTO_SYMMETRY_LINE and PD_AUTO_SYMMETRY_COLOR. The initial setting is ANY.

PD_AUTO_TANGENT_TOLERANCE function

```
-->(PD_AUTO_TANGENT_TOLERANCE) ----|number| --->
```

This function specifies the absolute distance tolerance used during PD_RESOLVE AUTOMATIC constraint extraction to check whether two elements are tangent. One of the elements extracted must be circular. The tolerance is used to compare the radius of the circular element to the distance from the circular element center point to the other element. The number supplied must be greater than or equal to zero. The initial setting is 0.000001 mm.

PD_AUTO_ZERO_DISTANCE_ TOLERANCE function

```
-->(PD_AUTO_ZERO_DISTANCE_TOLERANCE) ----|number| --->
```

This function specifies the absolute tolerance used during PD_RESOLVE AUTOMATIC constraint extraction to compare a distance to zero, such as detecting coincident points, collinear lines and point on element constraints. The number supplied must be greater than or equal to zero. The initial setting is 0.000001 mm.

PD_CLEAN_DRAWING command

```
+-----+
|
```

```

--> (PD_CLEAN_DRAWING) -v-+- (CLEAN_CLOSE_POINTS) -----+ |tol| +- (CONFIRM) -v->
      |                                     |
      +- (CLEAN_DUPLICATE_GEOMETRY) -+   |
      |                                     |
      +- (CLEAN_STACKED_GEOMETRY) -----+

```

This command is used to modify three geometrical situations which can cause inefficiency and confusion for parametric design.

`CLEAN_CLOSE_POINTS` is used to merge points that are within the user specified distance of each other. For each set of two or more points which are to be merged, a new location is chosen to move the entire set of points to. The new location will be within the tolerance zone specified. In general, there is no way to predict or control which elements will be adjusted. This should not be a limitation, since the tolerance used is usually very small.

`CLEAN_DUPLICATE_GEOMETRY` is used to delete elements that are duplicates of other elements. Most geometry types are duplicates if their model points match. Circles and construction circles are determined to be duplicates if the center points match and the radii are within the specified tolerance. In general, there is no way to control or specify which element will be deleted.

`CLEAN_STACKED_GEOMETRY` is used to split overlapping lines, arcs and circles. There is no user specified tolerance for this operation. The geometry must overlap with intersection tolerance.

Each option first scans the part for the desired modification. If there are instances of the defect then the offending geometry is highlighted and the user is prompted for a CONFIRM before actually performing the modification. If no cleanup is necessary then the prompt notes this fact while asking for a new option. As an aid to the macro writer, a CONFIRM at this point will be ignored.

PD_DEFAULT_DIM_COLOR command

```
--> (PD_DEFAULT_DIM_COLOR) --|color|-->
```

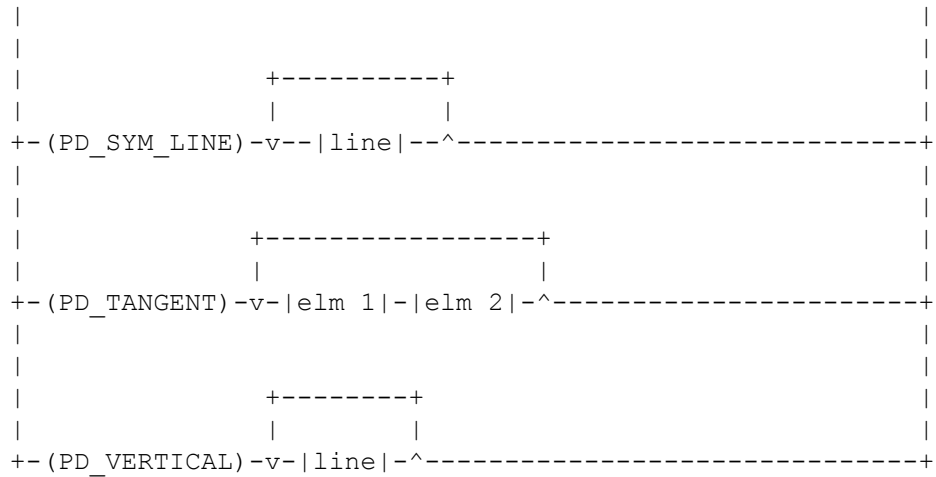
`PD_DEFAULT_DIM_COLOR` sets the display color of parametric dimensions drawn with `PD_MAKE_DIMENSIONS`.

PD_DEFAULT_DIM_TEXT_SIZE command

```
--> (PD_DEFAULT_DIM_TEXT_SIZE) --|text size|-->
```

This command is obsolete and has no effect on text size. Size and other attributes (except color) are controlled by the current dimension settings.

PD_FIX command



PD_FIX is used to define new constraints. These constraints will be used in future calls to PD_RESOLVE if the modify zone contains any of the elements involved. The meaning of each constraint type is described below in terms of how it effects PD_RESOLVE.

