

Allegro[®] Constraint Manager User Guide

Product Version 16.3
December 2009

© 1991–2010 Cadence Design Systems, Inc. All rights reserved.

Portions © Apache Software Foundation, Sun Microsystems, Free Software Foundation, Inc., Regents of the University of California, Massachusetts Institute of Technology, University of Florida. Used by permission. Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Constraint Manager contains technology licensed from, and copyrighted by: Apache Software Foundation, 1901 Munsey Drive Forest Hill, MD 21050, USA © 2000-2005, Apache Software Foundation. Sun Microsystems, 4150 Network Circle, Santa Clara, CA 95054 USA © 1994-2007, Sun Microsystems, Inc. Free Software Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA © 1989, 1991, Free Software Foundation, Inc. Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, © 2001, Regents of the University of California. Daniel Stenberg, © 1996 - 2006, Daniel Stenberg. UMFPACK © 2005, Timothy A. Davis, University of Florida, (davis@cise.ulf.edu). Ken Martin, Will Schroeder, Bill Lorensen © 1993-2002, Ken Martin, Will Schroeder, Bill Lorensen. Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts, USA © 2003, the Board of Trustees of Massachusetts Institute of Technology. All rights reserved.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Patents: Constraint Manager described in this document, is protected by U.S. Patents 5,481,695; 5,510,998; 5,550,748; 5,590,049; 5,625,565; 5,715,408; 6,516,447; 6,594,799; 6,851,094; 7,017,137; 7,143,341; 7,168,041, 7,464,358; 7,536,665; 7,562,330; 7,574,686.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1

<u>Welcome to Constraint Manager</u>	11
<u>The Allegro® Constraint Manager Information Set</u>	12
<u>What is Allegro® Constraint Manager?</u>	13
<u>Accessing Constraint Manager</u>	17
<u>Constraint Manager launched from Allegro PCB Series L, Performance L, and OrCAD PCB Editor</u>	18
<u>Constraint Manager launched from Allegro® Physical Viewer</u>	20
<u>Domains, Workbooks, Worksheets, and Cells</u>	21
<u>The Worksheet Selector</u>	23
<u>Workbooks</u>	26
<u>Physical and Spacing Workbook Views</u>	28
<u>Same Net Spacing DRC Modes</u>	30
<u>Constraint Manager's User Interface Controls</u>	33
<u>Accelerator Keys</u>	35
<u>Enhancements Done in 16.3</u>	37
<u>Online Formula Support</u>	37
<u>Differential Pair Naming</u>	39
<u>Changes in the Edit Via List Dialog Box</u>	39
<u>Advanced Tabbed View</u>	42
<u>New Look Analysis Modes Dialog Box</u>	43
<u>Goto Constraint Set Command</u>	44
<u>Object Membership Count</u>	46
<u>Filter Functionality</u>	47
<u>Suffix Indicating Number of Members</u>	47
<u>Miscellaneous Changes</u>	48

2

<u>Working with Constraint Objects</u>	49
<u>About Constraint Object Hierarchy</u>	50

Allegro Constraint Manager User Guide

<u>Types</u>	51
<u>About Objects</u>	53
<u>Designs and Systems</u>	53
<u>Net Class</u>	54
<u>Net Class Rules</u>	54
<u>Net Class-Class</u>	55
<u>Net Class-Class Rules</u>	55
<u>Differential Pairs</u>	56
<u>Differential Pair Worksheets</u>	57
<u>Using the Differential Calculator</u>	61
<u>Differential Pairs by Constraint Region</u>	63
<u>Differential Pair Rules</u>	65
<u>Match Groups</u>	67
<u>Generating pin pairs for the Match Group members</u>	67
<u>Defining Match Group requirements</u>	68
<u>Examples</u>	70
<u>Scope</u>	72
<u>Match Group Rules</u>	82
<u>Multi-group Membership</u>	82
<u>Buses</u>	84
<u>Bus Rules</u>	85
<u>Nets and Xnets</u>	85
<u>Pin Pairs</u>	86
<u>Examples of pin pairs</u>	87
<u>Pin Pair Rules</u>	88
<u>Ratsnest Bundle</u>	89
<u>Region</u>	90
<u>Region Rules</u>	90
<u>Region Class</u>	91
<u>Region Class Rules</u>	91
<u>Region Class-Class</u>	91
<u>Region Class-Class Rules</u>	92

