

MEDUSA⁴

VERSION 5.0

Parametrics

REFERENCE GUIDE



All rights reserved. No part of this documentation may be reproduced in any manner (print, photocopy or other) without the written permission of CAD Schroer GmbH.

CAD Schroer GmbH has made its best effort to ensure that the information in this document is accurate and reliable, but cannot guarantee the accuracy, timeliness, reliability or completeness of any of the information contained herein. CAD Schroer GmbH will not make any warranty nor accept legal responsibility or liability of any kind for consequences resulting from errors or omissions.

Registered Trademarks of CAD Schroer GmbH:
MEDUSA, STHENO

Trademarks of CAD Schroer GmbH:
MEDUSA₄, STHENO/PRO, MEDEA, MPDS

Third-Party Products and Trademarks:
Pro/ENGINEER, Pro/DETAIL and Pro/TOOLKIT are registered trademarks of Parametric Technology Corporation.

All other brand or product names are trademarks or registered trademarks of their respective owners.

July 2010

Copyright © CAD Schroer GmbH
Fritz-Peters-Str. 26 - 30
D - 47447 Moers

Germany

CAD Schroer GmbH
Fritz-Peters-Str. 26-30
47447 Moers

Tel. +49 2841 91 84 - 0
Fax +49 2841 91 84 - 44
e-mail: info@cad-schroer.de
www.cad-schroer.de

France

CAD Schroer France SAS
17, Rue du Docteur Lebel
94300 Vincennes

Tel. +33 1 41 94 51 40
Fax +33 1 43 77 17 68
e-mail: info@cad-schroer.fr
www.cad-schroer.fr

Italy

CAD Schroer Italia S.R.L.
Via Piave, 1/E
20030 Senago (MI)

Tel.: +39 02-38303267
Fax: +39 02-33303399
e-mail: info@cad-schroer.it
www.cad-schroer.it

Switzerland

CAD Schroer AG
Bettlistr. 35
8600 Dübendorf

Tel. +41 44 802 89 - 80
Fax +41 44 802 89 - 88
e-mail: info@cad-schroer.ch
www.cad-schroer.ch

United Kingdom

CAD Schroer UK Ltd
39 Newnham Road
Cambridge
CB3 9EY

Tel. +44 1223 460 408
Fax +44 1223 460 409
e-mail: info@cad-schroer.co.uk
www.cad-schroer.co.uk

USA

CAD Schroer US, Inc.
34 Rand Place
Pittsford, NY 14534

Tel: +1 866-SCHROER (866-724-7637)
Fax: +1 866-724-1701
e-mail: info@cad-schroer.com
www.cad-schroer.com

TABLE OF CONTENTS

Preface	7
Overview of Parametric Design	9
Preparing a Drawing for Parameterization	10
Parameterizing Object Geometry	14
Advanced Features of Parametric Design	15
Giving Commands in Parametric Design	17
Reference Points	19
Introduction	20
Prims	23
Static Baselines	24
Attachment Points	26
Dimensioning	27
Dimensioning for Parametric Design	28
Linear Dimension Types	30
Angular Dimensions	33
Radial and Diameter Dimensions	34
Dimensioning Fillets	35
Examples Showing PAR FIL Options	38
Tolerance Dimensioning	41
Geometric Constraints	47
Introduction	48
Perpendicular Points	49
Intersection Points	50

Tangent Points	52
Circular Baselines	53
Automatically Inferred Constraints	55
The Parametric Grid	59
Introduction	60
The Old Grid	62
The New Grid	63
The Potential Grid	65
Limited Grid Lines	66
Displaying Grid Lines	68
Adding Lines	71
Grid Tolerance	75
Variables and Expressions	77
Values in Dimensions	78
Replacing Dimension Text	80
Variables	81
Variable Scope Restrictions	82
Expressions	83
Operators and Functions	85
Output Dimensions	87
Parameterizing Geometry	89
Overview of Parameterization	90
PARS and PARS CAN	92
Switches and Layers	95
Parametric Switches	96
Setting Parametric Switches	98
Parametric Switch Examples	101
Layers	104
Layer Properties	105
Changing Layer Properties	107
Parametric Symbols	111
Creating Parametric Symbols	112
Loading Symbols Interactively	115

Rotating and Mirroring Parametric Symbols	118
Loading Symbols Using Instance Clumps	120
Using Instance Clumps - Gearbox Cover Example	123
Tables	129
Table Definitions	130
Constructing a Table	132
The TBL Command	135
Tables and Parametric Symbols	137
Parametric Groups	139
Introduction	140
Static Groups	143
Dynamic Groups	144
Rotating Parametric Groups.	147
The PAR PRE Command.	149
Changing Element Types	151
Parametric Design Element Defaults	152
Querying Element Defaults	153
Changing Element Types - PAR DDL	154
Changing Table Element Types - TBL DDL.	158
Mechanisms	159
Simulating Movement.	160
Running a Mechanism	162
Preparing the Definition Sheet.	163
Linear Motion	166
Rotary Motion.	169
Text Variable Commands.	170
Using Programs to Control Mechanisms	171
Plotting Motion Simulations	173
Examples	174
Appendix A Summary of Command Syntax	183
Parametric Switches	184
PAR DDL	187
PAR DIM	190

PAR FIL	191
PAR GRIS	192
PAR LOA	193
PAR SWI	194
PAR TOL	195
PARS	196
Q PAR	197
TBL	198
TBL DDL	199
VER FULL	200
Appendix B Error and Warning Messages	201
Error Messages	202
Warning Messages	208
List of Figures	209
Index	211

PREFACE

Book Conventions

The following table illustrates and explains conventions used in writing about MEDUSA applications.

Convention	Example	Explanation
Menu	Choose <i>Zoom</i> from the View menu Add button Choose the tool <i>Creates thin solid lines.</i>	Indicates a command, function or button that you can choose from a menu, dialog or tooltray.
Syntax	<code>acos 0.345</code> The <code>ciaddobj</code> command Return or Control-g	User input, commands, keywords and keys to press on a keyboard.
SyntaxBold	Enter command> plot_config	Where system output and user input are mixed, user input is in bold.
<i>SyntaxItalic</i>	<code>tar -cvf /dev/rst0 filename</code>	Supply an appropriate substitute for each variable; for the given example replace <i>filename</i> with an actual file name.
<i>Filename&path</i>	<code>medusa\med2d\m2d\src\</code>	Shows path and filenames.
UPPERCASE	MEDUSA or CADCONVERT	Names of products.
<i>italic</i>	<i>left mouse button</i> <i>Drafting User Guide</i>	Indicates the buttons to press on a mouse and names of books.
bold	A temporary group is a collection of ...	Emphasize text.

Please note: Illustrations showing menus and forms are taken from a window system. The display for other platforms can differ slightly.

Online User Documentation (HTML)

Online documentation for each book is provided in HTML format. You can view this online documentation in the MEDUSA installation directory and directly by calling it up within the MEDUSA user interface.

Installation Directory

1. Navigate to the directory where MEDUSA is installed.
<MEDUSA installation directory>/meddoc/doc/<language>/ (Unix)
<MEDUSA installation directory>\meddoc\doc\<language>\ (Windows)
where <language> is either *english*, *german* or *french*.
2. Click on the file *mainmenu.htm*.
3. Click the book title you want to view.

MEDUSA Interface

1. Click left on the entry *Help* inside the main menu.
2. Choose MEDUSA Documentation from the pulldown menu.
A browser opens showing the *mainmenu.htm* listing all available documents.

Printing Version of the Documentation (PDF)

A PDF (Portable Document Format) file is included for each online book. You must have Acrobat Reader installed to view and print PDF files. If you don't have the Acrobat Reader, you can download it for free from the Adobe homepage:

<http://www.adobe.com/products/acrobat/readstep.html>

In order to search for a keyword across PDF files you can use **Acrobat Reader**. Therefore **Acrobat Reader Version 6.0** or higher has to be installed. The Reader provides a multiple search function, i.e. you can specify complete directories containing several PDF files for searching.

OVERVIEW OF PARAMETRIC DESIGN

This chapter describes the general principles of the MEDUSA Parametric Design system.

- [Preparing a Drawing for Parameterization..... 10](#)
- [Parameterizing Object Geometry..... 14](#)
- [Advanced Features of Parametric Design 15](#)
- [Giving Commands in Parametric Design 17](#)

Preparing a Drawing for Parameterization

To parameterize geometry using the MEDUSA Parametric Design system you must prepare the geometry in a special way. In addition to the geometry that is to be parameterized the sheet must contain the following:

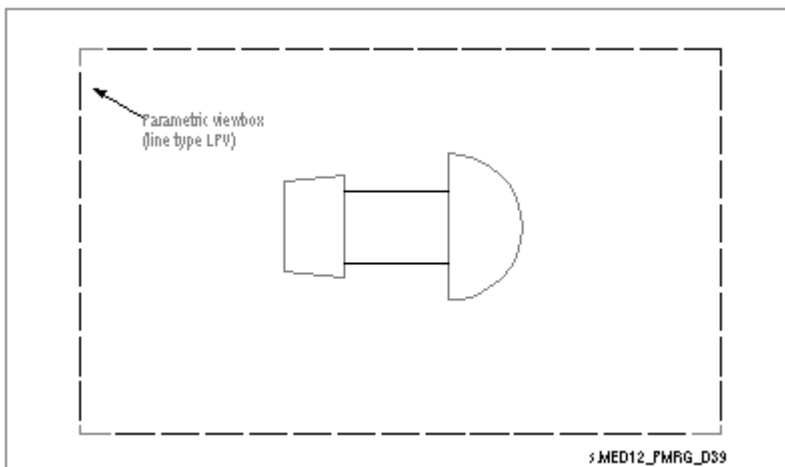
- A parametric viewbox
- A datum or reference point
- Dimensioning

The system uses a parametric grid to move points in the object geometry. You place some or all of the points in an object onto intersections on the grid. You then supply new parameters for the object geometry and the system uses these parameters to calculate the new position of each grid intersection when you give the PARS command. During parameterization, each point in the object geometry is positioned at the new grid intersection.

Parametric Viewboxes

Any geometry that you want to parameterize must be inside a Parametric viewbox. A viewbox can be any shape, but must consist of straight line segments of line type LPV. The following figure shows a drawing of a rivet, enclosed in a rectangular parametric viewbox.

Figure 1 Parametric Viewbox Containing Definition of a Rivet



Reference Points

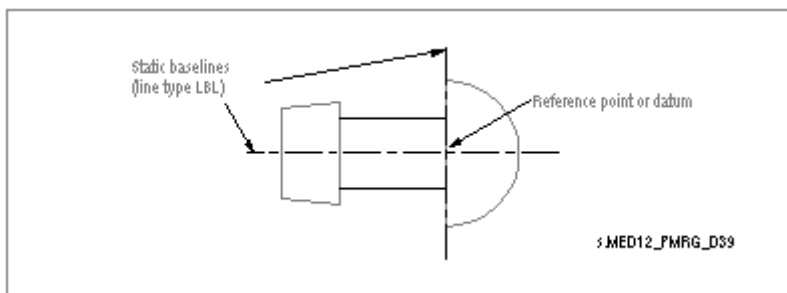
Inside every parametric viewbox you must place a reference point or datum. The reference point:

- Provides a static point of reference for all moving points during parameterization
- Can be anywhere inside the viewbox: it does not have to be a physical point on the object geometry.

Points of object geometry move during parameterization in relation to the position within the viewbox of the reference point. The reference point does not move during parameterization.

Creating a reference point: Use either a special prim or a pair of intersecting static baselines, line type LBL to create a reference point. In the following figure, a pair of static baselines define a reference point for the rivet drawing. In this example, the reference point does not coincide with a point on the object geometry but is at a center of symmetry.

Figure 2 Rivet With Reference Point



Please note: A grid line intersection is generated at the point on the sheet where you create a reference point. This is the starting point for the parametric grid.

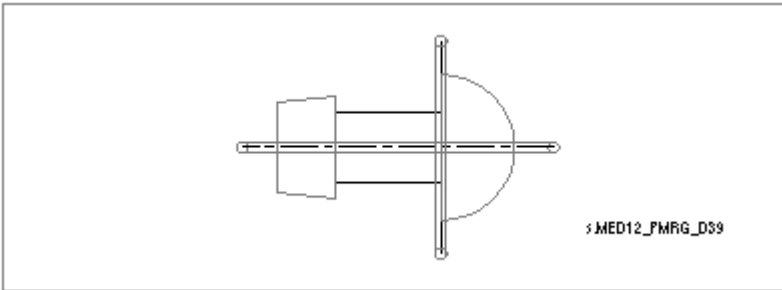
For more information about using reference points see [“Reference Points” on page 19](#).

Parametric Grid Lines

Parametric Design works not by moving individual points but by moving a grid upon which all movable points are placed. Grid lines are line type STK, and can be oriented at any angle and spacing. To display parametric grid lines, use the PAR GRIS command. [“The Parametric Grid” on page 59](#), describes the commands available for looking at different types of parametric grid lines.

The following figure shows the result of a PAR GRIS command. The static baselines generate two grid lines.

Figure 3 Rivet With Reference Point and Grid Lines

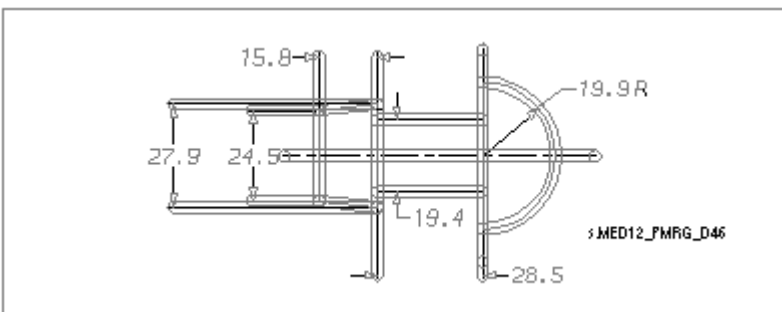


Reference points and the grid: A grid line intersection is created at the point where you place the reference point. This is the starting point for the rest of the parametric grid. To build the grid outwards from the reference point, you must dimension the object points.

Dimensioning the Object Points

Dimensions are necessary to build up the grid. To place a point on the grid you must dimension it using one of the standard MEDUSA dimensioning commands. Each dimension requires one or more existing grid lines to support it, so you usually begin dimensioning from the reference point.

Figure 4 Rivet With Dimensioning and Grid



Each time you add a new dimension clump, you can use the PAR GRIS command to check that the point you have dimensioned has been placed onto a grid intersection successfully. When you have dimensioned the object adequately, all moveable points will be at grid intersections, as in the above figure. You can now define a new set of object parameters. [“Dimensioning” on page 27](#), explains some useful dimensioning techniques.

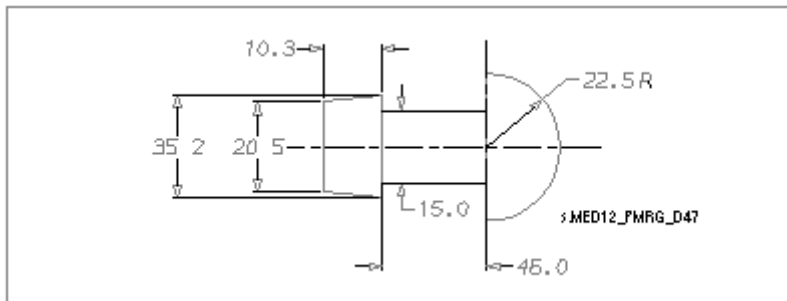
Defining your Own Parameters

To define new object parameters, you edit the dimension clump and replace the original dimension value with one of the following:

- Another numeric value
- A variable or an expression

The following figure shows the rivet with new parameters, before parameterization.

Figure 5 Original Rivet Definition with New Parameters



Parameterizing Object Geometry

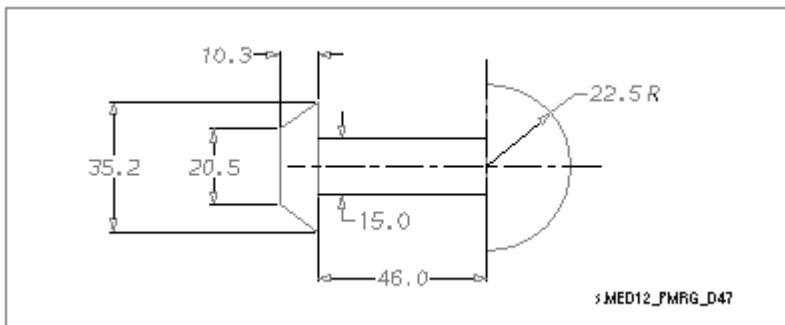
When you have checked that the object points are at grid intersections and you have defined some new object parameters, the geometry is ready for parameterization.

Parameterization Commands

You have to give just one command to parameterize geometry. This is the PARS command. When you give this command, the system tries to parameterize all geometry which is inside a parametric viewbox. The new parameters you have defined are used to calculate new positions for each grid intersections. The object points are then moved to the new grid intersection positions.

The following figure shows the rivet after parameterization with new dimension values.

Figure 6 Rivet After Parameterization



Parameterizing a Drawing Using Parametric Design

To create parameterized versions of a drawing using Parametric Design, follow this procedure:

1. Draw a parametric viewbox around the object you want to parameterize.
2. Choose a datum or reference point that will remain static.
3. Decide if all or part of the object is to be parameterized.
4. Dimension the object to put all the relevant points onto the parametric grid.
5. Check that all movable points of the object lie at grid intersections.
6. Exclude some parts from parameterization, if desired.
7. Replace dimension values with your own parameters.
8. Parameterize the object using the PARS command.
9. Check the result.

Advanced Features of Parametric Design

The following features enable you to perform more complex parameterization operations.

Parametric symbols

Parametric symbols enable you to parameterize complex assemblies using several simple part definitions. For example, the parameterized drawing of a crankcase may refer to a number of standard fittings such as bolts and washers. You can define each fitting as a parametric symbol which includes the part geometry and a reference to the values needed to provide the correct size of fitting. The whole range of symbols is then derived simply by supplying different parameters when loading the symbol onto a sheet. See [“Parametric Symbols” on page 111](#), for information on how to create and use parametric symbols.

Tables

Tables allow you to easily set several parameters of an object at the same time. After drawing and dimensioning a single part pattern and supplying variable dimension values, you can then place a table on the sheet with all the parameter values of that range of parts. During parameterization, the system reads the appropriate values from the correct line in the table and uses these values to parameterize the standard part definition.

The use of tables for parameterization is described in [“Tables” on page 129](#).

Parametric Groups

Only geometry that is inside a parametric viewbox will be transformed during parameterization. Geometry that is outside the viewbox will not be affected by any Parametric Design commands. You can change the way that parts of geometry within a viewbox are transformed with parametric groups. Using parametric groups you can:

- Completely exclude parts of the drawing from parameterization
- Parameterize parts of a drawing without having to dimension them fully

Geometry within a parametric group is treated as a single entity. Parametric groups can remain static or can be moved, magnified and rotated as a whole during parameterization. This allows you to handle details such as shaft ends and bolt holes simply.

See [“Parametric Groups” on page 139](#), for more information.

Mechanisms

You can use Parametric Design to study the movement of mechanisms. By repeatedly parameterizing a dimensioned drawing of a mechanism you can simulate its movement as it is redrawn on the screen at regular intervals. Whenever the mechanism is redrawn in a new position, its

previous positions remain visible, enabling you to investigate potential clashes between different parts of the mechanism.

See [“Mechanisms” on page 159](#), for more information on using Parametric Design to simulate the movement of mechanisms.

Giving Commands in Parametric Design

There are two ways of giving MEDUSA commands:

- By typing the command at the keyboard (interactively)
- By placing in-sheet command text, text type TCO, on the sheet

You can give any of the Parametric Design commands described in this manual (except the VER FULL command) both interactively and as in-sheet commands. When you enter a command interactively, the command is executed immediately. In-sheet commands are executed when you give the PARS command.

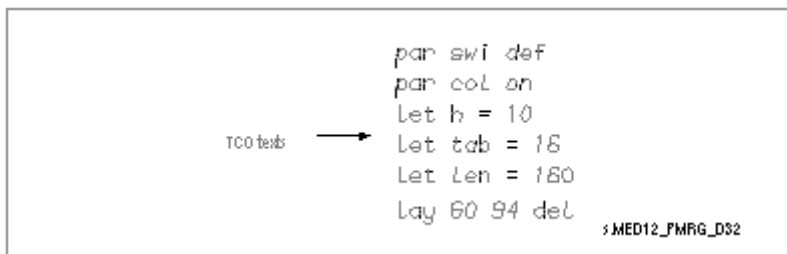
Why Use In-sheet Commands?

The advantage of using in-sheet commands is that they are stored with the sheet, so that, for example, special switch or layer settings or variable values are always used with a particular drawing. In-sheet commands only affect the viewbox in which you place them, so other viewboxes and other sheets are not affected by the commands in any one viewbox. After parameterization, parameters set with in-sheet commands are restored to the values held before parameterization. When you use Parametric Design, use in-sheet commands rather than interactive commands where the effects are global rather than local.

Execution Order

In-sheet commands are executed in sequence from the top left to the bottom right. If the sequence is critical, it is best to put them all in one vertical column. For example:

Figure 7 In-sheet Command Texts



Maximum Number of In-sheet Commands

The maximum number of in-sheet commands per MEDUSA sheet is 200.

REFERENCE POINTS

This chapter describes how to specify a reference point or datum. This is the point from which all movement occurs during parameterization.

- [Introduction](#) 20
- [Prims](#)..... 23
- [Static Baselines](#)..... 24
- [Attachment Points](#)..... 26

Introduction

The reference point or datum of a drawing provides a static point of reference for any object points that move during parameterization. Object points can move only in relation to the position on the sheet of the reference point. The reference point does not move during parameterization.

Creating a Reference Point

Use a special prim, two static baselines or an attachment point to define a reference point. Your choice of reference point may vary according to the nature of the drawing:

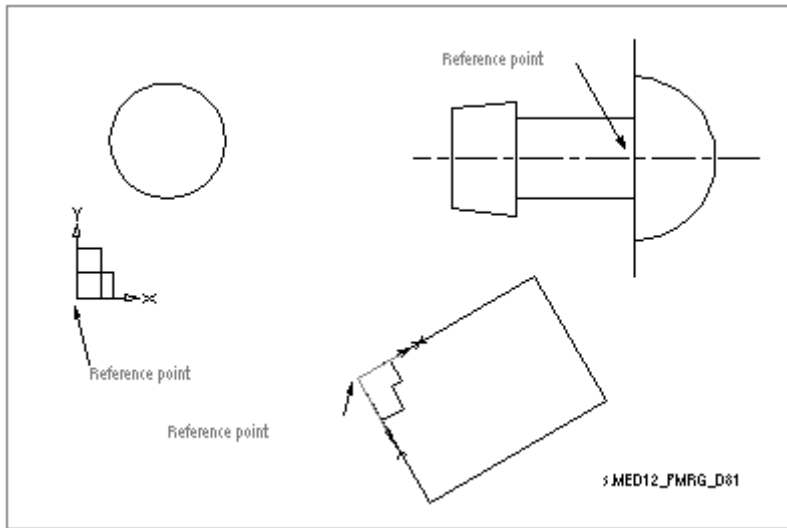
- In a regular parametric definition, the reference point can be defined using:
 - A view prim or a PVG prim
 - A pair of intersecting static baselines
- In a parametric symbol definition, you define the reference point using one or more attachment points (text type ATP)

A grid line intersection is created at the point in the viewbox where you place the reference point. This is the starting point or seed point for the parametric grid.

Positioning Reference Points

You can place a reference point anywhere inside the viewbox. The reference point does not have to be at a physical point on the object geometry. For example, it could be at a point of symmetry in the object definition, as in the rivet definition shown below. The following figure shows a number of ways reference points can be positioned in relation to object geometry.

Figure 8 Positioning Reference Points



Creating a Parametric Viewbox

Any geometry that you want to parameterize must be inside a Parametric viewbox. A parametric viewbox is a closed line of type LPV, consisting of straight line segments. Only elements wholly within a viewbox will be affected by Parametric Design commands.

Features

Viewboxes have the following features:

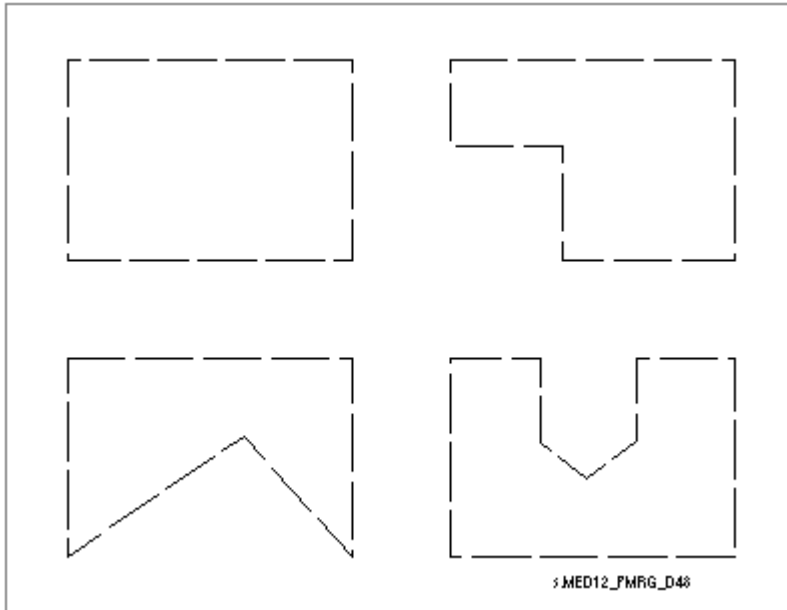
- They may be any shape
- They must not overlap or be nested

Viewboxes are processed in turn independently of one another, in the order that you place them on the sheet. The maximum number of viewboxes allowed per sheet is 20.

Examples

The following figure shows some parametric viewboxes.

Figure 9 Parametric Viewboxes



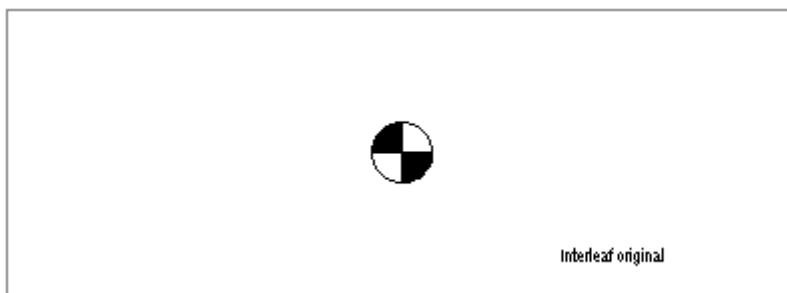
Prims

You can use either a view prim or a special parametric datum prim, type PVG, to define a reference point in your object definition. A grid intersection is created generated at the datum of the prim when you place it inside a parametric viewbox. This point is the static reference point for your object geometry.

Parametric datum prim

The following figure shows the parametric datum prim, type PVG.

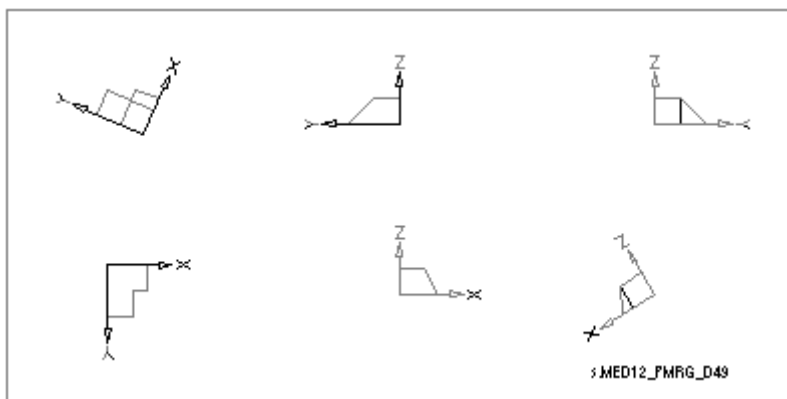
Figure 10 A PVG Prim



Orthogonal View Prims

The following figure shows the six orthogonal view prims you can use as reference points. They are type DXY, DYZ, DZX, DYX, DZY, and DXZ.

Figure 11 Orthogonal View Prims



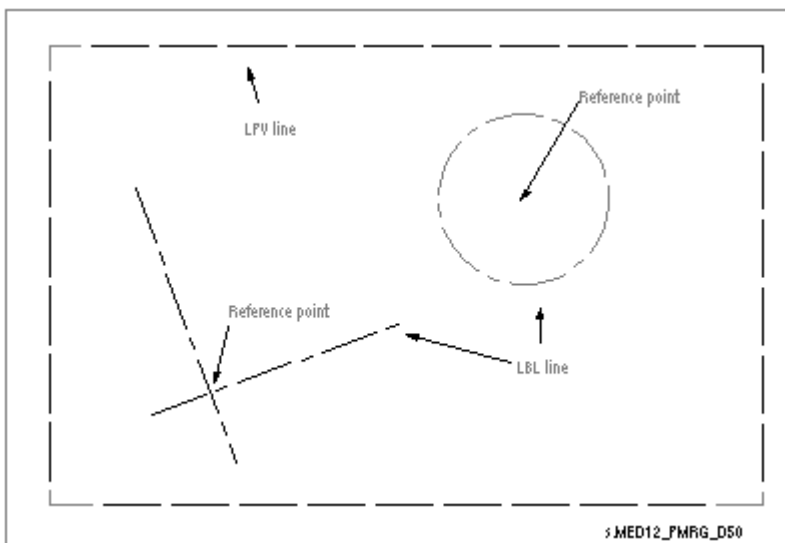
Static Baselines

In addition to the prim described on the previous page you can also define a reference point using one or more static baselines. A grid intersection is created in the viewbox at the point where the two straight static baselines intersect or at the center point of a circular static baseline. This point is the static reference point for your object geometry.

Constructing Static Baselines

A static baseline is a single LBL line segment. Each end point of the static baseline has a FUNV0 point function. A static baseline can be a straight line or a full circle created with either the CIR or CEN line functions. The following figure shows a viewbox containing static baselines.

Figure 12 Static Baselines



Using static baselines

Straight Static Baselines

A convenient and clear way to define a reference point is with two orthogonal straight baselines. The two baselines can be used as axes of the object with a fixed reference point defined by their intersection. Each baseline generates three grid lines:

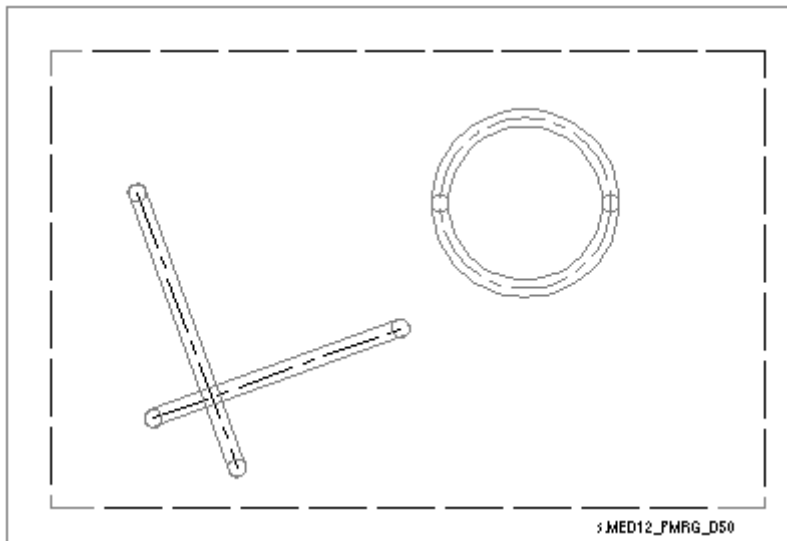
- One along the line
- One of zero length at each end of the line, perpendicular to the line

Circular Static Baselines

A circular static baseline generates a circular grid line, with a grid intersection at the circle center point.

The following figure shows the grid lines generated by the baselines in [Figure 12, “Static Baselines” on page 24](#). You can build the grid rest of the parametric grid starting from the grid intersection created at the reference point.

Figure 13 Grid Lines Generated by Static Baselines



Attachment Points

When creating a parametric symbol definition you can not use a prim or static baselines to define the reference point. Instead you must use one or more attachment points.

An attachment point is a text of type ATP. Attachment points define a set of X and Y-sheet coordinates that are used to:

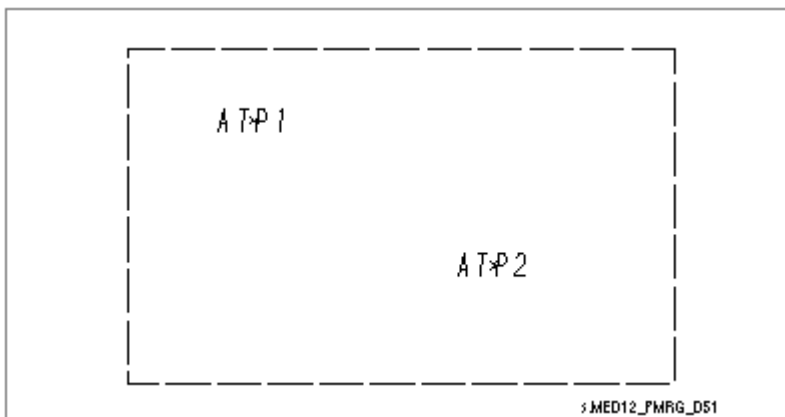
- Generate the parametric grid for the symbol definition before you unload it
- Position the parametric symbol when it is loaded onto a sheet

Two intersecting grid lines are generated through the datum of the text, one horizontal and one vertical.

Example

The following figure shows a viewbox with two attachment points, ATP1 and ATP2.

Figure 14 Attachment Points



More Information

“[Parametric Symbols](#)” on page 111 describes in detail how to use attachment points.

DIMENSIONING

This chapter describes how to dimension geometry for Parametric Design.

- Dimensioning for Parametric Design 28
- Linear Dimension Types 30
- Angular Dimensions 33
- Radial and Diameter Dimensions 34
- Dimensioning Fillets 35
- Examples Showing PAR FIL Options 38
- Tolerance Dimensioning 41

Dimensioning for Parametric Design

You place object points onto parametric grid intersections by dimensioning them. Dimensions specify the relationship between the different grid intersections. When you change dimension values, you define new positions for the grid intersections.

Points that are placed onto grid intersection are supported by the grid. By changing dimension values and supplying new object parameters, you define a new position for some or all of the grid intersections. During parameterization the system moves the grid intersections and corresponding object points to the new positions using the new parameters.

Creating Dimension Clumps

Dimensions are DIM clumps within a DMS super clump. DMS clumps must be left as sheet level elements when used as input for Parametric Design. The dimensions will not be transformed properly if they are within SET clumps. Note that the maximum number of DIM clumps allowed in a chain dimension is 50.

For more information on dimensioning refer to the entry for the DIM command in the *MEDUSA Basis1 Design Commands Guide*.

Dimension Types

You can parameterize geometry dimensioned using any of the following dimension types:

- Chain (*CHA*)
- Coordinate (*COO*)
- Datum (*DAT*)
- Angular (*ANG*)
- Radial (*RAD*)
- Diameter (*HOL*)
- Architectural (*AEC*)

Illegal Dimensioning

You can not parameterize geometry dimensioned with any of the following DIM command options:

Option	Description
ARC	Arc length dimensions
AXO	Axonometric orientation
LIM	Limit tolerancing

If you try to parameterize geometry dimensioned using any of these options you will receive the following error message:

```
Illegal dimension type
```

Dual Dimensions

Dimensions created with the DUAL option of the DIM command may be used as input to Parametric Design.

Dimension Tolerances

The VAR, ABS, UNS, and LIM options of the DIM command select how tolerances will be displayed on the sheet. You can parameterize any object with VAR, ABS, or UNS tolerances. The system will not accept LIM tolerances however, and will produce an error message if you try to parameterize geometry with LIM tolerances. You can easily convert LIM tolerances into VAR format before parameterization using the PAR DIM command, which is described in [“PAR DIM command syntax” on page 42](#).

Linear Dimension Types

Use any of the following linear dimension types as input to Parametric Design:

- CHA
- DAT
- COO
- AEC

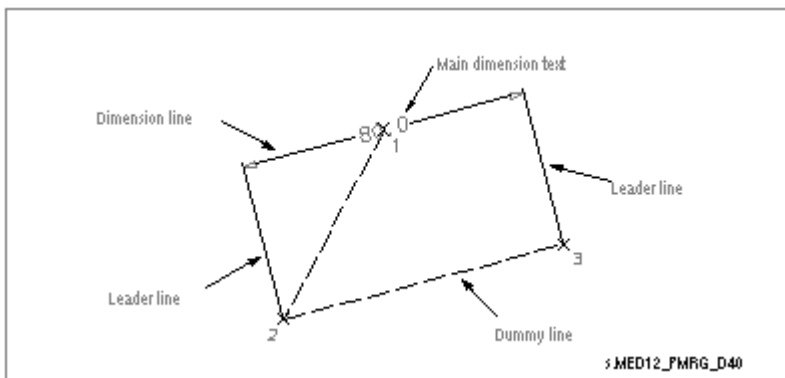
Creating Dimension Clumps

It is very important to use NEA probes for the dimension construction line. Except for the first point in the construction line (the dimension text point), all points in the construction line must be at grid intersections.

Significant Points

The following figure shows the significant points of a linear dimension, including the construction line used to create the dimension.

Figure 15 Significant Points of a Linear Dimension Type



Orientation of Linear Dimensions

Use the HOR, VER, PAR, and PER options of the DIM command to control the relative orientation of the components of the dimension clump:

- Use the HOR and VER options only if the geometry being dimensioned is to remain horizontal or vertical during parameterization
- Use the PAR and PER options if the geometry may need to change orientation during parameterization

If you use the HOR and VER options you will constrain the way the object points can move during parameterization. This is a particularly important consideration when creating parametric symbols, which are often loaded at different orientations. Parametric symbols are described in detail in [“Parametric Symbols” on page 111](#).

Chain Dimensions and Center Support

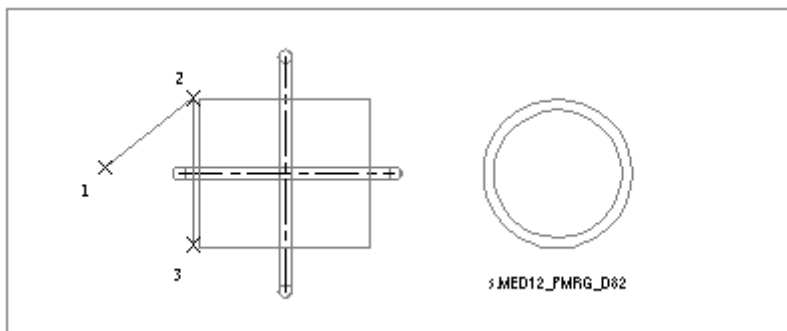
If you include the reference point (datum) in a dimension construction line, you can use any of the types of linear dimension listed in [“Linear Dimension Types” on page 30](#). If the reference point is at a center of symmetry rather than a point on the object geometry, then using a chain dimension you may be able to place some of the object points onto the grid using center support.

Center support allows you to dimension a point and place it onto the grid without including the reference point in the dimension construction line. Center support is possible when the system can find an existing grid line that passes through the center of the dimension, perpendicular to the dimension line.

Example

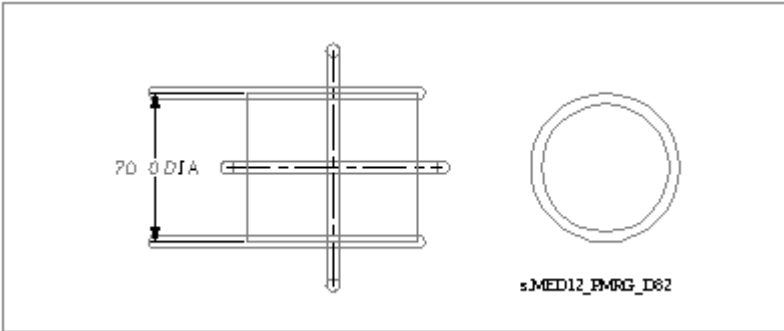
The following figure shows two views of a cylinder. The reference point is not a point of the geometry but is at the center of symmetry. The dashed line shows the three point dimension construction line that is used to create the chain dimension shown in [Figure 17, “Chain Dimensioning With Center Support” on page 32](#).

Figure 16 Two Views of a Cylindrical Object



There is no construction line point at the baseline intersection, but the system finds the horizontal baseline passing through the center of the diameter dimension and therefore places the dimensioned points on the grid.

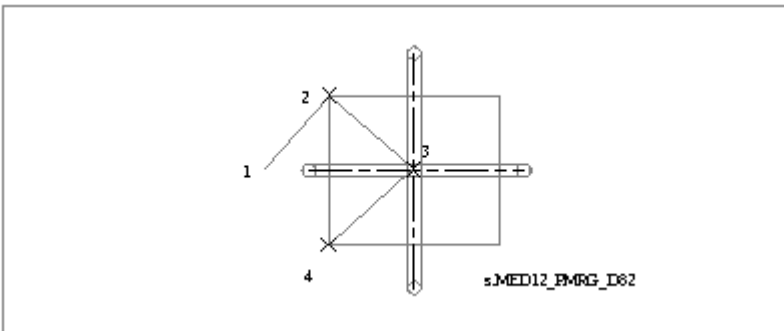
Figure 17 Chain Dimensioning With Center Support



Use *CHA* type dimensions when you need center support. Where you do not need center support, use *DAT* type dimensions and either:

- Move the reference point until it coincides with a point of geometry, or
- Create a new dimension clump that includes the reference point in the dimension construction line, as shown in the following figure.

Figure 18 Using The Reference Point to Create a DAT Dimension



Angular Dimensions

Angular dimensions are created using the ANG option of the DIM command.

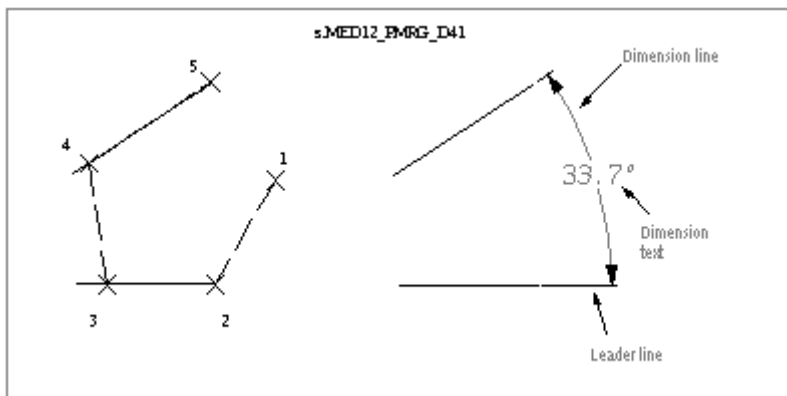
When creating angular dimensions, be careful to use NEA probes to create the second and last points in the dimension construction line. Ultimately, the second, last and center points of the dimension construction line must lie at grid intersections, though some of the grid lines used to define these intersections will be generated by the dimension itself.