PD_ANGLE_LINES specifies the relative angle between two linear elements. If a value or name is not specified then the current angle is preserved.

PD_COLLINEAR will force two linear elements to be collinear.

PD_DIMENSION will cause the geometry associated with the dimension to take on the given value. If a value or name is not given, then the current value of the dimension will be preserved. If a parameter name is defined then either the value of the parameter will be used to define the geometry or the geometry will be used to define the value of the parameter.

PD_DISTANCE specifies the distance between any two point, linear or circular elements. If a value or name is not specified then the current distance is preserved.

PD_ELEM will maintain the position and size of the element.

PD_FILLET will force two real geometry elements to be filleted by a given circular arc. Each end of the arc must share an end point with one of the elements.

PD_HORIZONTAL will force a linear element to be horizontal.

PD_MIRROR will force the extension of two elements to be mirror images of each other about a given line.

PD_POINT will force the point to remain in its current position, or in the position indicated by the parameter. Alternatively, other constraints can position the point which will be used to define the value of the parameter. See NOTE below for additional information about this constraint type.

PD_POINT_ON will force a point to slide along the indicated element or its extension. See NOTE below for additional information about this constraint type.

This is the simplest way to retain user defined constraints while removing all constraints generated by the system.

Example:

```
PD_FREE NOBAN ALLTYPES CONFIRM
```

This is the simplest way to free all parametric information from the part.

PD_HIDE_DIMENSIONS command

```
          +-----<-----+
          v                   ^
-->(PD_HIDE_DIMENSIONS) --v---|select dimensions|--^--->
```

PD_HIDE_DIMENSIONS removes selected parametric dimensions from the display. Parametric dimensions are displayed with PD_MAKE_DIMENSIONS.

PD_INFO_CONSTRAINT function

```
          +-----+
          |         |
-->(PD_INFO_CONSTRAINT) --v---|id_pnt|--^----->
```

PD_INFO_CONSTRAINT is used to highlight the geometry associated with a constraint. The constraint is identified by clicking its icon.

PD_INFO_CONSTRAINT is particularly useful for identifying quickly constrained pairs of elements (e.g. parallel lines, mirrored elements).

PD_INFO_ELEMENT function

```
          +-----+
          |         |
-->(PD_INFO_ELEMENT) --v---|id_pnt|--^----->
```

PD_INFO_ELEMENT is used to query how PD_RESOLVE determined the position and size of the element identified by id_pnt during the last call to PD_RESOLVE in the current session. All constraints that were used in the determination will have their icons displayed and all geometry on which those constraints rely will have their geometry highlighted as well.

Using this command on an element which was not solved for in the last call will produce no icons or highlights.

If the constraints have been edited since the last call to PD_RESOLVE then the information returned could be misleading.

PD_MAKE_DIMENSIONS command

```

+-----<-----+
  v               ^
-->(PD_MAKE_DIMENSIONS) --+--|sel elements|--+-->

```

PD_MAKE_DIMENSIONS displays parametric dimensions associated with selected elements. Parametric dimensions are created when COPILOT is turned on. Parametric dimensions are removed from the display with PD_HIDE_DIMENSIONS.

PD_MODIFY_DIMENSION command

```

+-----<-----+
  v               ^
-->(PD_MODIFY_DIMENSION) --+-- (IMMEDIATE) --+--+--|sel dim|--|new value|--+
|               |
+-- (DEFER) -----+

```

PD_MODIFY_DIMENSION sets a new value for a parametric dimension. Parametric dimensions are created when COPILOT is on, and they are displayed with PD_MAKE_DIMENSIONS.

IMMEDIATE indicates that all geometry should be updated as soon as the new value is entered.

DEFER indicates that the value of the parametric dimension should be changed, but the geometry should not be updated until later. In other words, DEFER allows several parametric dimensions to be modified at one time. Performance is improved, because the model is not updated several times. In some cases DEFER is necessary because some groups of modifications do not have valid intermediate geometry.

PD_NEW_C_LINE_COLOR function

```

-->(PD_NEW_C_LINE_COLOR) --|color|-->

```

PD_NEW_C_LINE_COLOR sets the color of construction lines created by the Parametrics module. This creation can occur when geometry is created with COPILOT turned on, when automatic constraint generation takes place, or when constraints are assigned.

PD_NEW_C_LINE_LINETYPE function

```

-->(PD_NEW_C_LINE_LINETYPE) --|linetype|-->

```

PD_NEW_C_LINE_LINETYPE sets the linetype of construction lines created by the Parametrics module. This creation can occur when geometry is created with COPILOT turned on, when automatic constraint generation takes place, or when constraints are assigned.

PD_NEW_C_LINE_VISIBILITY function

```
-->(PD_NEW_C_LINE_VISIBILITY) ---+--- (VISIBLE) -----+--->
                        |                               |
                        +--- (INVISIBLE) ---+
```

PD_NEW_C_LINE_VISIBILITY sets the visibility of construction lines created by the Parametrics module. This creation can occur when geometry is created with COPILOT turned on, when automatic constraint generation takes place, or when constraints are assigned.