3

Working With

<u>Reusable Constraint Objects — CSets</u>	93
<u>Reusable Constraints</u>	94
<u>Constraining Objects</u>	94
<u>Constraint Sets (CSets)</u>	94
<u>Methods of Constraining Nets</u>	95

4

ECSets and Topology Templates

<u>What is a Topology Template?</u>	100
<u>Importing ECSets</u>	102
<u>Mapping Templates and ECSets to Net-related Objects</u>	102
<u>Audits</u>	106
<u>Constraint Audit</u>	106
<u>Obsolete Objects Audit</u>	107
<u>Electrical CSets Audit</u>	108
<u>Topology Templates Audit</u>	108
<u>Exporting ECSets</u>	111
<u>Migrating Legacy Electrical Rule Sets</u>	111

5

Constraint Analysis

<u>How Allegro® Constraint Manager Performs Analysis</u>	114
<u>Viewing Worksheet Cells and Objects</u>	115
<u>Analyzing for DRC-based Constraints</u>	119
<u>DRC Constraint Modes</u>	120
<u>Analyzing for Simulation-based Constraints</u>	121
<u>Simulation-based Custom Stimulus</u>	122
<u>The Analysis Process</u>	123
<u>Analysis Results</u>	128
<u>Generated DRC Output</u>	129
<u>Waveforms</u>	129

Allegro Constraint Manager User Guide

<u>Reports</u>	129
<u>Worksheet cells</u>	129
<u>Interpreting Analysis Results Returned to a Worksheet</u>	130
<u>Constraints Across the System</u>	133

6

Using Constraint Manager with Other Tools Across the Allegro Platform

135

<u>Phases in the Design Flow</u>	136
<u>Design Exploration Phase (with SigXplorer)</u>	137
<u>Pin Scheduling</u>	138
<u>Design Capture Phase</u>	139
<u>Design Capture Phase (with System Connectivity Manager)</u>	139
<u>Front to Back Constraint Flow</u>	140
<u>Back to Front Constraint Flow</u>	142
<u>Design Floorplanning and Implementation Phases</u>	144
<u>Using SigXplorer in the capture, floorplanning and implementation phases</u>	146

7

Constraint Manager Your Way

149

<u>Customizing the User Interface</u>	150
<u>Customizing Visibility</u>	150
<u>Customizing Keyboard Shortcuts</u>	151
<u>Customizing Toolbars</u>	151
<u>Customizing Worksheets</u>	151
<u>Customize Mode</u>	152
<u>User-defined Properties</u>	153
<u>Customizing Simulations</u>	154
<u>Working with Custom Constraints</u>	155
<u>Managing Custom Measurements</u>	156
<u>Analyzing with Custom Measurements and Custom Stimulus</u>	158
<u>Customizing Design Rule Checks</u>	159
<u>Formulas</u>	159
<u>Inserting a formula in a cell</u>	160

Allegro Constraint Manager User Guide

<u>The Single-line Editor</u>	160
<u>Cell selection and operands</u>	161
<u>Calculating a Formula</u>	161
<u>Pre-defined Predicates</u>	163
<u>Programmatic Formulas</u>	164
<u>Support for Online Formula</u>	166
<u>Testing and Debugging Formulas</u>	167
<u>User-defined Constraints</u>	168
<u>User-defined Predicates and Measurements</u>	170
<u>User-defined Predicates</u>	170
<u>Defining a predicate</u>	171
<u>Predicate Parameters</u>	173
<u>Examples of predicate parameters</u>	174
<u>Testing and Debugging User-defined Predicates</u>	175
<u>User-defined Actuals</u>	178
<u>User-defined Measurements</u>	179
<u>Testing and Debugging User-defined Measurements</u>	182

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Topics in this chapter include

- [The Allegro® Constraint Manager Information Set](#) on page 12
- [What is Allegro® Constraint Manager?](#) on page 13
- [Accessing Constraint Manager](#) on page 17
- [Domains, Workbooks, Worksheets, and Cells](#) on page 21
- [Constraint Manager's User Interface Controls](#) on page 33
- [Enhancements Done in 16.3](#)

The Allegro[®] Constraint Manager Information Set

The Allegro[®] Constraint Manager information set consists of online books accessible from *Cadence Help* in both HTML and PDF formats. All documentation is accessible from Constraint Manager's help menu.

Refer to . . .

*Allegro[®]
Constraint Manager User Guide*
(this book)

for this level of information

This book is for users who want to know how to use Constraint Manager in the design flow.