The center point of the dimension is the point about which the angle is measured. The center point must lie within the viewbox, whether or not any line in the object passes through it.

Significant Points

The following figure shows the significant points of an angular dimension constructed using a five point construction line.

Figure 19 Creating an Angular Dimension



Radial and Diameter Dimensions

Radial and diameter dimension are created with the RAD and HOL options of the DIM command. The second point of the construction line used to create radial and diameter dimensions must be a point on the arc or circle circumference. This point does not have to be at a grid intersection.

Dimensioning Arcs with the Same Center and Radius

All arcs with the same center and radius must be dimensioned individually unless they overlap. You can avoid having to create separate dimension clumps for each arc by doing one of the following:

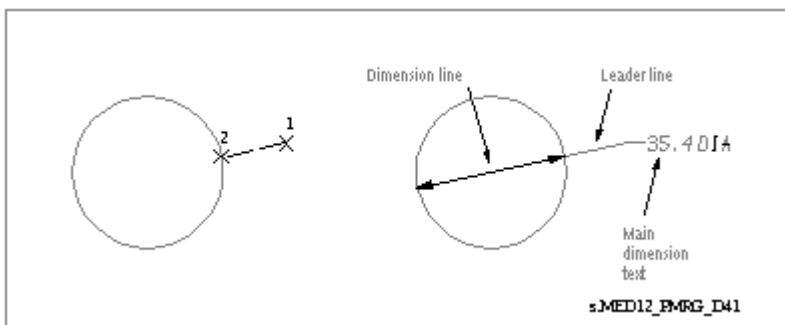
- Use a diameter instead of a radial dimension
- Add lines linking all the arcs together. See ["The Parametric Grid"](#), ["PAR GRIS CIR"](#) on [page 73](#) for details of how to do this using the Parametric Design command PAR GRIS CIR.
- Use a switch such as PAR CIR ON or PAR LIM OFF to extend all circular grid lines to full circles. These switches are explained in ["Switches and Layers"](#) on [page 95](#).

Only arcs of circles can be dimensioned. To parameterize a conic arc you must explicitly dimension the arc end points and the tangent point.

Significant Points

You can create radial and diameter dimensions using two or three point construction lines. The following figure shows the significant points in a diameter dimension created with a two point construction line. The significant points are the same for radial dimensions.

Figure 20 Significant Points of Radial and Diameter Dimensions



Dimensioning Fillets

It is not necessary to individually dimension every fillet in the object geometry. The PAR FIL command specifies a default radius for all undimensioned fillets in the viewbox during parameterization. This is particularly useful if the geometry contains several fillets of the same radius.

Creating the Correct Fillet Type

It is important to use fillets of the appropriate type when you create the object geometry:

Tangent point: Use a tangent point arc (FILT) between two straight segments. The grid lines along the two straight segments will intersect at the tangent point, which must be at a grid intersection.

Center point: Use a center point arc (FILC) if one or both of the line segments are circular. The circular grid line automatically gives a grid intersection at the circle center point.

How Fillets are Defined for PAR FIL

PAR FIL only affects arcs of circles that are supported by two existing tangential grid lines. Arcs supported in other ways are not considered to be fillets, and you must dimension them explicitly with radial dimensions as described in [“Dimensioning Arcs with the Same Center and Radius” on page 34](#).

You can specify a maximum original radius (`maxrad`) to restrict further what is considered to be a fillet.

Restrictions on the Scope of PAR FIL

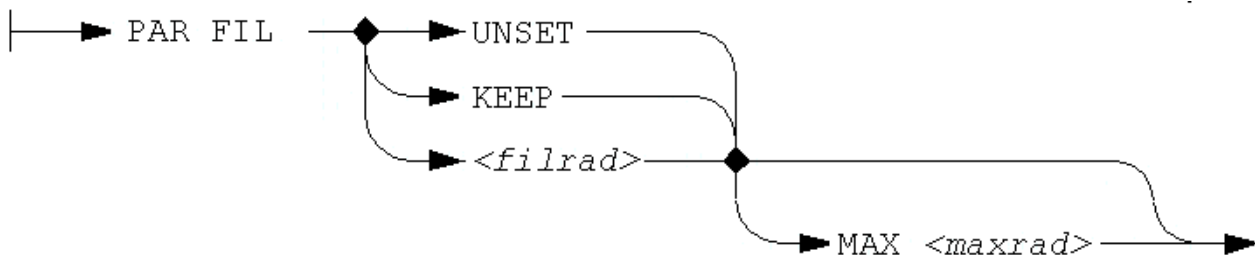
PAR BAS: If a fillet is fully supported by an automatically inferred baseline, as described in [“Geometric Constraints” on page 47](#), then the inferred radius will be used in preference to the value you specify with the PAR FIL command.

PAR LIM: The PAR FIL command has no effect if the PAR LIM switch is OFF.

In-sheet and Interactive Commands

If you give the PAR FIL command interactively, the default fillet radius is set for all viewboxes in the current sheet and for all sheets you subsequently call up. This may not be what you intend, so it is safer to place the PAR FIL command as an in-sheet command inside the appropriate viewbox. Information on using in-sheet commands is given in [“Overview of Parametric Design” on page 9.](#)

PAR FIL Command Syntax



Option	Description
UNSET	Specifies that all fillets must be dimensioned explicitly. This is the default setting for the PAR FIL command.
KEEP	Specifies that all fillets not explicitly dimensioned keep their original radius during parameterization.
<i>filrad</i>	<p>The default fillet radius of all undimensioned fillets.</p> <p>MAX Specifies that only fillets with original radius less than or equal to <i><maxrad></i> should be changed to the specified value. You must explicitly dimension all other fillets. This option provides a safeguard against accidentally dimensioning large radii.</p> <p>MAXRAD Maximum fillet radius.</p>

Displaying the Current PAR FIL Setting

Use the Q PAR FIL command to display the current PAR FIL setting.

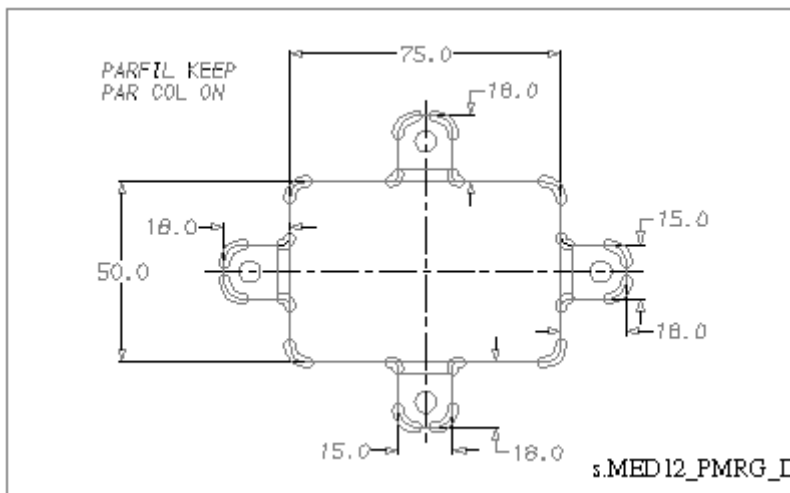
The syntax for this command is:

→ Q PAR FIL →

Checking the Effect of PAR FIL

You can check the effect of the PAR FIL command using the command PAR GRIS FIL. This draws grid lines over those fillets affected by the current PAR FIL setting. Grid lines are not drawn over fillets which are specifically dimensioned or which exceed the maximum value specified by `maxrad`. You can find out more about the PAR GRIS command in [“The Parametric Grid” on page 59](#).

Figure 21 Effect of the PAR GRIS FIL Command



Examples Showing PAR FIL Options

The following examples show the effect of each of the PAR FIL options.

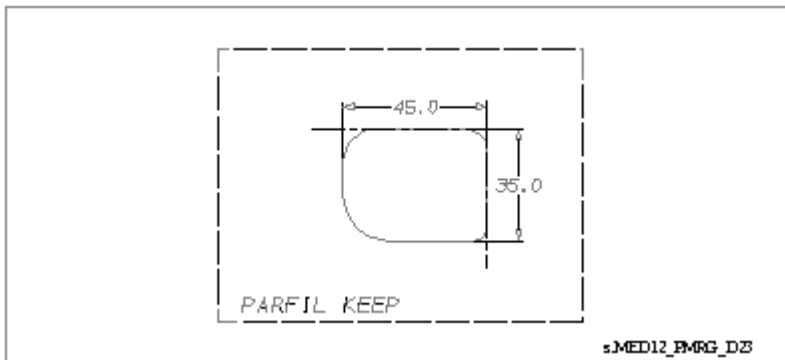
PAR FIL UNSET

This is the default option for PAR FIL. You must explicitly dimension all fillets in order to place them on the grid. Any undimensioned fillets will generate error messages.

PAR FIL KEEP

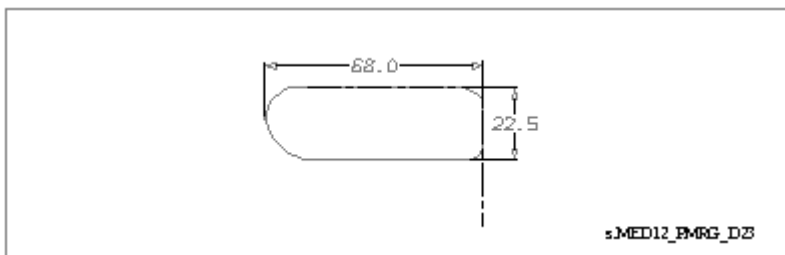
This command specifies that any fillets not explicitly dimensioned keep their original radius during parameterization. PAR FIL KEEP is placed on the sheet as an in-sheet command illustrated in the following figure.

Figure 22 In Sheet PAR FIL KEEP Command



The following figure shows the result of parameterizing the object shown in the figure above with new parameters. Note that all undimensioned fillets have kept their original radius.

Figure 23 After Parameterization

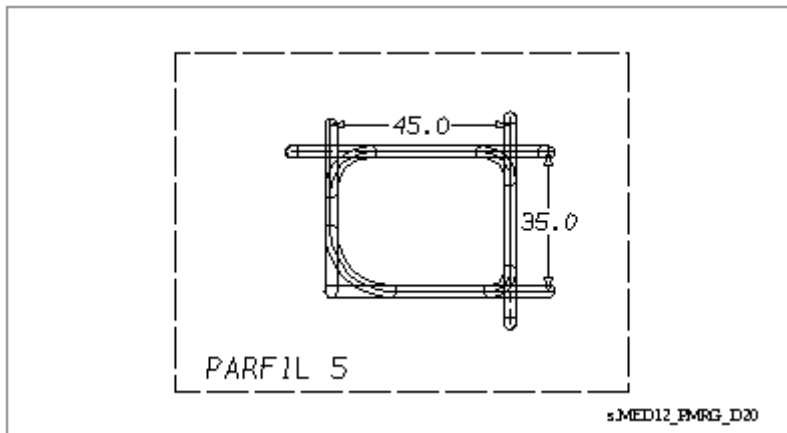


You can specify new parameters for any individual fillets by dimensioning them normally.

PAR FIL filrad

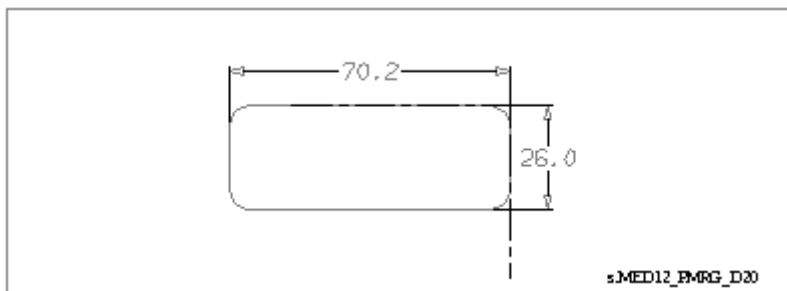
Use this option to specify a default fillet radius. In the following figure a default radius of 5 mm has been specified with an in-sheet command.

Figure 24 In-sheet PAR FIL 5 Command



The following figure shows the object after parameterization with new dimension values. The 5.0 mm, 7.0 mm, 10.0 mm and 15.0 mm fillets all have the same radius of 5 mm because of the in-sheet PAR FIL command.

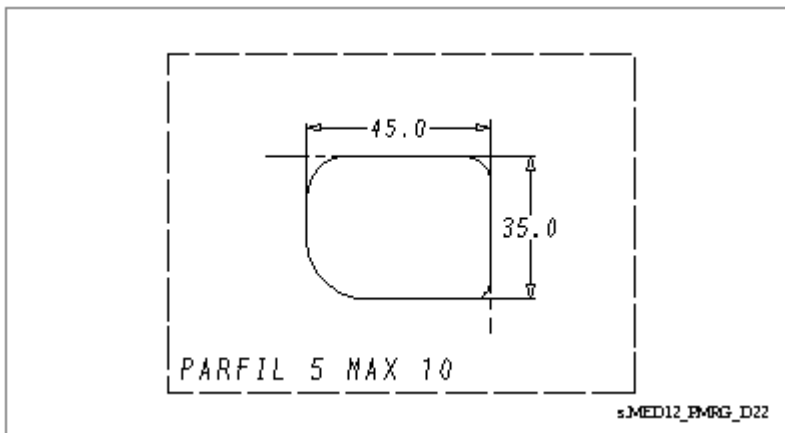
Figure 25 After Parameterization



PAR FIL filrad MAX maxrad

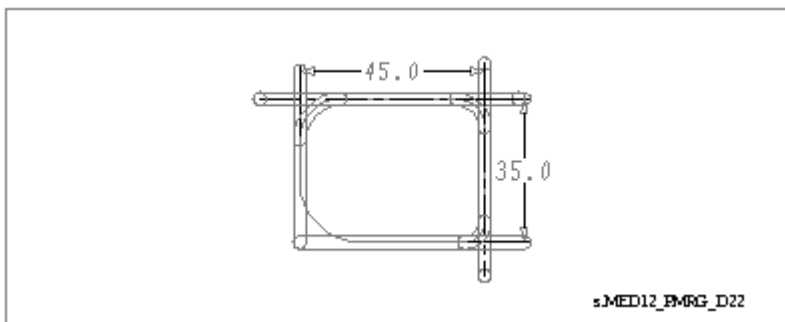
Specifies that only fillets with original radius less than or equal to maxrad should be changed to the specified value. You must explicitly dimension all other fillets. This option provides a safeguard against accidentally dimensioning large radii. The following figure shows how the default fillet radius is set to 5 mm, with a maximum radius of 10 mm using an in-sheet command.

Figure 26 In-sheet PAR FIL 5 MAX 10 Command



Any fillet with a radius exceeding the maximum specified value will not be supported on the parametric grid, but must be dimensioned explicitly. The effect is revealed by drawing the grid lines, as shown in the following figure. The 15 mm fillet exceeds the maximum specified value and it is not explicitly dimensioned, therefore it is not supported on the grid.

Figure 27 Unsupported Fillet Revealed by PAR GRIS Command



Tolerance Dimensioning

There are four kinds of tolerance dimensioning in MEDUSA. These are created using the following options with the DIM command:

- ABS
- UNS
- LIM
- VAR

The Parametric Design system needs the text in a dimension clump to evaluate to a single value in order to be able to parameterize the geometry normally. *ABS* and *UNS* tolerances give a single dimension value, but *LIM* and *VAR* tolerances specify a range of values and so have to be treated specially.

ABS and UNS Tolerances

ABS creates a dimension with a single boxed dimension text. This indicates that it is an absolute value. *UNS* creates a dimension with a dimension text that has a token value. The dimension text is underlined to show that the value given is unscaled, but it is a single value, so it can be parameterized normally.

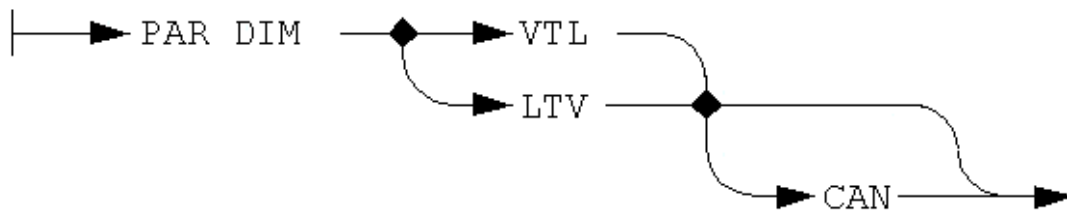
LIM Dimensioning

The *LIM* option of the DIM command creates a dimension with a maximum and minimum tolerance. Because the system requires a single, nominal value to calculate the new dimension, dimensions with *LIM* tolerances cannot be parameterized directly.

You can, however convert *LIM* tolerances into *VAR* tolerances before parameterization and then convert them back afterwards. The command to convert *LIM* tolerances into *VAR* format is `PAR DIM LTV`. After parameterization you can convert *VAR* tolerances back to *LIM* format with the command `PAR DIM VTL`.

PAR DIM command syntax

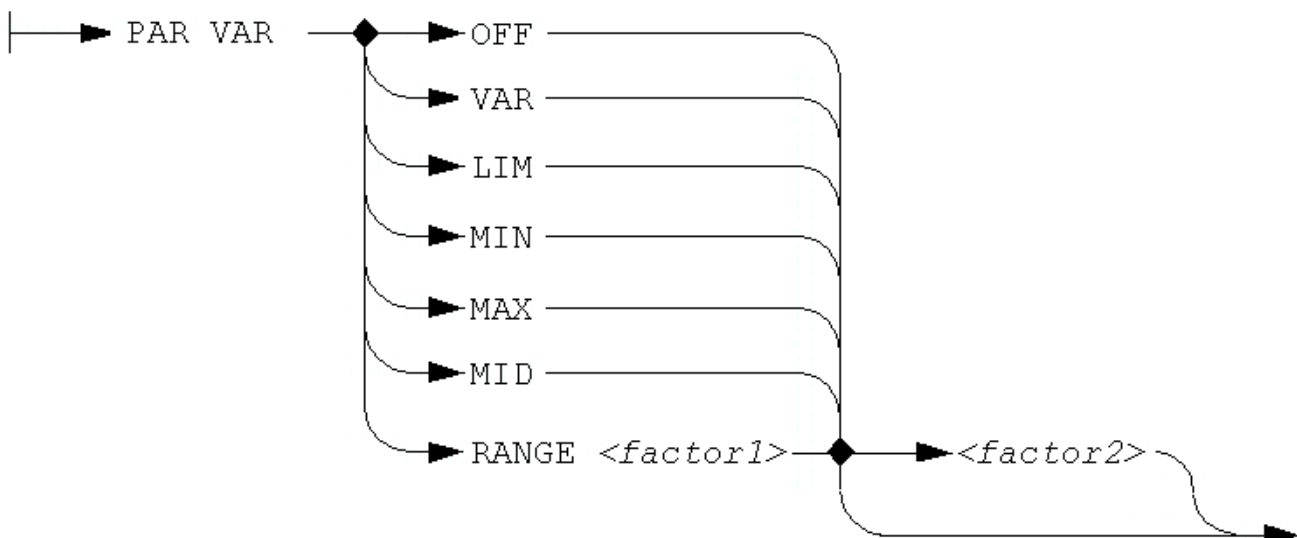
The syntax for the PAR DIM command is:



VAR Dimensioning

VAR creates a dimension with a nominal dimension value, and lets you specify permissible upper and lower tolerances. When you parameterize geometry containing VAR tolerances, you can control the way the system calculates the new dimension value with the PAR VAR switch.

PAR VAR Command Syntax



Options

The PAR VAR command has seven options: OFF, VAR, LIM, MIN, MAX, MID and RANGE. Each of these options specifies a tolerance which is used by the system to calculate the new dimension value during parameterization.

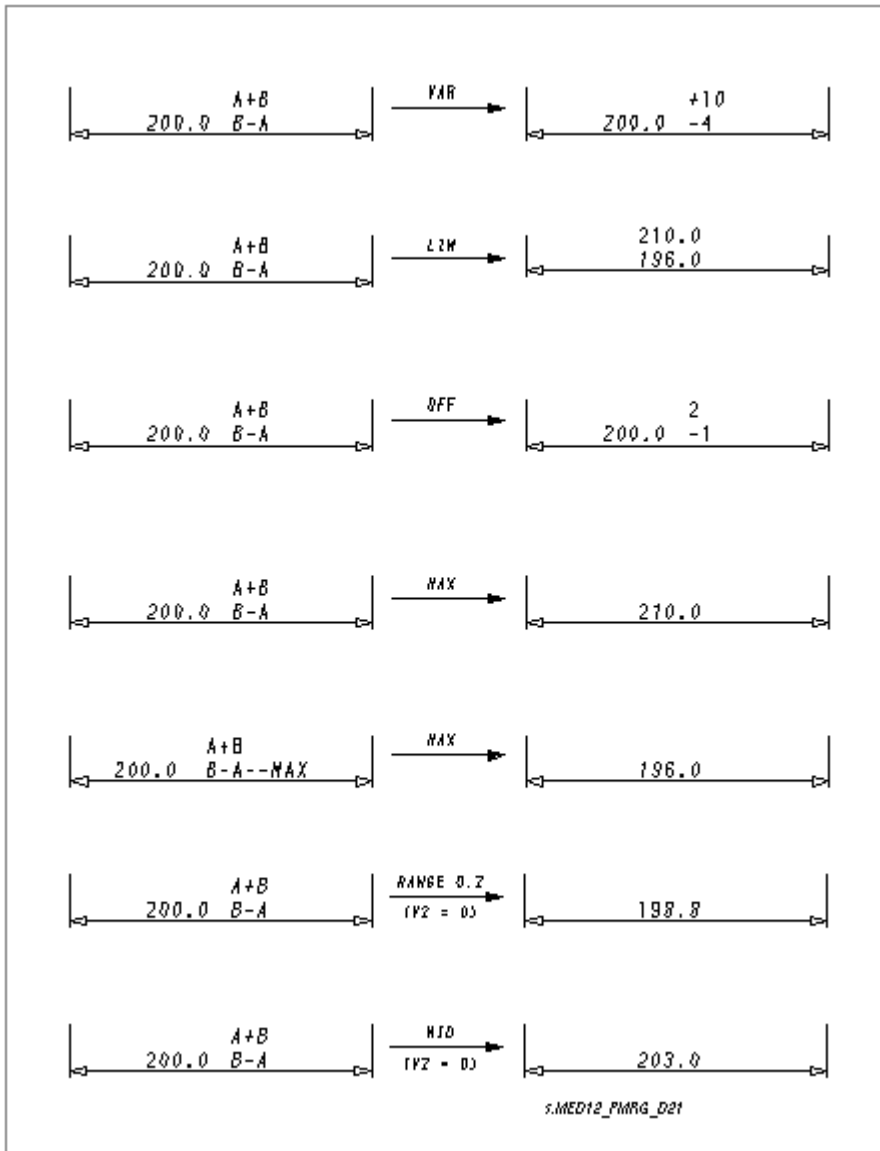
Option	Explanation
OFF	Specifies that any changes made to the original tolerance text are ignored. The original tolerance appears in the parameterized drawing.
VAR	Ensures that any expressions which have replaced original tolerance text are evaluated during parameterization. The resulting value is used to calculate the new dimension and resulting values are placed in the new dimension. This is the default setting.
LIM	Converts VAR tolerance dimensions to LIM format. The new tolerances are calculated by evaluating the original tolerance texts, and adding them to the nominal value.
MIN	Specifies that the lower tolerance is used for all tolerance dimensions. In some cases, such as holes, you may need to indicate maximum or minimum diameters explicitly. You can do this by editing the individual tolerance texts. This is described in “Specifying Individual Tolerances” on page 44 .
MAX	Specifies that the upper tolerance is used. In some cases, such as holes, you may need to indicate maximum or minimum diameters explicitly. You can do this by editing the individual tolerance texts. This is described in “Specifying Individual Tolerances” on page 44 .
MID	Calculates the tolerance using the following formula: $(MAX - MIN) * 0.5 + MIN$
RANGE	Calculates the tolerance using the following formula: $(MAX - MIN) \text{ factor1} + MIN \text{ factor2}$ factor1 must be a factor in the range 0 to 1. factor2 is an optional extra factor. You must supply factor1 as an argument here, but factor2 is optional.

Examples of all seven PAR VAR options are shown in the figure below opposite. The original upper tolerance used in these examples is 2 and the lower tolerance is -1. The tolerance variables A and B have the following initial values:

$$A = 7$$

$$B = 3$$

Figure 28 Effects of the PAR VAR Switch



Specifying Individual Tolerances

The options PAR VAR MAX and MIN set the tolerance for all tolerance dimensions during parameterization. In some cases you may need to indicate an individual tolerance explicitly, for example, to indicate the maximum diameter for a hole dimension. To do this, you can add --MAX or --MIN text to the relevant tolerance. This specifies that the tolerance indicated is to be used to calculate the new diameter dimension regardless of the current PAR VAR setting.

If you add --MIN to the higher tolerance, the higher tolerance will be used to calculate the new dimension when the PAR VAR switch is set to MIN. Conversely, with the switch set to MAX, you can add --MAX text to the lower tolerance and the lower tolerance will be used to calculate that dimension.

Procedure

Use the following procedure to place --MAX or --MIN text in a dimension:

1. Using a LET statement, set the new text variable TEXT\$ to the tolerance text followed by --MAX or --MIN, for example:

```
LET TEXT$ = '-.5 --MAX'
```

2. Make the relevant tolerance text current.
3. Replace the current text with the new text using the GOBC command.

For more information on using LET, refer to the *MEDUSA Bacis1 Design Commands Guide*.

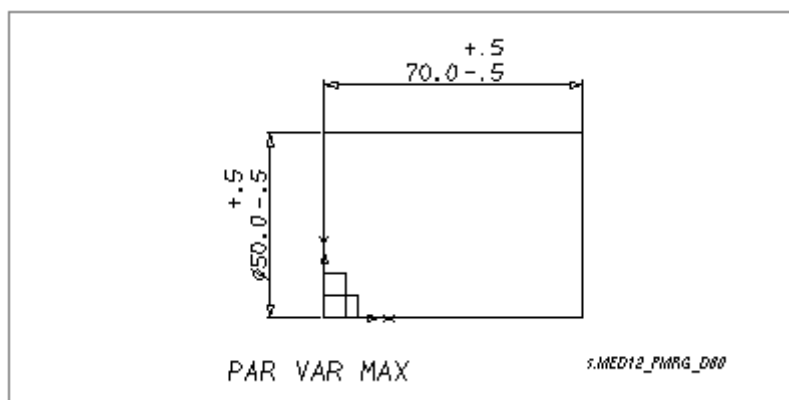
Example

To add --MAX to the lower tolerance of the diameter dimension shown in Figure 29 you would use the following command sequence:

```
*LET TEXT$ = '-.5 --MAX'
(position the cursor near the lower tolerance text)
*FINT CP
*GOBC
*
```

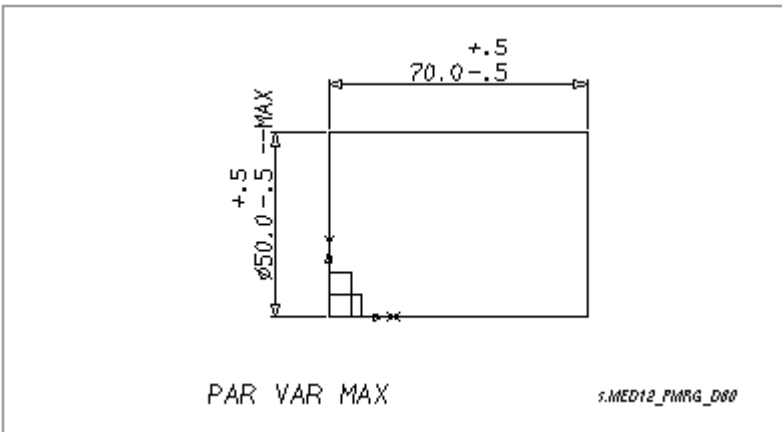
In the following figure the in-sheet command PAR VAR MAX specifies that the new dimension values will be calculated using the upper tolerance.

Figure 29 Tolerance Dimensions



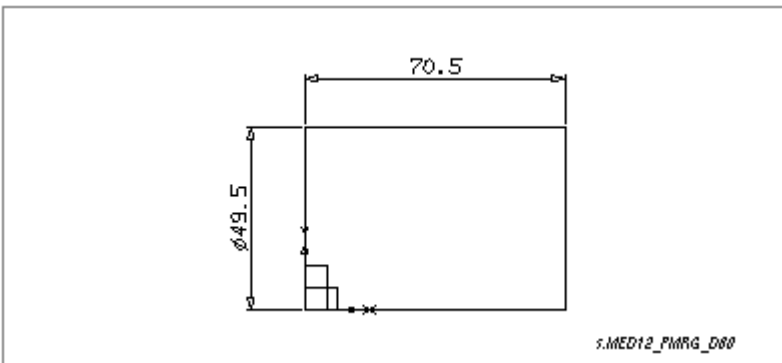
To ensure that the lower tolerance is used for the diameter dimension, --MAX text has been added to the lower tolerance text, as shown in the following figure. For all other dimensions in the viewbox, the upper tolerance will still be used during parameterization.

Figure 30 Inserting --MAX Text



The following figure shows the result of parameterizing the object shown in Figure 30. The lower tolerance has been used for the new diameter dimension and the upper tolerance for the other chain dimension.

Figure 31 Result of Parameterization



GEOMETRIC CONSTRAINTS

Most geometric constraints are normally deduced automatically by the Parametric Design system. If you need to specify constraints explicitly you can use dynamic baselines.

- [Introduction](#) 48
- [Perpendicular Points](#) 49
- [Intersection Points](#) 50
- [Tangent Points](#) 52
- [Circular Baselines](#) 53
- [Automatically Inferred Constraints](#) 55

Introduction

Most geometric constraints are normally deduced automatically by the Parametric Design system. However, there are times when you must specify constraints explicitly:

- Geometric constraints are not deduced automatically when the `PAR BAS` or `PAR LIM` switches are OFF
- Certain constraints may only be specified explicitly

You specify geometric constraints explicitly in the Parametric design system using dynamic baselines. Dynamic baselines are useful in situations where a dimension puts an unnecessary constraint on parameterization, for example, tangency. They may also be needed to place certain dimensions onto the grid.

Dynamic Baselines

A dynamic baseline consists of line segments of type LBL. If the line has several segments, each is considered independently. Each linear baseline generates three straight grid lines

- One along the baseline
- One through each end, perpendicular to the baseline

The grid lines generated by a dynamic baseline move relative to the end points. Thus, if the baseline is from an intersection to a tangent point, the new grid line will be from the new intersection tangential to the new circle.

Baseline Point Functions

The following five point functions have special meaning to the Parametric Design system when they are attached to a baseline:

- FUNV0 for a static point
- FUNV10 for a perpendicular point
- FUNV11 for an intersection point
- FUNV26 for the center of a circle

A baseline with a FUNV0 point function at each end is a static baseline. Static baselines define fixed reference points during parameterization. You can change the point function at one or both ends of a static baseline to convert the line into a dynamic baseline.

Setting Baseline Point Functions

There are two ways you can set a particular point function on a baseline:

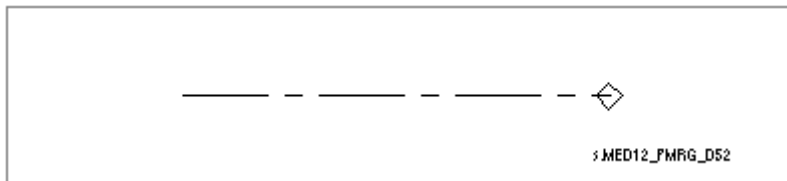
- Set the function while you are creating the line by placing the point with a probe and then immediately assigning the chosen point function
- Edit the baseline and, with the relevant point current, either probe the point function area on the MEDUSA 2D menu or type the command, for example, FUNV 11

Perpendicular Points

A perpendicular point must be supported by a single existing grid line perpendicular to the baseline, passing through the perpendicular point. The combination of perpendicular to perpendicular point functions is not allowed.

The following figure shows a dynamic baseline with a perpendicular point at one end and a static point at the other.

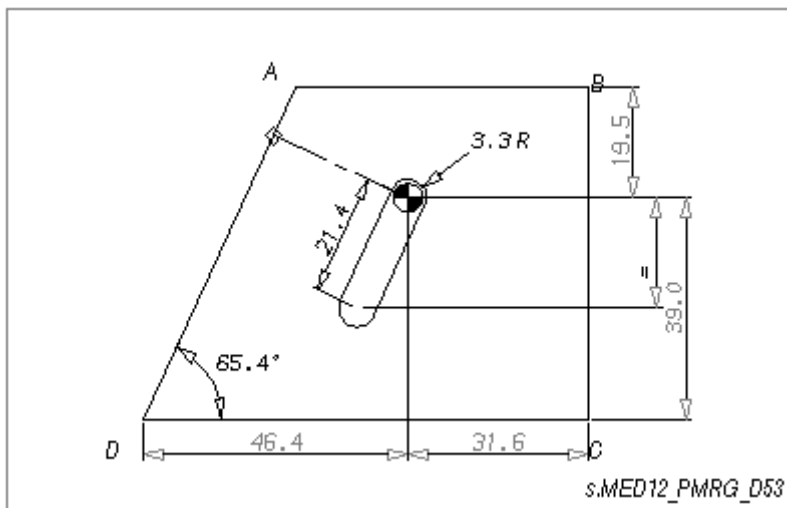
Figure 32 Perpendicular Point, FUNV10



Example

In the following figure a baseline with a perpendicular point is used to keep the slotted hole parallel to the line segment AD when the angle ADC is parameterized.

Figure 33 Dynamic Baseline With Perpendicular Point



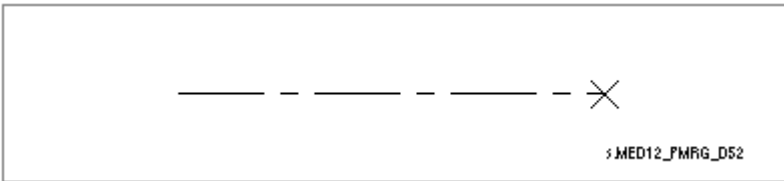
Intersection Points

An intersection point must be supported by an intersection of existing grid lines. Intersection points are FUNV11. You can place an intersection point at any of the following:

- The center of a circular grid line
- An intersection of straight grid lines
- An intersection of circular grid lines
- An intersection of a straight and circular grid line

The following figure shows a dynamic baseline with an intersection point at one end and a static point at the other.

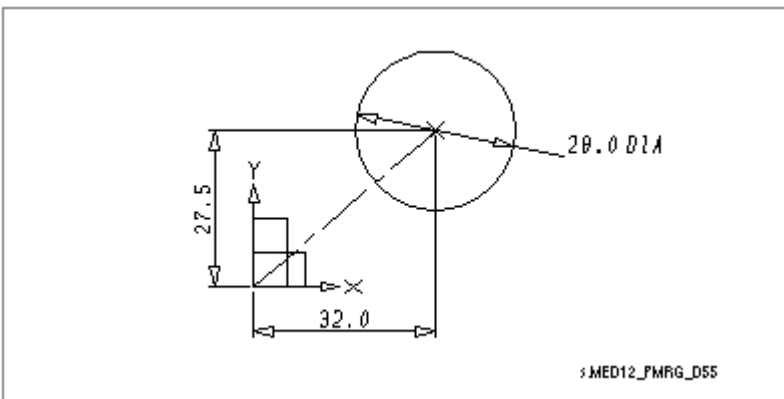
Figure 34 Intersection Point, FUNV11



Example

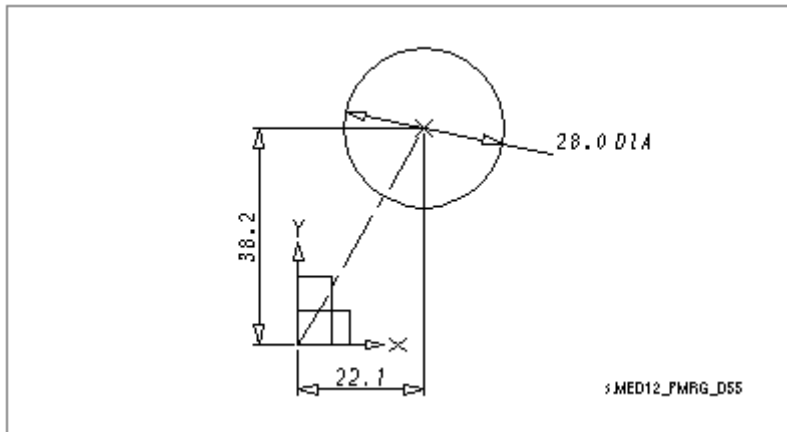
The baseline in the following figure has a static point at one end, attached to the reference point, and an intersection point at the other end, at the center of a circle. The intersection point is supported by an intersection of existing grid lines created by the dimensions.

Figure 35 Baseline With Intersection Point



The following figure shows the result of parameterization. The circle moves in relation to the reference point, but the intersection point ensures that the center of the circle always remains at a grid intersection.

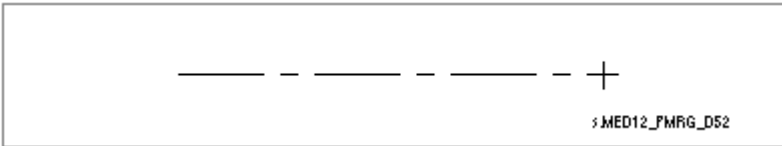
Figure 36 After Parameterization



Tangent Points

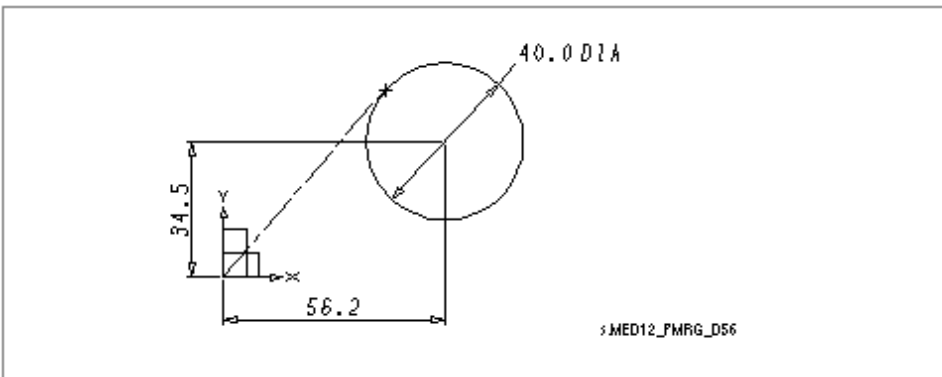
A tangent point remains at a tangent to a circular grid line when you shrink or expand the baseline. Tangent points are FUNV12. The following figure shows a dynamic baseline with a tangent point at one end and a static point at the other.

Figure 37 Tangent Point, FUNV12



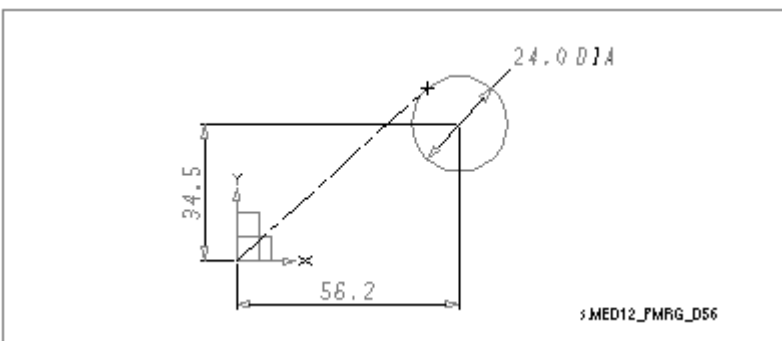
In the following figure, the baseline is tangential to a circle of 40 mm diameter.

Figure 38 Baseline With Tangent Point



When the diameter of the circle is reduced, the baseline tangent point remains tangential to the circle, as shown in the following figure.

Figure 39 After Parameterization



Circular Baselines

A circular dynamic baseline is a straight line segment with the following point functions:

- FUNV 26 at one end of the line segment
- FUNV 10 or 11 at the other end of the segment

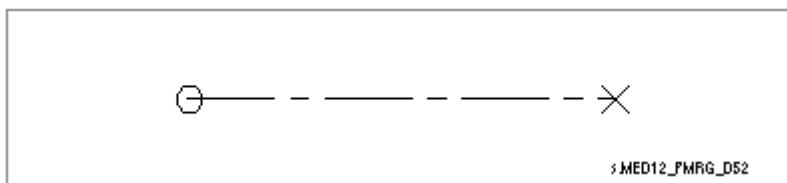
A circular baseline generates the following grid lines:

- A circular grid line centered on the FUNV26 point function and passing through the other end of the baseline
- A straight grid line along the line segment

Do not confuse circular dynamic baselines with circular static baselines, which have FUNV0 point functions.

The following figure shows a baseline with a circle center point at one end and an intersection point at the other.

Figure 40 Circular Base Line, FUNV26 and FUNV11



Circular Baseline Point Functions

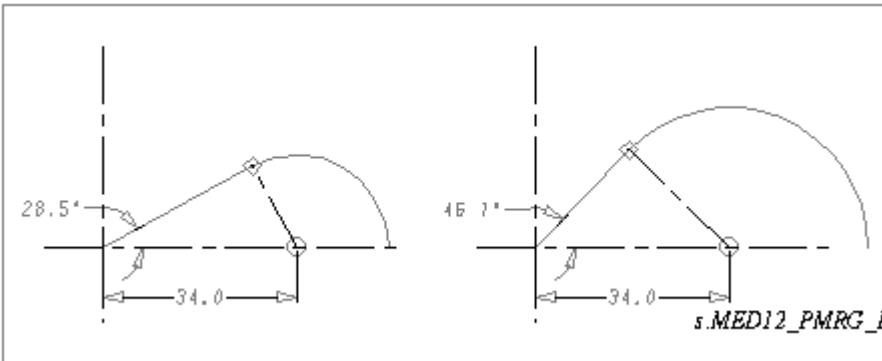
FUNV 26 One end of the baseline must have a FUNV 26 point function to indicate the center point. This end must lie at an existing grid line intersection and is used as the center of the circular grid line.

FUNV 11/10 One end of the baseline must be an intersection point, FUNV 11 or a perpendicular point, FUNV 10. The circular grid line passes through this point. Note that if you use a perpendicular point, the circular grid line will be tangential to the supporting grid line. It is the baseline, representing a radius of the circle, that is perpendicular to the existing grid line not the circle itself.