PD_NEW_POINT_COLOR function

```
-->(PD_NEW_POINT_COLOR) --|color|-->
```

This function specifies the color to be used for new points that are created by the system. This creation can occur in either the automatic constraint generator or when the user defines new PD_POINT and PD_POINT_ON constraints.

PD_NEW_POINT_LINETYPE function

```
-->(PD_NEW_POINT_LINETYPE) --|linetype|-->
```

This function specifies the linetype to be used for new points that are created by the system. This creation can occur in either the automatic constraint generator or when the user defines new PD_POINT and PD_POINT_ON constraints.

PD_NEW_POINT_VISIBILITY function

```
-->(PD_NEW_POINT_VISIBILITY) ---+--- (VISIBLE) -----+--->
                        |                               |
                        +--- (INVISIBLE) ---+
```

PD_NEW_POINT_VISIBILITY sets the visibility of points created by the Parametrics module. This creation can occur when geometry is created with COPILOT turned on, when automatic constraint generation takes place, or when constraints are assigned.

PD_PARAM_ADD command

```
-->(PD_PARAM_ADD) ---+--- (ANGLE_PAR) -----+---|parameter name|---->
```

```

|                                     |
+--- (LENGTH_PAR) ---+
|                                     |
+--- (POINT_PAR) -----+
|                                     |
+--- (USER_PAR) -----+

```

PD_PARAM_ADD adds a new parameter to the current part. The parameter must be one of four types: angle, length, point or user.

PD_PARAM_FIX command

```

--> (PD_PARAM_FIX) --|parameter name|---+-----|number|-----+---->
|                                     |
+--- (')--|expression|-- (')--+
|                                     |
+----- ()-----+

```

PD_PARAM_FIX defines what the value of the named parameter will be during the next call to PD_RESOLVE.

Inputting a |value| causes the new value of that parameter to be the given value. The value must be either a number or a point, depending on the parameter type.

Inputting a quoted expression will cause the expression to be evaluated during subsequent calls to PD_RESOLVE. The expression can include references to other parameters. Any valid Creo Elements/Direct Drafting expression is allowed.

Inputting a null string, , removes any previously defined value or expression for the parameter.

PD_PARAM_INQ function

```

--> (PD_PARAM_INQ) ----|parameter name|--->

```

PD_PARAM_INQ writes information about the parameter into the system inquiry array. It can then be retrieved with INQ (see INQ).

PD_PARAM_INQ set the following values:

function/command	<inq_index>	result
PD_PARAM_INQ	1	130 (inq code)
	2	0 : Named parameter does not exist
		1 : Named parameter does exist
	5	Conversion factor from parameter units to system units
	6	0 : Length parameter
		1 : Angle parameter

```

|                                     | 2 : User parameter          |
|                                     | 3 : Point parameter        |
|.....|
| if parameter type is not PD_POINT |
|.....|
|                                     | 3      | Current parameter value.   |
|.....|
| if parameter type is PD_POINT    |
|.....|
|                                     | 3      | X coordinate of point parameter. |
|                                     | 4      | Y coordinate of point parameter. |
+-----+

```

PD_PARAM_REMOVE command

```
-->(PD_PARAM_REMOVE) ----|parameter name|--->
```

PD_PARAM_REMOVE removes a parameter from the current part. The command will fail if the named parameter is used within an expression for another parameter or as the value for a constraint.

PD_PARAM_SAVE function

```
-->(PD_PARAM_SAVE) --->|output_spec|---->
```

PD_PARAM_SAVE saves the parameter settings to "output spec" (see help for OUTPUT_SPEC for details). Output consists of a PD_PARAM_FIX command for each parameter. The file can be read with INPUT to restore the parameter settings.

PD_PARAM_SHOW function

```
-->(PD_PARAM_SHOW) ---+--- (ALL) ---+----->
|                                     |
|                                     |
+--- (USER) ---+
```

PD_PARAM_SHOW displays constraints which reference parameters. If USER is specified then only those constraints which reference parameters set by value will be displayed.

PD_PREVIEW_COLOR function

```
-->(PD_PREVIEW_COLOR) ----|color|---->
```

PD_PREVIEW_COLOR is used to set the color of geometry displayed by subsequent calls to PD_RESOLVE PREVIEW.

PD_RIGID_ADD command

```
          +-----+-----+
          |               |               |
-->(PD_RIGID_ADD) --v--|rigid name|--v--|select|--^-->
```

This command is used to define rigid bodies, which is a collection of geometric entities which PD_SOLVE will translate and rotate as a group. No change of size or relative position will occur on members of a rigid body.

You can have as many rigid bodies as you desire. The rigid name can be any user string.

Members of a rigid body can be any geometric entity, including subparts and hatchings. Subparts will translate and rotate with the rigid body. Hatchings will rotate with the rigid body.