This book complements the information in the *Allegro[®] Constraint Manager Reference*.

*Allegro[®]
Constraint Manager Reference*

This book contains descriptions and procedures for all commands, organized by menu-sequence. Information about worksheet cells is also included.

If you click help in a dialog box or if you highlight a menu command and press F1, the appropriate command description from this book appears.

This book complements the information in the *Allegro[®] Constraint Manager User Guide*.

*Allegro[®] Platform
Constraints Reference*

This book contains information describing the constraints architecture, and it includes reference information for each constraint.

*Allegro[®] Platform
System Connectivity Manager
User Guide*

This book contains information describing the hierarchical, lower-level constraints used in the Constraint Manager-enabled, high-speed flow.

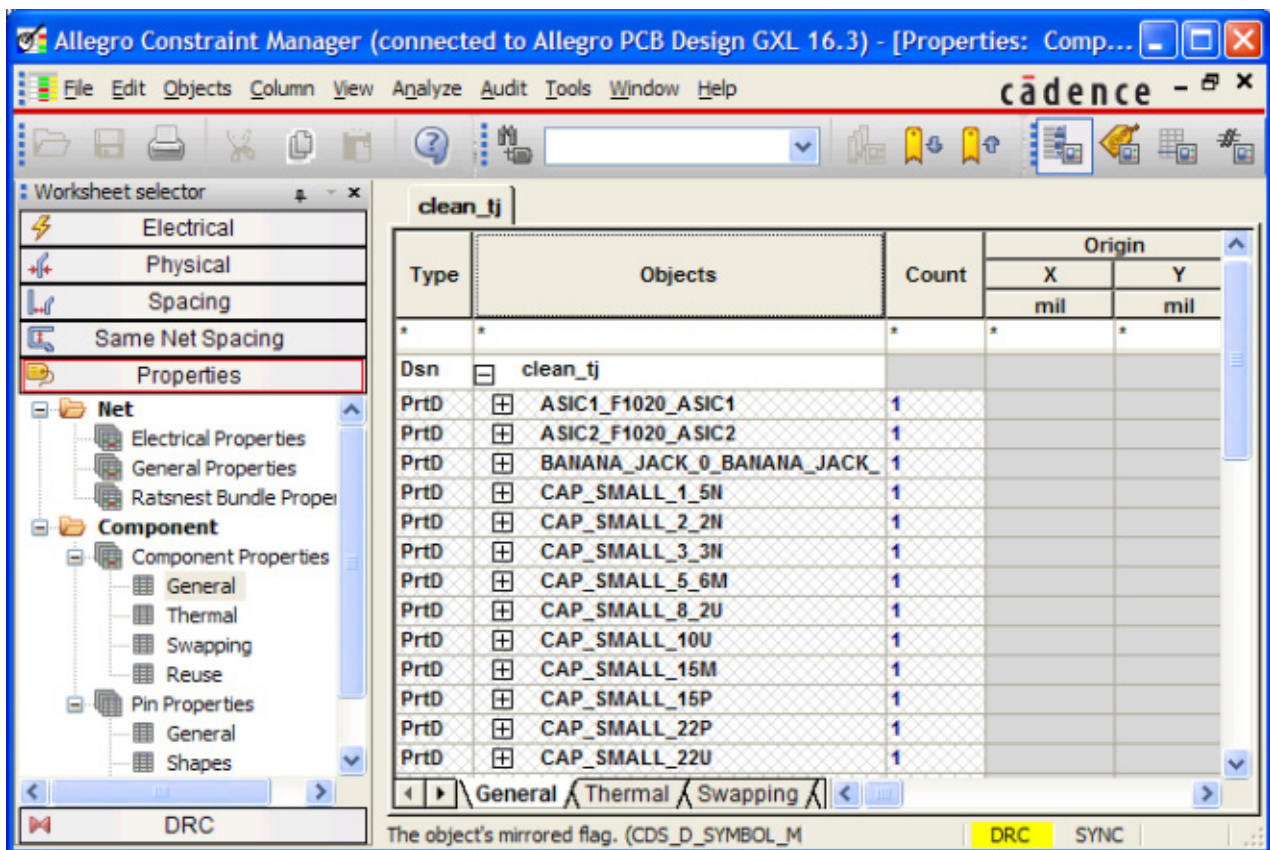
What is Allegro® Constraint Manager?