Examples

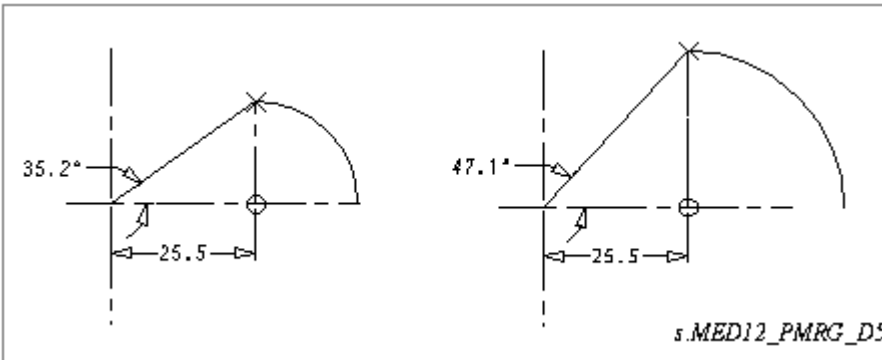
The following figure shows a drawing with a circular baseline, point functions FUNV 26 and FUNV 10. When the angle is increased, the perpendicular point remains at the intersection of the circle and the straight line, and the center point remains at the center of the circle.

Figure 41 Circular Baselines, FUNV26 and FUNV10



The following figure shows a baseline with FUNV 26 and FUNV 11 point functions. When the angle is increased, the baseline intersection point remains at the intersection of the arc and the straight line, and the center point remains at the center of the arc.

Figure 42 Circular Baselines, FUNV26 and FUNV11



Automatically Inferred Constraints

The Parametric Design system infers geometric constraints automatically when the `PAR BAS` switch is `ON`, provided that the `PAR LIM` switch is also `ON`. Every time you execute the `PARS` command with `PAR BAS ON`, the system looks to see where geometric constraints may be inferred from the geometry.

How Constraints are Inferred

The system infers geometric constraints automatically in the following way:

1. Unused potential grid lines are examined to see if they pass through any existing grid intersections, or are tangential or perpendicular to any existing grid lines.
2. If two points on a potential grid line are found that are intersection, tangent or perpendicular points, then a geometric constraint is inferred between the two points.

You can see what constraints are being inferred by the system using the `PAR GRIS BAS` command. This adds a baseline with the appropriate point functions wherever the system infers a constraint automatically.

Priority of Geometric Constraints

If the system finds more than two points on a potential grid line from which it can infer constraints, then two are chosen in the following preference order:

1. Intersection
2. Tangent
3. Perpendicular

If this preference order does not determine which two points to use, then the points are chosen arbitrarily.

When Constraints Are Inferred

Before any geometric constraints are inferred, the system places as much of the geometry as possible onto the grid. Then it attempts to infer geometric constraints in the following way:

1. The system infers any constraints it can from the existing grid.
2. The system looks to see if any more of the object geometry can be placed onto the grid.
3. The system looks again to see if it can infer any constraints from the existing grid.

This cycle continues until no further points of geometry can be placed on the grid and no more constraints can be inferred.

Baseline Inference and PAR FIL

If a default fillet radius is specified using PAR FIL, the specified radius is only used to support fillets if they cannot be supported by a grid line generated by an inferred constraint.

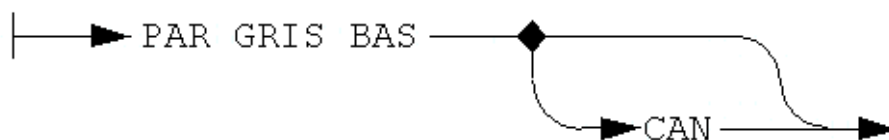
Baselines are inferred automatically, except when:

- The PAR BAS switch is OFF
- The PAR LIM switch is OFF

Displaying Inferred Baselines

You can look at inferred constraints using an option of the PAR GRIS command. The command PAR GRIS BAS draws automatically inferred baselines on the sheet. Some of the inferences may depend on the order of elements in the sheet, as there is often more than one way of adding baselines to provide all the necessary grid lines.

PAR GRIS BAS Syntax



The PAR GRIS BAS command can help to speed up parameterization. Once you have finished the definition drawing and successfully tested it, you can speed up subsequent parameterization commands by using PAR GRIS BAS to place the inferred baselines permanently on the sheet. You can then parameterize the sheet with the PAR BAS switch OFF. This prevents the system scanning for geometric constraints each time you parameterize the sheet.

Unwanted Inferences

In certain cases, unwanted inferences may be made by the system, placing unnecessary constraints on how the geometry is transformed during parameterization. To avoid this, you can apply all necessary constraints explicitly and then prevent the system applying constraints automatically. Use the following procedure:

1. Add all possible dynamic baselines using the PAR GRIS BAS command.
2. Delete any unwanted baselines.
3. Before parameterization, turn inferencing off by setting the PAR BAS switch to OFF.

For more information on the PAR GRIS command and how to draw parametric grids refer to [“The Parametric Grid” on page 59](#).

THE PARAMETRIC GRID

This chapter describes how MEDUSA Parametric Design uses a parametric grid to move individual object points during parameterization.

- Introduction 60
- The Old Grid 62
- The New Grid..... 63
- The Potential Grid..... 65
- Limited Grid Lines..... 66
- Displaying Grid Lines..... 68
- Adding Lines 71
- Grid Tolerance..... 75

Introduction

Parametric Design works not by moving individual points but by moving a grid upon which all movable points are placed. The grid consists of lines at any angles and also of circles. When you place a reference point in a viewbox, you create the starting point for the parametric grid. The reference point generates the first grid intersection. You build the parametric grid by dimensioning object points. Each time you dimension a point in the object geometry another grid intersection is created at that point. Dimensioned points are said to be supported by the grid.

All moveable points of object geometry must be placed at intersections on the parametric grid before parameterization. If a point is not at a grid intersection then the system does not know where to move it to during parameterization. When all points are dimensioned, the grid is complete.

Displaying Parametric Grid Lines

Grid lines are not visible until you give the PAR GRIS command. PAR GRIS displays grid lines corresponding to different versions of the grid:

OLD grid : Corresponds to the original object geometry plus the reference point and dimensioning.

NEW grid : Corresponds to the parameterized geometry.

POTENTIAL grid : Corresponds to the original object geometry without dimensioning. A grid intersection is generated at every point in the object geometry.

Limited and Unlimited Grid Lines

The final appearance of the grid lines depends on whether the lines are limited or unlimited. This is set using the PAR LIM switch, which is described fully under section [“Limited Grid Lines” on page 66](#).

Limited grid lines : Limited grid lines trace over only those parts of the object that are adequately dimensioned. You can clearly see which parts of the object have not yet been dimensioned.

Unlimited grid lines : Unlimited grid lines extend to the edge of the viewbox rather than simply tracing over the object, making it harder to detect errors.

Deleting Grid Lines

It is useful to display grid lines each time you add a new dimension clump to check that dimensioning is accurate and that the points you have dimensioned are placed on the grid properly. However, you should never leave the grid lines on the sheet while you are adding dimensions or other lines to the drawing. The probes used in dimension construction lines may find the ends of the grid lines, which are slightly displaced from the ends of the object. To prevent grid lines from interfering with dimensioning, either

- Cancel the PAR GRIS command immediately with CAN
- Delete the layer containing the grid lines (layer 99)

The Old Grid

The reference point and dimensions generate a grid consisting of lines at any angles and also of circles. Every point that will move during parameterization must lie on an intersection point in the grid. This grid is often referred to as the OLD grid because it corresponds to the original (old) object geometry. The lines of the OLD grid are drawn along the lines of the geometry when you give the PAR GRIS command.

Displaying OLD Grid Lines

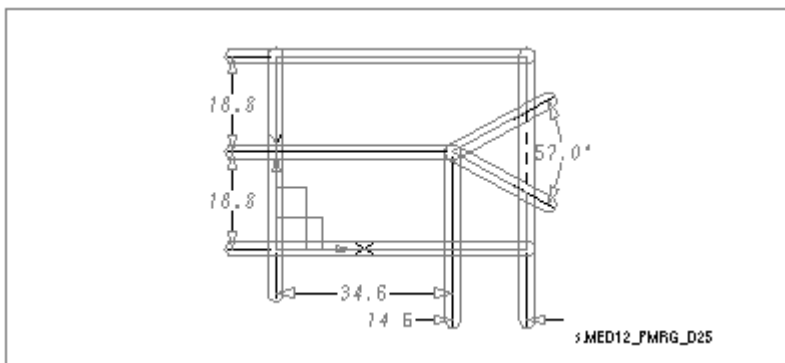
OLD grid lines are displayed using the commands PAR GRIS OLD or PAR GRIS. The grid lines are type STK lines 3 mm (0.118 inch) in width. Drawing the OLD grid is useful to reveal errors in the drawing, in particular, inaccurate dimensioning. Inadequately dimensioned points will generate the following error message:

```
Point not dimensioned
```

The datum of the error message text is centered on the point causing the problem.

Example: The following figure shows the effect of the PAR GRIS OLD command. Grid lines are drawn over all dimensioned points in the original object geometry. This is the OLD grid.

Figure 43 Old Grid Lines



Using PAR GRIS to Check the Grid

The order in which the OLD grid lines are drawn indicates the order in which the object points were placed on the grid. To examine the grid you are building, use the following procedure:

1. Draw the grid with the PAR GRIS command.
2. Redraw the sheet without the grid lines (grid lines are on layer 99).
3. Locate the first element in the sheet with the STA command.
4. Step through the grid with repeated NEXA STK commands.

This procedure should help you to diagnose errors and give you a better understanding of how the Parametric Design system works.

The New Grid

The system uses the parameters you supply to calculate the new position of each intersection in the OLD grid. The NEW grid shows the position of each OLD grid line based on the new parameters.

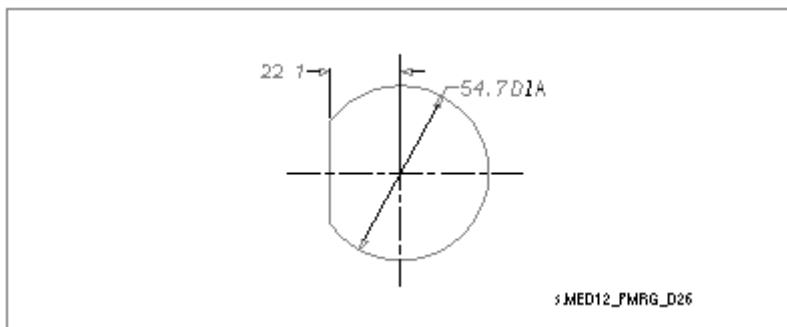
Displaying NEW Grid Lines

Use the command PAR GRIS NEW to display NEW grid lines. The lines are type L4 lines and are placed on layer 99. New grid lines extend to the edges of the viewport.

Investigating Errors

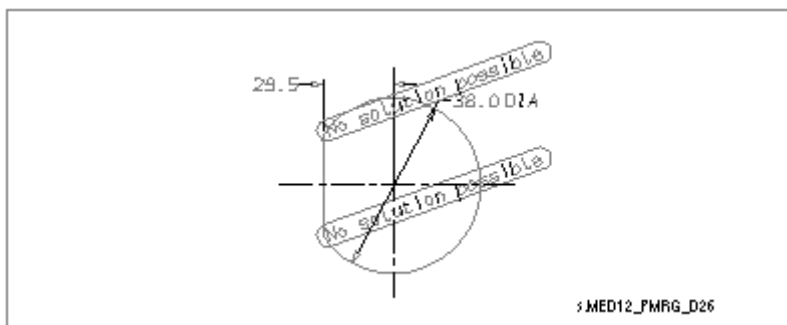
You can use PAR GRIS NEW to investigate errors relating to the new parameters. For example, in the following figure, the vertical line intersects with the circle. The system recognizes this as a geometric constraint: whenever the geometry is parameterized, the vertical line must intersect with the circle.

Figure 44 Parameterization Errors



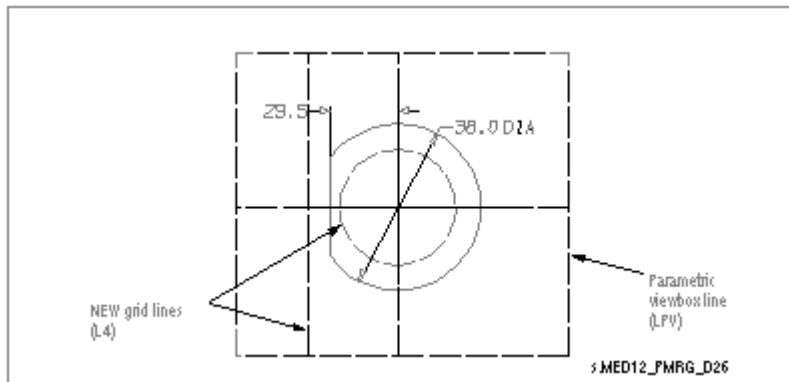
When the chain dimension is increased to 29.5 and the diameter is reduced to 38.0, a problem occurs, as shown in the following figure. Using the new parameters, the vertical line no longer intersects with the circle. Consequently, parameterization fails and error messages appear at the points causing the problem.

Figure 45 Parameterization Errors



Using PAR GRIS NEW you can examine the new grid and correct any problems with the new parameters. The NEW grid lines in the following figure show that the vertical line no longer intersects the circle.

Figure 46 Effect of PAR GRIS NEW Command



Removing NEW Grid Lines

To avoid leaving the NEW grid lines on the sheet, either cancel the PAR GRIS NEW command immediately or delete the layer containing the grid lines (layer 99).

The Potential Grid

The potential grid is generated from the object geometry alone, without reference to any dimensioning. You can look at potential grid lines before you dimension the geometry. Potential grid lines extend no further than the edges of the object geometry and are used by the system when creating the OLD grid from dimensions and other constructions in the drawing.

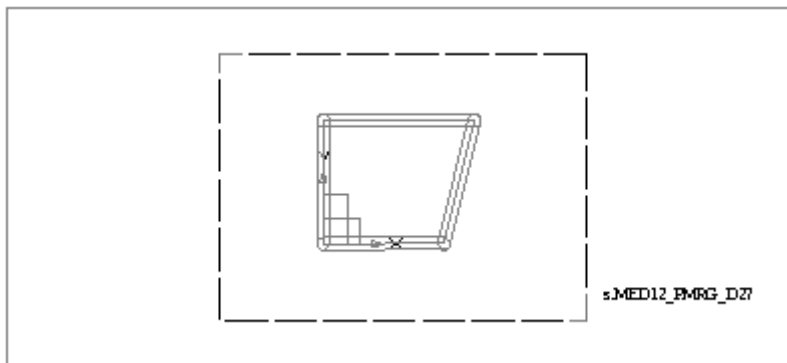
You might use potential grid lines to check, for example, whether the system recognizes colinearity constraints within the current grid tolerance setting.

Displaying Potential Grid Lines

Use the command PAR GRIS POT to display potential grid lines. The lines are type STK lines of 3 mm (0.118 inch) width, and are placed on layer 99.

You can only see the potential grid if the PAR LIM switch is ON (see [“Limited Grid Lines” on page 66](#)). The following figure shows an example where PAR GRIS POT is used to display the potential grid for an undimensioned object.

Figure 47 Effect of PAR GRIS POT Command



Parametric Switches and Potential Grid Lines

There are a number of switches that control the generation of potential grid lines. See [“Switches and Layers” on page 95](#), for more information about these switches.

Limited Grid Lines

The `PAR LIM` switch alters the appearance of OLD grid lines. These grid lines may be either limited or unlimited.

Limited Grid Lines

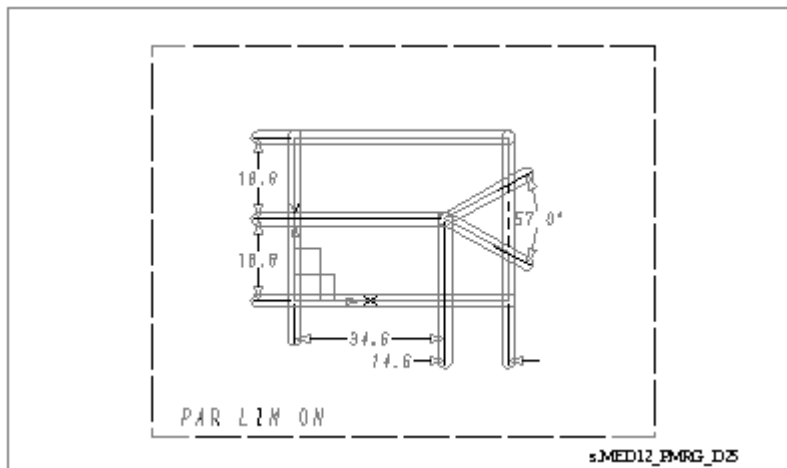
With `PAR LIM` set to `ON`, OLD grid lines are drawn along lines in the drawing, and do not extend beyond the end of the geometry. This is the default setting.

Potential grid lines are generated by the object geometry and extend no further than the edges of the object geometry. The system uses potential grid lines and dimensions to create the OLD grid. When the `PAR LIM` switch is `ON`, the length of each OLD grid line is limited to the extent of the corresponding potential grid line. Therefore, OLD grid lines are limited to the edges of the object geometry. If there is no corresponding potential grid line, a zero length grid line is created.

Example

The following figure shows a viewbox containing a fully dimensioned object. The in-sheet command `PAR LIM ON` has been placed inside the viewbox so that limited grid lines are displayed when the `PAR GRIS` command is given.

Figure 48 Limited Grid



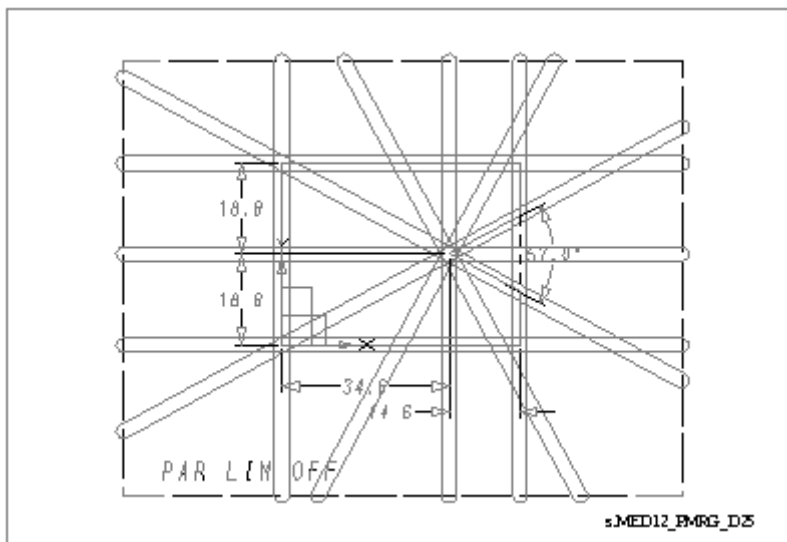
Unlimited Grids

When the `PAR LIM` switch is `OFF`, grid lines are unlimited. This means that they are not constrained to lie along the lines in the drawing, as represented by the potential grid lines, but are considered to be infinite in length. When drawn, the grid lines are limited only by the edges of the parametric viewbox. Unlimited grid lines have more grid intersections, which may sometimes be advantageous. However, there are a number of disadvantages to using unlimited grids:

- The grid lines do not simply trace over the part of the drawing that is adequately dimensioned, so it is harder to find errors
- Spurious intersections are easily generated, adding unwanted constraints to the drawing
- You cannot use default fillets (see "Dimensioning", "How Fillets are Defined for `PAR FIL`" on page 35) or automatic deduction of geometric constraints (see "Geometric Constraints", "Automatically Inferred Constraints" on page 55) with `PAR LIM OFF`

The following figure shows the object from Figure 48, "Limited Grid" on page 66, this time with the in-sheet command `PAR LIM OFF`. The OLD grid now extends past the edges of the object to the edges of the viewbox.

Figure 49 Unlimited Grid



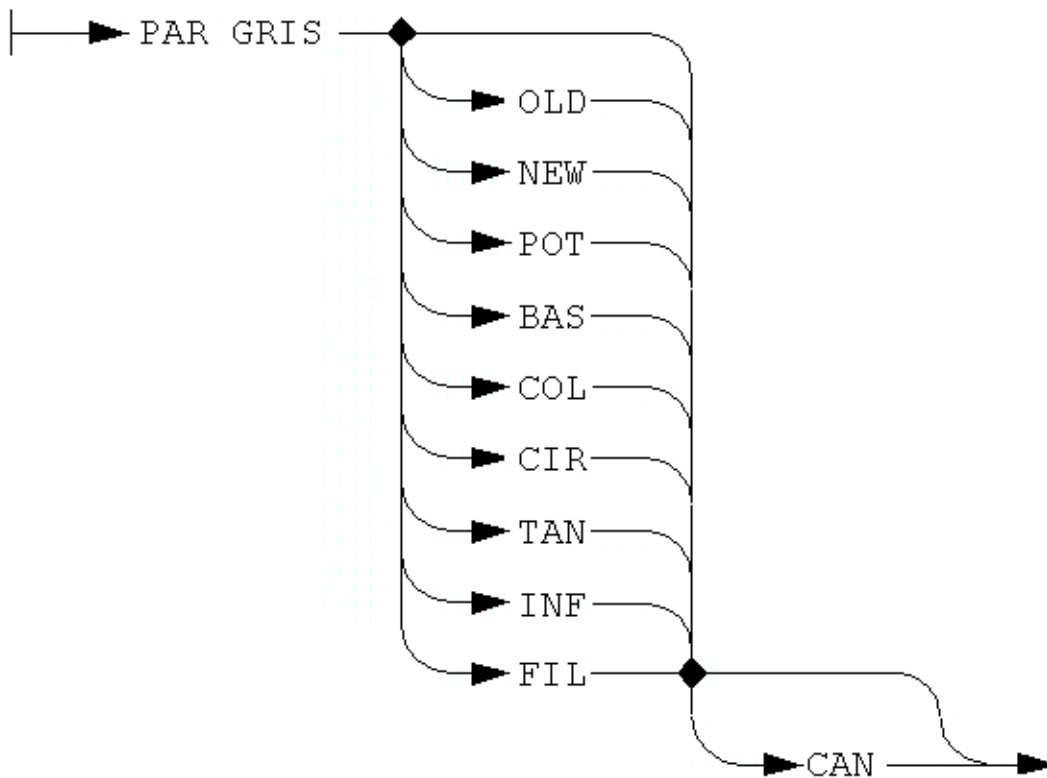
Displaying Grid Lines

The PAR GRIS command displays parametric grid lines. If the PAR LIM switch is OFF, you can only use the OLD and NEW options with the PAR GRIS command.

How to Display Grid Lines

To display grid lines, type PAR GRIS followed by the appropriate option.

PAR GRIS Syntax



Option	Explanation
OLD	Displays OLD grid lines, that is the grid corresponding to the original geometry. This is the default option. See “The Old Grid” on page 62 for more information about this option.

Option	Explanation
NEW	Displays the NEW grid, that is the grid corresponding to the parameterized drawing. NEW grid lines extend to the edges of the viewbox. See “The New Grid” on page 63 for more information about this command.
POT	Displays the potential grid. The potential grid is generated from the set of lines in the drawing. See “The Potential Grid” on page 65 for more information about this command.
BAS	Adds baselines corresponding to those automatically inferred by the system. Information on using the PAR GRIS BAS command is given under “Geometric Constraints”, “PAR GRIS BAS Syntax” on page 56 .
COL	Adds lines joining collinear straight line segments which do not overlap. See “PAR GRIS COL” on page 72 for more information about this command.
CIR	Adds complete circles for every arc of a circle in the viewbox. See “PAR GRIS CIR” on page 73 for more information about this command.
TAN	Adds tangent lines for any tangent point arcs in the viewbox. See “PAR GRIS TAN” on page 74 for more information about this command.
INF	Adds unlimited lines along each line segment in the viewbox and complete circles for every arc of a circle. An example is given in “PAR GRIS INF” on page 71 .
FIL	Adds grid lines along fillets that are affected by the current PAR FIL setting. PAR FIL is described in detail in “Dimensioning”, “Dimensioning Fillets” on page 35 .
CAN	Cancels the command. Note that cancelling the PAR GRIS command has a slightly different action from cancelling most commands in the Drafting System. The grid lines are removed, but the graphics are not redrawn to reflect this. This is also a feature of the PARS and PARS CAN commands, described in “Parameterizing Geometry” on page 89 .

Commands Between PAR GRIS and CAN

You can use only a limited number of commands between any of the PAR GRIS options and the CAN command. The permitted commands belong to the following categories. To find out more about these commands refer to the *MEDUSA Bacis1 Design Commands Guide* or *MEDUSA Bacis1 Guide*.

Category	Command
Drawing commands	CLE, DRA, REDRA
Windowing commands	WIN, PAN, ZOO
Parametric switch commands	PAR SWI
Bacis1 internal commands	QVAR

Please note: If you use any commands other than these, the CAN command will not be

effective and grid lines will be placed permanently on the sheet. The same categories of command may be used between the PARS command and CAN.

Grid Line Types and Layer Defaults

Grid lines created with the PAR GRIS command have the following element types:

Grid Command	Line Type	Layer
PAR GRIS OLD,FIL,POT	STK, width 3 mm (0.118 inch)	99
PAR GRIS COL,CIR,TAN,INF	L3	17
PAR GRIS NEW	L4	99
PAR GRIS BAS	LBL	16

You can change the default line types and layers used for parametric grid lines using the PAR DDL command, which is described in ["Changing Element Types"](#), ["Changing Element Types - PAR DDL"](#) on page 154.

Making Grid Lines Untransformable

Note that the end points of lines created with the COL, CIR, TAN, and INF options of the PAR GRIS command may not be at grid line intersections. Make sure that you either delete them or make them untransformable before you parameterize the drawing. You can easily do this using an in-sheet layer command, for example `LAY 17 UNTRN`. Refer to ["Switches and Layers"](#) on page 95, for more information on how to change layer properties.

Adding Lines

The PAR GRIS command has four options which provide a quick way of adding lines to your drawing to help to place all required points on the grid. These options; INF, COL, CIR, and TAN, are described below.

The line segments added with the INF, COL, CIR and TAN options are line type L3 and are placed on layer 17. It is important that you make the lines untransformable before you try to parameterize the sheet. If the end points of these extra lines are not at grid line intersections, the following error message will appear at each point during parameterization:

Point not dimensioned

To prevent this, make layer 17 untransformable using a layer command as described in [“Making Grid Lines Untransformable”](#) on page 70.

PAR GRIS INF

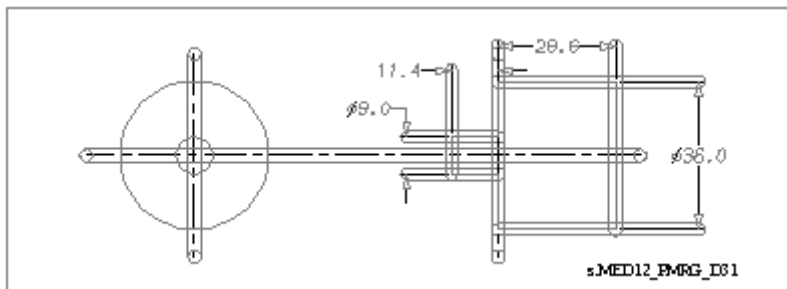
This command adds the following line segments:

- Straight lines along each line segment in the viewbox, extending to the edges of the viewbox
- Complete circles for every arc in the viewbox

Example

The following figure shows two views of the same object. The relationship between the two views must be maintained during parameterization. For example, if one of the diameter dimensions in the right-hand view changes, then the corresponding diameter in the left-hand view should change by the same amount.

Figure 50 Default Grid Setting



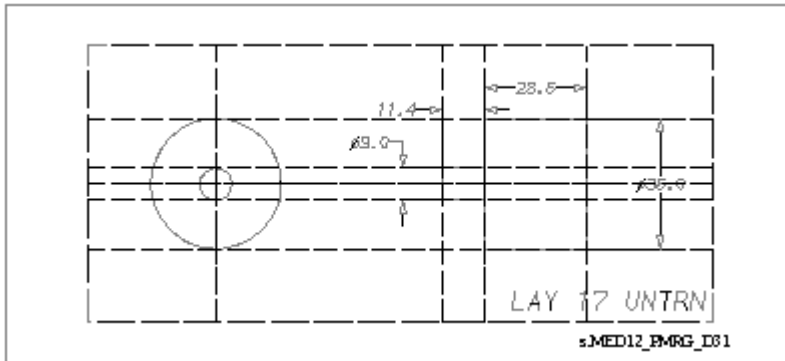
Displaying the grid lines as in the above figure reveals that with the horizontal grid lines generated by the 9.0 and 36.0 diameter dimensions to not extend to the circles in the left-hand view. Dimensioning the left-hand view separately would place the circle onto the grid but:

- The information specified by the two existing diameter dimensions would be repeated

- The system would fail to identify the relationship between the two views: only by identifying this relationship can the system impose the necessary geometric constraints during parameterization

The grid lines can be extended using PAR GRIS INF. This generates infinite lines that extend along the length of the horizontal grid lines to the edges of the viewport and connect the two views. The result is shown in the following figure.

Figure 51 Infinite Lines Added With PAR GRIS INF



PAR GRIS INF or Unlimited Grids

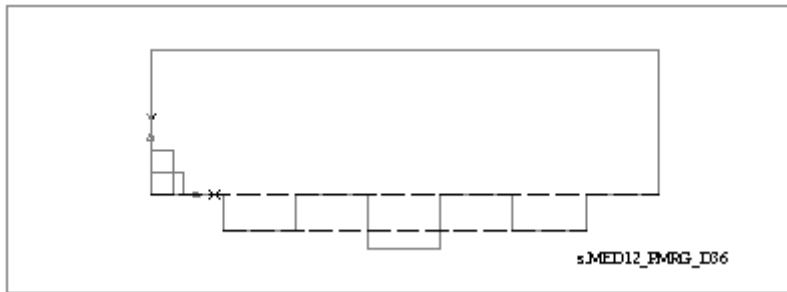
Although you could use unlimited grids (PAR LIM OFF) to extend the grid lines from the right-hand view to the circles in [Figure 50, “Default Grid Setting” on page 71](#), the necessary geometric constraints will not be automatically inferred. Using PAR GRIS INF has the advantage that it:

- Does not add lines for each zero length grid line while, with unlimited grids, full grid lines would be generated for each zero length grid line found
- Allows you also to use automatic inference of baselines (PAR BAS ON) and default fillet settings (PAR FIL), which are not possible with limited grids

PAR GRIS COL

This command adds lines connecting all non-overlapping collinear line segments. PAR GRIS COL adds all possible lines: you may want to delete lines that you do not require. In the following figure, several collinear line segments have been linked using PAR GRIS COL.

Figure 52 Collinear Lines Added With PAR GRIS COL



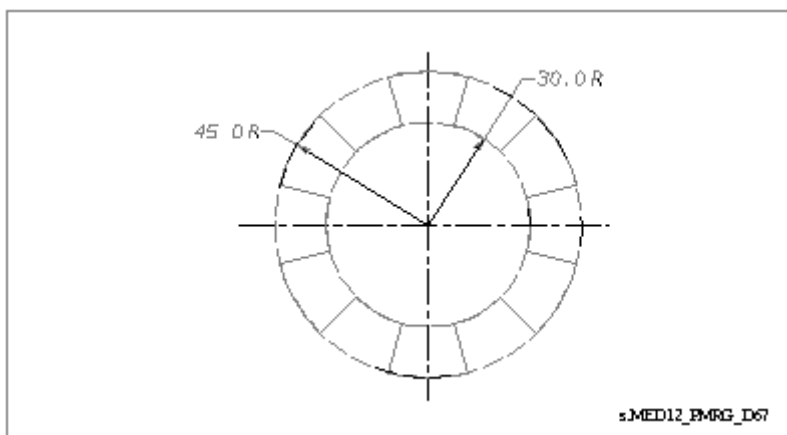
When to Use PAR GRIS COL

This command offers more control than simply turning the `PAR COL` switch ON, as you can take different decisions for different sets of collinear lines. If you use the `PAR COL` switch, the `PAR GRIS COL` command is still useful to show graphically the assumptions that are being made. Sometimes you will discover that lines that were meant to be collinear are not collinear within the parametric tolerance, or vice versa.

PAR GRIS CIR

This command adds a circle for each arc in the viewbox. This can be useful where geometry contains several arcs with the same center and radius. For example, in the following figure only the arcs crossed by the radial dimension lines are supported on the grid. To avoid having to dimension each arc individually, circular L3 lines linking arcs with the same center and radius have been added with `PAR GRIS CIR`.

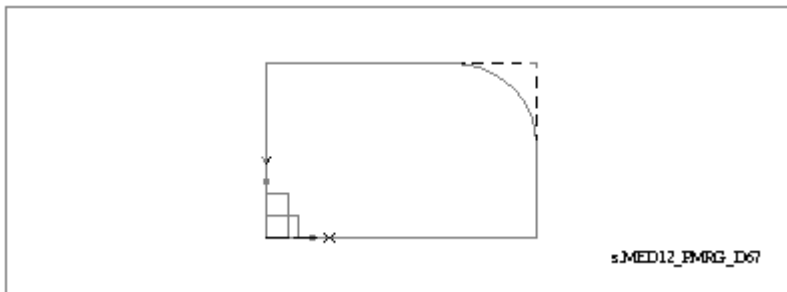
Figure 53 Circles Added With PAR GRIS CIR



PAR GRIS TAN

This command adds tangential lines to the tangent point of all tangent point arcs in the viewport, as shown in the following figure. This is useful if the `PAR TAN` switch is `OFF`, when grid lines extend along the arc of tangent point arcs but not to the tangent point itself.

Figure 54 Tangential Lines Added With PAR GRIS TAN



Grid Tolerance

The Parametric Design system uses a tolerance to test for intersections and other conditions. The grid is extended by this tolerance to allow for inaccuracies in the drawing. This means that zero length grid lines appear as short lines twice the length of the parametric tolerance.

Using the Grid Tolerance: The tolerance is used primarily as a test distance; points in the drawing must lie within the tolerance of an old grid line intersection, so that they can move to the new grid line intersection. Overlapping line segments must be within each other's set tolerance.

Changing Grid Tolerance

You can change the grid tolerance with the PAR TOL command. The default value is 0.1 mm (0.00394 inches), which should be suitable for most drawings, but you may find that you need to exaggerate some details. For example, almost collinear lines should be drawn at a significant angle if you do not intend them to be collinear.

If the system does not find points to be collinear that you intend to be collinear, then you may need to increase the tolerance. Conversely, you may need to decrease the tolerance if you find that points not intended to be collinear are being recognized as collinear. However, it is generally better to adjust the drawing to avoid such problems, since changing the tolerance may upset other parts of the drawing. The default value has been found to work best in most cases.

The PAR TOL Command

You can set parametric grid tolerance with the PAR TOL command. If you give a value of zero, the tolerance will be set to the default of 0.1 mm.

PAR TOL command syntax is shown below:

| → PAR TOL <tolerance> →

Variable	Description
<i>tolerance</i>	The tolerance required. This must be a real value expressed as a decimal, a fraction, or an expression.

Q PAR TOL

Use Q PAR TOL to query the current parametric tolerance value. The command syntax is shown below:

|→ Q PAR TOL →

VARIABLES AND EXPRESSIONS

This chapter describes how MEDUSA Parametric Design evaluates the parameters you supply in dimension clumps and moves the parametric grid in accordance with these values. New parameters may be simple numeric values, or variables or expressions that evaluate to simple numeric values.

- Values in Dimensions 78
- Replacing Dimension Text 80
- Variables 81
- Variable Scope Restrictions 82
- Expressions 83
- Operators and Functions 85
- Output Dimensions 87

Values in Dimensions

The Parametric Design system uses the parameters you supply to calculate the new position of the parametric grid intersections. You specify parameters for Parametric Design by replacing the original dimension value with a simple numeric value, or a variable or expression that evaluates to a simple numeric value. Before parameterization, you must specify each parameter using one of the following methods:

- Directly replacing the value of the dimension text
- Setting a variable and giving it a value either directly or using an expression, for example:

```
LET A = 60  
LET B = A/2
```
- Specifying a set of values from a table and using these values to replace variables on the sheet

Replacing Dimension Values

The values in dimensions are part of the dimension. When you change the text for use in Parametric Design, it must remain part of the dimension. Therefore you should not delete the old text and place new text in the same position; you should replace the value of the original text.

Avoiding Ambiguity in Dimension Texts

When you replace dimension values with your own parameters, be careful which part of the dimension text you replace. Characters that do not represent values should be PRE and POS texts rather than main dimension texts, for example, R, DIA and j. PRE text is placed before and POS text is placed after the numeric value in the DIM clump.

If PRE and POS texts are confused with the dimension value, the dimension text becomes potentially ambiguous. For example, radial dimensions, such as R21, are often problematic. R21 is valid both as a variable name *R21*, or as the value 21 with PRE text of R. In cases such this, the dimension text is always interpreted as a simple value with a PRE text. The following message will be displayed on the screen to warn you:

```
Warning  
ambiguous dimension texts interpreted as values
```

If you intend it to be a variable, enclose the name with parentheses (...). For example, use (R21) for the variable name R21.

Other Dimension Text Formats

Apart from the simple format of a number with or without decimal places, two other formats are allowed in dimension texts. These are degrees, minutes, and seconds for angular dimensions, and feet and inches for imperial units. These formats can also have any PRE and POS texts.

Dimension Text Format After Parameterization

The Parametric Design system only uses the dimension value to parameterize geometry. PRE and POS text is ignored during parameterization and you can delete it if you want to make the definition drawing clearer. Note however that the dimensioning on the new drawing will appear in the style used originally; the value or expression used to specify the new value does not affect the final format, which will have PRE and POS texts corresponding to the original drawing, even if you deleted them before parameterization.

How Dimension Texts Are Evaluated

Dimension text is evaluated in the following ways to produce a simple numeric value which can be used as input for Parametric Design:

- Any PRE text is stripped from the beginning of the text string. If this leaves a simple numeric value then this value is used. For example, R15.5, becomes 15.5. Obviously 15.5 with no PRE or POS text is also valid.
- Any variables or expressions are evaluated to yield a simple numeric value, for example `LENGTH`, `ANGLE* .6`.

Replacing Dimension Text

The new geometry is calculated from the parameters you supply when you replace the original dimension values with new ones. You can replace dimension value text with a different numerical value, for example 55.0, 17.25, 2, or with a variable or an expression.

Procedure

To replace a value in a dimension use the following procedure:

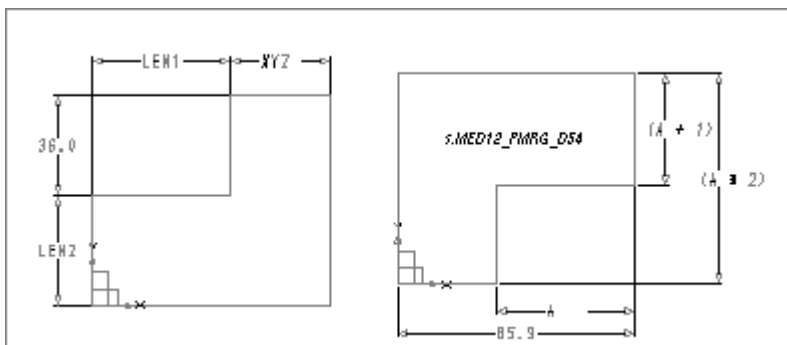
1. Put the new value into the text buffer.
2. Make the dimension text to be replaced current.
3. Substitute the contents of the text buffer for the current text using the GOBC command.

Repeat the procedure until you have replaced all the values required.

Variables and Expressions in Dimension Clumps

In the following figure, some of the original dimension values have been replaced with variable names or expressions. To do this, place the variable name into the text buffer, for example LEN2 or XYZ, and then replace the dimension text using the procedure described above.

Figure 55 Variables in Dimensions



The variables and expressions must evaluate to a single numeric value during parameterization. The variables A, LEN1, LEN2, and XYZ must be specified before parameterization using the LET command.

A more complex example of how expressions can be used, involving logical operators, is given later in this chapter in [“Example” on page 84](#).

Variables

Once you have placed variables in dimension clumps you assign a value to each one using the LET command. You can use the LET command either interactively or as an in-sheet command.

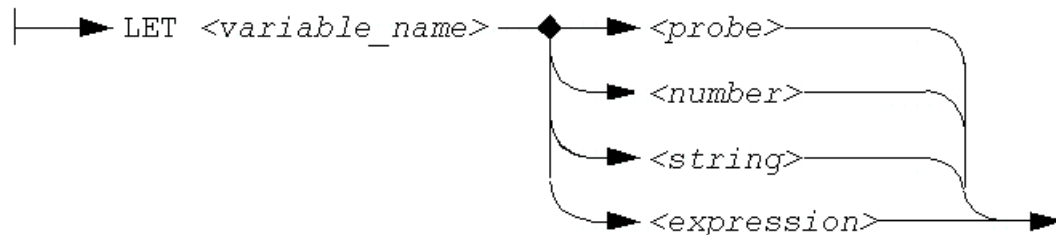
To enter values from the keyboard, type the whole command string, and then press the `Return` key. For example:

```
*LET A = 45
*LET LONG = 100
```

In-sheet LET Commands

You can also place a LET command inside the viewbox as an in-sheet command, text type TCO. The commands are executed in order from top left to the bottom right. If the order of execution is important, it is best to arrange LET commands in a vertical column in the appropriate order. In-sheet commands are described in ["Overview of Parametric Design"](#), ["Giving Commands in Parametric Design"](#) on page 17.

LET Command Syntax



Displaying Current Variable Values

Option	Description
<i>variable_name</i>	The name of the variable. The name can be up to six characters, but the first character must be alphabetic. Note that variable names must not contain spaces.
<i>probe</i>	A positional probe, assigning a set of X and Y sheet coordinates to the name <i>variable_name</i> . Use this to position attachment point texts during parametric symbol loading.
<i>number</i>	A numeric value. The number specified must be a signed real value expressed as a decimal, a fraction, or an exponential.
<i>string</i>	Any text string enclosed in single quotes ('...').
<i>expression</i>	An expression that evaluates to a simple numeric value, for example: LENGTH*(WIDTH/2)

You can query the value of variables and expressions you have created with the QVAR command.

Variable Scope Restrictions

Any variable that exists before parameterization may be accessed at any time. However, there are restrictions on the scope of variables created using the following methods:

- In-sheet commands (text type TCO)
- Output dimensions
- Parametric symbols

It may not be possible to access variables created using these methods as they exist only temporarily for the duration of the PARS or PAR LOA commands. If you need to access such a variable, in a Bacis1 program, for example, then the variable must exist as a protected variable before parameterization.