Rigid body membership is recorded as an info string on the member entity. The form of the info string is always "PD_RIGID " + |rigid name|.

PD_RIGID_REMOVE command

```
          +-----+-----+
          |               |               |
-->(PD_RIGID_REMOVE) --v--|rigid name|--v--|select|--^-->
```

This command is used to remove members from rigid bodies.

PD_RIGID_REFRESH_TABLE function

```
-->(PD_RIGID_REFRESH_TABLE) --->
```

This command is used to synchronize the contents of the logical table, rigids.ltab, with the names of all rigid bodies in the current part. This synchronization is done automatically whenever PD_RIGID_ADD, PD_RIGID_REMOVE, or PD_RIGID_SHOW is invoked, or when the current part is changed. However, since a rigid body is simply an info string, which can be edited by other means, this command is occasionally useful in its own right.

PD_RIGID_SHOW function

```
          +-----+
          |               |
-->(PD_RIGID_SHOW) --v--|rigid name|--^-->
```

This command is used to highlight the members of a rigid body.

PD_SHOW function

```

--> (PD_SHOW) --+-----+-----+ (ALLTYPES) ----->
                |         |         |
+-- (BANNED) ----+ +-- (PD_ANGLE_LINES) ----+
                |         |         |
+-- (NEW) -----+ +-- (PD_COLLINEAR) -----+
                |         |         |
+-- (SYSTEM) ----+ +-- (PD_DIMENSION) ----+
                |         |         |
+-- (UNUSED) ----+ +-- (PD_DISTANCE) ----+
                |         |         |
+-- (USED) -----+ +-- (PD_ELEM) -----+
                |         |         |
+-- (USER) -----+ +-- (PD_FILLET) -----+
                |         |         |
+-- (VIOLATED) ---+ +-- (PD_HORIZONTAL) ----+
                                |         |
                                +-- (PD_MIRROR) ----+
                                |         |
                                +-- (PD_PARALLEL) ----+
                                |         |
                                +-- (PD_PERPENDICULAR) ---+
                                |         |
                                +-- (PD_POINT) -----+
                                |         |
                                +-- (PD_POINT_ON) -----+
                                |         |
                                +-- (PD_SAME_DISTANCE) ---+
                                |         |
                                +-- (PD_SAME_SIZE) -----+
                                |         |
                                +-- (PD_SIZE) -----+
                                |         |
                                +-- (PD_SLOPE) -----+
                                |         |
                                +-- (PD_SYM_LINE) -----+
                                |         |
                                +-- (PD_TANGENT) -----+
                                |         |
                                +-- (PD_VERTICAL) -----+

```

PD_SHOW is used to add a subset of the constraint icons to the display.

The set of constraints to be added can be restricted in two ways. First, an optional constraint kind modifier can be given. Second, a constraint type or ALLTYPES must be given.


```

| |
+-- (PD_SYM_LINE) -----+
| |
+-- (PD_TANGENT) -----+
| |
+-- (PD_VERTICAL) -----+

```

PD_SHOW_CLEAR is used to remove a subset of the constraint icons which are currently being displayed.

The set of constraints to be removed follows the same syntax as PD_SHOW.

Example:

```
PD_SHOW_CLEAR ALLTYPES
```

This is the simplest way to remove all constraint icons from the display.

PD_SHOW_COLOR function

```
-->(PD_SHOW_COLOR) ----|color|---->
```

PD_SHOW_COLOR is used to set the color of constraint icons displayed by subsequent calls to PD_SHOW.

PD_SHOW_LABEL_SIZE function

```
-->(PD_SHOW_LABEL_SIZE) ---|size|--->
```

PD_SHOW_LABEL_SIZE is used to set the size of constraint icons displayed by subsequent calls to PD_SHOW.

PD_SHOW_MOVE_TEXT function

```

+-----+
| |
-->(PD_SHOW_MOVE_TEXT) --v--|id_pnt|---|move_pnt|--^-->

```

PD_SHOW_MOVE_TEXT is used to move the set of constraint icons from their current position to a new position. A set of constraint icons is all of the constraint icons attached to a particular geometry element. The new position will be used by all subsequent calls to PD_SHOW that cause constraint icons to be displayed for the element, as long as the current session is active.

PD_SHOW_USE_POSTFIX function

```
-->(PD_SHOW_USE_POSTFIX) --+---- (NO) --+---->
| |
```

+-- (YES) --+

PD_SHOW_USE_POSTFIX is used to set the show mode for dimension constraints. If YES, the constraint value is displayed in the postfix field of the dimension. If NO, the constraint value is displayed in the main value field of the dimension.

PD_USE_DIMENSION command

```
+-----+-----<-----+
|                                     |
v                                     |
-->(PD_USE_DIMENSION)--+---|parametric dimension|--|new dimension|+----->
```

PD_USE_DIMENSION exchanges a parametric dimension with a non-parametric dimension. The parametric dimensions which are created when COPILOT is on may not be the preferred ones in all cases. A dimension may be created with the normal dimension commands and then swapped via PD_USE_DIMENSION so that it becomes a parametric dimension.