Allegro® Constraint Manager is a cross-platform, workbook- and worksheet-based application used to manage constraints across all tools in the Cadence PCB and IC Package design flow.

Constraint Manager lets you define, view, and validate constraints at each step in the design flow, from design capture (in Allegro® Design Entry HDL or System Connectivity Manager) to floorplanning (in Allegro® PCB SI L, XL, and GXL to design realization (in Allegro® PCB L, XL, GXL, and OrCAD PCB Editor). You can also use Constraint Manager with SigXplorer to explore circuit topologies and derive electrical constraint sets which can include custom constraints, custom measurements, and custom stimulus.

Note: Figure 1-2 depicts Constraint Manager worksheets as launched from PCB Editor, OrCAD PCB Editor, or APD. The worksheet hierarchy is different for Constraint Manager when launched in exploration mode, from Allegro Design Entry HDL, or from System Connectivity Manager.

Figure 1-1 The Constraint Manager User Interface



Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Constraint Manager uses familiar user interface controls. See [Table 1-2](#) on page 33 for more information.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

In Constraint Manager, you work with objects and constraint sets, which capture your design requirements.

Constraint Manager organizes constraints and Constraint Sets into the *Physical*, *Spacing*, *Same Net Spacing*, and *Electrical* domains. You then assign the appropriate constraint set to objects in your design, changing references (or re-defining the currently assigned constraint set) as your design requirements change. A constraint set can be referenced by any number of objects in your design.

For more information on design objects and the object hierarchy, see [Chapter 2, “Working with Constraint Objects.”](#)

For more information on how to define constraint sets and how to assign them to objects in your design, see [Chapter 3, “Working With Reusable Constraint Objects — CSets.”](#)

Constraint Manager affords you the following features and benefits:

Table 1-1 Constraint Manager Features

Feature	Benefit
Object Grouping	You can organize objects into easily-managed units, such as a <i>Class</i> , <i>Bus</i> , <i>Differential Pair</i> , or <i>Match Group</i> to make it easier to apply constraints to member objects.
Conceptual Definition	You can define constraints in a Constraint Set and later apply those constraints to net-related objects.
Redefinable Constraints	Rather than changing individual net-related constraints one-by-one, you can redefine a constraint set and all objects that reference that constraint set get updated all at once.
Cross-Probing	<ul style="list-style-type: none">■ You can run Constraint Manager with companion tools such as Allegro Design Entry HDL, Allegro SI, or Allegro Package Design and select a net in Constraint Manager and see the associated object update dynamically in the schematic, floorplanner, or layout, respectively. <p>Conversely, Constraint Manager updates its values when they are modified in a companion tool.</p> <ul style="list-style-type: none">■ When you cross probe cells that contain constraint violations, the respective DRC marker (bowtie) becomes highlighted in the design entry- or PCB-Editor, or in APD.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Feature	Benefit
Topology Exploration	You can access SigXplorer from Constraint Manager to schedule pins and derive generic or net-specific constraints, which may include custom constraints, custom measurements, and custom stimulus. The resulting topology template data can be imported into Constraint Manager as an Electrical CSet.
Design Reuse	You can group constraints that satisfy a specific design requirement into an constraint set, which can be referenced within the active design or exported for reuse in a subsequent design.
Cloning Constraints	In addition to importing constraint sets or creating them from scratch, you can copy it, modify its parameters, and save it under a new name.
Analysis	Constraint Manager performs design rule checks, and simulations as necessary, to analyze the design. Analysis results are communicated by DRC markers, results populated in worksheet cells, simulation waveforms, and reports. Analysis results (actuals) can be compared to defined constraints to derive margins.
System-level Constraints	Constraint Manager can capture board-to-board interconnect constraints.
Persistent Storage	Constraint Manager maintains constraint information in either the board or the schematic database.
Net, Component, and Pin Properties	The <i>Properties</i> workbooks let you add and edit certain properties for nets, components, or pins.
Customizable User Interface	You can create a custom workbook and worksheets that suit your work habits.

Accessing Constraint Manager

You access Constraint Manager in Exploration mode through the Windows *Start* menu or by entering `consmgr` in a Unix or Linux shell.

Constraint Manager can also be invoked from a host tool as follows:

From this tool	Choose this menu command
Allegro PCB Editor, Allegro Package Design, or Allegro SI	<i>Setup – Constraints – Electrical Physical Spacing Constraint Manager</i>
Allegro Design Entry HDL	<i>Tools – Constraints – Edit</i>
System Connectivity Manager	<i>Design – Edit Constraints</i>

You can also click the Constraint Manager icon in the host tool's toolbar.