Protected Variables

A protected variable:

- Can be accessed at any time, no matter what the current operation
- Cannot be deleted with the DELM command or overwritten with the DEFM command
- May be edited using the EDIM command or overwritten using the LET command

Creating a Protected Variable

The DEFM command creates a protected variable. DEFM has the following syntax:

| —▶ DEFM —▶ *<variable_name>* —▶ *<string>* —▶

Argument	Description
<i>variable_name</i>	Is the name of the variable being generated. It can contain from one to six alphanumeric characters, the first of which must be alphabetic.
<i>string</i>	Is the text string the variable contains. Text strings that contain spaces must be bounded by either single or double quotes. For example: DEFM BUF1 "this string has spaces" DEFM BUF2 thisstringhasnospaces

Numeric variables: To create a protected variable which contains a number, you must use a slightly different procedure. You first create the protected variable using the DEFM command, then use the LET command to assign a numeric value to it. For example:

```
DEFM FRED ''  
LET FRED = 42.5
```

You can find out more about variables and expressions, together with the commands for their manipulation, in the *MEDUSA Bacis1 Guide*. The DEFM command is described in detail in the *MEDUSA Bacis1 Design Commands Guide*.

Expressions

Using expressions, you can express relationships between different dimensions of the same object. For example, you may know that the length of a component is always twice its width. Using an expression, you can define this relationship, which will be maintained whenever the object is parameterized.

The DEF Command

The DEF command generates a variable containing an unevaluated expression. The expression is only evaluated when the variable is referenced, when the current values of the other variables will be used. You can use DEF both interactively and as an in-sheet command.

Command Syntax

The syntax for the DEF command is:

| —▶ DEFM —▶ *<variable_name>* —▶ *<expression>* —▶

Option	Description
<i>variable_name</i>	Is the name of the variable being generated. It can contain from one to six alphanumeric characters, the first of which must be alphabetic.
<i>expression</i>	Is an unevaluated expression, enclosed in single or double quotes. The value of expression is evaluated each time the variable is referenced using the current values of any variables named in the expression. Variables named in the expression need only be present when the expression is evaluated.

Each DEF command must begin on a new line. Examples of DEF commands are:

```
DEF SHORT = 'LONG/10'
DEF XYZ = "WIDTH + (A.GT.B)*LENGTH*SIN(3*a)"
```

The DEF command is described in detail in the *MEDUSA Bacis1 Design Commands Guide*.

Brackets in Expressions

Use rounded brackets to enclose those parts of the expression to be calculated first. Brackets can be nested to any level. On all MEDUSA platforms, expressions can be enclosed in round

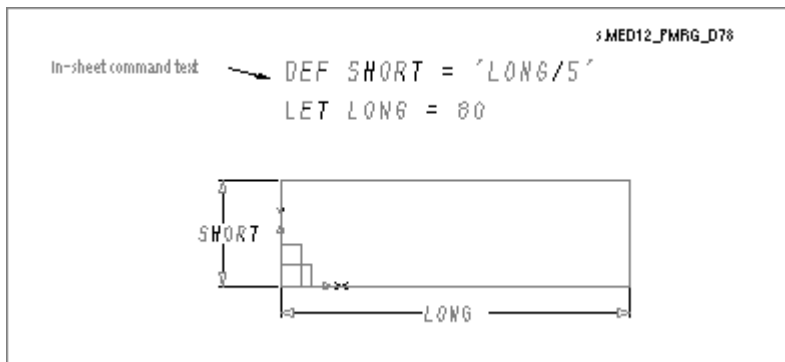
brackets ((...)) or angle brackets (<...>). Angle brackets cannot be used within expressions, as the following example shows:

Legal	Illegal
<LEN + 10>	<A*<B-4>>
<A*(B-4)>	(A*<B-4>)

Example

The following example shows a drawing where in-sheet commands define the variables *LONG* and *SHORT*. The variable *SHORT* contains an unevaluated expression, *LONG/5*. The expression is only evaluated when the variable *SHORT* is referenced, when the current values of the *LONG* will be used.

Figure 56 Using Variables and Expressions



Operators and Functions

MEDUSA has a variety of arithmetical and logical operators that you can use to create expressions and conditions to be passed as variables during parameterization.

Arithmetic Operators

You can use the following arithmetic operators:

Symbol	Operation
+	Add
-	Subtract
/	Divide
*	Multiply
**	Raise to the power of

You can use any of these arithmetic operators in expressions. For example, this is an interactive sequence:

```
*DEF X = (A + B)
*DEF Y = '(B ** 2) - (X / 2) '
*DEF Z = '3 * X'
*PARS CAN
*
```

Logical Operators

You can include simple conditional statements in expressions using one or more of the following logical operators:

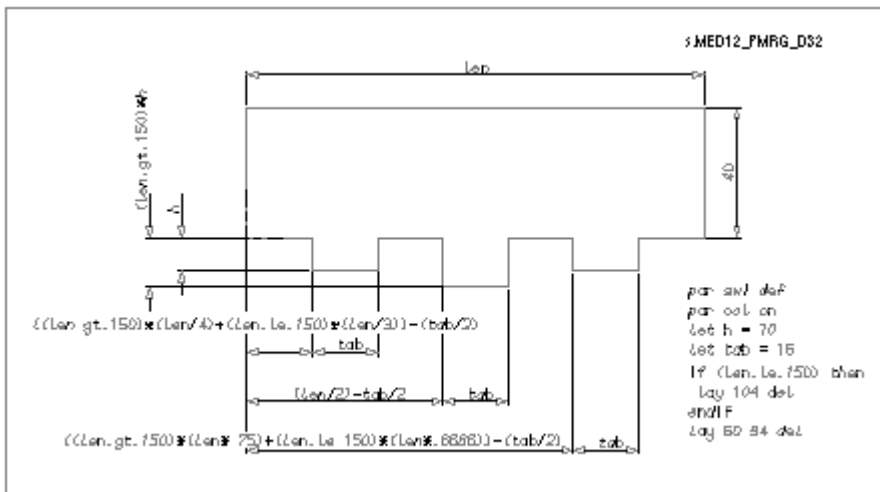
Symbol	Operation
.AND.	And
.OR.	Or
.NOT.	Not
.LT.	Less than
.LE.	Less than or equal to
.EQ.	Equal to
.GT.	Greater than
.GE.	Greater than or equal to

These operators work by being set to either 0 or 1. If the expression is `TRUE` then the operator is set to 1. If the expression is `FALSE` the operator is set to 0.

Example

In the following drawing, two tabs are required when *len* is less than or equal to 150, otherwise three tabs are needed. To ensure this, one of the dimensions is set to $(len * .75) - (tab/2)$ when *len* is greater than 150, but to $(len * .6666) - (tab/2)$ when *len* is less than or equal to 150.

Figure 57 Drawing Using Logical Operators



This is done by replacing the dimension text with the following expression:

$$((len.gt.150)*(len*.75) + (len.le.150)*(len*.6666)) - (tab/2)$$

When *len* is greater than 150: For example, if *len* is equal to 160 and *tab* is equal to 16 then:

$$\begin{aligned} \text{dimension} &= (1*(len*.75) + 0*(len*.6666)) - (tab/2) \\ &= (len*.75) - (tab/2) \\ &= 112 \end{aligned}$$

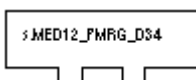
The result is shown below.



When *len* is less than 150 : If *len* is equal to 135 and *tab* is equal to 16 then:

$$\begin{aligned} \text{dimension} &= (0*(len*.75) + 1*(len*.6666)) - (tab/2) \\ &= (len*.6666) - (tab/2) \\ &= 82 \end{aligned}$$

The result is shown below.



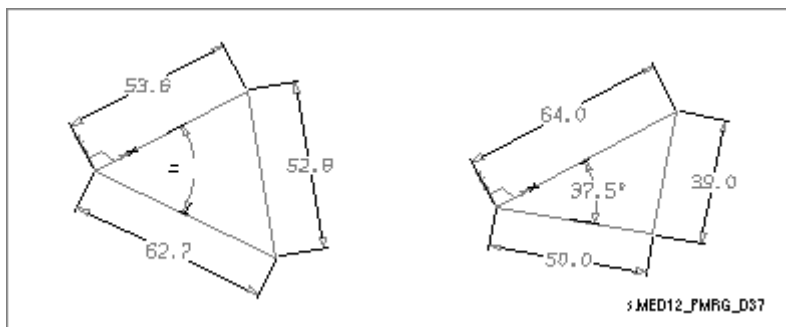
Output Dimensions

Output dimensions display values for information. Such dimensions are not necessary for specifying the size and shape of the object. To make an output dimension, replace the dimension text with an equals sign (=). The original dimension format is used when the final value is output.

Example

In the following figure, the triangle is completely specified by the three chain dimensions. The angular dimension is an output dimension. The result after parameterization is shown on the right, where the angle has been calculated and put into the dimension.

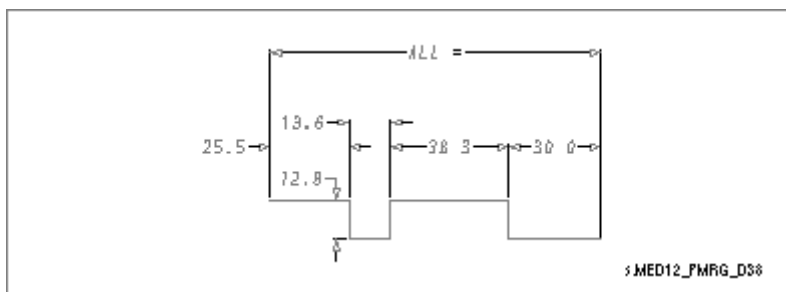
Figure 58 Output Dimension



Variables in Output Dimensions

You can use a variable to hold an output dimension value by replacing the dimension text with a variable name followed by an equals sign, for example: ALL = .

Figure 59 Output Dimension With Variable



During parameterization the value of the variable ALL in the above figure will be calculated from the other dimension values and this number will replace the variable in the parameterized drawing.

Uses: Output variables are often used in programs that executes the PARS command. Output variables in programs must be protected variables.

PARAMETERIZING GEOMETRY

This chapter describes how the parameterization command PARS works.

- [Overview of Parameterization](#) 90
- [PARS and PARS CAN](#) 92

Overview of Parameterization

When you have prepared the definition sheet, you can parameterize the geometry using the PARS command. Until you are confident that your drawing is correct, it is best to ensure that parameterization is temporary by cancelling the PARS command immediately with CAN.

What Happens When You Give the PARS Command

When you execute the PARS command, the system performs the following operations in each viewbox on the sheet:

1. Executes in-sheet commands.
2. Generates the parametric grid.
3. Replaces dimension values with new parameters.
4. Moves any points of the geometry.
5. Deletes specified elements.
6. Processes instance clumps, if any, and loads parametric symbols.
7. Restores any variables, switches or layer properties changed by in-sheet commands to their pre-parameterization values.

Executing In-sheet Commands

Any in-sheet commands you have placed inside the viewbox are executed, for example, setting variables or changing switch settings.

Generating the Grid

The potential, OLD and NEW grids are generated using the reference point, geometry, and dimensions. Dimensioned points are placed onto grid intersections then the system checks to see if the dimensioned points can be moved successfully using the new parameters.

Replacing Dimension Values

The original dimension values are replaced by the new parameters you have specified. These may be simple numeric values, or evaluated variables or expressions.

Moving Points

Dimensioned points in the object geometry are moved in the following ways:

Lines : Each point in the line must lie at a grid line intersection, and points are moved with the associated grid intersection.

Arcs : The weight at the tangent point of a tangent point arc is transformed so that arcs of circles continue to be arcs of circles. Arcs of ellipses created by the ELL command are also preserved. For other arcs, the weight is transformed by the ratio of the weights needed to give ELL type arcs.

Text and prims: If the datum point of a text or prim element lies at a grid intersection, it is moved with the intersection. Texts and prims that are not placed on the grid are left unmoved. Those that lie on grid lines, but not at a grid intersection, are moved with the grid line provided that the grid line intersects other grid lines on both sides of the text or prim. The movement is a proportioned translation, that is, the ratio of the distances from the text or prim to the nearest grid intersections on each side is preserved.

Deleting Elements

After moving the object, elements on layers that can be deleted are deleted from the sheet.

Loading Parametric Symbols

Any instance clumps in the viewbox are processed and the specified symbols are loaded and parameterized. For detailed information about parametric symbols, refer to [“Parametric Symbols” on page 111](#).

Restoring Values

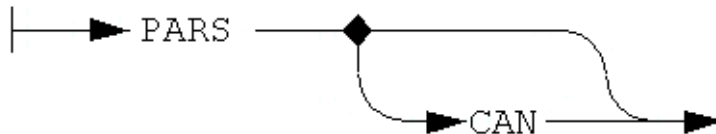
Any changes made by in-sheet commands are reversed. For example, variables, switches, and layer properties restored to the values they held before parameterization.

PARS and PARS CAN

PARS parameterizes all geometry enclosed in a parametric viewbox. The contents of each viewbox are scaled according to the dimensions present. Each viewbox is processed independently.

By cancelling PARS immediately with CAN, you can parameterize geometry temporarily. This allows you to check the accuracy of new parameters without changing the geometry permanently. You can then alter the definition if necessary.

Command Syntax



Option	Description
CAN	Cancels the parameterization command. Unlike cancelling most 2D Design commands, cancelling PARS will not undraw the new graphics and redraw the old graphics. The graphics corresponding to the parameterized geometry are displayed until you refresh the screen with the REDRA command.

Using PARS CAN

Always use PARS CAN unless you are quite sure that your drawing is ready to be parameterized permanently. If the geometry is parameterized permanently:

- You will not be able to go back and change parameters if they are wrong
- You may not be able to parameterize geometry that is itself the result of parameterization

If you use PARS CAN you will avoid any danger of accidentally forgetting to cancel the parameterization.

Commands Between PARS and CAN

You can use only certain commands between PARS and CAN. The permitted commands belong to the following categories. To find out more about these commands refer to the *MEDUSA Bacis1 Design Commands Guide* or *MEDUSA Bacis1 Guide*.

Category	Command
Drawing commands	CLE, DRA, REDRA
Windowing commands	WIN, PAN, ZOO
Parametric switch commands	PAR SWI
Bacis1 internal commands	QVAR

If you use any commands other than these, the CAN command will not be effective and parameterization will be permanent. The same categories of command may be used between the PAR GRIS command and CAN. Permanent parameterization

Always save the sheet before parameterizing geometry permanently. It is not always easy to reverse the effect of PARS once the opportunity to cancel parameterization has passed. Remember, if you enter commands after PARS, you may not be able to cancel the parameterization. Use the EXIT command immediately after PARS to explicitly remove the possibility of cancelling.

Parameterization Errors

Appendix D provides a list of possible error messages. Error messages are written onto the sheet on layer 99. Errors which occur when using Parametric Design fall into the following categories:

- Errors relating to the old grid, such as problems in the definition geometry. These are text type TS1
- Errors relating to the new grid lines, such as problems in the new parameter values. These are text type TR1
- Errors relating to 2D Drafting, such as typing errors

Associated error lines are type L6. If error messages are written onto the sheet, a warning message is output to the screen in the normal way.

2D Design Errors

In addition to error messages from the Parametric Design system, any relevant 2D Design error message can appear in the sheet or on the screen. Refer to the *MEDUSA Basis1 Design Commands Guide* for more information on these error messages. Drawing the grid is often very helpful in finding errors in the drawing. See [“The Parametric Grid” on page 59](#) for more information on drawing grid lines.

SWITCHES AND LAYERS

This chapter describes how you can change the way geometry is parameterized by changing various parametric switches settings and layer properties.

- [Parametric Switches](#) 96
- [Setting Parametric Switches](#) 98
- [Parametric Switch Examples](#) 101
- [Layers](#) 104
- [Layer Properties](#) 105
- [Changing Layer Properties](#) 107

Parametric Switches

The switches outlined below affect the operation of the MEDUSA Parametric Design system.

PAR BAS

When `PAR BAS` is `ON`, the system uses the potential grid lines to infer geometric constraints automatically. Constraints are not inferred automatically when the `PAR LIM` switch is `OFF`. For more information see ["Geometric Constraints"](#), ["Automatically Inferred Constraints"](#) on page 55 which describes how to use the `PAR BAS` switch to control inferencing of geometric constraints.

PAR COL

When `PAR COL` is `ON`, potential grid lines span the gaps between non-overlapping collinear straight line segments. Provided that you want the segments to remain collinear, you do not then have to dimension each of the line segments separately.

PAR CIR

The `PAR CIR` switch affects the generation of potential grid lines from arcs of circles. If the switch is `ON`, then complete circular grid lines are generated for all arcs in the viewbox. This enables you to place all arcs with the same center and radius onto the grid without dimensioning each one separately.

PAR LIM

The `PAR LIM` switch makes `OLD` grid lines either limited or unlimited. Limited grid lines drawn along lines in the drawing, and do not extend beyond the end of the geometry. Unlimited grid lines are limited only by the edges of the parametric viewbox. For more information see ["The Parametric Grid"](#), ["Limited Grid Lines"](#) on page 66 which describes limited and unlimited grid lines and gives examples.

PAR MOV

When the `PAR MOV` switch is `OFF`, dimension texts are replaced by new parameters but the original geometry is not transformed to reflect the new parameters. If `PAR MOV` is `OFF` when loading parametric symbols, the symbol will be loaded at the same position it had in the definition sheet.

PAR PRE

When the `PAR PRE` switch is `OFF`, dimensioning within parametric groups is ignored and points move with the group. For more information see ["Parametric Groups"](#), ["The PAR PRE Command"](#) on page 149 which describes the `PAR PRE` switch in more detail.

PAR SUP

PAR SUP ON enables newer forms of dimension support on Prime and VAX systems.

PAR TAN

When the PAR TAN switch is turned OFF, the tangent lines of tangent point arcs are not used to create the potential grid. The default setting for PAR TAN is ON because the tangential grid lines are almost always needed to place the tangent points of tangent point arcs at grid intersections.

PAR TEX

When PAR TEX is OFF, geometry is transformed according to the new parameters in the dimension clumps but the dimension texts display their original values after parameterization.

PAR UND

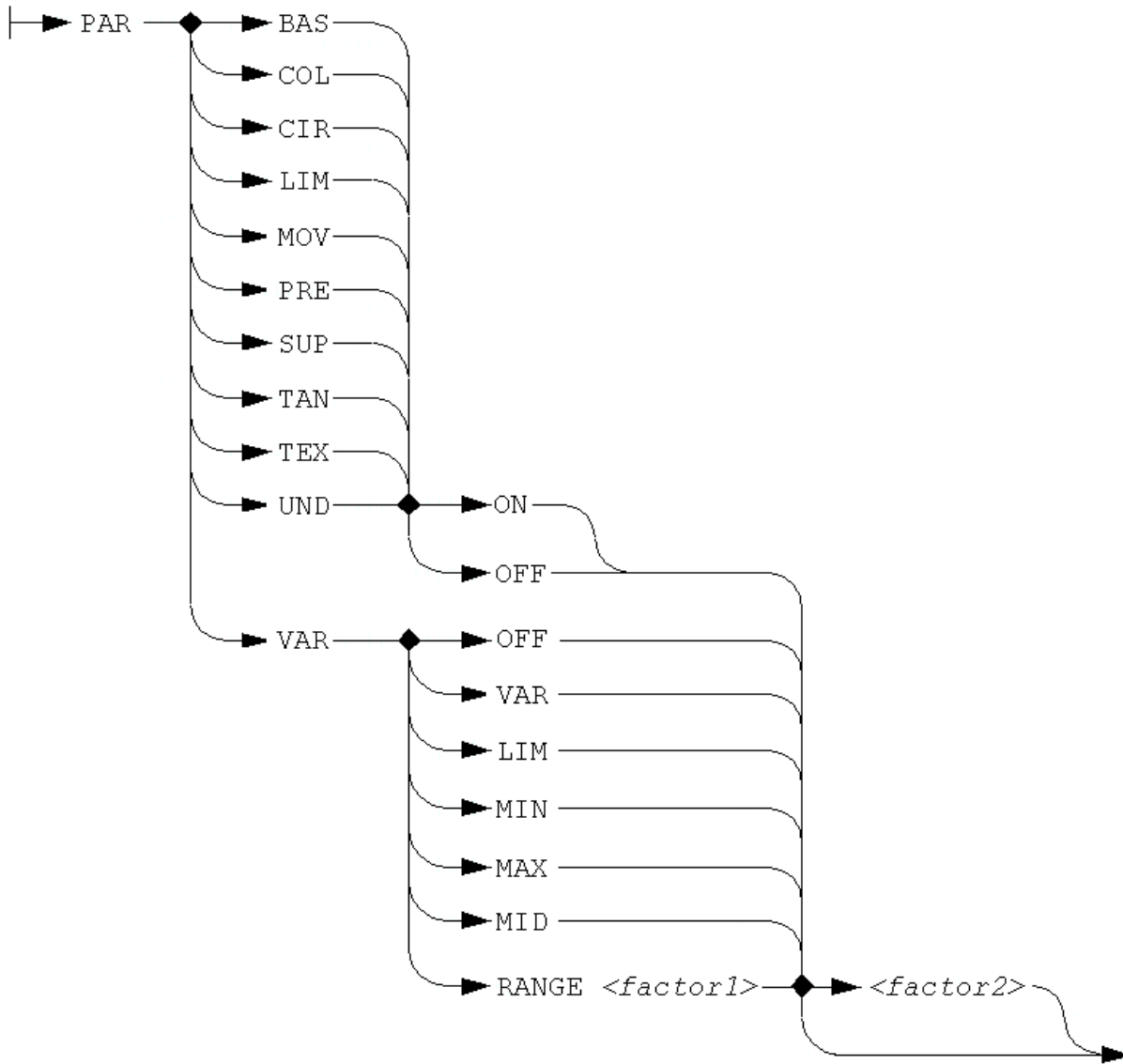
The parametric switch PAR UND controls undrawing (erasing) of the geometry during parameterization. The default setting for this switch is ON, so that when geometry is parameterized, the original image is removed before the new image is drawn. When the PAR UND switch is OFF, the old image remains visible. This switch is used in mechanism simulations, allowing you to see all stages of movement at the same time. For more information see "[Mechanisms](#)", "[Undrawing the Image](#)" on page 165 that describes using PAR UND OFF in mechanisms simulations.

PAR VAR

The PAR VAR switch controls the way the Parametric Design system evaluates tolerance variation dimensions. This switch is described in detail under the section "[Dimensioning](#)", "[PAR VAR Command Syntax](#)" on page 42.

Setting Parametric Switches

The graph below shows the command syntax for changing parametric switch settings. The options are explained below and opposite, and some examples are given on the following pages.



Option	Default	Explanation
BAS	ON	When the PAR BAS switch is ON, geometric constraints are automatically inferred, provided that the PAR LIM switch is also ON. Geometric constraints are described in “Geometric Constraints” on page 47 .
COL	OFF	When PAR COL is ON, grid lines extend along non-overlapping collinear line segments. An example is shown under “The PAR COL Switch” on page 101 .

Option	Default	Explanation
CIR	OFF	When PAR CIR is ON, complete circular grid lines are generated from circle arcs. An example is shown under “The PAR CIR Switch” on page 102.
LIM	ON	When PAR LIM is ON, grid lines do not extend further than the points in the object geometry. When PAR LIM is OFF, grid lines are unlimited and extend to the edges of the viewbox. This switch is explained under section “The Parametric Grid” , “Limited Grid Lines” on page 66.
MOV	ON	When PAR MOV is OFF, dimension texts are replaced by new parameters but the original geometry is not changed.
PRE	OFF	The PAR PRE switch affects the way in which points within parametric groups move during parameterization. This switch is described in detail under section “Parametric Groups” , “The PAR PRE Command” on page 149.
SUP	ON	Enables newer forms of dimension support on VAX and Prime platforms.
TAN	ON	When PAR TAN is ON, then the tangent lines from tangent point arcs are included in the potential grid. An example is shown in “The PAR TAN Switch” on page 103.
TEX	ON	When PAR TEX is OFF, geometry is transformed but dimension texts display their original values after parameterization.
UND	ON	When PAR UND is ON, the original geometry is undrawn during parameterization. This is done before drawing the new object. When PAR UND is OFF, the original definition remains visible on the screen after parameterization.
VAR	VAR	The PAR VAR switch controls the way the Parametric Design system evaluates tolerance variation dimensions. This switch is described in detail under section “Dimensioning” , “PAR VAR Command Syntax” on page 42.

Using In-sheet Commands to Set Switches

The best way of setting switches is using in-sheet commands. Use the following procedure:

1. Select new text of type TCO.
2. Place the command into the text buffer, for example: `/PAR TAN ON`
3. Position the text on the sheet using a probe.

Querying Switch Settings

Use the Q PAR SWI command to find out the current parametric switch settings. The syntax for this command is:

```
| → Q PAR SWI →
```

Resetting Switch Defaults

Use the PAR SWI command to reset all parametric switches to their default settings. The syntax for this command is:

```
| → PAR SWI → ◆ → DEF →  
| → PAR SWI → ◆ → OLD →
```

Option	Explanation
DEF	Resets switches to the following defaults: ON -->LIM, BAS, TAN, MOV, TEX, SUP, and UND OFF --> COL and CIR
OLD	Resets switches to default settings compatible with pre-6.0 CIS MEDUSA software. In particular, it sets the LIM, BAS, SUP and TAN switches to OFF.

Undrawing Graphics

When the 2D Design graphics switch is turned OFF using the SWI GRA command, then the definition object is not undrawn and the new object is not drawn after parameterization. See the *MEDUSA Bacis1 Design Commands Guide* for more information on this switch.

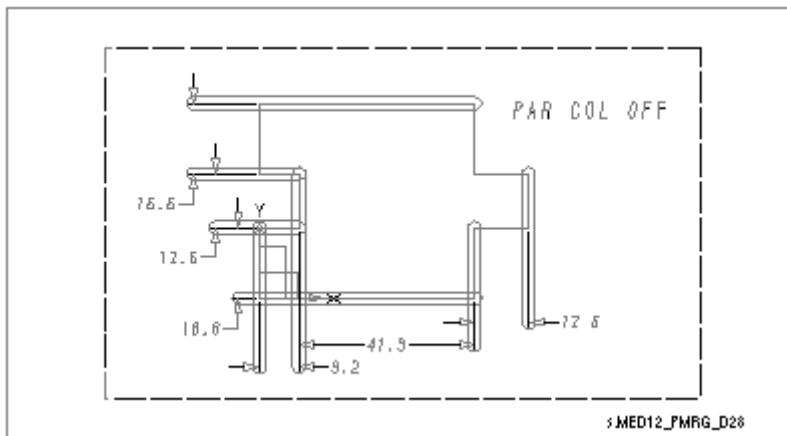
Parametric Switch Examples

The following examples show the effect of the switches `PAR COL`, `PAR CIR` and `PAR TAN`.

The `PAR COL` Switch

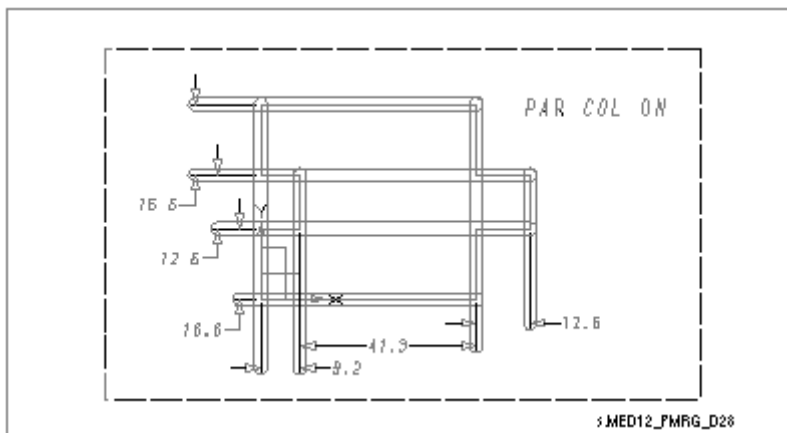
In the following example, the grid lines do not extend from the dimensioned line segments to collinear line segments.

Figure 60 Effect of `PAR COL OFF`



In the following figure, the in-sheet command `PAR COL ON` ensures that the grid lines extend across all collinear line segments without having adding any further dimensions.

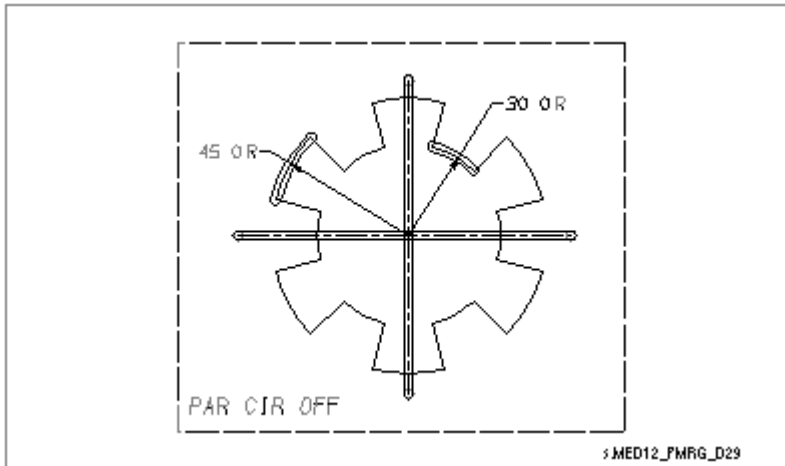
Figure 61 Effect of `PAR COL ON`



The PAR CIR Switch

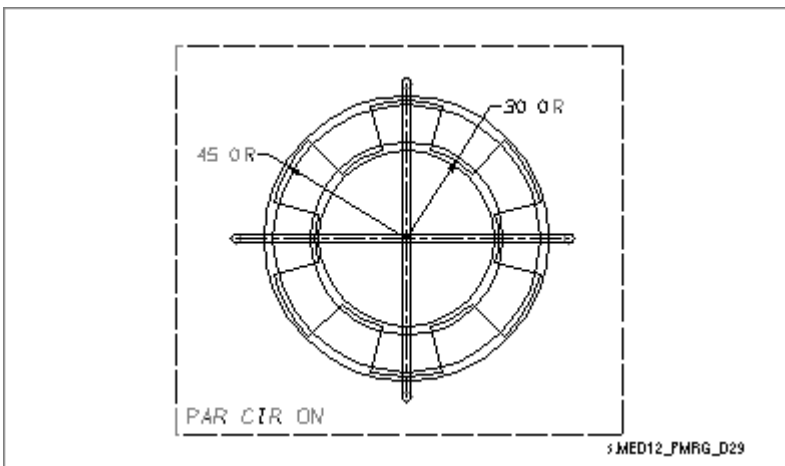
In the following figure radial dimensions generate grid lines along only the dimensioned arcs: the other arcs with the same center and radius are not placed onto the grid.

Figure 62 Effect of PAR CIR OFF



In the following figure the effect of the in-sheet command PAR CIR ON is to generate a circular grid line which places all arcs with the same center and radius onto the grid.

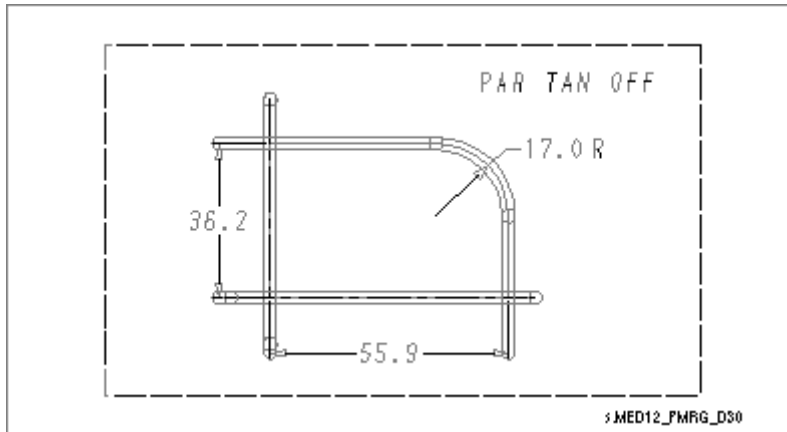
Figure 63 Effect of PAR CIR ON



The PAR TAN Switch

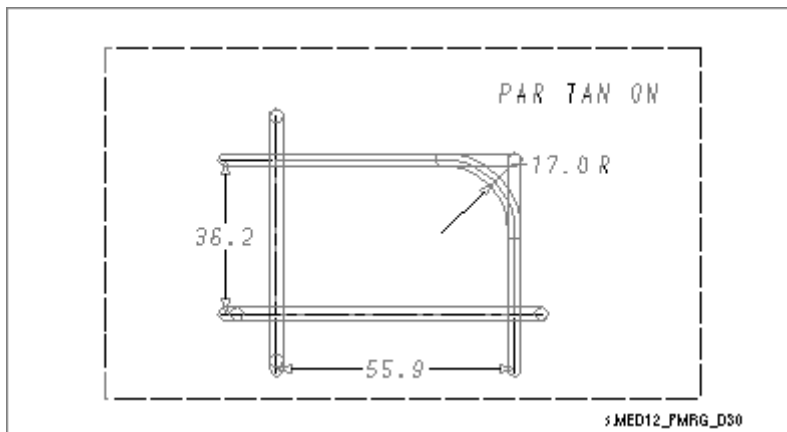
In the following figure, although the arc is supported on the grid, the tangent point is not because PAR TAN is set to OFF by an in-sheet command.

Figure 64 Effect of PAR TAN OFF



In the following figure, the in-sheet command PAR TAN ON ensures that both the tangent point of the arc and the arc itself are supported on the grid.

Figure 65 Effect of PAR TAN ON



Layers

Layers have properties that affect whether or not they are drawn and whether elements on them can be changed. There are two layer properties that are used only in Parametric Design

- TRN/UNTRN
- DEL/UNDEL

Other layer properties are also relevant to other commands in MEDUSA 2D Design. This manual only documents the effect of layer commands in relation to Parametric Design. For further information on layer properties and commands, refer to the *MEDUSA Basis1 Design Commands Guide*.

Layer Defaults

Elements created with Parametric Design commands are placed onto the following layers:

Layer	Element type
4	Dimensions, baselines
13	Attachment points, instance clump elements
14	In-sheet command text, table elements
15	Parametric group elements
16	Automatically inferred baselines
17	Lines created with the COL, CIR, and TAN options of the PAR GRIS command.
28	Viewbox lines
56	Orthogonal 3D view prims used as reference points.
99	Error messages, OLD, NEW, and POT

You can change these layer defaults with the PAR DDL command. See ["Changing Element Types"](#), ["Changing Element Types - PAR DDL"](#) on page 154 for information on how to use PAR DDL.

Querying Layer Properties

Use the Q LIS command to display the current layer properties.

Layer Properties

You can change the way geometry is parameterized by changing any of the following layer properties:

Hittable/Unhittable

Elements on unhittable layers (UNHIT) are not used to generate the parametric grid and they are not transformed during parameterization. Elements on hittable layers (HIT) are used to generate the grid. Whether they are transformed or not depends on the other layer properties.

Transformable/Untransformable

Elements on transformable layers (TRN) may be transformed during parameterization, if the layers are also hittable and unprotected. Elements on untransformable layers (UNTRN) may be used to build up the grid, but cannot be transformed.

Deletable/Undeletable

Elements on deleteable layers (DEL) are automatically deleted at the end of parameterization, provided the layers are also hittable and unprotected. This is useful for deleting dimensions and other items that are required to build up the grid, but are not wanted in the final drawing. Elements on undeleteable layers (UNDEL) are left on the sheet.

Protected/Unprotected

Elements on protected layers (PRO) can not be transformed or deleted, but may be used to build up the grid. This property overrides the TRN and DEL layer properties.

Visible/Invisible

Elements on visible layers (VIS) are undrawn before they are moved, and then redrawn in their new positions during parameterization. Elements on invisible layers (INVIS) are neither undrawn nor redrawn. Note that the setting of the `PAR UND` switch affects the undrawing of all elements (see [“PAR UND” on page 97](#) for more on the `PAR UND` switch).

Default Layer Settings

When you run the Parametric Design system the default setting for all layers is:

- Hittable
- Transformable
- Undeletable
- Unprotected

- Visible

Layer 99, which is used for error messages and grid lines, is ignored during parameterization.

Dimension Clumps

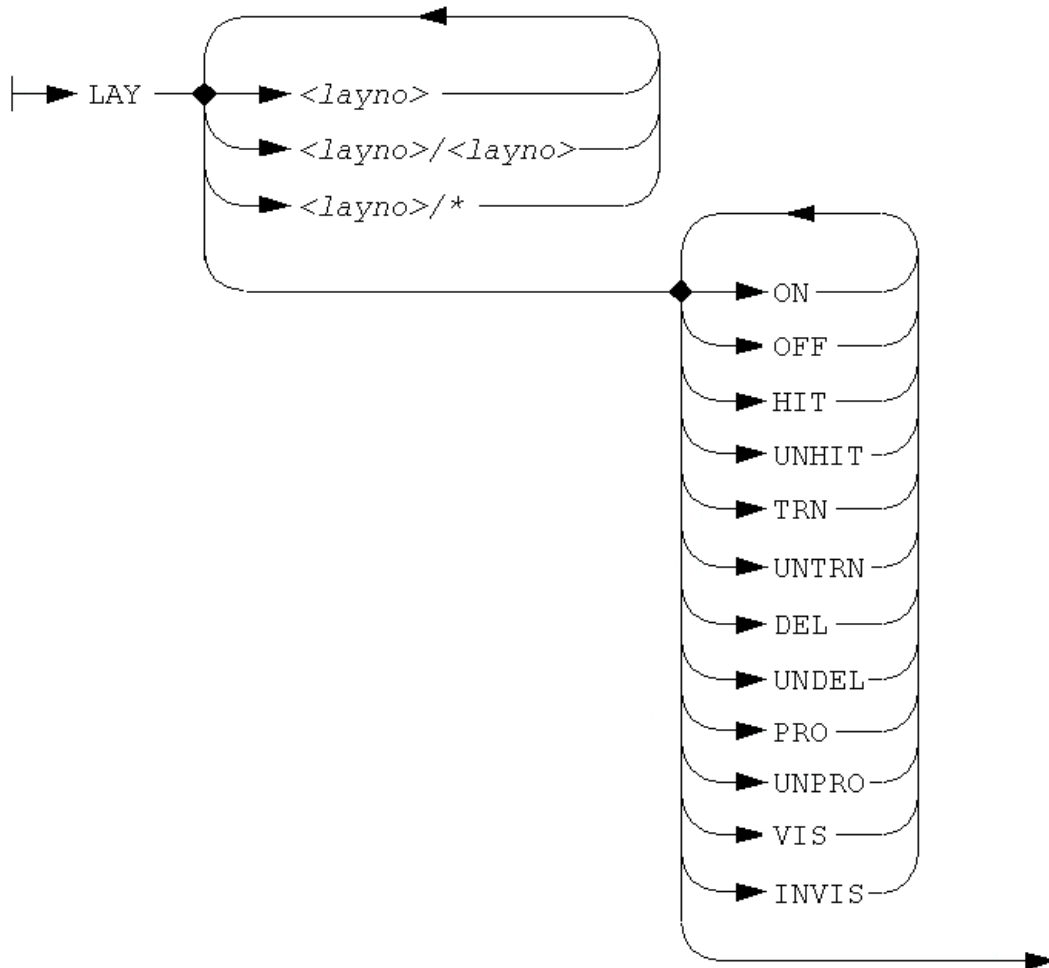
Dimension clumps are treated as a single unit by the Parametric Design system. Each clump is ignored, transformed or deleted as a single unit according to the layer of the dimension text (layer 4).

Changing Layer Properties

You can change layer properties using the LAY command. LAY can be used both interactively and as an in-sheet command.

LAY Command Syntax

The syntax of the LAY command is shown below and the options are explained below and on the next page:



Options	Description
<i>lay_no</i>	Specifies the number of the required layer. Must be an integer.
<i>lay_no</i> / <i>lay_no</i>	Specifies a range of layers. For example, layers 18 through 23 would be specified as 18/23.

Options	Description
<i>lay_no/*</i>	Specifies all layers from the layer specified to layer 1023. For example, layers 99 through 1023 would be specified as 99/*.
ON	Layers that are ON are visible, hittable and transformable. This is the default.
OFF	Layers that are OFF are invisible, unhittable and untransformable.
HIT	Elements on hittable layers are used to generate the grid. This is the default.
UNHIT	Elements on unhittable layers are completely ignored by the Parametric Design system.
TRN	Elements on transformable layers are transformed during parameterization. This is the default.
UNTRN	Elements on untransformable layers can be used to build up the grid, but cannot be transformed.
DEL	Elements on deletable layers are deleted during parameterization.
UNDEL	Elements on undeletable layers remain on the sheet after parameterization. This is the default.
PRO	Elements on protected layers cannot be transformed or deleted but can be used to generate grid lines.
UNPRO	Elements on unprotected layers can be transformed and deleted by the Parametric Design system. This is the default.
VIS	Elements on visible layer are visible on the screen. This is the default.
INVIS	Elements on invisible layers are neither undrawn nor redrawn during parameterization.

Using In-sheet Commands to Change Layer Properties

You can change layer properties both interactively and using in-sheet command text, though it is better to use in-sheet commands. Use the following procedure to place an in-sheet layer command on the sheet:

1. Select new text of type TCO.
2. Place the command into the text buffer, for example: `/LAY 4 13 DEL`
3. Position the text inside the viewbox using a probe.

LAY DEL Command and Viewboxes

The LAY DEL command does not delete viewboxes which are on the given layer by default. For this a control variable named PARIVB (PARAmetric Ignore ViewBox) is available.

- If PARIVB is UNSET or if its value is zero or less than zero, viewboxes will be deleted.
- If PARIVB is set to a value larger than zero, viewboxes will not be deleted.
- If PARIVB is placed, as a TCO text, on a sheet within a viewbox (for example `LET PARIVB = 0`), deleting the viewbox will only be applied for that one viewbox.
- If you wish viewboxes never to be deleted, PARIVB can be set in the *draft.mac* and/or the *draft.ini/.drafterc* file. In this case an individual viewbox could be made an exception to this by placing a TCO text (for example `LET PARIVB = 0`) inside its viewbox.

As with any other geometry selected for DELEtion, the viewbox line will only be a candidate for deletion if it is "inside" the relevant viewing area, which does (in this case) include the possibility of being coincident with the viewbox.

PARAMETRIC SYMBOLS

This chapter describes how to create and use parametric symbols. A parametric symbol is a fully dimensioned drawing, complete with reference point, which is stored in a symbol file. Parametric symbols are parameterized as you load them onto a MEDUSA sheet.