In order for an exchange to occur, both dimensions must control the same geometry.

PD_ZONE_ADD command

```
+-----+
|                                     |
+-----+
|                                     |
-->(PD_ZONE_ADD) --+-----+---v---|select|--^--->
|                                     |
+---(NEIGHBORS)---+
```

PD_ZONE_ADD adds elements to the modify zone. If the NEIGHBORS qualifier is not given, then the elements added are the elements selected. If the NEIGHBORS qualifier is given then the elements added to the zone are the elements selected and any elements that share a model point with one of the elements selected.

The modify zone is the set of elements which might be modified by the PD_RESOLVE command.

PD_ZONE_REMOVE command

```
+-----+
|                                     |
-->(PD_ZONE_REMOVE) --v---|select|--^--->
```

PD_ZONE_REMOVE removes elements from the modify zone.

PD_ZONE_SHOW function

-->(PD_ZONE_SHOW)----->

PD_ZONE_SHOW highlights all elements in the current modify zone. The highlight is removed on the next screen redraw.

parameter name

```
-->|letter|---+-----+--->
      |           |
      +---|letter|---+
      |           |
      +---|digit|----+
      |           |
      +-----(_)-----+
```

This describes the syntax of a parameter name. A parameter name follows the same restrictions and guidelines as macro names. Parameter names must not conflict with existing command names, qualifiers, macros, etc.

C

Parametric Design Defaults

Setting Icon Defaults From the Screen Menu.....	176
Move Icon.....	176
Icon Size	176
Set Color	176
The PD_Defaults File.....	177
PD_AUTO_SYMMETRY_*	177
PD_AUTO_*_TOLERANCE	178
PD_RESOLVE_MERGE_TOLERANCE.....	178
PD_SHOW_*	179
ADD_CURRENT_INFO	179
PD_NEW_C_LINE_*	179
PD_NEW_POINT_*	179
PD_PREVIEW_COLOR.....	179
PD_DEFAULT_DIM_COLOR	179

User-changeable defaults exist for several entities used by Parametric Design.

All of these defaults may be set at Creo Elements/Direct Drafting startup time by including the appropriate Parametric Design functions in the file `pd_def.mac`.

The default constraint icon size and color may also be set through Parametric Design screen menu commands.

Setting Icon Defaults From the Screen Menu.

Use **Move Icon** and **Icon Setting** to set the location, size and colors of Parametric Design information labels and highlighting.

Move Icon

Move Icon allows you to change the location of constraint icons on the part. To move icons:

1. Click **Parametric** ► **Modify** ► **Move Icon** or type:
`PD_SHOW_MOVE TEXT [Enter]`
2. Click the icon to be moved.
3. Drag the cursor to the new location and click again.

Note that you can only change the position of the constraint icon itself; you can't change the position of the callout line on the referenced geometry.

Icon Size

To change the icon size:

1. Click **Parametric** ► **Settings** ► **Icon** to bring up the **Icon Setting** dialog box, or type:
`PD_SHOW_LABEL_SIZE [Enter]`
2. Type in the new size.

Subsequent **Show** or **Inquire Icon** functions will draw constraint icons in the new size. Existing icons can be cleared and re-displayed to get the new size. The Creo Elements/Direct Drafting `NEW_SCREEN` function will also re-display icons in the new size and location. The default icon size is the same as the Creo Elements/Direct Drafting default text size, 3.5 mm.

Set Color

To set the icon color:

1. Click **Parametric** ► **Settings** ► **Icon** to bring up the **Icon Setting** dialog box, or type:
`PD_SHOW_COLOR [Enter]`
2. Enter the new color.

Subsequent **Show** or **Inquire Icon** functions will draw constraint icons in the new color. Existing icons must be cleared and re-displayed to get the new color. The initial show color is CYAN.