There are two ways you can load parametric symbols:

- Using the PAR LOA command, which enables you to parameterize and load a single parametric symbol interactively
- By parameterizing geometry which has been prepared with instance clumps

This chapter describes the following:

- [Creating Parametric Symbols](#) 112
- [Loading Symbols Interactively](#) 115
- [Rotating and Mirroring Parametric Symbols](#) 118
- [Loading Symbols Using Instance Clumps](#) 120
- [Using Instance Clumps - Gearbox Cover Example](#) 123

Creating Parametric Symbols

Essentially, a parametric symbol definition is the same as any other object definition in Parametric Design. However, when creating a parametric symbol definition:

- Always use attachment points to define reference points
- Always rotate the symbol definition before dimensioning it
- Do not unload the viewbox with the symbol geometry

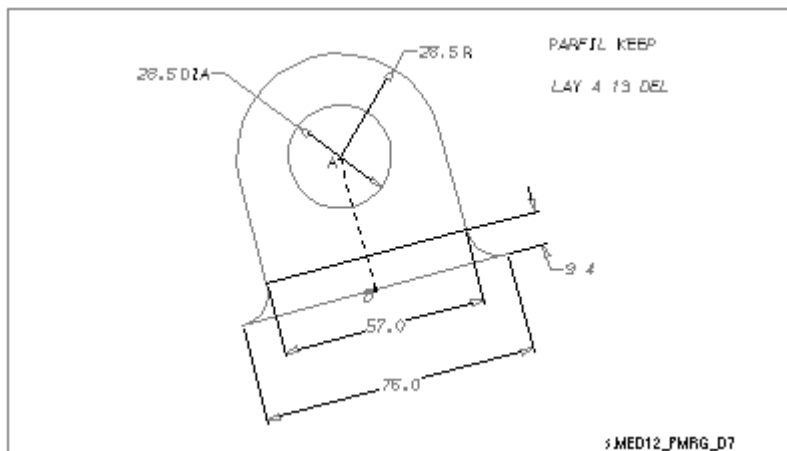
Reference Points

You must use attachment points and not static baselines or prims to define reference points in parametric symbol definitions. An attachment point is a text of type ATP, and you can place any number of attachment points in a parametric symbol definition. The ATP text contains a variable name that evaluates to the X and Y-coordinates of the reference point. Two grid lines are generated through the datum of the text, one horizontal and one vertical.

Example

The following figure shows a parametric symbol. The attachment points A and B generate grid intersections at the datum of the texts, and the line linking the attachment points generates a grid line through the attachment points and along the orientation of the rotated object. The orientation and length of the tab when it is loaded will be determined by the relative positions of the attachment points. The in-sheet layer command `LAY 4 13 DEL` ensures that the dimensioning and the extra line are deleted during parameterization.

Figure 66 Symbol Definition With Attachment Points



Symbol Orientation

The symbol shown in the Figure 66 has been rotated. This is because horizontal and vertical lines grid lines can constrain the orientation of the symbol when you try to load it. If you intend to load the symbol at different orientations, avoid creating horizontal and vertical lines in the symbol definition. Note that attachment points generate horizontal and vertical grid lines through the datum point. An easy way of avoiding constraints on symbol orientation is to rotate the entire symbol definition, including any attachment points, and then dimension it in a rotated position.

Testing the Symbol Definition

Before you unload the symbol definition, test it with the PARS CAN command. Use the following procedure:

1. Make sure the object geometry is inside a parametric viewbox.
2. Specify positions for the attachment point texts using LET commands and probes, for example:

```
LET A = FRE$RP
```
3. Parameterize the geometry temporarily with PARS CAN.

If there are no problems with the definition sheet, then you are ready to unload the symbol.

Unloading the symbol

Use the UNLOA command to unload the dimensioned symbol. Full details of UNLOA are given in the *MEDUSA Bacis1 Design Commands Guide*. The point probed when unloading the symbol is not significant: this point is not used at all when the system loads the parametric symbol.

To avoid unloading the viewbox with the geometry, use either of the following methods:

- Draw a group line around the geometry then use the UNLOAG command
- Make the viewbox line current then use the UNLOAIC command

In-sheet Commands and Parametric Symbols

Parametric symbol definitions may include in-sheet commands and tables as well as object geometry. When you load a symbol onto a sheet, it takes the switch settings, layer attributes, grid tolerance value, and fillet radius default for the sheet onto which you are loading it. You can change the settings for the symbol being loaded with in-sheet commands in the symbol definition, but after the symbol is loaded the settings are restored to their original values. In-sheet commands within a symbol definition affect only the symbol geometry and not the rest of the geometry on the master sheet.

Deleting Layers

Parametric symbols may include in-sheet layer commands, for example, to delete elements that are required to parameterize the symbol, but are not wanted on the final drawing. You can

remove unwanted elements automatically using an in-sheet layer command such as LAY 4 14 DEL in the symbol definition.

Tables and Parametric Symbols

Tables can be useful when using parametric symbols. The entries in a table row or column define values for several symbol variables at once. See [“Tables” on page 129](#), for more information about using tables in Parametric Design.

Variables and Parametric Symbols

Any variables created within a parametric symbol definition exist only temporarily for as long as it takes to load the symbol onto the sheet. These variables cannot be accessed afterwards. If the value of a variable used within a symbol is required outside the symbol, in a Bacis1 program, for example, you must define it as a protected variable.

Loading Symbols Interactively

The PAR LOA command loads and scales a parametric symbol. Use the following procedure to load a single symbol:

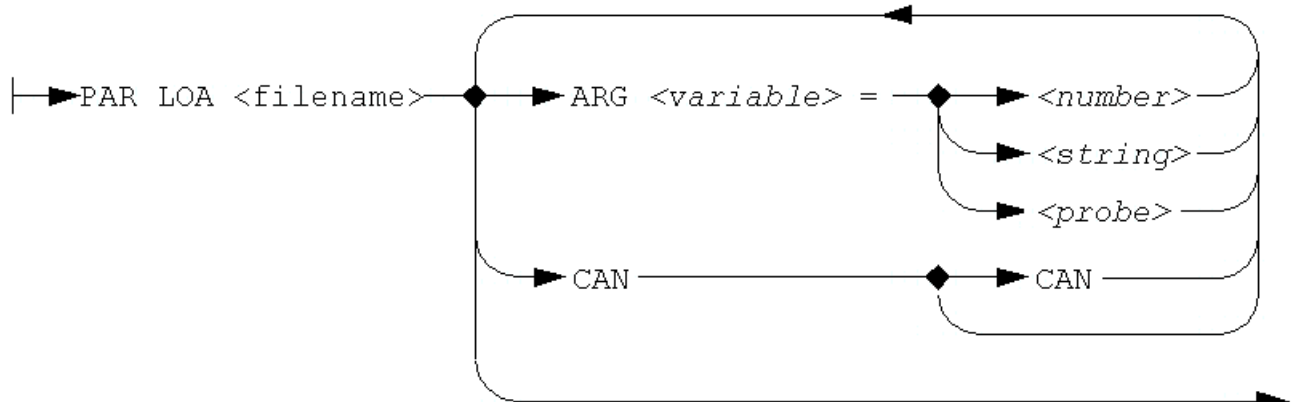
1. Specify the symbol you want to load by typing PAR LOA followed by a filename.
2. Specify all variables in the symbol definition with the ARG command.
3. Load the symbol with the CAL command.

For example:

```
*PAR LOA bearing1.sym
*ARG ANG = 45
*ARG A = FRE$RP
*CAL
```

Only protected variables and variables defined using ARG can be accessed by the symbol as it is loaded. More information about variable scope restrictions is given under section "[Variables and Expressions](#)", "[Variable Scope Restrictions](#)" on page 82.

PAR LOA Syntax



Option	Explanation
filename	The name of the file containing the symbol definition.
ARG	Specifies variables in the parametric symbol, including symbol attachment points.
variable	The name of the required variable. For example, <i>LENGTH</i> , <i>X1</i> , <i>RADIUS</i> .
number	Value assigned to the variable name variable. The number specified must be a signed real value expressed as a decimal, a fraction, or an exponential.
string	Any text string enclosed in single quotes. For example, ' <i>LENGTH</i> '.
probe	A probe positioning attachment points in the symbol definition. Probe syntax is described in the <i>MEDUSA Basis1 Design Commands Guide</i> .

CAL	Initiates the symbol loading and parameterizing (if any) after you have specified all the arguments. You can load several copies of the symbol by respecifying the arguments before using CAL again.
CAN	Cancels the last copy of the symbol.

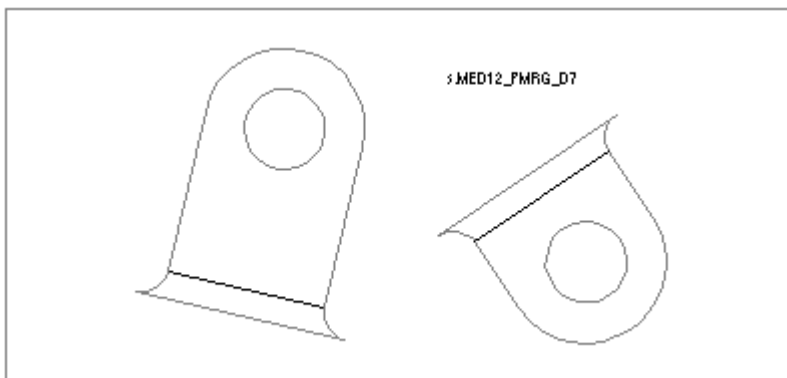
Loading the Same Symbol Several Times

PAR LOA works in a loop, enabling you to load a parametric symbol in several places on the same sheet. You can redefine the attachment point position and any other variables before loading another copy of the symbol with the CAL command. For example:

```
*PAR LOA bolt.sym
*ARG A = FRE$RP
*CAL
(move the cursor to a new position)
*ARG A = FRE$RP
*CAL
(move the cursor to a new position)
*ARG A = FRE$RP
*CAL
```

In the following figure, the symbol defined in [Figure 66, “Symbol Definition With Attachment Points” on page 112](#) has been loaded onto a sheet at different positions and orientations. The different orientations are achieved by moving the two attachment points A and B in relation to each other. The in-sheet commands in the original symbol definition ensure that the dimensioning, the attachment points, the line linking the attachment points, and the in-sheet command text itself do not appear on the final drawing.

Figure 67 Symbol Loaded at Different Orientations



Cancelling the Last Copy

If you are not happy with the new drawing you can cancel the last copy of the symbol with the CAN command and then respecify the arguments before loading another copy. For example:

```
*PAR LOA bearing2.sym
*ARG A = NEA$RP
*ARG B = SEG$RP
```

```
*CAL  
*CAN  
*ARG A = NEA$RP  
*ARG B = INT$RP  
*CAL
```

Unlike cancelling PARS and PAR GRIS, the new graphics are undrawn when you cancel PAR LOA.

Errors During Loading

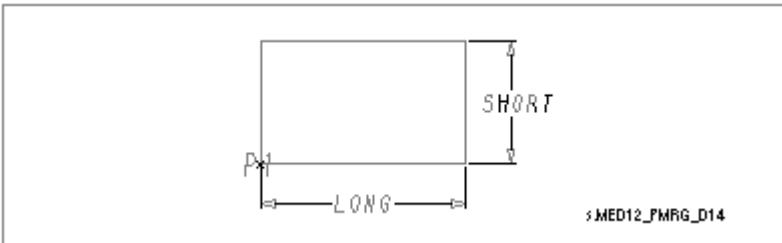
If you make a mistake when loading a parametric symbol, the unparameterized symbol will appear at an arbitrary location on the screen with error messages indicating the problem areas. However, although you can see the symbol on the screen, it is not loaded onto the sheet and if you refresh the screen with a REDRA command, the image will disappear. The reason for this is that it could be difficult to delete a symbol which has been placed into a drawing. To investigate the problem, examine the file containing the original symbol definition.

Rotating and Mirroring Parametric Symbols

It is often useful to rotate or mirror a symbol about its attachment point as it is loaded onto the sheet. Use ROT or MIR before you give the CAL command. See the *MEDUSA Basis1 Design Commands Guide* for more details on rotating and mirroring elements.

You can rotate and mirror only parametric symbols which contain one attachment point. The following examples show how the symbol definition in the following figure can be rotated and mirrored.

Figure 68 Original Symbol



Rotating Symbols

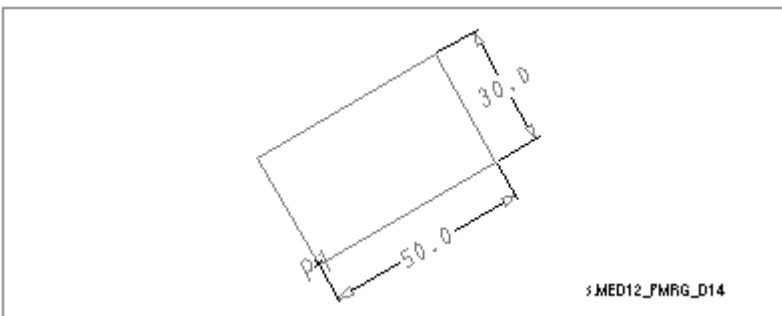
You can rotate parametric symbols use any of these commands:

- ROTLN angle (text remains legible)
- ROTFN angle (text remains fixed)
- ROTRN angle (text remains rigid)

For example, to load and rotate the symbol definition shown in Figure 68, you could use the following commands. The result is shown in the following figure.

```
*PARLOA RECT.SYM
*ARG P1 = FRE$RP
*ARG LONG = 50
*ARG SHORT = 30
*ROTLN 30
*CAL
```

Figure 69 Symbol Rotated by 30 Degrees



Mirroring Symbols

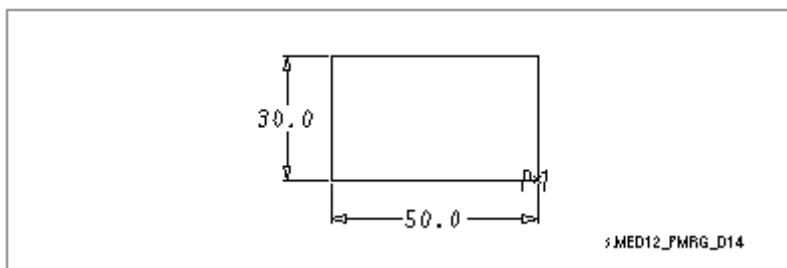
Use any of the following commands to mirror symbols about the horizontal or vertical axis.

- MIRVLN (mirror vertical, text legible)
- MIRVRN (mirror vertical, text rigid)
- MIRVFN (mirror vertical, text fixed)
- MIRHLN (mirror horizontal, text legible)
- MIRHRN (mirror horizontal, text rigid)
- MIRHFN (mirror horizontal, text fixed)

These commands work only if the symbol has a single attachment point. For example, the following sequence gives the mirroring shown in the following figure:

```
*PARLOA RECT.SYM  
*ARG P1 = FRE$RP  
*ARG LONG = 50  
*ARG SHORT = 30  
*MIRVFN  
*CAL  
*
```

Figure 70 Symbol Mirrored with MIRVFN Command



Loading Symbols Using Instance Clumps

So far, to load and parameterize a symbol, you have used three commands: PARLOA, a series of ARG commands to specify variable values and finally CAL to load and parameterize. Using instance clumps, you can parameterize a sheet and load parametric symbols onto it with one command, PARS.

When you parameterize a sheet containing instance clumps, the system does the following for you:

1. Parameterizes any geometry that is inside a viewport.
2. Processes any instance clumps, loading and parameterizing parametric symbols.

If you use the PAR GRIS command in a sheet containing instance clumps, grid lines are only drawn for the main definition. Instance clumps will not be processed.

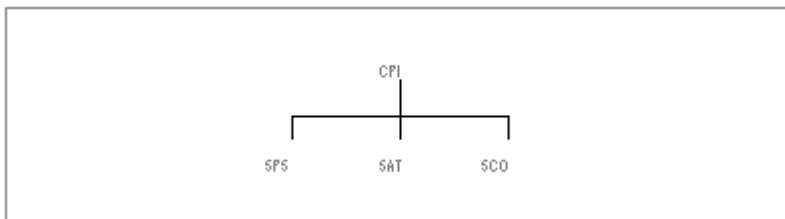
Instance Clump Elements

Each instance clump contains the following elements:

- SPS text containing a symbol file name. The named symbol is loaded onto the sheet when you give the PARS command.
- One or more instance clump attachment points (SAT texts) that define the position at which the symbol attachment points (ATP texts) are placed
- Optional in-sheet commands (text type SCO) that specify variables in the symbol definition

The following figure shows the structure of an instance clump:

Figure 71 **Structure of an Instance Clump**



Creating an Instance Clump

To create an instance clump, use the following procedure:

1. Open a new CPI clump.
2. Place text into the text buffer, for example:
ATP1, hole1.sym, arg len1 = 30.5.
3. Select the appropriate text type: SPS text for the symbol filename, SAT text for attachment point text or SCO text for commands.
4. Place the text onto the sheet using a probe.

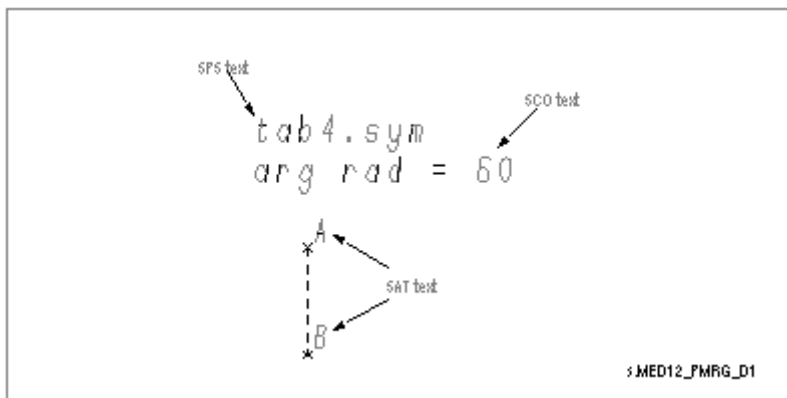
Instance Clump Element Types

The following element types are used in instance clumps:

- Symbol filename SPS text
- Symbol attachment points SAT text
- Symbol commands SCO text

The following figure shows an example of an instance clump in detail.

Figure 72 Instance Clump Elements



For each CPI clump on the sheet, a copy of the symbol in the file specified by the SPS text is loaded. You can use only one SPS text in each instance clump, and this must be a valid file-name.

Variables in Instance Clumps

The symbol attachment points, text type *SAT*, are variables which are used to position the symbol on the sheet. The text must contain a valid variable name (up to six characters, the first character being alphabetic). *SAT* texts in instance clumps usually correspond directly to the attachment points, *ATP* texts, in the symbol definition. Symbol attachment points are normally positioned at grid intersections so that the symbol is loaded at the appropriate place in the parameterized object.

You can define any other variables required for the symbol using ARG commands in *SCO* type texts. Only protected variables and argument variables you define in instance clumps can be accessed by parametric symbols during parameterization.

See "[Variables and Expressions](#)", "[Variable Scope Restrictions](#)" on page 82 for more information about variable scope restrictions and protected variables

Commands in Instance Clumps

You can place the following commands in instance clumps. They should be of text type SCO:

- VER FULL
- ARG

VER FULL: Place the VER FULL command in an instance clump to generate more detailed error message information when an error occurs during symbol loading. The syntax for VER FULL is:

| → VER FUL →

VER FULL causes the symbol to be drawn on the screen and enlarged, with error messages indicate the areas on the symbol causing problems.

ARG : Use ARG to assign values to variables in parametric symbols. Values can be numbers or strings. The following graph shows the syntax for the ARG command:

| → ARG <variable> = —◆— → <number> —→ <string> —→

Variable	Description
variable	The name of the required variable. The name can be up to six characters, but the first character must be alphabetic.
number	The number assigned to the variable <i>variable</i> . The number you specify must be a signed real value expressed as a decimal, a fraction, or an exponential.
string	Any text string enclosed in single quotes ('...').

Loading Errors

Errors that occur when you load symbols using instance clumps are indicated by error messages drawn on the SPS text. If you include VER FULL in an instance clump, more detailed error message information is provided if an error is found during symbol loading.

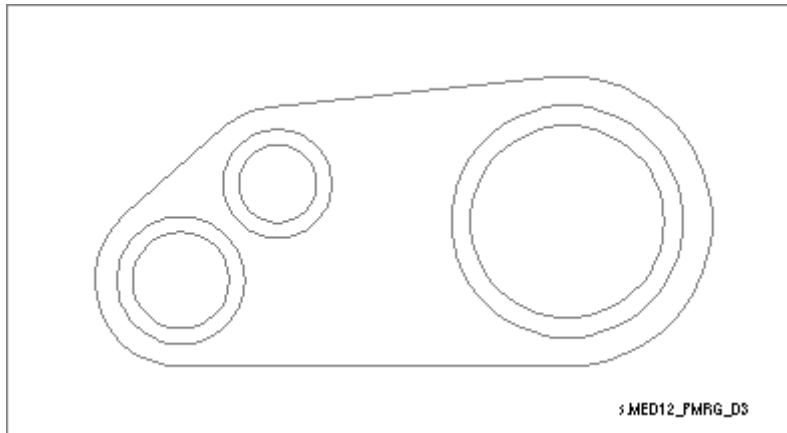
It is important to note that the Parametric Design system first parameterizes the object in the viewbox fully and, when it has finished, then processes any instance clumps in the viewbox. The instance clumps are processed independently of each other. If errors are encountered when loading a symbol specified by one of the instance clumps, the symbol will be left in the sheet at the wrong position, and the rest of the object will be parameterized normally. This means you can use only the limited range of commands listed under ["Parameterizing Geometry"](#), ["Commands Between PARS and CAN"](#) on page 93 before cancelling the parameterization.

It is therefore recommended that you use the temporary parameterization command PARS CAN before parameterizing the sheet irreversibly with PARS.

Using Instance Clumps - Gearbox Cover Example

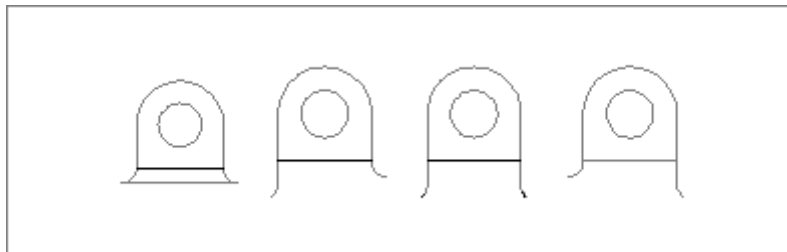
This example shows instance clumps being used to add tabs onto a drawing of a gearbox cover, shown in the following figure.

Figure 73 Gearbox Cover



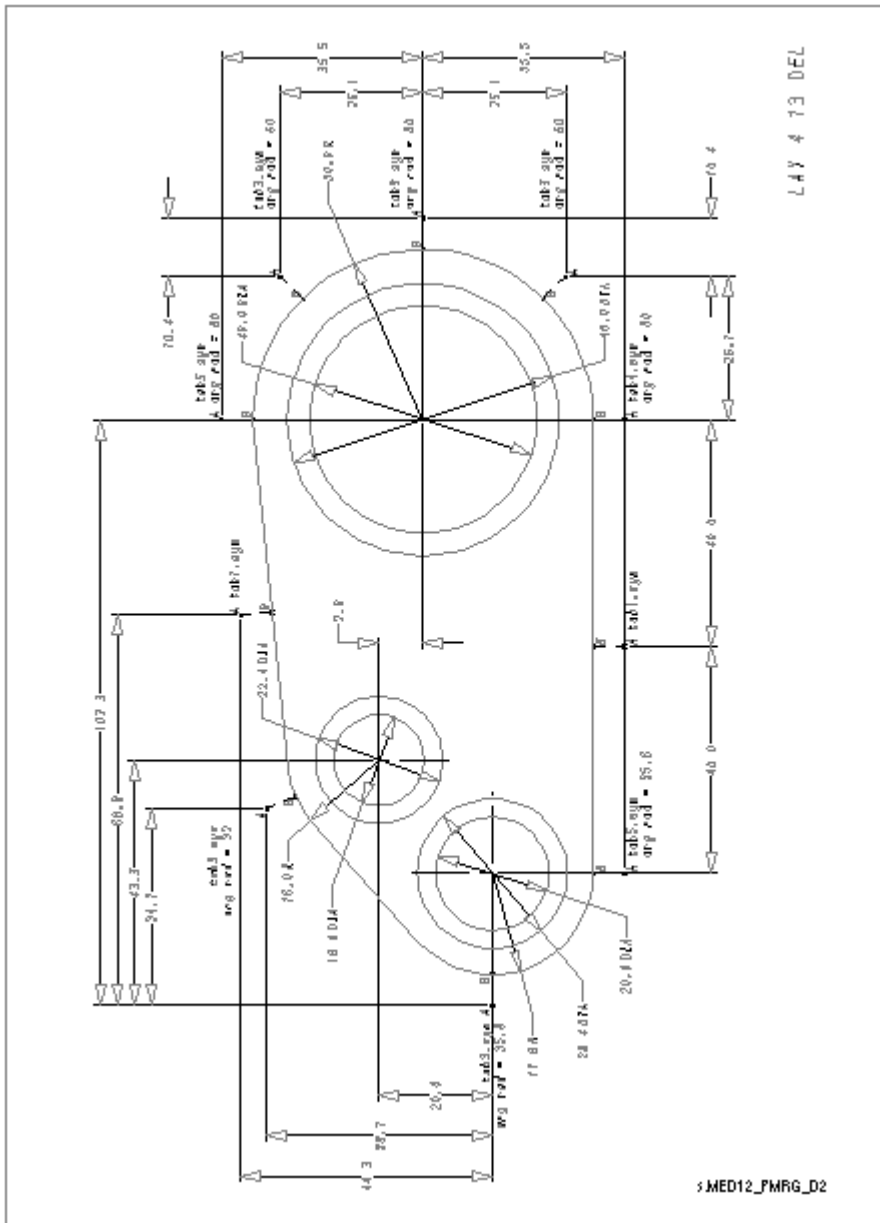
Four different symbols are defined representing four tabs, shown in the following figure. These symbols must be positioned on the main drawing during parameterization. One tab fits along the straight edges of the gearbox cover and three fit onto curved edges of the cover. A variable is used in these three to specify each different radius of curvature.

Figure 74 Four Tabs



The [Figure 75, “Gearbox Cover Definition”](#) shows the definition sheet for the gearbox cover. The geometry of the main part of the cover is drawn on the sheet and fully dimensioned, and instance clumps specify where the tabs are to be placed when the sheet is parameterized. The sheet also contains an in-sheet command `LAY 4 13 DEL` which deletes all dimensions and instance clump elements from the final drawing. Note that the symbol attachment points are all fully dimensioned.

Figure 75 Gearbox Cover Definition



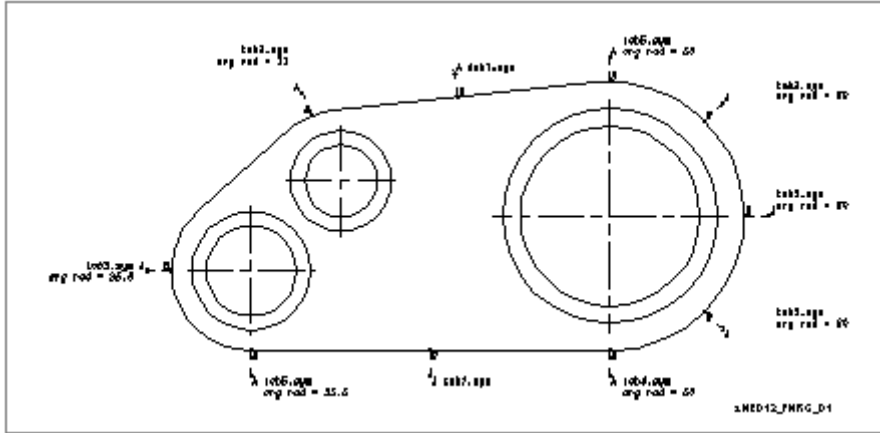
Instance Clumps

Instance clumps are used in this example to add the tabs to the main drawing of the gearbox cover. Each instance clump on the main component definition contains the following information:

- The name of the symbol file containing the tab definition
- Two symbol attachment points (*SAT* texts)
- For the symbols containing the variable *RAD*, *SCO* text to specify radius of curvature

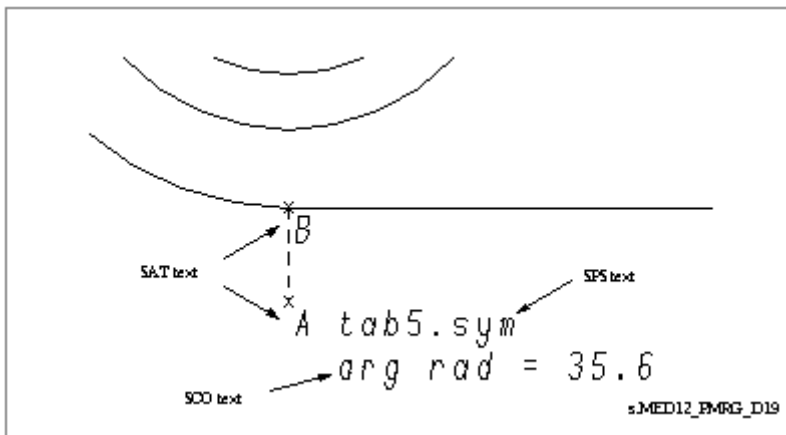
The following figure shows the gearbox cover definition with the dimensioning removed so that you can see the instance clumps more clearly.

Figure 76 Instance Clumps



The following figure shows one of the instance clumps in detail.

Figure 77 Detail of Instance Clump



Symbol Definitions

Four different symbols are defined, three of which contain an additional variable, *RAD*, to specify the radius of curvature. The symbols are:

- *tab1.sym* for use on flat surfaces
- *tab3.sym* for use on curved surfaces
- *tab4.sym* and *tab5.sym* for use on partially curved surfaces

The following figures show the symbol definitions for each of the tabs used in this example.

Figure 78 Definition for tab1.sym

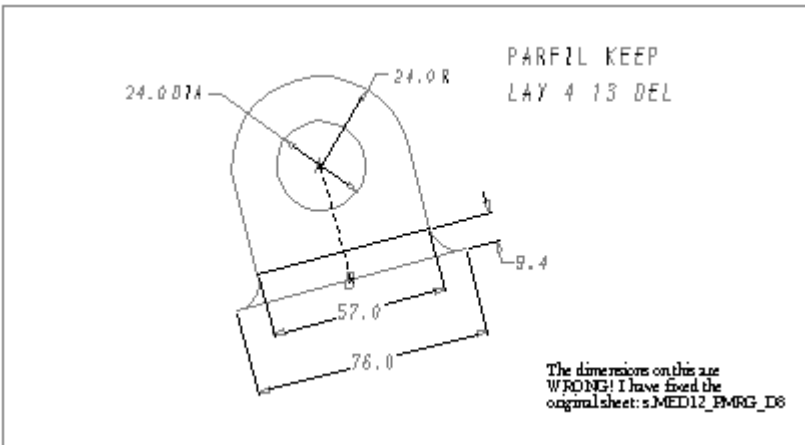


Figure 79 Definition for tab3.sym

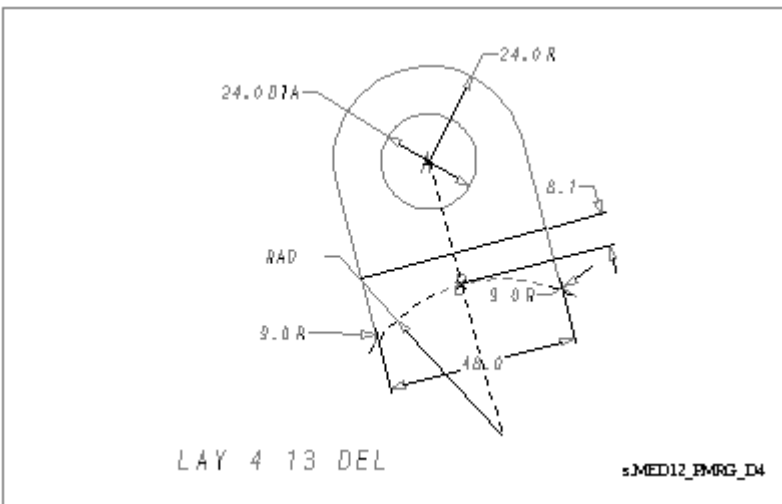
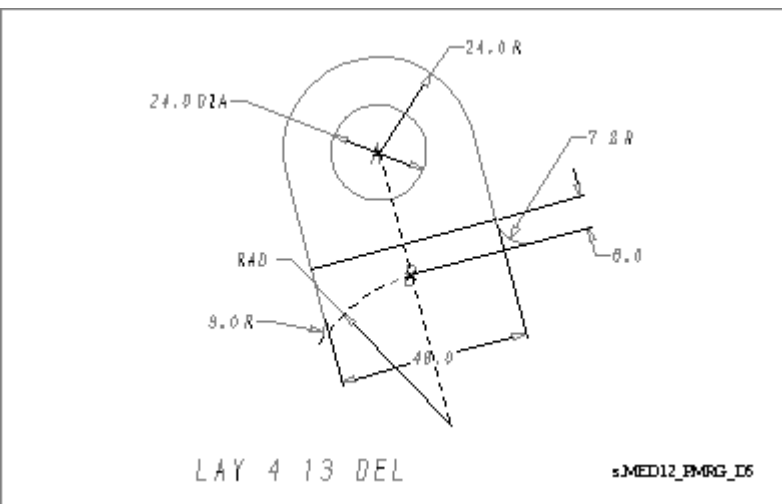


Figure 80 Definition for tab4.sym



TABLES

This chapter shows how to use tables to store one or more groups of values on a sheet and set a number of related variables during parameterization.

- [Table Definitions.....](#) 130
- [Constructing a Table](#) 132
- [The TBL Command.....](#) 135
- [Tables and Parametric Symbols](#) 137

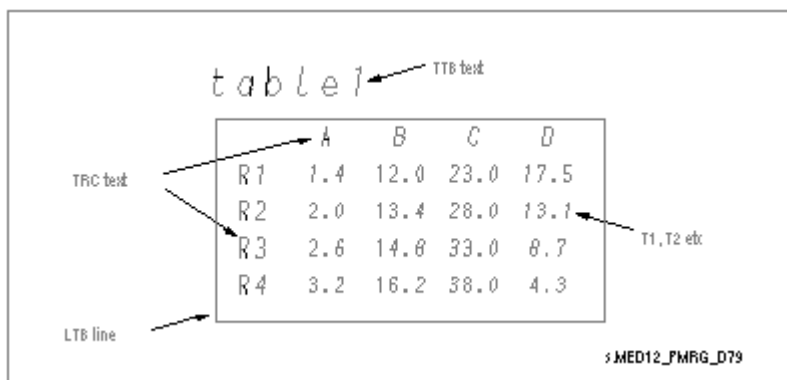
Table Definitions

A table is composed of special elements. Tables contain a series of values arranged in rows and columns which you can access using the TBL command. Using a table you can set several variable parameters at once with a single TBL command. You can have more than one table on a sheet.

Accessing Table Values

When you execute the TBL command, the system reads values from the table and uses them to parameterize the geometry. Specifying a table row or column has the same effect as giving a LET command for each entry in that row or column.

Figure 83 The Parts of a Table



Parts of a Table

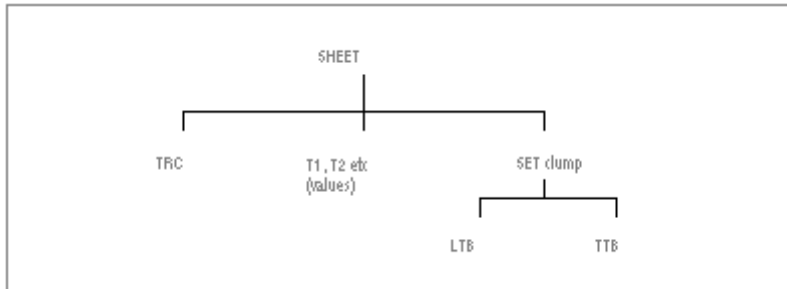
A table consists of the following elements:

Part	Element Type	Function
Title	TTB text	The table name. This text must be in the same SET clump as the table boundary line. TTB text is optional if there is only one table on the sheet.
Box	LTB line	Defines the table boundary. The LTB line must be in a SET clump.
Column names	TRC text	Row and column names define a set of values. TRC texts
Row names		are sheet level texts. The should not be part of the clump containing the LTB line ands must be within the table area defined by the LTB line.
Values	Any text type	The values are sheet level texts, placed at the intersection of horizontal and vertical lines from the row and column text datums. Use either construction lines or the grid to place values. Any valid expression may be used in a table.

Structure of Table Elements

The diagram below shows the relationship between table elements.

Figure 84 The Structure of Table Elements



Adding Elements to Tables

You can add lines to the table, for example to divide the rows and columns. These lines can be of any type and will be ignored by the TBL command. However, you must put the LTB line defining the table boundary and any extra lines on an untransformable layer before parameterization.

Positioning Tables on the Sheet

A table can be placed anywhere on the sheet. It does not have to be in the same viewbox as the object geometry, though if it is inside the viewbox make the table boundary and any extra lines untransformable.

Changing the Element Types Associated with a Table

If you use the default element types used for tables (types LTB, TTB and TRC) in other applications, conflicts between element types and their uses may occur. You can avoid this by redefining element types for the parts of a table with the command TBL DDL. This command is described under ["Changing Element Types"](#), ["Changing Table Element Types - TBL DDL"](#) on [page 158](#).

Constructing a Table

You can construct a table both by entering the commands directly using the keyboard. In the following example the commands are entered directly:

```
*NEWC SET NEWL LTB LAYN14
*FRE $RP      Probe the bottom left corner of the box
*FRE $RP      Probe the top right corner of the box
*BOX ENDL
*/table1
*NEWT TTB LAYN14 FRE $RPPosition the table name text with a probe
*END          End the clump
*/R1
*NEWT TRC LAYN14
*FRE $RP      Position the row and column texts with probes
```

Entering Values in a Table

Table values can be of any text type. Use either a MEDUSA 2D Design grid or construction lines to place the table values in correct alignment with the row and column texts. If they are not aligned you will receive the following error message when you try to parameterize the drawing:

```
Cannot resolve table
```

This error means that the system cannot separate the table into rows and columns with the same number of entries in each row and in each column during parameterization.

Example

The table shown in the following figure contains the dimensions for a joist section. Lines and text (Nominal size, Thickness) have been added between the rows and columns to make the information in the table clearer. Remember that if you add lines for this reason, you must make them untransformable before parameterizing the drawing. This table can be used to create a set of joists. The definition sheet for the joist is shown in [Figure 87, "In-sheet TBL Command"](#) on [page 136](#). Because this is the only table on the definition sheet, there is no table title text (text type TTB).

Figure 85 Table of Dimensions For Joist Section

JOISTS to BS4 : Part 1 : 1972								
Ref	Nominal size #x#	Mass /metre kg	Section		Thickness		Radius	
			Depth #	Width #	Web #	Flange #	Root #	Toe #
	NON	W	D	B	S	T	R	Y
1	'254X203'	81.85	254.0	203.2	10.2	19.9	19.6	9.7
2	'254X114'	37.20	254.0	114.3	7.6	12.8	12.4	6.7
3	'203X152'	52.88	203.2	152.4	8.9	16.5	15.5	7.8
4	'203X102'	25.33	203.2	101.6	5.8	10.4	9.4	3.2
5	'178X102'	21.54	177.8	101.6	5.3	9.8	9.4	3.2
6	'152X127'	37.20	152.4	127.0	10.4	13.2	13.5	6.8
7	'152X89'	17.89	152.4	88.9	4.9	8.3	7.9	2.4
8	'152X76'	17.86	152.4	76.2	5.8	9.6	9.4	4.6
9	'127X114'	29.76	127.0	114.3	10.2	11.5	9.9	4.8
10	'127X76'	26.79	127.0	76.2	7.4	11.4	9.9	5.0
11	'127X76'	16.37	127.0	76.2	5.6	9.6	9.4	4.6
12	'127X76'	19.36	127.0	76.2	4.5	7.8	7.9	2.4
13	'114X114'	26.79	114.3	114.3	9.5	10.7	14.3	2.4
14	'102X102'	29.87	101.6	101.6	9.5	10.3	11.1	3.2
15	'102X64'	9.65	101.6	63.5	4.7	6.6	6.9	2.4
16	'102X44'	7.44	101.6	44.4	4.3	6.1	6.9	3.3
17	'89X89'	19.95	88.9	88.9	9.5	9.9	11.1	3.2
18	'76X76'	14.67	76.2	80.0	6.9	8.4	9.5	3.2
19	'76X76'	12.85	76.2	76.2	5.7	8.4	9.4	4.6

MED12_PMRG_D75

Variables and Expressions in Tables

You can use variables and expressions in tables. Variables are evaluated during parameterization and the numeric value is used to calculate the new dimension.

Please note: The sequence of variables and expressions in the table is important.

In the table shown in the following figure, some of the values are expressions rather than numeric values.

Figure 86 Table with Variables

	A	B	C
Row1	10	A+D	B/2
Row2	20	30	40

MED12_PMRG_D79

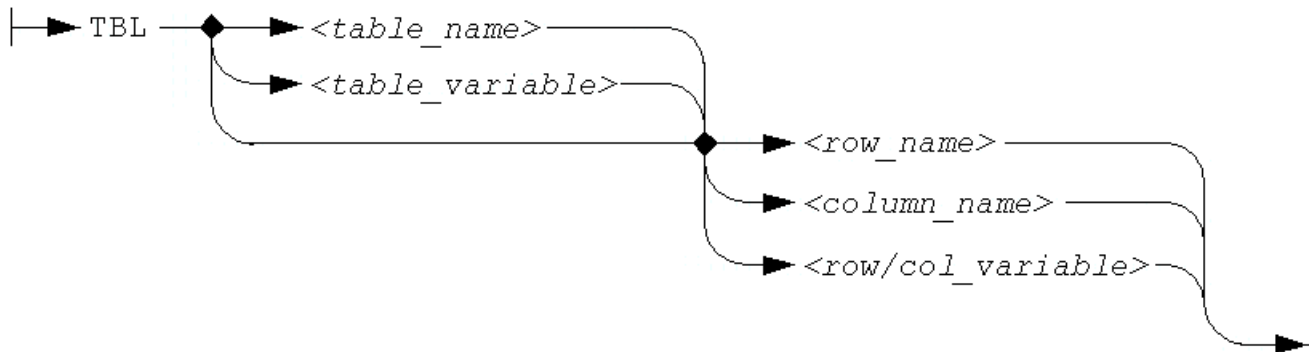
The value of each variable used in a table refers to its value before the execution of the TBL command and not to its value calculated during the execution of the TBL command.