The PD_Defaults File

Parametric Design reads the file `pd_def.mac` at startup to determine defaults. You can change these defaults by editing this file. As supplied, the `pd_def.mac` file contains the following lines:

```
PD_AUTO_SYMMETRY_LINE YES           Auto symmetry lines are enabled and
PD_AUTO_SYMMETRY_COLOR YELLOW       defined to be yellow,
PD_AUTO_SYMMETRY_LINETYPE DOT_CENTER dot-centerlines.

PD_AUTO_ZERO_DISTANCE_TOLERANCE 1E-6 Zero-distance tolerance used by AUTO
PD_AUTO_SAME_DISTANCE_TOLERANCE 1E-6 Distance tolerance used by AUTO
PD_AUTO_ANGLE_TOLERANCE 1E-6       Same-angle tolerance used by AUTO
PD_AUTO_TANGENT_TOLERANCE 1E-6     Tolerance for determining tangency in AUTO
                                     Tolerance for merging elements in AUTO
PD_RESOLVE_MERGE_TOLERANCE 1E-6    Default SHOW color for icons is CYAN
                                     Default Icon size is 3.5
PD_SHOW_COLOR CYAN                  Dimensional constraints replace dimension
PD_SHOW_LABEL_SIZE 3.5              ZONE Auto On feature is enabled
PD_SHOW_USE_POSTFIX NO              Default color for generated points
                                     Linetype for generated points
                                     Visibility of generated points
ADD_CURRENT_INFO "PD_ZONE" END     Default color for generated construction l
                                     Linetype for generated construction lines
                                     Visibility of generated construction line
PD_NEW_POINT_COLOR GREEN            Default SOLVE Preview color is MAGENTA
PD_NEW_POINT_LINETYPE DOT_CENTER   Default color of parametric dimensions is
PD_NEW_POINT_VISIBILITY INVISIBLE  Design Intent Capture enabled by default

PD_NEW_C_LINE_COLOR GREEN
PD_NEW_C_LINE_LINETYPE DOT_CENTER
PD_NEW_C_LINE_VISIBILITY INVISIBLE

PD_PREVIEW_COLOR MAGENTA

PD_DEFAULT_DIM_COLOR GREEN

Design_intent_on
```

PD_AUTO_SYMMETRY_*

The `PD_AUTO_SYMMETRY_*` options enable and disable automatic symmetry line constraining. When `PD_AUTO_SYMMETRY_LINE` is set to YES, the constraint generator and solver will automatically assign symmetry-line constraints to all lines that have the `PD_AUTO_SYMMETRY_COLOR` and `PD_AUTO_SYMMETRY_LINETYPE`.

PD_AUTO*_TOLERANCE

The PD_AUTO*_TOLERANCE options set various tolerances used by **Complete** to determine the relationships among elements in a part. These tolerances are used to compensate for minor inconsistencies in a part (e.g., fillets that are not quite tangent, slightly disconnected elements, lines slightly out of parallel, etc.).

PD_AUTO_ZERO_DISTANCE_TOLERANCE

This function specifies the absolute tolerance used by **Complete** to compare a distance to zero. This tolerance is used to detect coincident points, collinear lines and point-on-element constraints. The number supplied must be greater than or equal to zero. The initial setting is 0.000001 mm.

PD_AUTO_SAME_DISTANCE_TOLERANCE

This function specifies the absolute tolerance used by **Complete** to compare two non-zero distances. This tolerance may be used, for example, to check whether two circular elements have the same radius. The number supplied must be greater than or equal to zero. The initial value is 0.000001 mm.

PD_AUTO_ANGLE_TOLERANCE

This function specifies the absolute tolerance used by **Complete** to compare angles. This tolerance is applied, for example, when checking whether two lines are parallel or a single line is horizontal. The number supplied must be greater than or equal to zero. The initial value is 0.000001 radians.

PD_AUTO_TANGENT_TOLERANCE

This function specifies the absolute distance tolerance used by **Complete** to check whether two elements are tangent. This tolerance is applied only if one of the extracted elements is circular. The tolerance is used to compare the radius of the circular element to the distance from the circular element center point to the other element. The number supplied must be greater than or equal to zero. The initial setting is 0.000001 mm.

PD_RESOLVE_MERGE_TOLERANCE

PD_RESOLVE_MERGE_TOLERANCE specifies the absolute tolerance used by the Parametric Design solver to decide whether the modification should result in the merging of two or more points, or in the deletion of small elements. The merge process can be eliminated by setting this tolerance to a negative number. The initial setting is 0.000001 mm.

PD_SHOW_*

The PD_SHOW_COLOR and PD_SHOW_LABEL_SIZE options correspond to the color and size settings in the **Icon Setting** dialog box.

PD_SHOW_USE_POSTFIX controls the display of dimensional constraints. When set to the default value of NO, the constraint value replaces the normal dimension text. If set to YES, the constraint value is displayed as a postfix to the main dimension text. Earlier versions of the software always displayed dimensional constraints as postfixes.

ADD_CURRENT_INFO

ADD_CURRENT_INFO "PD_ZONE" END causes the "PD_ZONE" info to be included in all new geometry. As a result, all new geometry is added to the zone.

PD_NEW_C_LINE_*

The PD_NEW_C_LINE_* options determine the default color, linetype, and visibility for any new construction lines that are created when constraints are created. The default color and linetype causes these construction lines to be displayed as green lines. Visibility of these construction lines can be set so they are either always visible or visible only when constraints are displayed.

PD_NEW_POINT_*

The PD_NEW_POINT_* options determine the default color, linetype, and visibility for any new points that are created. Typically, these points are created only when needed to fix point-on or reference-point constraints. The default color and linetype causes these points to be displayed as green ×'s. Visibility of these points can be set so they are either always visible or visible only when constraints are displayed.

PD_PREVIEW_COLOR

PD_PREVIEW_COLOR sets the default color used by the **Preview** mode of **Solve**. This color cannot be set from the screen menu.

PD_DEFAULT_DIM_COLOR

PD_DEFAULT_DIM_COLOR sets the default color used when parametric dimensions are displayed for Dimension Driven Modification. This color cannot be set from the screen menu.

Index

A

- ADD CURRENT INFO function, 179
- Add Multiple command, 26
- Add Single command, 26
- All option, 36
- Angle command, 70
- Angle constraint, 46
- Assign command, 30, 71
- assigning
 - constraints, 28
 - parameters, 71
 - to the zone, 26
- automatic constraint generation, 94, 106

B

- Ban command, 32
- banning constraints, 32

C

- Clean commands, 24
- cleaning geometry, 23-24, 94
- Clear command, 31
- clearing constraints, 31
- Collinear constraint, 48
- Complete command, 32-33, 94, 106
- constraints
 - Angle, 46
 - assigning, 28
 - automatic generation, 94, 106
 - banning, 32
 - clearing, 31
 - Collinear, 48
 - Dimension, 48

- Distance, 50
- Fillet, 53
- freeing, 31
- generating with Design Intent
 - Capture, 86
- Horizontal, 54
- icons, 46
- Mirror, 55
- Parallel, 56
- parametizable, 71
- Perpendic, 58
- Point on, 59
- Refelem, 60
- Refpoint, 61
- Samedist, 61
- Samesize, 62
- showing, 31
- Size, 63
- Slope, 63
- Symmline, 64
- Tangent, 65
- tools, 28
- Vertical, 67
- X-Distance, 67
- Y-Distance, 67
- creating
 - parameter value tables, 22
 - parameters, 70
 - rigid body, 28

D

- D_CLEAN_DRAWING command, 156
- defaults, 175
- defining

- parameters, 72
- zones, 23
- deleting
 - parameters, 75
 - zone elements, 27
- design intent capture
 - benefits, 86
 - constraint creation, 87
 - disabling, 86
 - elements affected, 87
 - enabling, 86
 - inquiring on/off status, 86
 - overview, 86
- Dimension constraint, 48
- dimension driven modification
 - benefits, 80
 - defer, 82
 - displaying parametric dimensions, 80
 - immediate, 82
 - overview, 80
- dimensions
 - controlling position, 51
- Display command, 27, 36
- displaying
 - icons, 23
 - zones, 27
- Distance constraint, 50
- Duplicate command, 25

E

- editing parameter dimensions, 72
- expression, defined, 69

F

- Fillet constraint, 53
- Free command, 31, 71
- freeing constraints, 31

G

- Generate Rigids commands, 27
- geometry
 - cleaning, 94
- geometry cleaning, 23

H

- Horizontal constraints, 54

I

- Icon Color command, 176
- Icon command, 176
- Icon commands, 23
- Icon Size command, 176
- icons
 - coloring, 176
 - constraint, 46
 - manipulating, 23
 - modifying, 105
 - moving, 176
 - purpose, 30
 - removing, 31
 - resizing, 176
- Inconsistent option, 36
- infos
 - PD RIGID, 27
 - PD STATUS, 37
 - PD ZONE, 26
- Infos
 - PD ZONE, 98
- Input command, 21, 40
- input parameter value tables, 21
- Inquire command, 34
- Inquire Element command, 34
- Inquire Icon, 34
- invisible geometry, 109
 - display attributes, 109, 179

K

Keep command, 20, 36

L

Length command, 70
loading master parts, 18

M

master part
 creating, 39
 defined, 18
 evaluating, 42
 loading, 18
 preparation, 24
master parts
 and parameters, 40
Mirror constraint, 55
Modify command, 104
modifying
 parameter values, 19
modifying parameter value tables, 19
Move Icon command, 176
multiple parts, 104

N

naming rules for parameters, 71
No Keep command, 21, 36

P

Parallel constraint, 56
parameter definition table
 fields, 72
parameter name, 174
parameter value table
 application, 21, 76
 creating, 22
 illustrated, 19
 modifying, 19
 restoring, 21

 saving, 21
parameters
 assigning, 71
 changing values, 19
 creation, explicit, 70
 creation, implicit, 70
 defining, 72
 definition table, 69
 editing, 69, 72, 74-75
 expressions, 69
 naming rules, 71
 removing, 75
 showing, 19, 76
 units, 72
 value table, 19
 value type, 73
Parametric Design
 starting, 13
Parametric Design software
 defaults, 175
 overviewed, 12
parametric dimensions
 display attributes, 81
 modifying, 82
 swapping, 83
 viewing, 80
Partially Determined option, 38
PD RIGID info text, 27
PD STATUS info text, 37
pd_auto_angle_tolerance, 94
PD_AUTO_ANGLE_TOLERANCE
 function, 155, 178
pd_auto_same_distance_tolerance, 94
PD_AUTO_SAME_DISTANCE_
 TOLERANCE function, 155, 178
PD_AUTO_SYMMETRY_COLOR
 function, 155, 177
PD_AUTO_SYMMETRY_LINE
 function, 155, 177
PD_AUTO_SYMMETRY_
 LINETYPE function, 156, 177
pd_auto_tangent_tolerance, 94