The TBL Command

You can use the TBL command both interactively and as an in-sheet command.

- When you specify a row name in the TBL command, the values for that row are assigned to the variables specified in the columns
- When you specify a column name in the TBL command, the values for that column are assigned to the variables specified in the rows

The syntax of the TBL command is shown below:



Option	Description
<i>table_name</i>	Specifies the table name. This is only required if there is more than one table in the relevant sheet or symbol.
<i>table_variable</i>	A variable name that evaluates to a valid table name. The name can be any text string, but must be enclosed in angled brackets (< ... >).
<i>row_name</i>	Specifies the row name. The name specified can be any text string, but must be enclosed in single quotes ('...').
<i>column_name</i>	Specifies the column name. The name specified can be any text string, but must be enclosed in single quotes ('...').
<i>row/col_variable</i>	A variable which evaluates to a table row or column name. The name specified can be any text string, but must be enclosed in angled brackets (< ... >).

Using Variables to Access Rows

The command `TBL table1 'row1'` accesses values from the row named row1 each time you load the symbol. Instead of identifying a row or column name, you can use a variable. This enables you to specify a different row or column each time you load the symbol. For example, the following command creates the variable name var:

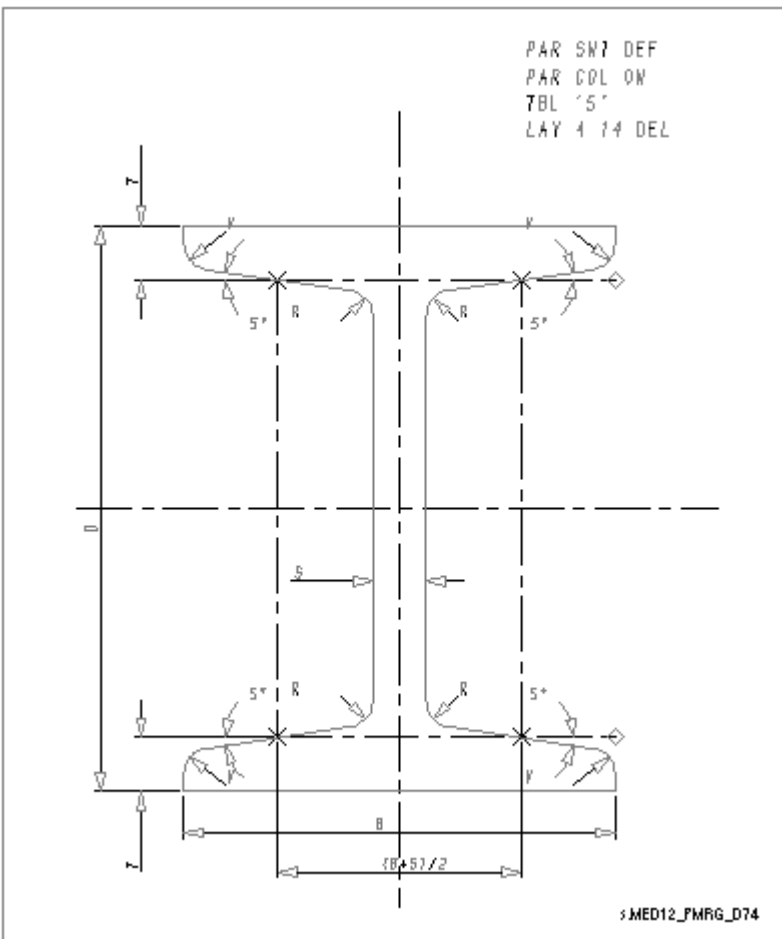
```
TBL table1 <var>
```

The value you specify for this variable must be a table row or column name.

Example

The following figure shows an example with an in-sheet TBL command. The table named in the TBL command is shown in [Figure 85, "Table of Dimensions For Joist Section" on page 133](#). The row name, 5, contains a set of values for all the variables in this joist definition. Because there is only one table on this definition sheet, no table name needs to be specified.

Figure 87 In-sheet TBL Command



Tables and Parametric Symbols

In the last chapter parametric symbols were introduced as a useful way of storing part definitions that dimensioned in terms of variable parameters. Tables can be used with parametric symbols to set all the variables in a parametric symbol at once with a single TBL command. A table can store the values for variables in the main component and in any symbols. Using tables to store variable values in this way helps to keep the main drawing simpler.

If you want to use a table when loading a parametric symbol, simply include the table in the symbol definition. Create the table in the usual way. You must unload the table with the symbol geometry.

In-sheet Commands

The following are the in-sheet commands.

TBL command : Include an in-sheet TBL command in the symbol definition and unload this with the table and symbol geometry.

Layer commands : Use a layer command to make the table outline and table value texts untransformable, or the system will try to parameterize them when you load the symbol. Use the same in-sheet command to prevent the table elements and in-sheet commands from the symbol definition appearing on the master drawing. For example,

```
LAY 13 14 UNTRN DEL
```

Unloading the Symbol

When you unload the symbol make sure you unload:

- All of the geometry
- The table
- All in-sheet commands

Please note: Do not unload the parametric viewbox.

PARAMETRIC GROUPS

This chapter describes how to use parametric groups in MEDUSA Parametric Design to isolate geometry where dimensions are to remain unchanged during parameterization. This enables you to parameterize parts of an object.

- Introduction 140
- Static Groups 143
- Dynamic Groups 144
- Rotating Parametric Groups 147
- The PAR PRE Command..... 149

Introduction

Parametric groups provide a way of moving and scaling parts of geometry without having to dimension every detail. The detail enclosed in a parametric group is either scaled or ignored during parameterization, depending on the number of PPG prims in the group.

Points within a parametric group move in different ways depending on whether or not they are dimensioned:

- All points within the group line that are not explicitly dimensioned are moved and scaled together with the group. You need to explicitly dimension only a few datum points.
- Points within the group line that are explicitly dimensioned are moved according to the dimension moves and not according to how the group moves.

Static and Dynamic Groups

There are two types of parametric group:

- Static
- Dynamic

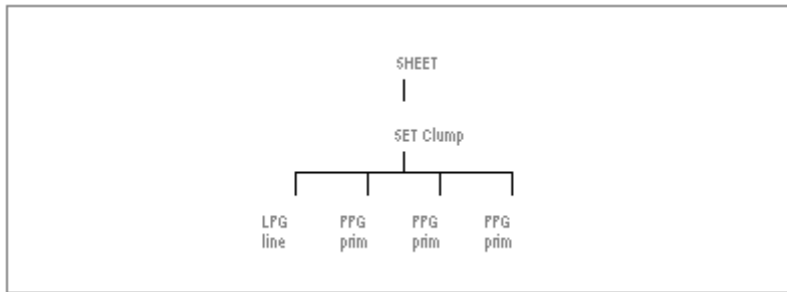
Static groups : The simplest form of parametric group is a static group. This consists of a closed line of type LPG in a SET clump, and does not include any parametric group prims. Undimensioned points inside a static group remain in their original position on the sheet during parameterization.

Dynamic groups : A parametric group that contains an LPG line and one or more PPG prims is known as a dynamic group. Points within a dynamic group can be moved, scaled, rotated or differentially scaled depending on how many PPG prims the group contains.

Parametric Group Elements

A parametric group is a clump of type SET containing a parametric group line. This is a single closed line of type LPG composed of straight line segments. The clump may also contain one, two or three prims of type PPG.

Figure 88 Structure of a Parametric Group



Using tangent point arcs to define the parametric group line is not recommended, as the tangent points lie outside the line.

Creating a Parametric Group

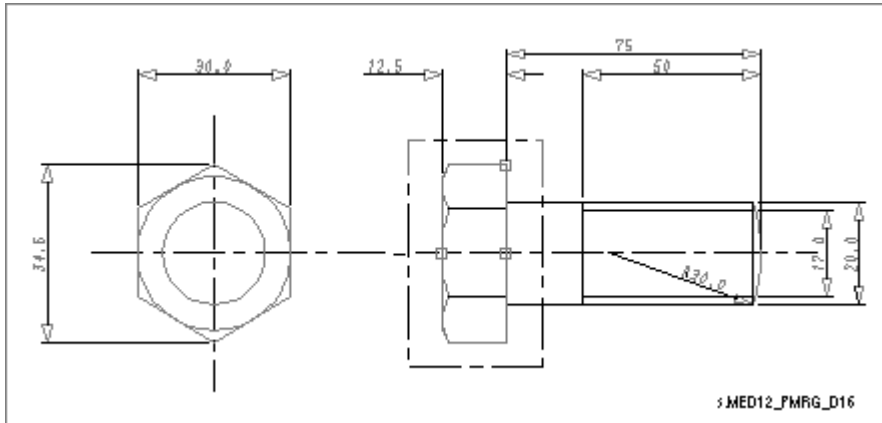
Use the following procedure to create a parametric group.

1. Enclose the geometry that will not vary in dimension by a parametric group line of type LPG.
2. If necessary, draw a further parametric group around any parts of the geometry.
3. If required, create one, two, or three parametric group prim(s) of type PPG inside the same clump as the parametric group line.
4. Ensure that all PPG prims are at grid line intersections. You may need to dimension them to place them on the grid.

Parametric Group Example

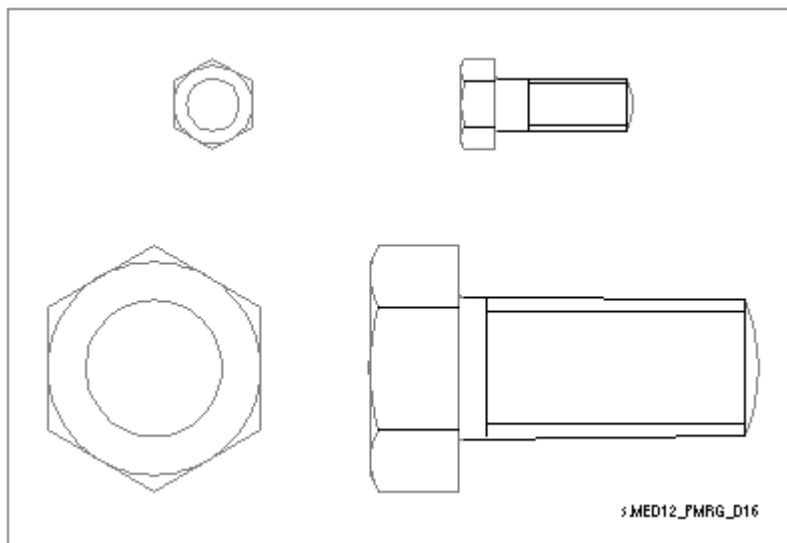
In the drawing shown in the following figure, placing a parametric group around the bolt head removes the need to detail the bolt head chamfer. The LPG line surrounding the bolt head defines a parametric group containing three PPG prims. All three prims lie on grid line intersections.

Figure 89 Bolt Definition With Parametric Group



The following figure shows two examples obtained by parameterizing the definition in Figure 89. In each case, the bolt head has been scaled in proportion to new parameters.

Figure 90 Results of Parameterization



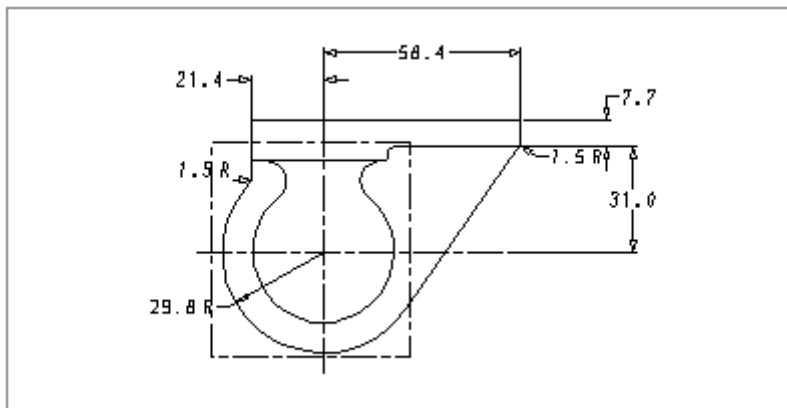
Static Groups

A static group consists of a closed line of type LPG in a SET clump. Static parametric groups do not contain parametric group prims. Any point within the LPG line that is not fully dimensioned will not be moved during parameterization.

Example

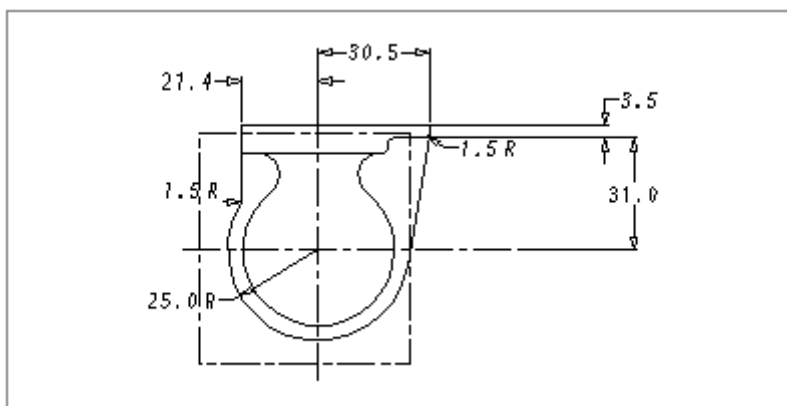
The following figure shows an object where most of the geometry has been enclosed in a static group line. This means that only the dimensions shown have to be added to enable the drawing to be parameterized.

Figure 91 Static Parametric Group



The following figure shows the same object after parameterization. Only one dimension has been changed, and the geometry inside the static group is untransformed.

Figure 92 After Parameterization



Dynamic Groups

A parametric group containing one or more prims of type PPG is known as a dynamic group. PPG prims must be part of the same SET clump as the LPG line, and the datum of each prim must lie at a grid intersection. These datum points are used to define the transformation that is applied to any undimensioned points lying within the group line.

The PPG prims do not need to lie inside the area defined by the group line. The prims are associated with the group line, because they are all in the same clump. You can include more than one closed group line in a clump. However, the group lines should not overlap or be nested.

How Dynamic Groups Work

What happens to a dynamic group during parameterization depends on how many prims it contains.

Translating - one prim: If the group has only one prim, then the group is simply translated in the same way as the datum point with no rotation or scaling.

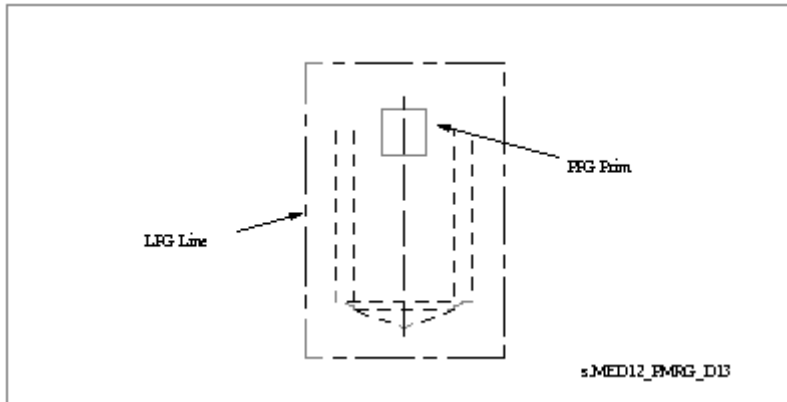
Scaling and rotating - two prims : With two prims the group will be scaled according to how the distance between the two prims changes, and rotated according to how one datum rotates about the other.

Differential scaling - three prims : Group with three prims can be scaled differentially, as well as translated and rotated. Different scale factors can be used along different axes of the object. Normally the prims are positioned so that the object is scaled differently in two orthogonal axes. If the axes are not orthogonal, you may have to link the prims with one or more lines to make sure they lie at grid intersections.

Example

The following figure shows an example of a dynamic group consisting of an LPG line and one PPG prim.

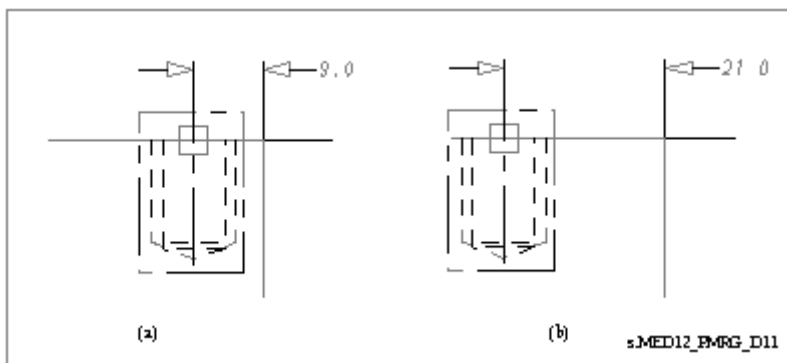
Figure 93 Parametric Group Definition



Transforming Using One Prim

The drawing (a) in the following figure shows a parametric group positioned near to the corner of an object. The PPG prim in the group is placed on the parametric grid by the 9.0 dimension. When this parameter is changed, the bolt hole moves horizontally during parameterization, as shown in (b).

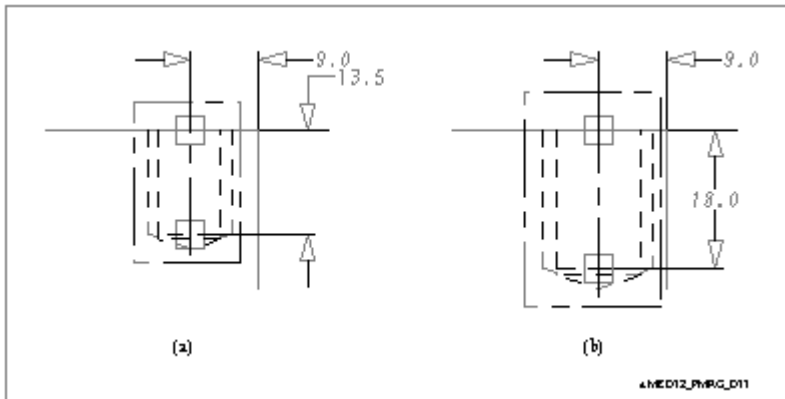
Figure 94 Group with One Prim



Transforming Using Two Prims

The drawing (a) in the following figure shows a parametric group with two prims. An additional PPG prim at the bottom of the bolt hole is placed on the grid by dimensioning the length of the bolt hole. By increasing the length of the hole, the whole of the bolt hole is scaled proportionally, as shown in (b).

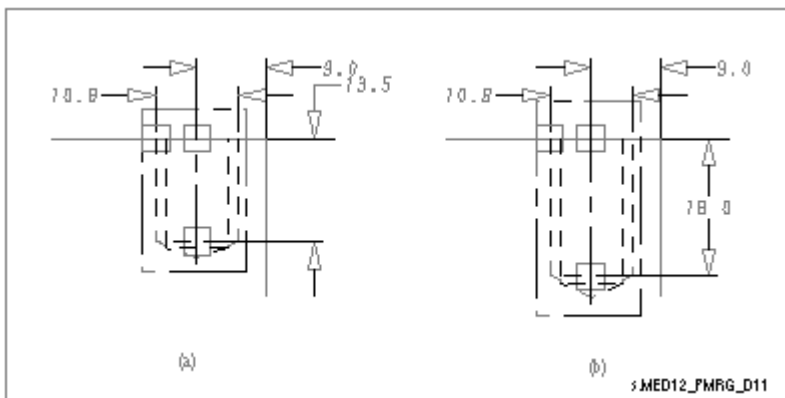
Figure 95 Group with Two Prims



Transforming Using Three Prims

If you add a third PPG prim as shown in (a) below, the length and width of the bolt hole can be changed independently. The third prim is placed at a grid line intersection by dimensioning the width of the bolt hole. The result of lengthening the bolt hole is shown in (b). The hole has been stretched in the Y-axis but left unchanged in the X-axis. Note that this may still not be exactly what you want as the angles in the geometry at the bottom of the bolt hole will have changed.

Figure 96 Group with Three Prims

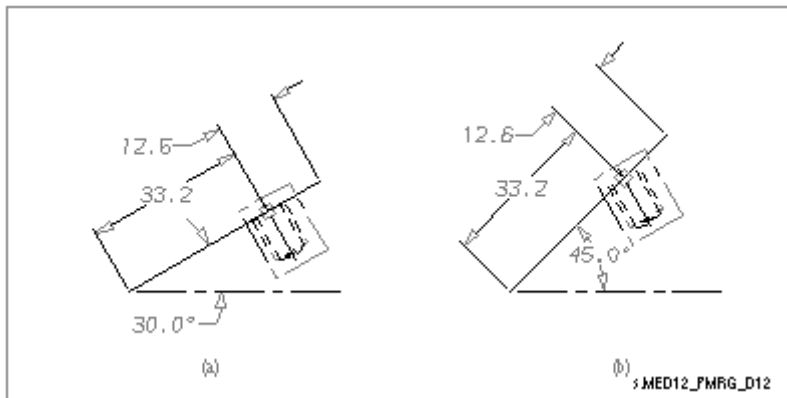


Rotating Parametric Groups

Using two or three prims, you can rotate geometry within parametric groups. This allows you to keep geometry within parametric groups at the same orientation as the geometry outside the group during parameterization.

Drawing (a) in the figure below shows a bolt hole on a part of an object that is to be rotated. With only one prim in the group, the hole will be translated as shown in (b) where the hole is no longer at the same orientation as the rest of the object.

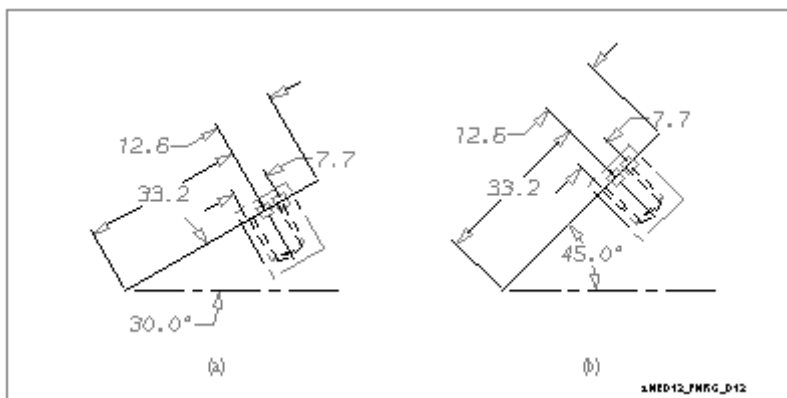
Figure 97 Parametric Group with One Prim



Rotating Groups Using Two Prims

The following figure shows that by placing two prims inside the parametric group, the hole rotates with the rest of the geometry.

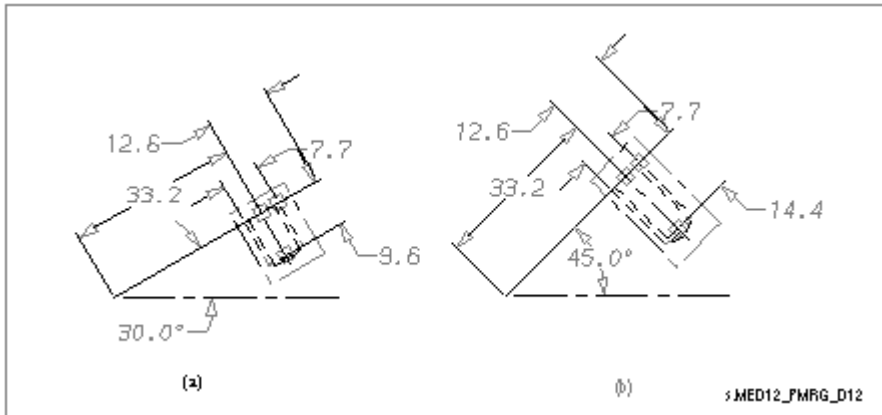
Figure 98 Parametric Group with Two Prims



Rotating Groups Using Three Prims

In the following figure, three prims in the parametric group allow the bolt hole to be scaled differentially as well as rotated.

Figure 99 Parametric Group with Three Prims

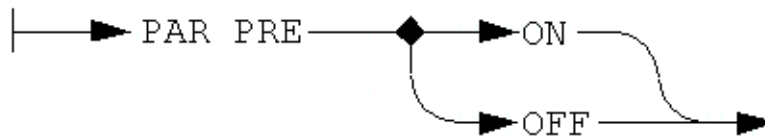


The PAR PRE Command

When you create a parametric group, points within the group line that are explicitly dimensioned are moved during parameterization according to the parameters specified in the dimension and not according to how the group moves.

Using the `PAR PRE` switch you can change this so that all dimensioning within the groups will be ignored and explicitly dimensioned points within a parametric group will move according to the group and not the dimension. The command syntax for changing the `PAR PRE` switch is shown below.

Syntax



Option	Description
OFF	When <code>PAR PRE</code> is OFF, all dimensioning within parametric groups is ignored.
ON	With <code>PAR PRE</code> ON, points within the parametric group line that are explicitly dimensioned are moved according to the dimension and not according to how the group moves. This is the default setting.

CHANGING ELEMENT TYPES

This chapter describes how to change the default element types used in MEDUSA Parametric Design with the PAR DDL and TBL DDL commands.

- [Parametric Design Element Defaults 152](#)
- [Querying Element Defaults 153](#)
- [Changing Element Types - PAR DDL 154](#)
- [Changing Table Element Types - TBL DDL 158](#)

Parametric Design Element Defaults

The variables listed below are the Parametric Design system element type defaults. You can change these defaults using the PAR DDL command, which is described under section [“Changing Table Element Types - TBL DDL”](#) on page 158.

Variable	Description	Default
ACO	Instance clump command text	SCO
ARP	Instance clump attachment point text	SAT
BOX	Viewbox line	LPV
BSL	Baselines	LBL
COL	Lines created by PAR GRIS COL	L3
COM	In-sheet command text	TCO
CIR	Lines created by PAR GRIS CIR	L3
DAT	Datum point prim	PVG, DXY, DYZ, DXZ DYX, DZY, DZX
ERR	Line and text types used for error messages	
FIL	Lines created by PAR GRIS FIL	STK
GPD	Parametric group datum prim	PPG
GPL	Parametric group line	LPG
INF	Lines created by PAR GRIS INF	L3
INS	Instance clump name text	SPS
NDA	Attachment point text	ATP
NEW	NEW grid lines	STK
OLD	OLD grid lines	STK
POT	Potential grid lines	STK
POI	Baseline point functions	
TAN	Lines created by PAR GRIS TAN	L3

Querying Element Defaults

Use Q PAR DDL to display the current Parametric Design element defaults.

Example

When you give the command Q PAR DDL, a complete list of current element type defaults is displayed. For example:

```
*
*qparddl

Parametrics element types :-
  BOX - LPV
  DAT - LBL
  NDA - PVG DXY DYZ DZX DYX DZY DXZ
  COM - ATP
  INS - TCO
  ARP - SPS
  ACO - SAT
  GPL - SCO
  GPD - LPG
  SLN - PPG
  SPM - SWL
  POT - SWP
Grid lines :-
  POT - STK Layer: 99 Width: 3.00
  OLD - STK Layer: 99 Width: 3.00
  NEW - L4 Layer: 99 Width: 0.00
  BAS - LBL Layer: 16 Width: 0.00
  COL - L3 Layer: 17 Width: 0.00
  CIR - L3 Layer: 17 Width: 0.00
  TAN - L3 Layer: 17 Width: 0.00
  INF - L3 Layer: 17 Width: 0.00
More?y
  FIL - STK Layer: 99 Width: 3.00
Errors :-
  Input      : TS1
  Output     : TR1
  Line Type: L6
  Layer      : 99
  Width      : 0.00
Point functions :-
  Intersection : 11
  Tangent      : 12
  Perpendicular: 10
  Circle center: 26
```

Changing Element Types - PAR DDL

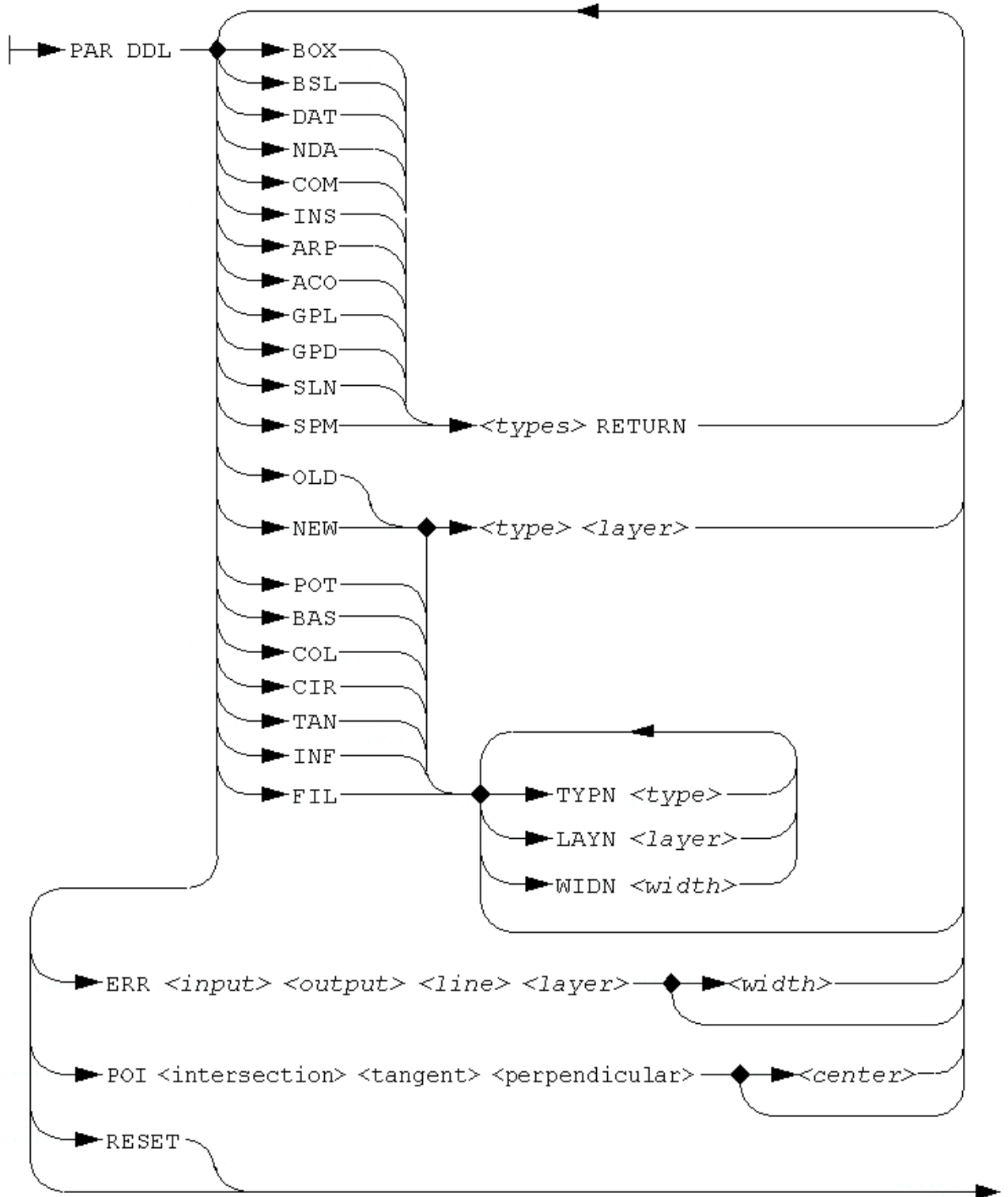
Use PAR DDL to redefine the element types that are used in parametric constructions. You can redefine the type, and for grid lines also the layer and width, used for output from Parametric Design commands such as PAR GRIS. The syntax graph for PAR DDL is shown on the next page.

Any element types you define using PAR DDL replace the default element types for constructions shown under section [“Parametric Design Element Defaults” on page 152](#).

General Parametric Design Elements

The following series of options define various Parametric Design element types.

Option	Default	Description
BOX	LPV	Viewbox line
BSL	LBL	Baselines
DAT	PVG	Datum point prim DXY DYZ DXZ DYX DZY DZX
NDA	ATP	Attachment point text
COM	TCO	In-sheet command text
INS	SPS	Instance clump name text
ARP	SAT	Instance clump attachment point text
ACO	SCO	Instance clump command text
GPL	LPG	Parametric group line
GPD	PPG	Parametric group datum prim
SLN	SWL	Static window line. This is a deprecated line type: use a parametric group line instead.
SPM	SWP	Static window datum prim. This is prim: use a parametric group datum prim instead. types --> One or more valid element types



Grid Line Elements

The following series of options define Parametric Design grid line elements.

Option	Default	Description
OLD	STK	Old grid lines
NEW	STK	New grid lines type --> Any valid element type, for example TS1, LPG, L5. layer --> The number of the required layer. Must be an integer.
POT	STK	Potential grid lines
COL	L3	Lines created by PAR GRIS COL
CIR	L3	Lines created by PAR GRIS CIR
TAN	L3	Lines created by PAR GRIS TAN
INF	L3	Lines created by PAR GRIS INF
FIL	STK	Lines created by PAR GRIS FIL TYPN <i>type</i> A valid element type, for example TS1, LPG, L5. LAYN <i>layer</i> The number of the required layer. <i>layer</i> must be an integer. WIDN <i>width</i> Width of a superline. <i>width</i> is a signed real value expressed as a decimal, fraction, or an exponential.

Point Functions and Error Messages

The following options define baseline point functions and error message elements.

Option	Description
POI	Baseline point functions: intersection Intersection point function. The default is 11. tangent Tangent point function. The default is 12. perpendicular Perpendicular point function. The default is 10. center Center point function. The default is 26.
ERR	Line and text types used for error messages: input The text type used for input errors, that is, errors concerning the new grid. The default is TS1. output The text type used for output errors, that is, errors concerning the new parameters you have specified. The default is TR1. line The line type used to indicate bad constructions. The default is L6. layer The layer used for error messages. This defaults to 99. width The width of the error lines if you defines a superline in line.

Element Type Defaults

RESET: Resets all element types to the default.

PAR DDL Example

The following example shows several element types being changed using PAR DDL:

```
*PAR DDL BOX LGR
*OLD L3 52 NEW STK 52 3.0
*PAR DDL POI 8 12 18 24
*FIL LAYN 100 TYPN L4
*
```

Changing Table Element Types - TBL DDL

Use the TBL DDL command to redefine the element types used in tables. Any element types you redefine in this way replace the types currently in use.

Syntax

| → TBL DDL → <line_type> <name_type> <row/column> →

Variable	Description
<i>line_type</i>	The line type used for the table boundary line. The default is LTB.
<i>name_type</i>	The text type used for the table name text. The default is TTB.
<i>row/column</i>	The text type used for the row and column name texts. The default is TRC.

Example

```
*TBL DDL L6 T2 T3*
```

MECHANISMS

This chapter describes how to simulate the movement of parts or objects on a MEDUSA sheet by repeatedly parameterizing an object and changing selected dimensions. This allows you, for example, to investigate potential clashes between different parts of a mechanism.

- [Simulating Movement](#) 160
- [Running a Mechanism](#) 162
- [Preparing the Definition Sheet](#) 163
- [Linear Motion](#) 166
- [Rotary Motion](#)..... 169
- [Text Variable Commands](#) 170
- [Using Programs to Control Mechanisms](#) 171
- [Plotting Motion Simulations](#)..... 173
- [Examples](#) 174

Simulating Movement

Parametric Design allows you to simulate movement by repeatedly parameterizing geometry. You set a dimension variable and then repeat the following sequence as many times as required:

1. Parameterize the geometry temporarily with PARS CAN.
2. Increment the dimension variable.

This allows you to produce several versions of a single drawing, each representing a stage of motion.

Simulating Different Types of Motion

The type of motion simulated may be linear, rotary, or a combination of the two:

- Linear movement is simulated by moving geometry relative to a fixed point
- Rotary movement is simulated by moved geometry around a fixed or moving center

To repeat the sequence of commands needed to simulate motion you can do any of the following:

- Use the macros provided with the Parametric Design menus
- Execute a text variable command
- Execute a `Bacis1` program

Setting Parameters

The following parameters are used in motion simulation:

- A *controlling variable*, an angular, a linear, or a radial/hole dimension whose value changes with each parameterization operation
- An *increment setting*, the amount by which the controlling variable is incremented each parameterization operation

The starting value of the controlling variable must be specified before parameterization begins. The increment can be a positive or negative value.

Running the Simulation

There are three ways of running a motion simulation once you have prepared your sheet and specified the variables:

- One step at a time
- For a specified number of steps
- Continuously (until you decide to stop)

Single step : To run a motion simulation one step at a time, increment the controlling variable by the amount you set parameterize the prepared geometry with the variable. If you cancel the PARS command immediately, the scaling is not permanent, although the variable will be incremented by the specified amount.

Several steps : To parameterize the sheet a specific number of times, use a program. Each PARS command in the program must contain a CAN command. When the program ends, REDRA displays the original object only. An example of using a program to parameterize a mechanism is given later in this chapter under section [“Parameterizing the Loader Mechanism Repeatedly”](#) on page 181.

Continuously : To parameterize a sheet continuously, use either a program or a text variable command. If you use a text variable command, use the following procedure to stop parameterization:

- `Control-p` for Prime
- `Control-c` for Sun and VAX

The process is aborted at the earliest possible opportunity, and automatically cancelled. When you refresh the screen with a REDRA command, the original object definition is drawn.

Running a Mechanism

Use the following general procedure for motion simulation:

1. Ensure that all parts of the mechanism definition are inside the parametric viewbox.
2. Place an in-sheet command inside the viewbox to prevent dimensioning, in-sheet commands and dynamic baselines from being redrawn each time the drawing is parameterized. For example:
`LAY 4 14 UNTRN INVIS`
If you have included any parametric groups in the drawing, make them untransformable.
3. Test the drawing by clearing the workstation screen with the CLE command and then use PARS CAN to check for problems in the mechanism definition. Correct any errors before continuing.
4. Replace the dimension which is to change and any other dimensions with variable names.
5. Specify all variables using LET commands.
6. Test the drawing again using PARS CAN.
7. Set the PAR UND switch to OFF.
8. Clear the workstation screen using the CLE command.
9. Execute a text variable command or program to repeatedly parameterize the drawing.

Preparing the Definition Sheet

You can use the Parametric Design system to simulate the movement of a mechanism drawn on any MEDUSA sheet. Before you can start simulating motion, however, you must prepare the definition drawing in the following way:

1. Specify any parameters that will change during parameterization.
2. If necessary, add dynamic baselines to connect reference points on rotating linkages.
3. Add in-sheet commands to place dimensioning on an invisible or deletable layer.
4. Make sure the `PAR UND` switch is `OFF`.

Drawing and Dimensioning the Geometry

Dimension the object as you would for regular parameterization. Enclose parts of the object that will not move within a parametric group or leave their dimensions unchanged. Make sure that all points required to move are at grid line intersections.

Setting up Variables and Expressions

Replace the text of dimensions that will change during parameterization with variable names or expressions, for example:

- `LEN`
- `ANG + 15`
- `LEN * .5`

Specify a starting value for each variable name before you test the drawing. Full details of how to use variables and expressions are given in [“Variables and Expressions” on page 77](#).

Using Dynamic Baselines

Many mechanisms involve two or more rotating linkages. The position of a linkage at any one time usually depends on the position of the linkage to which it is connected and you can use dynamic baselines to connect the reference points on each linkage. Refer to [“Geometric Constraints” on page 47](#), for more information about using dynamic baselines.

Making Dimensions Invisible

Place dimensioning on an invisible or deletable layer to clarify each stage of a mechanism during repeated parameterization. Use an in-sheet layer command, for example:

```
LAY 4 INVIS
```

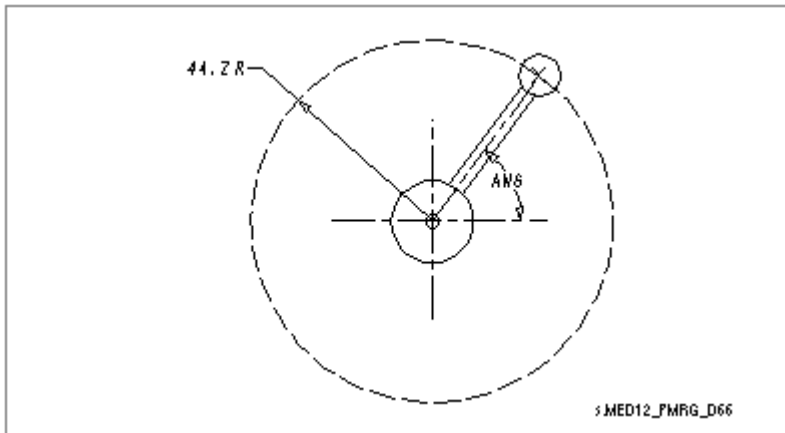
Parameterization is slightly quicker if dimensioning is made invisible.

Adding Lines

Adding lines to the mechanism definition can help to create the dimensions necessary for repeated parameterization. For example, it may be necessary to add a horizontal or vertical line to a drawing so that you can specify a variable angle. Place all additional lines on a layer where they can be easily deleted or made invisible during parameterization.

You may wish to use additional lines to improve visualization of the motion to be simulated. For example, when simulating rotary movement, it is helpful to construct a dummy circle that connects the center of rotation with a point on the outline of the object or with the corresponding dynamic reference point. When used to show the path of a dynamic reference point, the dummy line also serves to dimension the reference point radially. An example of this is shown in the following figure.

Figure 100 Using Additional Lines



Undrawing the Image

To monitor the movement of an object, the parameterized version of the master drawing created by each parameterization must remain visible on the screen. This allows you to see and investigate potential clashes between different parts of an object or mechanism.

The parametric switch `PAR UND` controls undrawing (erasing) of the image. The default setting for this switch is `ON`, so that when an object is repeatedly parameterized, the old image is removed when the new image is drawn. When the `PAR UND` switch is `OFF`, the old image remains visible. You can then see all stages of movement at the same time.

Cancelling the Parameterization Command

When simulating movement you must use the `PARS CAN` command. If you parameterize the geometry permanently, the parameterized version of the master will be used as the input for the next parameterization. This may cause errors as geometry produced by parameterization often cannot itself be parameterized.

Linear Motion

You can simulate linear movement by parameterizing geometry so that the geometry moves relative to a fixed reference point. The reference point can be either a pair of static baselines, an orthogonal view prim or a PVG prim.

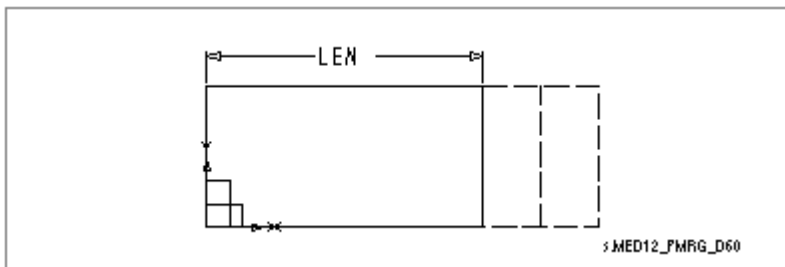
Magnification

To simulate magnification of an object in one or more directions, place a reference point on part of the geometry being parameterized and replace dimensions with variables or expressions. The reference point will not move during parameterization and therefore repeated parameterization will simulate magnification of the object in one or more directions rather than movement.