PD_AUTO_TANGENT_TOLERANCE function, 156, 178
 pd_auto_zero_distance_tolerance, 94
 PD_AUTO_ZERO_DISTANCE_TOLERANCE function, 156, 178
 PD_DEFAULT_DIM_COLOR command, 81, 157, 179
 PD_DEFAULT_DIM_TEXT_SIZE command, 157
 PD_FIX command, 157
 PD_FREE command, 161
 PD_HIDE_DIMENSIONS command, 163
 PD_INFO_CONSTRAINT function, 163
 PD_INFO_ELEMENT function, 163
 PD_MAKE_DIMENSIONS command, 163
 PD_MODIFY_DIMENSION command, 164
 PD_NEW_C_LINE_COLOR function, 164, 179
 PD_NEW_C_LINE_LINETYPE function, 164, 179
 PD_NEW_C_LINE_VISIBILITY function, 109, 165, 179
 PD_NEW_POINT_COLOR function, 165, 179
 PD_NEW_POINT_LINETYPE function, 165, 179
 PD_NEW_POINT_VISIBILITY function, 109, 165, 179
 PD_PARAM_ADD command, 165
 PD_PARAM_FIX command, 166
 PD_PARAM_INQ function, 166
 PD_PARAM_REMOVE command, 167
 PD_PARAM_SAVE function, 167
 PD_PARAM_SHOW function, 167
 PD_PREVIEW_COLOR function, 31, 35, 167, 179
 PD_RESOLVE command, 168
 pd_resolve_merge_tolerance, 94
 PD_RESOLVE_MERGE_TOLERANCE function, 168, 178
 PD_RIGID_ADD command, 169
 PD_RIGID_REFRESH_TABLE function, 169
 PD_RIGID_REMOVE command, 169
 PD_RIGID_SHOW function, 169
 PD_SHOW function, 170
 PD_SHOW_CLEAR function, 171
 PD_SHOW_COLOR function, 172, 179
 PD_SHOW_LABEL_SIZE function, 172, 179
 PD_SHOW_MOVE_TEXT function, 172
 PD_SHOW_USE_POSTFIX function, 172, 179
 PD_USE_DIMENSION command, 173
 PD_ZONE info text, 26, 98
 PD_ZONE_ADD command, 173
 PD_ZONE_REMOVE command, 173
 PD_ZONE_SHOW function, 174
 Perpendic constraints, 58
 Point command, 71
 Point on constraint, 59
 Points command, 25
 Preview command, 20, 35

R

Refelem constraint, 60
 Refpoint constraint, 61
 Remove command, 27, 75
 removing
 parameters, 75
 zone elements, 27
 replication, 102
 rigid bodies, 101-102
 rigid body
 clearing, 28

-
- creating, 28
 - defined, 27
 - showing, 28
- ## S
- Samedist constraint, 61
 - Samesize constraint, 62
 - Save command, 21, 40
 - saving parameter value tables, 21
 - Show command, 19, 31, 76
 - showing
 - constraints, 31
 - parameters, 19, 76
 - Size constraint, 63
 - sketch input, 95
 - Slope constraint, 63
 - Solve command, 20, 35
 - Solved option, 37
 - solving
 - errors, 36
 - Keep mode, 20, 36
 - modes, 20, 35
 - No Keep mode, 21, 36
 - Preview mode, 20, 35
 - solver described, 108
 - tolerances, 94
 - SPLINE CONVERSION function, 100
 - splines
 - parametric control, 99, 101
 - Stacked command, 25
 - strategy
 - for constraining, 33, 35
 - for master parts, 39, 42
 - Symmline constraint, 64
 - system defaults, 175
- ## T
- Tangent constraint, 65
 - tolerances
 - defined, 94
 - pd_auto_angle_tolerance, 94
 - pd_auto_same_distance_tolerance, 94
 - pd_auto_tangent_tolerance, 94
 - pd_auto_zero_distance_tolerance, 94
 - pd_resolve_merge_tolerance, 94
- ## U
- UA_DESIGN_INTENT command, 86
 - UA_GET_DESIGN_INTENT
 - arithmetic function, 86
 - Unsolved option, 37
 - User command, 71
- ## V
- variation, generating, 18
 - Vertical constraint, 67
- ## W
- working methods
 - advanced, 94
 - for designer, 14
 - for draftsman, 13
 - summarized, 13
- ## X
- X-Distance constraint, 67
- ## Y
- Y-Distance constraint, 67
- ## Z
- zone
 - implementation, 98
 - manipulation of, 98-99
 - multiple, 99
 - Zone commands, 24

zones

adding elements to, 26

defined, 24

removing elements, 27

showing contents, 27