Example

The following figure represents a rectangle that has been repeatedly parameterized. The value of the variable LEN is increased each time, while the lower left-hand corner of the rectangle remains stationary because of the prim. The dashed lines show where the right edge of the rectangle is redrawn each time LEN is incremented.

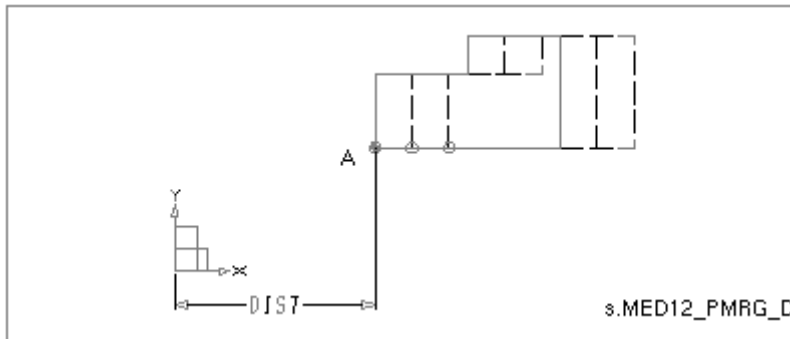
Figure 101 Magnification in One Direction



Movement in One Direction

To simulate movement in one direction, replace a dimension in the direction required with a variable or expression. The following figure shows how the whole object geometry moves when the value the variable, $DIST$, is increased.

Figure 102 Linear Motion



DIST specifies the position of Point A relative to the reference point (a DXY prim). By increasing *DIST* with each PARS command, the rectangle moves to the right. If *DIST* were decreased, the rectangle would move to the left.

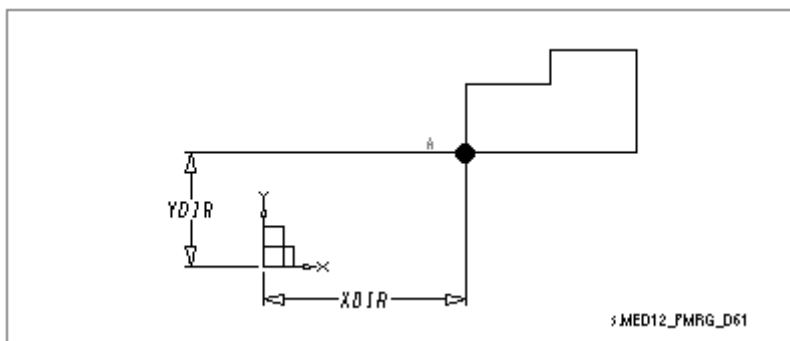
Movement in Two Directions

You can simulate movement in two directions in two different ways:

- By replacing a dimension in each direction with separate variables or expressions, as in the following figure.
- By replacing a dimension in one direction with a variable and a dimension in the other direction with an expression containing the same variable, as in [Figure 104, “Varying a Single Parameter” on page 168](#).

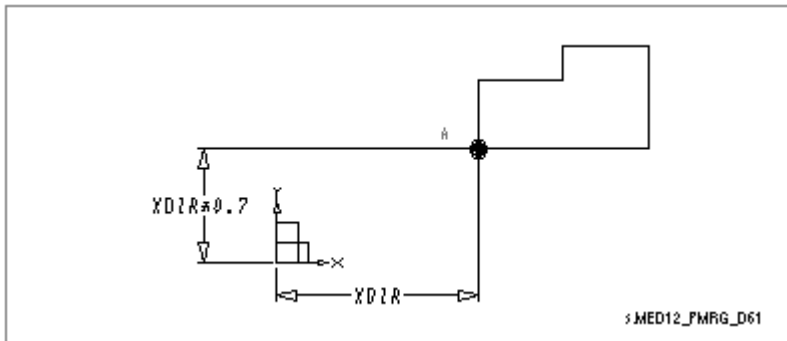
In the following figure, the aim is to move the rectangle in both the X and Y axes. To do this you must increment the two variables, *XDIR* and *YDIR*, which specify the position of the geometry point A relative to the prim.

Figure 103 Varying Two Parameters



In the following figure, any change in the value of the variable *XDIR* will result in movement in both directions because one of the dimensions is expressed in terms of the other.

Figure 104 Varying a Single Parameter



Rotary Motion

You can simulate rotary movement by moving geometry around a fixed or moving center. This is achieved by repeatedly parameterizing geometry containing an angle dimension. The angle must be specified as a variable or expression whose value changes each time the geometry is parameterized. As with linear motion, you can change any of the other mechanism dimensions into variables.

You might use rotary motion simulation to represent an object turning around a fixed center, for example, the movement of a crank and crank arm.

Rotation About a Fixed Center

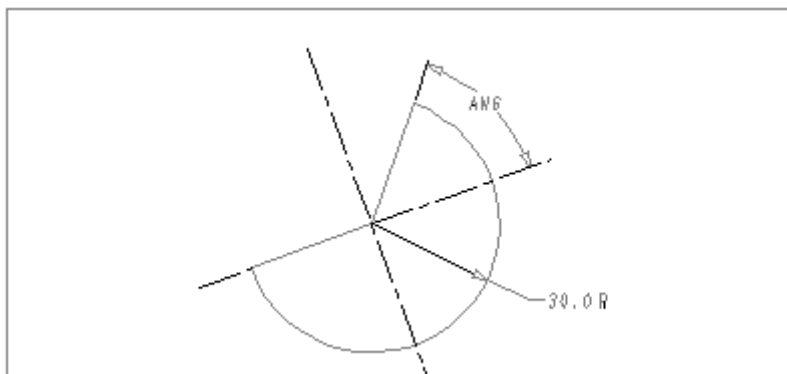
Use the following procedure when you define an object that is to be rotated about a fixed center:

1. Draw the object geometry.
2. Place a reference point at the point around which the object is to be rotated.
3. Dimension the drawing and add additional lines to the drawing if necessary (see ["Preparing the Definition Sheet"](#), ["Adding Lines"](#) on page 164).
4. Rotate the object and baselines to avoid the constraints imposed by horizontal and vertical baselines. Do not use an angle of 30, 45, or 60 degrees as these are considered as special cases by the system.
5. There must be at least one angle dimension between one of the static baselines and a point on the outline of the object. If such an angle dimension is not present, add one.
6. Perform steps one through nine of the general procedure described under section ["Running a Mechanism"](#) on page 162.

Example

The following figure shows a fully dimensioned cam that can be rotated about its center. The angle of rotation is specified by the angle ANG. Note that the cam and the static baselines have been rotated through an arbitrary angle to avoid the constraints associated with horizontal and vertical baselines.

Figure 105 Rotation About a Fixed Center



Text Variable Commands

Using a text variable command, you can repeatedly perform two operations:

- Temporarily parameterize geometry
- Increment the controlling variable

Only one variable, the controlling variable, can be incremented using a text variable command. This transforms a critical dimension, for example the angle in the cam in [Figure 105, “Rotation About a Fixed Center” on page 169](#). Dimensions that are functions of the controlling variable may also be transformed during parameterization, for example, $ANG + 50$ or $ANG * .5$. You cannot change any other dimension variables with a text variable command.

Creating a Text Variable Command

A text variable command should be created and stored in a buffer. Use either the input text buffer (which is also used when creating or editing text) or create an exclusive buffer:

- Commands stored in the input text buffer are executed when you give the command XQT. For example:

```
/PARS CAN LET ANG = (ANG - 10) XQT
```

- An exclusive buffer is a variable name containing a text variable command. For example, an exclusive buffer ABC might be declared in the following way:

```
LET ABC = 'PARS CAN LET ANG = (ANG - 10) XQT ABC'
```

Executing a Text Variable Command

The XQT command executes a text variable command.

- For input text buffer commands, type XQT then press Return.
- For text variable commands stored in an exclusive buffer, type XQT followed by the variable buffer name, for example XQT `buffer_name` then press Return.

Once started, a text variable command will run the commands in the text buffer continuously until you stop it.

Interrupting a Text Variable Command

To stop a text variable command loop, use the key sequence described under section [“Simulating Movement”, “Continuously” on page 161](#). If you stop a text variable command in this way and then want to restart it, you must first refresh the screen with a REDRA command and then enter a new starting value for the variable.

Querying the Content of a Text Buffer

To display the contents of text buffers type QVAR `TEX` for the input text buffer and QVAR `buffer_name` for an exclusive buffer.

Using Programs to Control Mechanisms

Programs are a powerful tool for simulating movement using Parametric Design. Using programs you can:

- Change several dimension variables at once
- Use logical operators during parameterization

Creating a Program

You can create a program in two ways:

- Using your operating system text editor
- Using MEDUSA program mode

Details of how to create programs in MEDUSA program mode are given in the *MEDUSA Basics 1 Guide*.

Running a Program

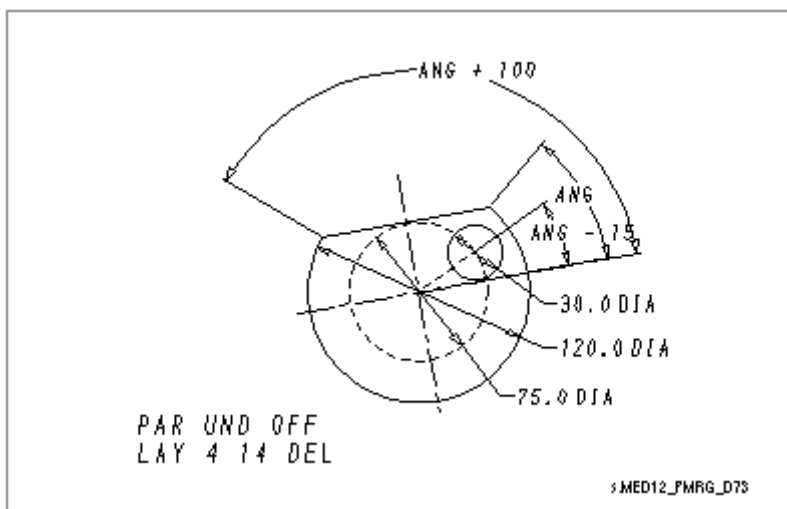
When you run a program to parameterize an object, first clear the screen with the CLE command and then type RUN followed by the name of the program. For example:

```
*RUN MECH1.PRG Return
*
```

Example

The drawing shown in the following figure is ready to be parameterized using a program which will rotate the component several times, increasing the value of the variable ANG by 30 degrees each time.

Figure 106 Cam Definition Drawing



To begin with, the program sets the value of the variable `ANG` to 40 degrees. The `PARS CAN` command is inside a loop which parameterizes the component until `ANG` is greater than 100 degrees.

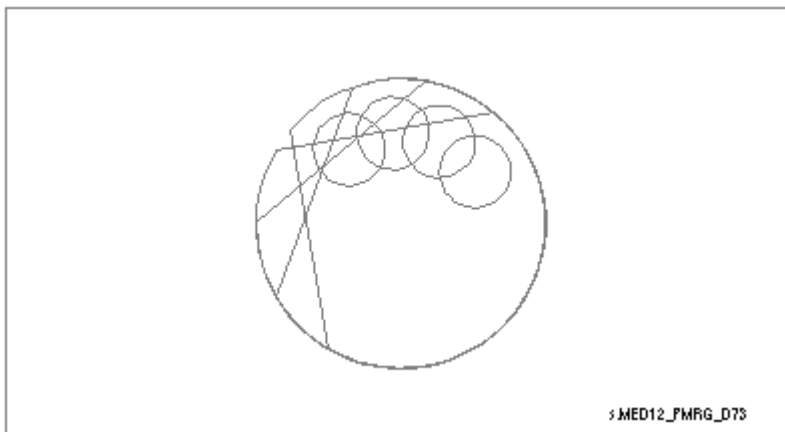
```
10 LET ANG = 40
20 LOOP
30 PARSCAN
40 BREAK IF (ANG.GT.100)
50 LET ANG = (ANG + 30)
60 ENDLLOOP
70 ENDRUN
```

With each loop, the program does the following:

1. Parameterizes the component temporarily with the `PARS CAN` command
2. Tests whether `ANG` is greater than 100 using the logical operator `.GT`. The result of the test is true or false:
 - If false, `ANG` is increased by 30 and the loop is reentered.
 - If true, the loop is exited and the program stops.

The following figure shows the result of repeated parameterization using the program above.

Figure 107 Result of Repeated Parameterization



Plotting Motion Simulations

It is not possible to plot the screen display to show all the movement of the mechanism. You can only plot the master drawing that is used for each parameterization.

The reason for this is that you parameterize the drawing using PARS CAN, not PARS. The master drawing is redrawn with new dimensions, but the new dimensions are cancelled in memory. When you plot a sheet, you get the information in memory, not the information on the screen.

It is not possible to use PARS instead of PARS CAN in command loop. The PARS command must be cancelled, otherwise errors would be caused. Using PARS, the drawing produced by one parameterization would be used as input for the next. This causes problems as there is no guarantee that a drawing produced by parameterization is itself capable of being parameterized. A drawing that is the result of parameterization may contain geometric constraints, such as coincident points or horizontal and vertical lines, which prevent it from being parameterized

Getting a Screen Dump

One way to get around this problem is to get a screen dump. A screen dump can only be obtained if there is a hardcopy device attached to your workstation.

Creating a Composite Drawing

If you cannot get a screen dump, you can create a composite drawing of the mechanism using the following procedure.

1. Increment the variable in the master drawing.
2. Parameterize the drawing permanently.
3. Save the sheet as a symbol.
4. Increment the variable on the master drawing by a further step.
5. Parameterize the drawing again.
6. Save the sheet as a symbol, using a new filename.

Continue this procedure until you have a symbol representing each stage in the movement of the mechanism. Then load all the symbols onto the same sheet to form a composite drawing which can be plotted.

Examples

The examples in the following sections show both linear and rotary motion.

Piston mechanism : The first example uses a piston mechanism which uses both linear and rotary movement. Static baselines define the reference point and a text variable command is used to repeatedly parameterize the mechanism.

Loader mechanism : The second example is of a lift arm assembly for a loader shovel bucket. The rams, linkages etc are stored as parametric symbols and are loaded onto the sheet during parameterization using instance clumps. All but two of the dimensions are constant, and a program is used to repeatedly change the values of the two variable dimensions.

Each example shows:

- The basic mechanism definition
- The method used to repeatedly parameterize the mechanism (text variable command or program)
- An illustration of the mechanism following repeated parameterization

Procedure

When defining a mechanism that involves more than one type of motion, use the following procedure:

1. Create and dimension the basic mechanism. Do not include any detail.
2. Define starting values for each of the variables and test the mechanism with PARS CAN.
3. Create a text variable command or program to change the values of the variables.
4. Test the mechanism using the commands in the text variable command or program. Do not proceed until the mechanism can be repeatedly parameterized successfully.
5. If required, make adjustments to the mechanism and repeat the previous step.
6. If required, edit the text variable command or program and repeat step 4.
7. Add details to the basic mechanism and fully dimension them.
8. Verify that the mechanism will still parameterize and correct any errors.
9. Execute the text variable command or program to move the mechanism.

Piston mechanism

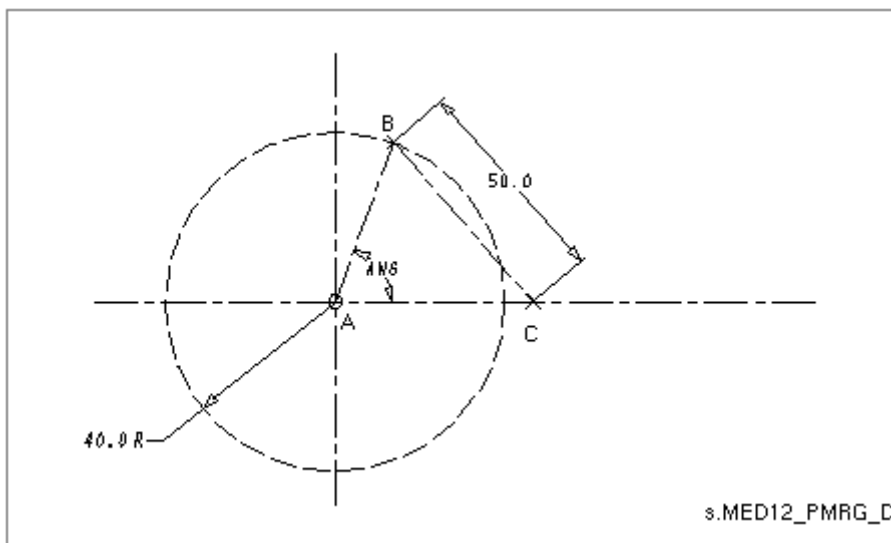
In this example a piston mechanism shows both linear and rotary movement. By repeatedly changing the value of the dimension *ANG*, the line segment *AB* rotates about Point *A*. The movement of Point *C* is constrained in two ways:

- It is always at a fixed distance from Point *B*
- It can only move horizontally because a horizontal static baseline passes through it

The basic mechanism shown in the following figure contains the following important points:

- A single static reference point defined by intersecting static baselines
- Two dynamic baseline segments define three further points:
 - *A* is a circle center point (FUNV 26) defining the center of rotation
 - *B* and *C* are intersection points (FUNV 11)

Figure 108 Basic Mechanism for a Piston



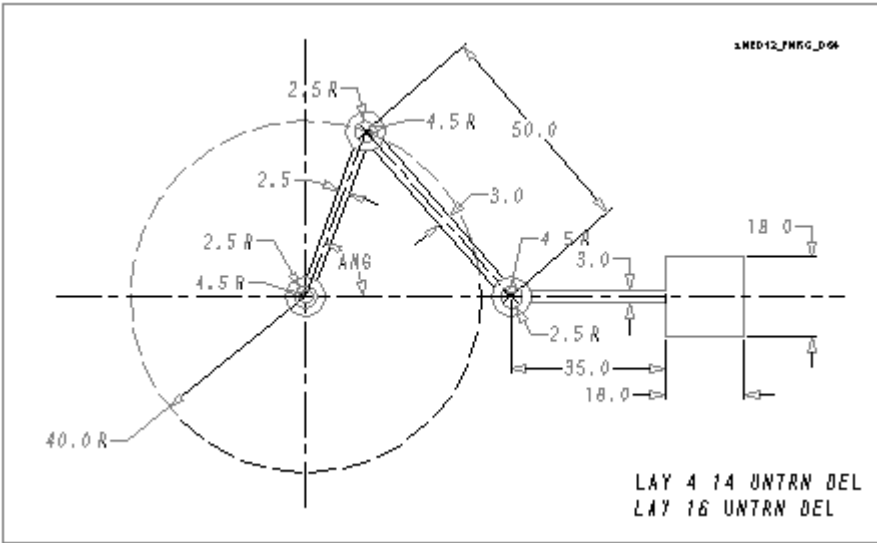
Text Variable Command

The following sequence of commands, using a text variable command, is used to repeatedly parameterize the mechanism:

```
* / PARS CAN LET ANG = <ANG + 10> XQT
* LET ANG = 0
* CLE
* XQT
*
```

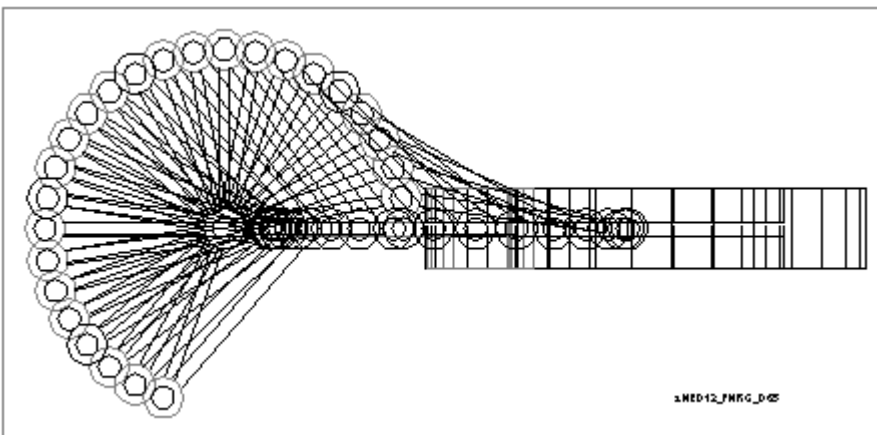
The following figure shows the fully-defined mechanism definition, ready for parameterization.

Figure 109 Detailed Mechanism Definition



The following figure shows a possible result of repeated parameterization.

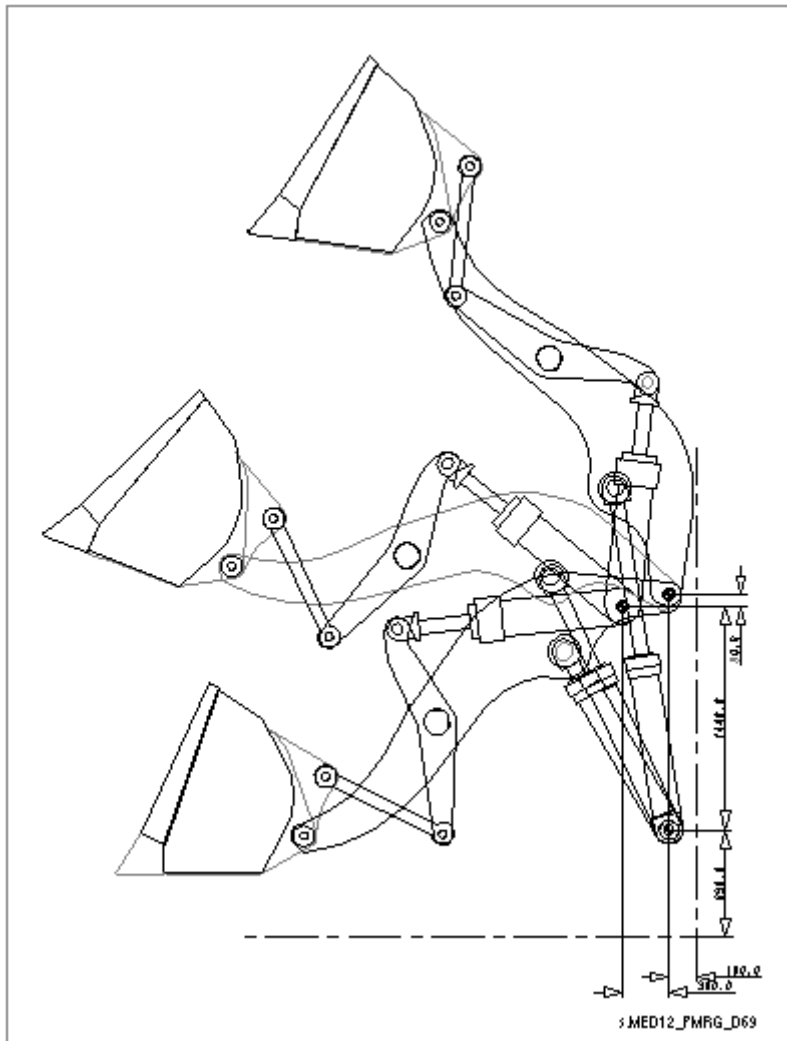
Figure 110 Repeated Parameterization of Piston Mechanism



Loader Mechanism

The following figure shows a lift arm assembly for a loader shovel. When the lift rams are extended, the length of the bucket control ram remains constant while you can manipulate the angle of the bucket floor.

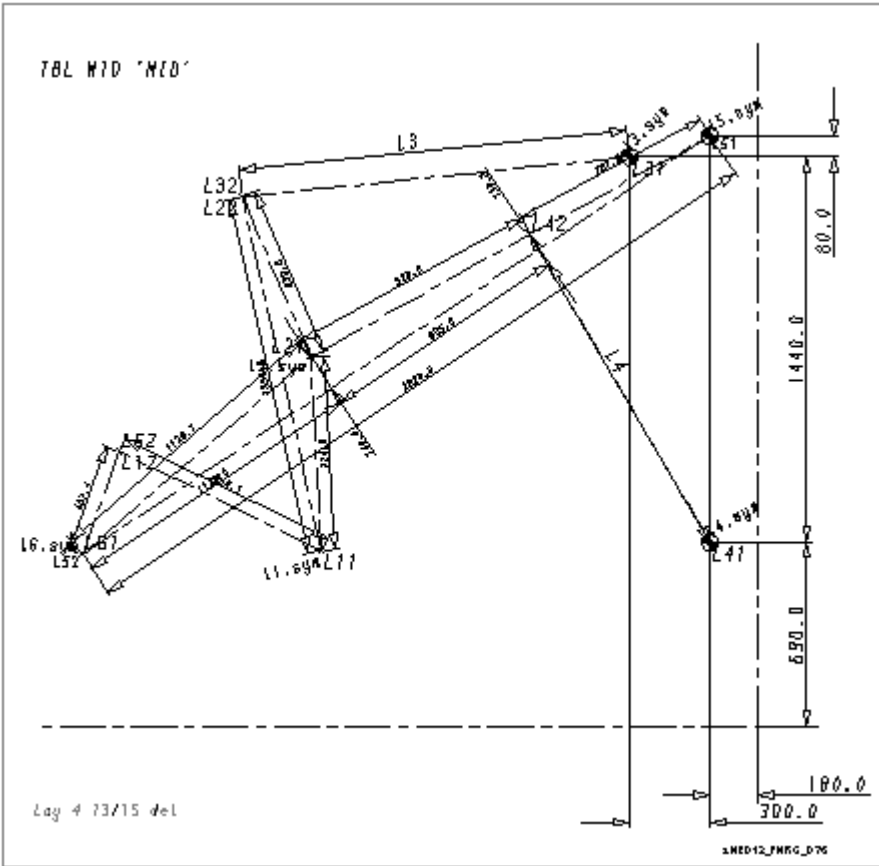
Figure 111 Simulation Showing Lift Arm Assembly Motion



Loader Definition Sheet

The following figure shows a detail from the loader definition sheet. This contains a skeletal outline of the lift arm assembly. All the dimensions are constant with the exception of two identified by the variable names L3 and L4. These variables control the length of the bucket control and lift rams respectively.

Figure 112 Loader Definition



Tables

The following figure shows three tables containing values which can be used to produce a single parameterization of the loader definition drawing. The in-sheet TBL command specifies row MID from the table MID. This sets L3 to 1455 and L4 to 1800 when the drawing is parameterized.

Figure 113 Tables LO, MID, and HIGH

LO		
	L4	L3
MIN	1332	1205
MID	1332	1455
MAX	1332	1705
MID		
	L4	L3
MIN	1800	1205
MID	1800	1455
MAX	1800	1705
HIGH		
	L4	L3
MIN	2232	1205
MID	2232	1455
MAX	2232	1705

1MED12_PMRG_077

Mechanism Components

Drawings of the rams, linkages and other components are stored as parametric symbols and are loaded onto the sheet during parameterization using instance clumps. The following figure shows the symbol definition for the bucket and Figure 115 shows the lever arm.

Figure 114 Bucket Definition

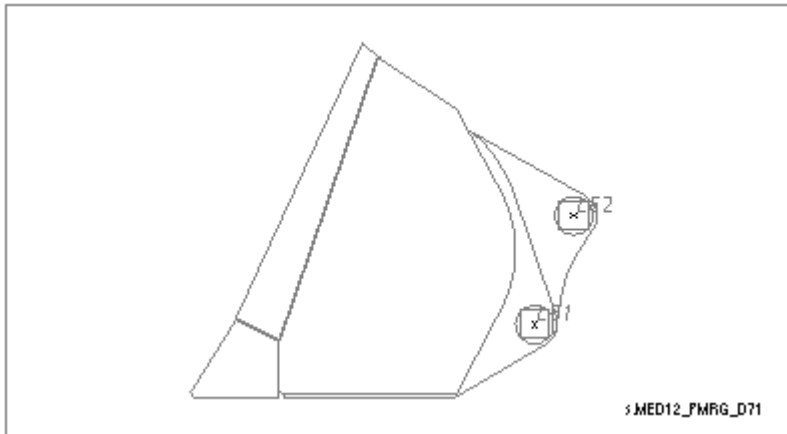
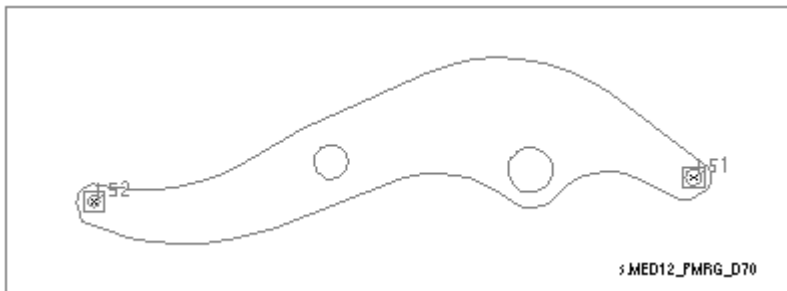


Figure 115 Lever Arm Definition



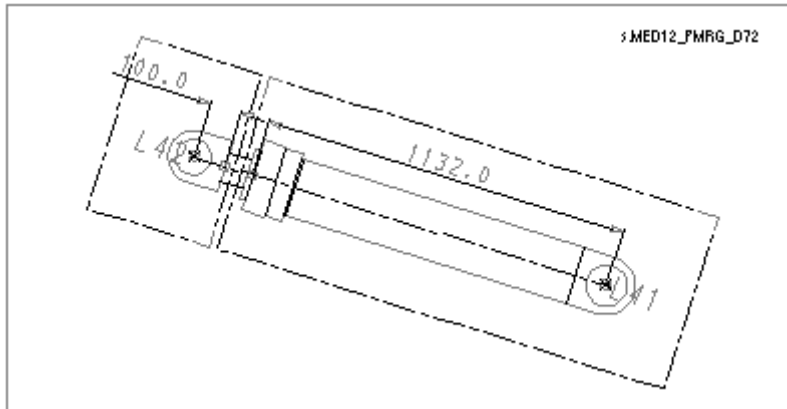
With the exception of the hydraulic rams, the components loaded into the parameterized mechanism are undimensioned parametric groups. The whole symbol has been enclosed in a parametric group line and therefore does not need to be dimensioned as it loaded. In the above figure, there are two parametric group prims, at the same position as the ATP texts. These will be used when the symbol is loaded onto the sheet during parameterization.

Attachment Points

All the component symbols have two attachment points which are used to locate the symbol in the correct position within the mechanism. Attachment point texts (ATP texts) on the symbols correspond to labelled points (SAT texts) of the same name on the skeletal outline shown in [Figure 112, "Loader Definition" on page 178](#).

Symbol definitions for the hydraulic rams have to be both located and scaled within the mechanism drawing. The ram cylinder and rod eye are included in separate parametric groups as they do not need to be scaled when they are loaded. Because they are inside a parametric group line they do not have to be fully dimensioned. The definition for the lift ram is shown below.

Figure 116 Lift Ram Definition



During parameterization the ram rod is extended when loaded into the sheet as the ends of the ram assembly are placed at the appropriate attachment points on the mechanism, which move according to the current value of the variables L3 and L4.

Parameterizing the Mechanism Once

After executing PARS CAN once the parameters specified in the in-sheet TBL command TBL MID 'MID' will move the mechanism to the mid-position for both lift and bucket control rams.

Parameterizing the Loader Mechanism Repeatedly

Using a program that increments a variable after each PARS CAN command, you can both raise the loader arm and tip the bucket. Before doing this, the in-sheet TBL command which sets the value of L4 and L5 permanently (see [Figure 112, "Loader Definition" on page 178](#)) must be either deleted or commented out, to allow one of the variables to be incremented during parameterization.

In the program below, the value of L3 is constant and L4 is incremented after each PARS CAN. This raises the loader arm. [Figure 111, "Simulation Showing Lift Arm Assembly Motion" on page 177](#) gives an idea of what happens when you run this program.

```
20 let l3 = 1455
30 let l4 = 1332
40 parscan
```

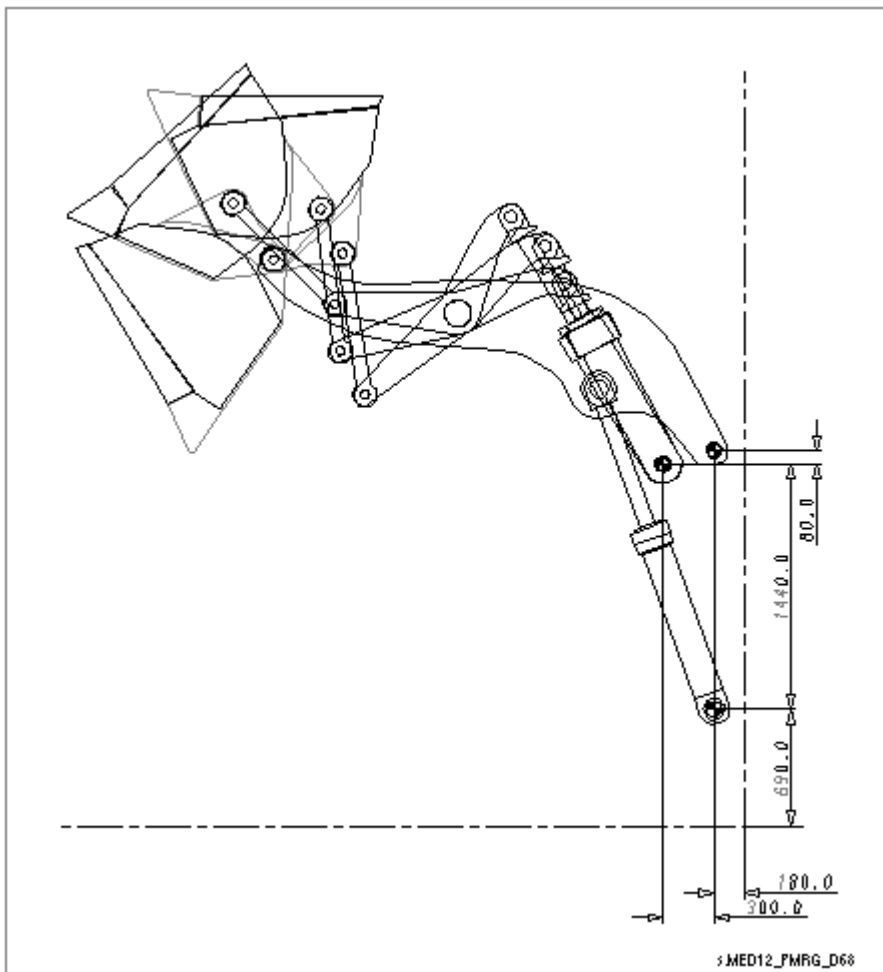
```
50 let 14 = 1650
60 parscan
70 let 14 = 2000
80 parscan
```

A different program can be run which tips the bucket. In this program, L4 remains constant and L3 is incremented after each PARS CAN:

```
20 let 14 = 2000
30 let 13 = 1705
40 parscan
50 let 13 = 1455
60 parscan
70 let 13 = 1205
80 parscan
```

The following figure gives an impression of what happens when you run this program.

Figure 117 Tipping the Bucket



APPENDIX A SUMMARY OF COMMAND SYNTAX

This appendix summarizes all the MEDUSA Parametric Design system command syntax. MEDUSA 2D commands which are used during parameterization, for example, LET and LAY, are not covered here. For more information on 2D commands refer to the *MEDUSA Bacis1 Design Commands Guide*.

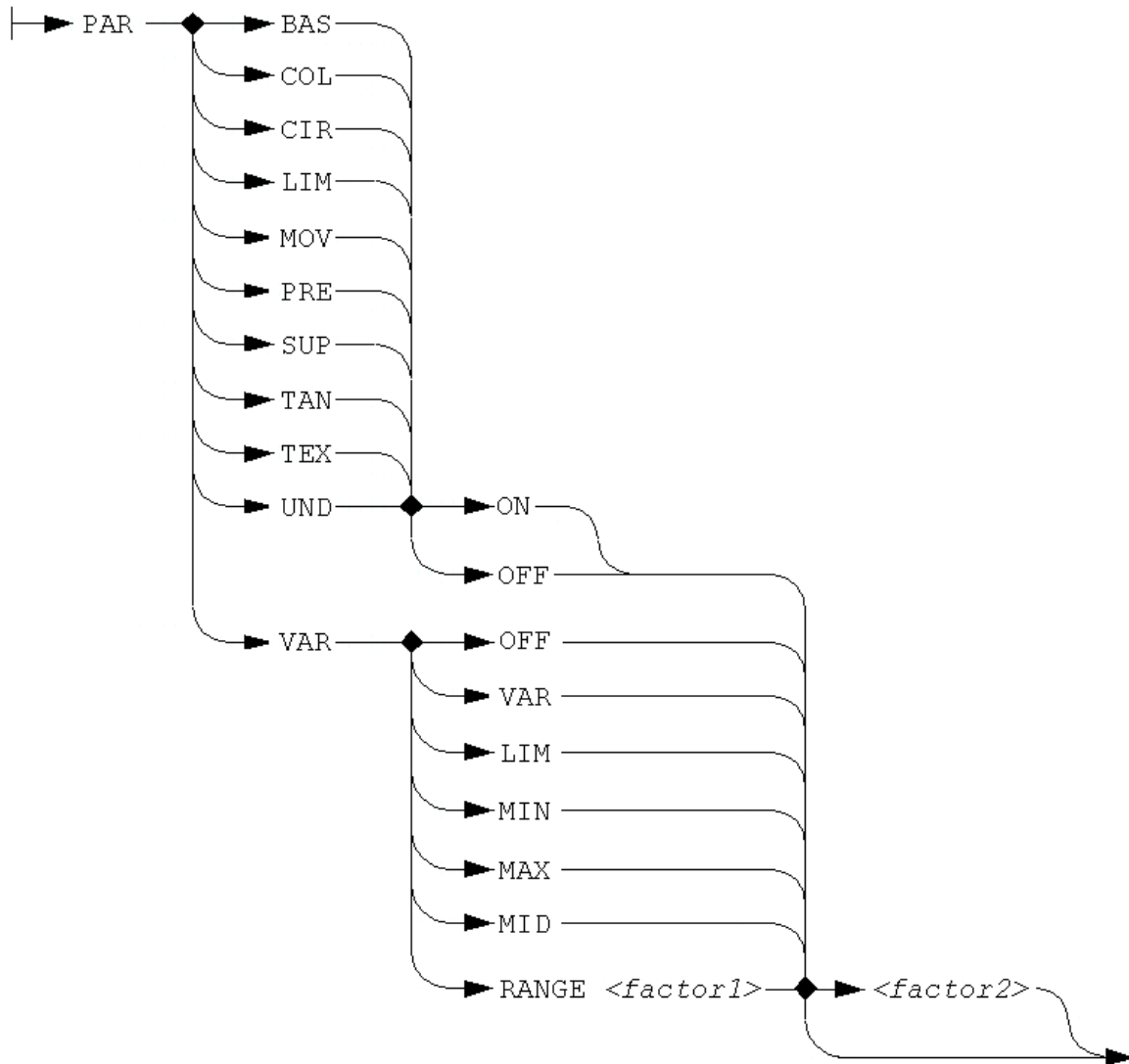
You can issue all Parametric Design commands both interactively and as in-sheet commands except for VER FULL, which can appear only inside instance clumps. A brief description is given of each command, followed by syntax graph and information about command arguments, subcommands and defaults.

- Parametric Switches 184
- PAR DDL..... 187
- PAR DIM 190
- PAR FIL..... 191
- PAR GRIS 192
- PAR LOA..... 193
- PAR SWI 194
- PAR TOL..... 195
- PARS 196
- Q PAR..... 197
- TBL 198
- TBL DDL 199
- VER FULL..... 200

Parametric Switches

There are a number of switches which affect the operation of the Parametric Design system. You can set switches both interactively and using in-sheet commands. The command syntax for changing parametric switch settings is shown below.

Syntax



Options

Option	Default	Description
BAS	ON	If the PAR BAS switch is ON, baselines are automatically inferred, provided that the PAR LIM switch is also ON.
COL	OFF	If the PAR COL switch is ON, non-overlapping collinear line segments are merged to create potential grid lines.
CIR	OFF	If the PAR CIR switch is ON, complete circle grid lines are generated from arcs of circles in the viewbox.
LIM	ON	If the PAR LIM switch is ON, grid lines are limited by the line segments in the viewbox. If the switch is OFF, then baselines cannot be automatically inferred, and fillets cannot be given a default radius.
MOV	ON	If the PAR MOV switch is OFF, elements are not scaled, but dimension texts are replaced by scaled values. If PAR MOV is OFF when loading parametric symbols the symbol will be loaded at the same position it had in the definition sheet.
PRE	ON	The PAR PRE switch allows you to set the precedence rules for parametric groups to those of static windows. When PAR PRE is OFF, points that are dimensioned, fully or partially, within the parametric group are totally unaffected when you parameterize the sheet, just as when using static windows. With PAR PRE ON, static windows take precedence over dimensioning, and dimensioning takes precedence over parametric groups. See "Parametric Groups" , "The PAR PRE Command" on page 149 for more information.
SUP	ON	This switch applies to Prime and VAX systems only. If the PAR SUP switch is ON it enables newer forms of dimension support.
TAN	ON	If the PAR TAN switch is ON, then the tangent lines from tangent point arcs are included in the potential grid lines.
TEX	ON	If the PAR TEX switch is OFF, elements are scaled but dimension texts retain their original values.
UND	ON	If the PAR UND switch is ON, then the drawing definition is undrawn when you parameterize the sheet with the PARS command. This is done before drawing the new object. With PAR UND set to OFF, the original definition remains visible on the screen when you parameterize the sheet.

Tolerance Variation Options

The PAR VAR switch has seven options: OFF, VAR, LIM, MIN, MAX, MID and RANGE. Each of these options specifies a tolerance used by the system to calculate the new dimension value during parameterization.

Options	Description
OFF	Any changes made to the original tolerance text are ignored. The original tolerance appears in the parameterized drawing.
VAR	Any expressions which have replaced original tolerance text are evaluated during parameterization. This is the default setting.
LIM	Converts VAR tolerance dimensions to LIM format.
MIN	Specifies that the lower tolerance is used for all tolerance dimensions.
MAX	Specifies that the upper tolerance is used for all tolerance dimensions.
MID	Calculates the tolerance used for all tolerance dimensions from the following formula: $(MAX - MIN) * 0.5 + MIN$
RANGE	Calculates the tolerance used for all tolerance dimensions from the following formula: $(MAX - MIN) \textit{factor1} + MIN + \textit{factor2}$ <i>factor1</i> must be a factor in the range 0 to 1. <i>factor2</i> is optional.

Further Information

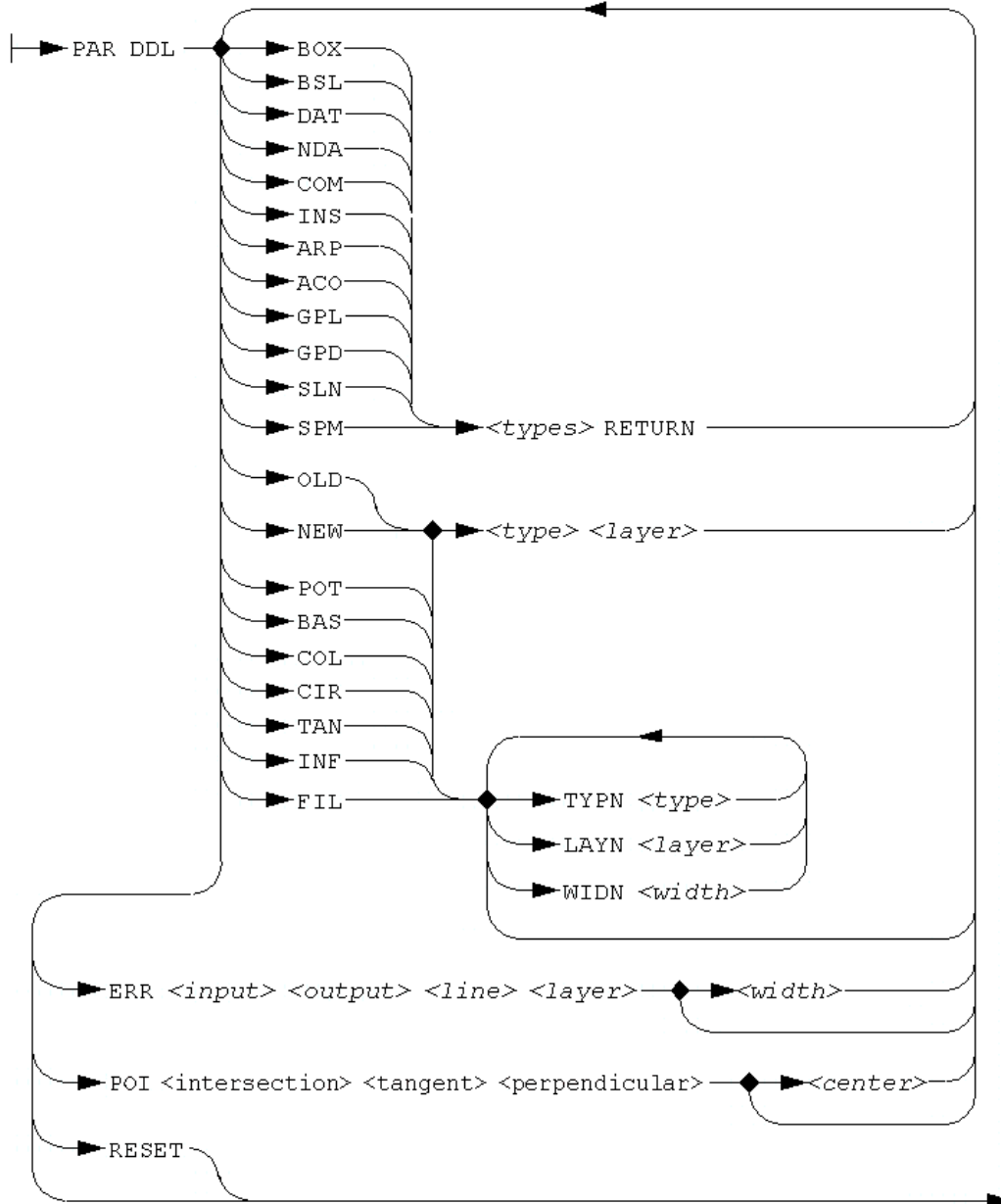
Refer to “[Switches and Layers](#)” on page 95 for examples and further details of how to change parametric switch settings. The PAR VAR switch is described under section “[Dimensioning](#)”, “[PAR VAR Command Syntax](#)” on page 42.

PAR DDL

Redefines the following element types:

- Element types that are used in parametric constructions
- The type, layer, and, for superlines, the width of parametric grid lines
- The line and text types used for error messages
- The point functions used in baselines

Syntax



Further Information

Refer to “[Changing Element Types](#)” on page 151, for more information on using this command.

General Parametric Design Elements

The following series of options define various Parametric Design element types such as view-boxes, baselines and parametric group lines.

Option	Default	Description
BOX	LPV	Viewbox line
BSL	LBL	Baselines
DAT	PVG	Datum point prim: DXY DYZ DXZ DYX DZY DZX
NDA	ATP	Attachment point text
COM	TCO	In-sheet command text
INS	SPS	Instance clump name text
ARP	SAT	Instance clump attachment point text
ACO	SCO	Instance clump command text
GPL	LPG	Parametric group line
GPD	PPG	Parametric group datum prim
SLN	SWL	Static window line. This is a deprecated line type: use a parametric group line instead.
SPM	SWP	Static window datum prim. This is prim: use a parametric group datum prim instead. types: One or more valid element types.

Grid line Elements

The following series of options define Parametric Design grid line elements. Use these options to redefine the type, layer and, for superlines, the width of the grid lines drawn by the PAR GRIS command.

Option	Default	Description
OLD	STK	Old grid lines
NEW	STK	New grid lines <i>type</i> Any valid element type, for example TS1, LPG, L5. <i>layer</i> The number of the required layer. Must be an integer.
POT	STK	Potential grid lines
COL	L3	Lines created by PAR GRIS COL
CIR	L3	Lines created by PAR GRIS CIR

Option	Default	Description
TAN	L3	Lines created by PAR GRIS TAN
INF	L3	Lines created by PAR GRIS INF
FIL	STK	Lines created by PAR GRIS FIL <i>TYPN type</i> A valid element type, for example TS1, LPG, L5. <i>LAYN layer</i> The number of the required layer. layer must be an integer. <i>WIDN width</i> Width of a superline. width is a signed real value expressed as a decimal, fraction, or an exponential.

Point Functions and Error Messages

The following options define baseline point functions and error message elements.

POI	Baseline point functions: <i>intersection</i> Intersection point function. The default is 11. <i>tangent</i> Tangent point function. The default is 12. <i>perpendicular</i> Perpendicular point function. The default is 10. <i>center</i> Center point function. The default is 26.
ERR	Line and text types used for error messages: input The text type used for input errors, that is, errors concerning the new grid. The default is TS1. <i>output</i> The text type used for output errors, that is, errors concerning the new parameters you have specified. The default is TR1. <i>line</i> The line type used to indicate bad constructions. The default is L6. <i>layer</i> The layer used for error messages. This defaults to 99. <i>width</i> The width of the error lines if you defines a superline in line.

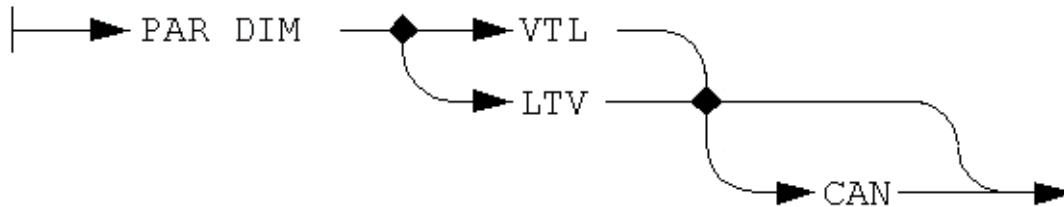
Element Type Defaults

You can reset the standard element types and layers to those used by default when the Parametric Design system is first used with PAR DDL RESET.

PAR DIM

Converts tolerance dimension types. Using PAR DIM, you can convert LIM tolerances into VAR format before parameterization, and then convert them back into LIM format again after parameterization. You cannot parameterize dimensions created with the DIM command option LIM.

Syntax



Option	Description
LTV	Converts LIM tolerance dimensions to VAR format.
VTL	Converts VAR tolerance dimensions to LIM format.

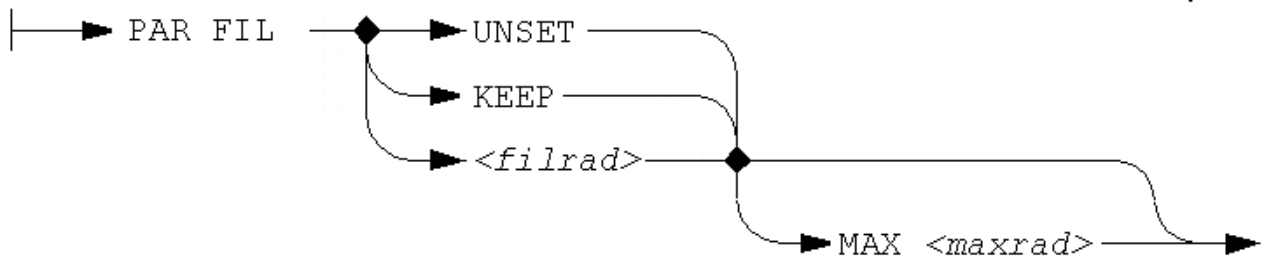
Further Information

Refer to ["Dimensioning"](#), ["PAR DIM command syntax"](#) on page 42 for more information on using this command.

PAR FIL

Defines a default fillet radius. You can use PAR FIL both interactively and as an in-sheet command.

Syntax



Option	Description
UNSET	Specifies that all fillets must be dimensioned explicitly. This is the default.
KEEP	Specifies that all fillets that are not explicitly dimensioned keep their radii unchanged.
<i>filrad</i>	<p>MAX</p> <p>Specifies that only fillets with original radius less than or equal to maxrad should be changed to the specified value. All other fillets must be explicitly dimensioned. If this qualifier is not provided and large radial dimensions are omitted, unexpected results may occur.</p> <p><i>maxrad</i></p> <p>Specifies a maximum fillet radius. The value specified must be a signed real value that is expressed as a decimal, a fraction, or an exponential.</p>

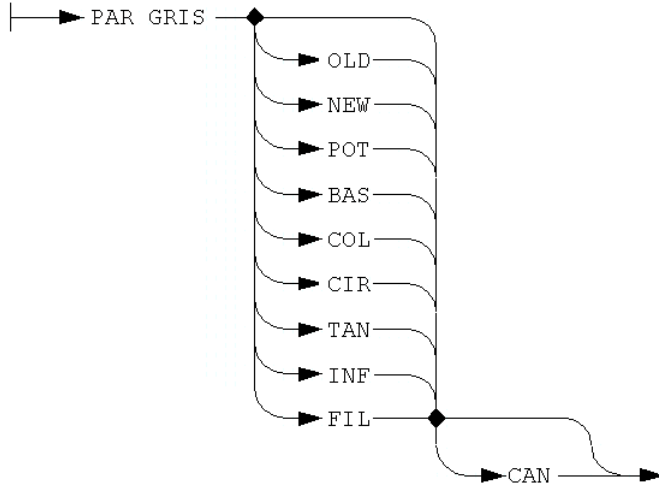
Further Information

Refer to ["Dimensioning"](#), ["Dimensioning Fillets"](#) on page 35 for more information about how to use PAR FIL.

PAR GRIS

Displays parametric grid lines.

Syntax



Option	Description
OLD	Draws the old grid (lines of type STK), that is, the grid corresponding to the original drawing. This is the default option for the PAR GRIS command.
NEW	Draws the new grid (lines of type STK), that is, the grid corresponding to the parameterized drawing.
POT	Draws the potential grid (lines of type STK).
BAS	Draws baselines (lines of type LBL) corresponding to those that are automatically inferred if the PAR BAS switch is ON.
COL	Draws lines of type L3 over non-overlapping collinear straight line segments.
CIR	Draws complete circular lines of type L3 for all arcs.
TAN	Draws lines of type L3 tangentially to tangent point arcs.
INF	Draws infinite lines of type L3 along each line segment in the viewbox, and complete circles for each arc of a circle.
FIL	Draws grid lines (line type STK) along only the fillets that are affected by the current PAR FIL setting.

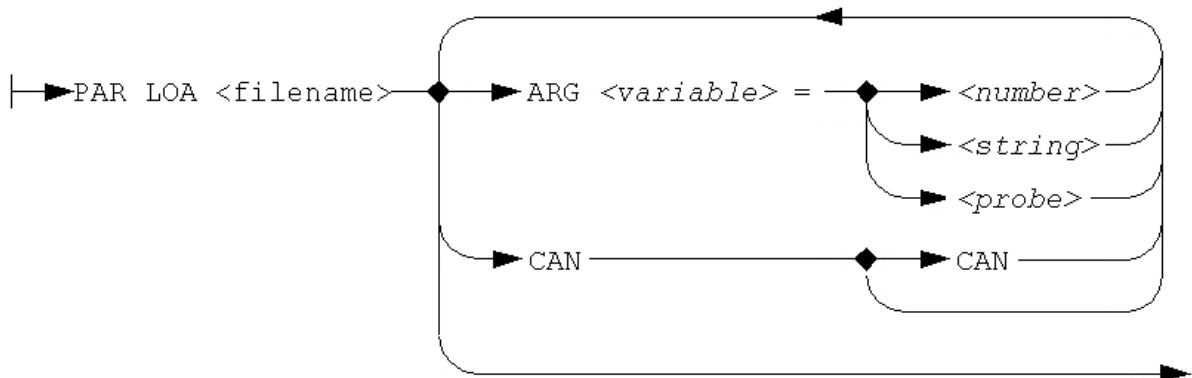
Further Information

See [“The Parametric Grid” on page 59](#) for more information about grids and the PAR GRIS command.

PAR LOA

Loads and parameterizes a parametric symbol. *filename* is the name of the symbol file you want to load.

Syntax



Option	Description
ARG	<p>Assigns values to variables in the parametric symbol during parameterization. You can use ARG interactively or as an in-sheet command of text type SCO in an instance clump.</p> <p><i>variable</i> The name of the required variable.</p> <p><i>number</i> A number assigned to the variable <i>variable</i>.</p> <p><i>string</i> Any text string enclosed in single quotes.</p> <p><i>probe</i> A positional probe. This assigns sheet coordinate values to attachment point variables in the symbol definition.</p>
CAL	Initiates the symbol loading and scaling (if any) after you have specified all the arguments. You can load several copies of the symbol by respecifying the arguments before typing CAL again.
CAN	Cancels the last copy of the symbol loaded.

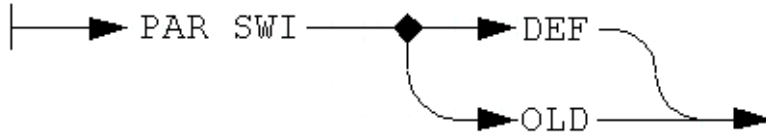
Further Information

See “[Parametric Symbols](#)” on page 111 for more information about using PAR LOA and ARG.

PAR SWI

Resets parametric switches to their default settings.

Syntax



Option	Explanation
DEF	Resets switches to the following default settings: ON LIM, BAS, TAN, MOV, TEX, SUP and UND OFF COL and CIR
OLD	Resets switches to default settings compatible with pre-6.0 CIS MEDUSA Parametrics software. In particular, it sets the LIM, BAS, SUP and TAN switches to OFF.

PAR TOL

Sets the parametric grid tolerance.

Syntax

| → PAR TOL *<tolerance>* →

Variable	Description
<i>tolerance</i>	The value of the tolerance required. The value specified must be a signed real value expressed as a decimal, a fraction, or an expression.

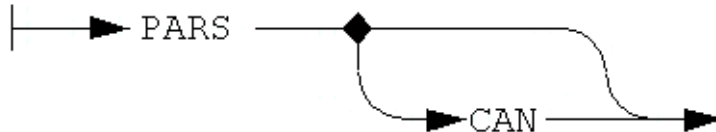
Further Information

Refer to ["The Parametric Grid"](#), ["Grid Tolerance"](#) on page 75 for more information on using PAR TOL.

PARS

Scales an object according to the values in dimensions clumps. If you do not want parameterization to be permanent, cancel PARS with CAN. When you use this command you can type either PARS CAN or PARSCAN.

Syntax



Option	Description
CAN	Cancels the last parameterization command. Always cancel the PARS command with CAN unless you are sure you will not need to change the definition drawing again.

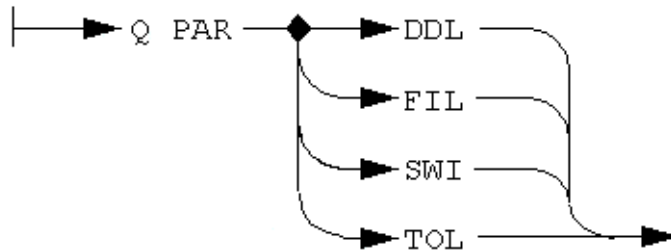
Further Information

See [“Parameterizing Geometry” on page 89](#), for more information about using PARS.

Q PAR

Displays the current grid tolerance, fillet, switch, and element default settings.

Syntax



Option	Description
DDL	Gives information about the current default types for the elements the Parametric Design system is using.
FIL	Gives information about the current fillet settings.
SWI	Gives information about the current parametric switch settings.
TOL	Gives information about the current parametric grid tolerance.

Further Information

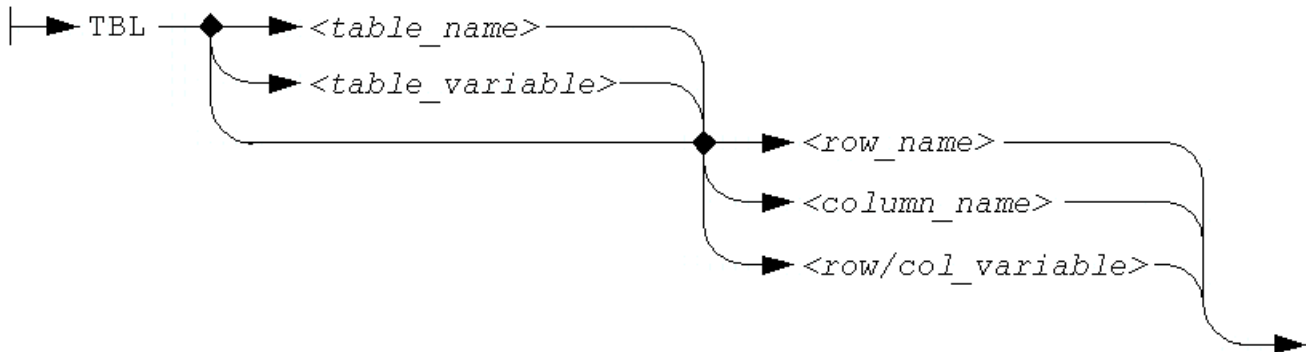
Details of grid tolerance, fillet, switch, and element defaults are given on the following pages:

- Default element types under ["Changing Element Types"](#), ["Parametric Design Element Defaults"](#) on page 152
- Fillets under ["Dimensioning"](#), ["Displaying the Current PAR FIL Setting"](#) on page 36
- Switches under ["Switches and Layers"](#), ["Setting Parametric Switches"](#) on page 98
- Grid tolerance under ["The Parametric Grid"](#), ["Grid Tolerance"](#) on page 75

TBL

References the values in a table. The table can create and overwrite variables for use in parameterizing objects. You can use the TBL command both interactively or as an in-sheet command.

Syntax



Option	Explanation
<i>table_name</i>	Specifies the table name. This is only required if there is more than one table in the relevant sheet or symbol.
<i>table_variable</i>	A variable name that evaluates to a valid table name. This must be a text string enclosed in angled brackets (< ... >).
<i>row_name</i>	Specifies a table row name. This can be any text string enclosed in single quotes ('...').
<i>column_name</i>	Specifies a column name. This can be any text string enclosed in single quotes ('...').
<i>row/col_variable</i>	A variable which evaluates to a valid table row or column name. This must be a text string enclosed in angled brackets (< ... >).

Further Information

Refer to [“Tables” on page 129](#) for more information about tables and the TBL command.

TBL DDL

Redefines the element types used in tables. The element type specified must be a CAN-code of up to 3 characters: the first character must be a letter, the others can be letters, numbers, \$, %, or #.

Syntax

| → TBL DDL → <line_type> <name_type> <row/column> →

Option	Default	Explanation
<i>line_type</i>	LTB	Specifies the line type to be used for the table frame line.
<i>name_type</i>	TTB	Specifies the text type to be used for the table title.
<i>row/column</i>	TRC	Specifies the text type to be used for the row and column headings.

Further Information

See “[Changing Element Types](#)” on page 151, for examples and for more information about using this command.

VER FULL

Displays detailed error information if an error occurs during symbol loading.

Syntax

| → VER FUL →

Further Information

Refer to the section "[Parametric Symbols](#)", "[Commands in Instance Clumps](#)" on page 122 for more information about using this command.

APPENDIX B ERROR AND WARNING MESSAGES

This appendix lists alphabetically the error and warning messages that can occur when you use MEDUSA Parametric Design.

Any relevant Drafting System error message can also appear either in the sheet or on the screen. For information on Drafting System error messages, refer to the *MEDUSA Basis1 Design Commands Guide*.

- [Error Messages..... 202](#)
- [Warning Messages 208](#)

Error Messages

Ambiguous Construction

The new grid on which the construction is based does not result in a unique set of grid lines. When a construction has unnecessary supporting grid lines, Parametric Design uses only the minimum necessary to support the construction. The remaining grid lines are then checked for consistency. This error is normally caused either by over-dimensioning, or by using a tolerance that is too large. Use the PAR GRIS command to identify the grid lines that are used to support the construction.

Ambiguous Point

The point indicated is at a point on the old grid that does not transform to a unique point on the new grid. This is because the grid lines which intersect at the point in the old grid do not all intersect at the same point in the new grid. This error is normally caused by over-dimensioning or by using a tolerance that is too large. Use the PAR GRIS command to identify the grid lines that intersect at the point.

Cannot resolve table

The table cannot be separated into rows and columns with the same number of entries in each row and in each column. Check the types, positions and number of texts in the table.

Element types insufficiently specified

This error occurs if one of the construction elements used by the Parametric Design system does not have a valid type specified for it. This is most likely to be a problem in the DDL, as the Parametric Design system has a complete set of defaults.

Fatal error during dimensioning

This error is produced when the `PARS` command is executed and an error is found in your dimensioning, for example a corrupt dimension clump.

Group datums define impossible transformation

This error is normally caused by coincident or colinear datum points in a parametric group.

Illegal context for element

This error is produced when you execute the `PAR LOA` command and an element is found in the symbol definition which is not valid as input to Parametric Design.

Illegal dimension type

Axonometric dimensioning and tolerance limit dimensioning are not supported by Parametric Design, that is, the DIM command options AXO and LIM. See [“Dimensioning” on page 27](#), to see which dimensioning types you can use with Parametric Design.

Illegal expression

An expression has been used to define a variable which includes an invalid operator or function. See [“Variables and Expressions” on page 77](#), for a list of valid operators and functions.

Illegal point/line function

The point function used on the baseline indicated is not recognized by the system. See [“The Parametric Grid” on page 59](#), for more information on point functions.

Illegal variable name

The text indicated should be a valid variable name. This message may occur with symbol attachment points and table row and column names.

Illegal viewbox definition

The viewbox indicated interferes with other viewboxes in the sheet. Viewboxes may not overlap or be nested. The X and Y limits of each viewbox line are used when testing for overlap. The other viewboxes are processed as normal.

No constructions in viewbox

No grid was generated because there was no valid construction in the viewbox. This is either because there are no dimensions or other constructions in the viewbox, or the constructions are on unhittable layers.

No current sheet

There must be a current sheet before you can execute any commands

No solution possible

The point or construction has no solution in the new grid. For example, a point may lie at the intersection of two circular grid lines in the old grid, but in the new grid the circles are too far apart to intersect.

No supporting grid(s)

The construction indicated does not have sufficient grid lines to support it. Use the PAR GRIS command to examine the existing grid. This should reveal the missing link that is required to build up the grid. If the PAR BAS switch is OFF, a dynamic baseline may be needed.

No table found

No table could be found with the name you specified in the TBL command. If the TBL command is given as an in-sheet command in a symbol, the search is restricted to the symbol containing the command.

No viewboxes in sheet

The Parametric Design system only scales objects which are inside a parametric viewbox.

Obsolete command GLMT, use PAR LIM

Obsolete command GLME, use PAR LIM

Obsolete command GLRAD, use PAR LIM

Obsolete command QGLRAD, use PAR LIM

These error messages relate to obsolete Prime Variational Geometry commands for displaying parts of the grid. Use the PAR LIM switch instead. When the PAR LIM switch is ON, the grid lines are limited by the line segments in the viewbox.

Point not dimensioned

The point indicated does not lie on a grid intersection in the old grid. All line points must lie on grid intersections. This message is the result of under-dimensioning, but can also be caused by inaccurate drafting, where points that should be coincident are not quite coincident. The error can also occur if the end points of dimensions are not at grid intersections. Be careful to probe the correct points when you are dimensioning objects.

Problem with construction

Some unidentified problem has occurred with a construction. This is often due to zero length baselines or expressions that evaluate to the wrong data type.

Problem with parametric group datums

An internal problem has occurred with the parametric group datum points.

Problem with parametric group line

A problem has occurred in creating a test polygon from the parametric group line.

Problem with VAR to LIM conversion

This message is displayed following the unsuccessful completion of a PAR GRIS or PARS CAN. A problem has occurred in attempting to convert a VAR dimension to a LIM dimension.

Row/col not found

The specified text string cannot be found as a row or column name in the relevant table.

Row/col text not outside body texts

The row and column texts must be outside the area of the table containing all the entries.

Specified type is not a line**Specified type is not a prim****Specified type is not a text**

These errors are due either to a problem in the DDL or an invalid type specified in the PAR DDL command.

System error - automatic baseline buffer overflow

The buffer used for storing grid lines that cross the potential grid line indicated has overflowed. This means that no baseline will be automatically generated. This buffer is also used with output radial dimensions. This problem can be avoided by adding an explicit baseline, if required.

System error - grid finding buffer overflow

The buffer used for finding grid lines that pass through a specified point has overflowed because too many grid lines pass through the point.

System error - element type storage is full

The maximum number of element types you can define with the PAR DDL command is 50. Sixteen are defined by default. See [“Changing Element Types” on page 151](#), for more information about the PAR DDL command.

System error - parametric workspace is full

This error is normally caused by too many dimensions in one viewbox. The workspace is used for storing information on the constructions and the grid lines.

System error - too many in-sheet commands

The maximum number of in-sheet commands in a viewbox is 200. Less may be allowed if large numbers of variables are used.

Too few rows/columns/entries

There must be at least one row, one column and one entry in a table.

Too many datums in parametric group

The maximum number of prims in a parametric group is three.

Too many entries in table

The maximum number of entries in a table is 2500.

Too many parametric groups

The maximum number of parametric groups in a viewbox is 100.

Too many rows and columns in table

The maximum total number of rows and columns in a table is 100.

Too many viewboxes

The maximum number of viewboxes in a sheet is 20.

Trying to move point outside max. area

When you parameterize an object, no points can be moved outside the maximum drawing area.

Typing mistake!

This error is the result of bad command syntax either in an in-sheet command or in a command that you have entered directly.

Unable to convert dimension

The PAR DIM command option VTL has failed. Either the VAR dimension being converted or the resulting LIM dimension is invalid.

Unable to load sheet file

This error occurs when you try to load a parametric symbol file which does not exist or is not readable. Make sure you have given the correct pathname for the symbol file.

Unknown element type

This error is generated when you try to specify an illegal element type in the TBL DDL command. See [“Changing Element Types” on page 151](#), for information about the element types you can specify using TBL DDL.

Unset variable

The construction indicated includes a variable which has not been assigned a value. The command QVAR may be used to query the current value of a variable.

Warning Messages

ambiguous dimension texts interpreted as values

One or more dimension texts may have been interpreted as simple values with a PRE text, but the texts could also have been interpreted as variable names. Alternatively, the dimension text may have been interpreted as both a value produced by automatic dimensioning and also as an expression. See [“Geometric Constraints” on page 47](#) for more information about how the Parametric Design system interprets dimension text.

messages written into sheet

This message is output to the screen whenever error messages are written into the sheet.

no grid lines drawn, as PAR LIM switch is OFF

Certain PAR GRIS options are only available with limited grid lines.

PARDIM command interrupted

PARGRIS command interrupted

PARLOA command interrupted

PARS command interrupted

These messages will appear when you interrupt any of these commands.

point(s) scaled outside viewbox

One or more points were moved outside the viewbox when the elements were moved.

protected elements not transformed or deleted

Elements on protected layers cannot be transformed or deleted by the Parametric Design system. However, they can be used during the creation of the grid. This warning occurs if there are elements on a hittable but protected layer that may also be transformed or deleted.

LIST OF FIGURES

Figure 1	Parametric Viewbox Containing Definition of a Rivet	10	Figure 36	After Parameterization	51
Figure 2	Rivet With Reference Point	11	Figure 37	Tangent Point, FUNV12	52
Figure 3	Rivet With Reference Point and Grid Lines	12	Figure 38	Baseline With Tangent Point	52
Figure 4	Rivet With Dimensioning and Grid	12	Figure 39	After Parameterization	52
Figure 5	Original Rivet Definition with New Parameters	13	Figure 40	Circular Base Line, FUNV26 and FUNV11	53
Figure 6	Rivet After Parameterization	14	Figure 41	Circular Baselines, FUNV26 and FUNV10	54
Figure 7	In-sheet Command Texts	17	Figure 42	Circular Baselines, FUNV26 and FUNV11	54
Figure 8	Positioning Reference Points	21	Figure 43	Old Grid Lines	62
Figure 9	Parametric Viewboxes	22	Figure 44	Parameterization Errors	63
Figure 10	A PVG Prim	23	Figure 45	Parameterization Errors	63
Figure 11	Orthogonal View Prims	23	Figure 46	Effect of PAR GRIS NEW Command	64
Figure 12	Static Baselines	24	Figure 47	Effect of PAR GRIS POT Command	65
Figure 13	Grid Lines Generated by Static Baselines	25	Figure 48	Limited Grid	66
Figure 14	Attachment Points	26	Figure 49	Unlimited Grid	67
Figure 15	Significant Points of a Linear Dimension Type 30	30	Figure 50	Default Grid Setting	71
Figure 16	Two Views of a Cylindrical Object	31	Figure 51	Infinite Lines Added With PAR GRIS INF	72
Figure 17	Chain Dimensioning With Center Support	32	Figure 52	Collinear Lines Added With PAR GRIS COL	73
Figure 18	Using The Reference Point to Create a DAT Dimension	32	Figure 53	Circles Added With PAR GRIS CIR	73
Figure 19	Creating an Angular Dimension	33	Figure 54	Tangential Lines Added With PAR GRIS TAN	74
Figure 20	Significant Points of Radial and Diameter Dimensions	34	Figure 55	Variables in Dimensions	80
Figure 21	Effect of the PAR GRIS FIL Command	37	Figure 56	Using Variables and Expressions	84
Figure 22	In Sheet PAR FIL KEEP Command	38	Figure 57	Drawing Using Logical Operators	86
Figure 23	After Parameterization	38	Figure 58	Output Dimension	87
Figure 24	In-sheet PAR FIL 5 Command	39	Figure 59	Output Dimension With Variable	87
Figure 25	After Parameterization	39	Figure 60	Effect of PAR COL OFF	101
Figure 26	In-sheet PAR FIL 5 MAX 10 Command	40	Figure 61	Effect of PAR COL ON	101
Figure 27	Unsupported Fillet Revealed by PAR GRIS Command	40	Figure 62	Effect of PAR CIR OFF	102
Figure 28	Effects of the PAR VAR Switch	44	Figure 63	Effect of PAR CIR ON	102
Figure 29	Tolerance Dimensions	45	Figure 64	Effect of PAR TAN OFF	103
Figure 30	Inserting --MAX Text	46	Figure 65	Effect of PAR TAN ON	103
Figure 31	Result of Parameterization	46	Figure 66	Symbol Definition With Attachment Points	112
Figure 32	Perpendicular Point, FUNV10	49	Figure 67	Symbol Loaded at Different Orientations	116
Figure 33	Dynamic Baseline With Perpendicular Point	49	Figure 68	Original Symbol	118
Figure 34	Intersection Point, FUNV11	50	Figure 69	Symbol Rotated by 30 Degrees	118
Figure 35	Baseline With Intersection Point	50	Figure 70	Symbol Mirrored with MIRVFN Command	119
			Figure 71	Structure of an Instance Clump	120
			Figure 72	Instance Clump Elements	121
			Figure 73	Gearbox Cover	123
			Figure 74	Four Tabs	123
			Figure 75	Gearbox Cover Definition	124

Figure 76	Instance Clumps	125	Figure 98	Parametric Group with Two Prims.	147
Figure 77	Detail of Instance Clump	125	Figure 99	Parametric Group with Three Prims	148
Figure 78	Definition for tab1.sym	126	Figure 100	Using Additional Lines.	164
Figure 79	Definition for tab3.sym	126	Figure 101	Magnification in One Direction	166
Figure 80	Definition for tab4.sym	126	Figure 102	Linear Motion.	167
Figure 81	Definition for tab5.sym	127	Figure 103	Varying Two Parameters.	167
Figure 82	Gearbox Cover - Final Drawing	127	Figure 104	Varying a Single Parameter	168
Figure 83	The Parts of a Table	130	Figure 105	Rotation About a Fixed Center	169
Figure 84	The Structure of Table Elements.	131	Figure 106	Cam Definition Drawing.	171
Figure 85	Table of Dimensions For Joist Section	133	Figure 107	Result of Repeated Parameterization	172
Figure 86	Table with Variables.	133	Figure 108	Basic Mechanism for a Piston.	175
Figure 87	In-sheet TBL Command.	136	Figure 109	Detailed Mechanism Definition	176
Figure 88	Structure of a Parametric Group	141	Figure 110	Repeated Parameterization of Piston Mechanism.	176
Figure 89	Bolt Definition With Parametric Group	142	Figure 111	Simulation Showing Lift Arm Assembly Motion 177	
Figure 90	Results of Parameterization.	142	Figure 112	Loader Definition	178
Figure 91	Static Parametric Group.	143	Figure 113	Tables LO, MID, and HIGH.	179
Figure 92	After Parameterization	143	Figure 114	Bucket Definition	180
Figure 93	Parametric Group Definition.	145	Figure 115	Lever Arm Definition	180
Figure 94	Group with One Prim	145	Figure 116	Lift Ram Definition	181
Figure 95	Group with Two Prims	146	Figure 117	Tipping the Bucket.	182
Figure 96	Group with Three Prims	146			
Figure 97	Parametric Group with One Prim.	147			

INDEX

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Numeric

2D Design Errors 94

A

ABS Tolerance 41
Accessing Table Values 130
adding grid lines 71
advanced features 15
Angular dimensions 33
ARC 29
arithmetical operators 85
attachment point 26
Avoiding Ambiguity in Dimension Texts 78
AXO 29

B

Baseline Inference 56
Baseline Point Functions 48
Baselines 24
brackets in expressions 83

C

CAN 196
Center point 35
Center Support 31
Chain Dimensions 31
Changing
 Element Types 154
 Table Element Types 158
Checking the Effect of PAR FIL 37
circular dynamic baseline 53
Circular Static Baselines 25
Commands Between PAR GRIS and CAN 69
Commands Between PARS and CAN 93
Constraints
 how they are Inferred 55
Constructing a Table 132
Creating
 a Parametric Group 141
 a Parametric Viewbox 21
 a Reference Point 20
 Dimension Clumps 28, 30
 Parametric Symbols 112

Current PAR FIL Setting 36

D

DAT type dimensions 32
datum prim 23
DEF Command 83
Defaults
 layer 104
Defining your Own Parameters 13
Deletable 105
deleting grid lines 61
Diameter Dimensions 34
Dimension Clumps 106
Dimension Texts
 avoiding ambiguity 78
 Format After Parameterization 79
 replacing 80
Dimension Tolerances 29
Dimension Types 28
 linear 30
Dimensioning
 Filletts 35
 illegal 29
 LIM 41
 the Object Points 12
 Tolerance - 41
dimensions
 output 87
 values 78
displaying
 grid lines 68
 Inferred Baselines 56
 Parametric Grid Lines 60
Dual Dimensions 29
Dynamic Baselines 48
dynamic groups 140
 how they work 144

E

Effects of the PAR VAR Switch 44
Element Defaults 152
 query 153

element types
 changing 154
 of grid lines 70
Entering Values in a Table 132
error messages 202
Example
 circular baselines 54
 drawing using operators 86
 dynamic groups 145
 in-sheet TBL command 136
 instance clumps 123
 intersection point 50
 limited grid lines 66
 Loader Mechanism 176
 magnification 166
 Old Grid Lines 62
 output dimensions 87
 PAR CIR 102
 PAR COL 101
 PAR DDL 157
 PAR FIL filrad 39
 PAR FIL filrad MAX maxrad 40
 PAR FIL KEEP 38
 PAR TAN 103
 parametric group 141
 parametric symbol 112
 perpendicular point 49
 Piston mechanism 175
 program to control mechanisms 171
 simulating a rotary motion 169
 static groups 143
 table 132
 TBL DDL 158
 tolerance dimensioning 45
 variables and expressions 84
expressions 83
 brackets 83
 in Dimension Clumps 80

F

Fillet Type 35
FUNV0 48

G

geometric constraints 48
graphics
 undraw 100
grid 60
 adding lines 71
 new 63
 investigating errors 63
 old 62
 potential 65
 Tolerance 75
grid lines
 deleting 61
 displaying 68
 element types 70
 limited 66
 making untransformable 70
 unlimited 67

H

Hittable 105
How Constraints are Inferred 55
How Dynamic Groups Work 144

I

Illegal Dimensioning 29
Individual Tolerances 44
In-sheet Commands
 maximum number 17
 to Set Switches 99
 why to use 17
Instance Clump 120
 commands in - 122
 Element Types 121
 Gearbox Cover Example 123
 Loading Errors 122
 variables 121
intersection point 50
Investigating Errors in new grid 63
Invisible 105

L

LAY Command 107
Layers 104
 change Properties 107
 defaults 104
 properties 105
 query properties 104
LET Command 81
LIM 29
LIM Dimensioning 41
Limited Grid Lines 66
Limited grid lines (definition) 60
Linear Dimension Types 30
Linear Motion 166
Loader mechanism 174
loading symbols 115
logical operators 85

M

Magnification 166
Maximum Number of In-sheet Commands 17
Mechanism
 running 162
Mirroring Symbols 119
Movement in One Direction 166
Movement in Two Directions 167
Moving Dimensioned Points 90

N

NEA probes 30
New Grid 63
NEW grid (definition) 60

O

Old Grid 62
OLD grid (definition) 60

operators 85
Orthogonal View Prims 23
Other Dimension Text Formats 79
Output dimensions 87

P

PAR BAS 96
PAR CIR 96
 example 102
PAR COL 96
 example 101
PAR DDL 154, 187
PAR DDL RESET 189
PAR DIM 42, 190
PAR FIL 36, 191
PAR FIL filrad (example) 39
PAR FIL filrad MAX maxrad (example) 40
PAR FIL KEEP (example) 38
PAR GRIS 68, 192
PAR GRIS BAS 56
PAR GRIS CIR 73
PAR GRIS COL 72
 when to use 73
PAR GRIS INF 71
PAR GRIS NEW 64
PAR GRIS POT 65
PAR GRIS TAN 74
PAR GRIS to Check the Grid 62
PAR LIM 96
PAR LOA 115, 193
PAR MOV 96
PAR PRE 96, 149
PAR SUP 97
PAR SWI 194
PAR TAN 97
 example 103
PAR TEX 97
PAR TOL 75, 195
PAR UND 97
PAR VAR 97
 Command 42
 examples 44
 options 43, 186
Parameterization
 Errors 93
 overview 90
Parameterizing 14
Parametric Design
 advanced features 15
 overview 9
Parametric Grid Lines 11
Parametric Group
 creation 141
 Elements 140
 Example 141
 introduction 140
 rotating 147

parametric switches 96, 184
 in-sheet commands 99
 setting 98
parametric symbols
 creating 112
 loading - using instance clumps 120
 mirroring 119
 Reference Points 112
 rotate 118
Parametric Viewboxes 10
PARIVB 109
PARS 92, 196
 what happens when giving the - command 90
parts of a table 130
Perpendicular Point 49
Piston mechanism 174
place a reference point 20
Plotting Motion Simulations 173
Potential Grid 65
POTENTIAL grid (definition) 60
preference order of geometric constraints 55
Preparing a Drawing for Parameterization 10
prim 23
Printing documentation from Portable Document Format
 (PDF) files 8
Priority of Geometric Constraints 55
Programs to Control Mechanisms 171
Properties
 of layers 105
 change 107
Protected 105
protected variable 82
PVG Prim 23

Q

Q PAR 197
Q PAR TOL 76
query
 element defaults 153
 Layer Properties 104
 Switch Settings 100

R

Radial Dimensions 34
Reference Points 11
 of Parametric Symbols 112
 positioning 20
Resetting Switch Defaults 100
Rotary Motion 169
rotate parametric groups 147
Rotating Symbols 118
Running a Mechanism 162
Running the Simulation 161

S

Setting Parametric Switches 98
Significant Points 30, 33, 34

Simulating

- Different Types of Motion 160
 - examples 174
 - linear motion 166
 - magnification 166
 - overview 160
 - plot motion simulations 173
 - prepare the definition drawing 163
 - programs 171
 - rotary motion 169
 - running 161
- Static Baselines 24
- Static groups 140
- Straight Static Baselines 24
- Structure of Table Elements 131
- switches 96, 184
 - query its settings 100
 - reset to defaults 100
- symbols
 - load interactively 115
 - loading - Using Instance Clumps 120
 - mirroring 119
 - rotate 118
 - test definition 113

T

- table 130
 - change element types 131
 - create 132
 - enter values 132
 - s and Parametric Symbols 137
 - variables and expressions 133
 - variables to access rows 135
- tangent point 35, 52
- TBL 135, 198
- TBL DDL 158, 199

- Testing the Symbol Definition 113
- text variable command 170
- Tolerance Dimensioning 41
- Tolerance Variation Var Options 186
- Tolerances
 - individual 44
- Transformable 105

U

- Undeletable 105
- Undrawing Graphics 100
- Unhittable 105
- Unlimited grid lines (definition) 60
- Unlimited Grids 67
- Unprotected 105
- UNS Tolerance 41
- Untransformable 105
- Unwanted Inferences 57

V

- Values in Dimensions 78
- variables 81
 - in Dimension Clumps 80
 - protected 82
 - Scope Restrictions 82
 - to Access Rows 135
- Variables and Expressions in Tables 133
- VER FULL 200
- Visible 105

W

- warning messages 208
- Why Use In-sheet Commands 17