Constraint Manager maintains constraint information in the board database when used with Allegro PCB or SI, in the package database when used with Allegro Package Design, or in the schematic database when used with Allegro Design Entry HDL.

Important

The appearance of the Worksheet Selector, worksheets, and commands differ depending on whether Constraint Manager is launched in Exploration mode, invoked from a front-end application, or a backend-application. For example, *By Layer* view of *Physical* and *Spacing* cells is not available in Constraint Manager, when launched from OrCAD PCB Editor or Allegro PCB Editor, Performance L option.

The name of the tool from which you launch Constraint Manager appears in the banner atop the Constraint Manager user interface. For example:

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Constraint Manager (Connected to Allegro Design Entry HDL)

See [Chapter 6, “Using Constraint Manager with Other Tools Across the Allegro Platform”](#) for using Constraint Manager with other Cadence tools.

Constraint Manager launched from Allegro PCB Series L, Performance L, and OrCAD PCB Editor

This manual covers all functionality available in Constraint Manager when invoked from Allegro[®] PCB Series GXL and XL. When invoked from an Allegro[®] PCB Series L, Performance L, or OrCAD PCB Editor, Constraint Manager launches with these limitations:

- Scripting Scripting is limited to the commands and constraints that are supported in Allegro[®] PCB Series L or OrCAD PCB Editor.
- Match Groups You can define Match Groups only in net-level worksheets; you cannot define match groups at the Constraint Set-level.
- Pin Pairs You can define pin pairs only in net-level worksheets; you cannot define pin pairs at the at the Constraint Set-level
- Signal Integrity Signal integrity analysis is not supported. The *Signal Integrity* workbook has been removed.
- Timing Timing analysis is not supported. The *Timing* workbook has been removed.
- Custom measurements and custom stimulus Custom measurements and custom stimulus are not supported. The *Custom Measurements* tab (*Analyze – Analysis Modes*) has been removed; only the *DRC Modes* tab remains. The *Custom Measurements* workbook is not visible.
- Crosstalk DRC Crosstalk analysis is not supported. The `max_xtalk` and `max_peak_xtalk` design rule checks have been removed from the *Analysis Modes* dialog box.
- Topology Templates Topology import and export are not supported. As such, the *Tools* menu has also been removed prohibiting access to topology exploration tools including SigXplorer and SigWave.
- Analysis Simulation-based analysis is not supported. Only design rule checks can be performed.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

- Xnet Creation
You create an Xnet (extended net) through a signal model. using a Series XL or GXL PCB Editor or high tier legacy PCB Editor.

- PCB Performance Series L
 - In the *Electrical* domain, custom measurements, pin delay, and Z-axis delay are not supported. You also cannot control same net crosstalk and parallelism checks.
 - *By Layer* view of *Physical* and *Spacing* cells is not supported.
 - Ratsnest Bundle worksheets are not supported.
 - Microvias are not supported.

- PCB Series L
 - In the *Electrical* domain, custom measurements, pin delay, and Z-axis delay are not supported. You also cannot control same net crosstalk and parallelism checks.
 - For the *Physical*- and *Spacing*-domains, regions, pin pairs, Xnets, differential pairs, buses, and by-layer worksheets are not supported.
 - Ratsnest Bundle worksheets are not supported.
 - Microvias are not supported.

- OrCAD PCB Editor
 - *By Layer* view of *Physical* and *Spacing* cells is not available.
 - Ratsnest Bundle worksheets are not supported.
 - Microvias are not supported.

When you select an object in Constraint Manager and right-click, a context pop-up menu appears. Keep in mind that this guide depicts all available options. If you launched Constraint Manager from Allegro® PCB Series L Editor (Performance), or OrCAD PCB Editor, some options will be removed or dimmed to inhibit functionality.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Constraint Manager launched from Allegro® Physical Viewer

PCB collaboration tools lack constraint management access, yet companies with co-design partners may require design constraint information as specified by contract or agreement. Use Constraint Manager in conjunction with Allegro® Physical Viewer as a back-end validation tool that lets design partners view electrical constraint information and analysis results and communicate it without requiring interpretation or conversion if an Allegro flow is used.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

When invoked from the Allegro® Physical Viewer *Setup* menu, read-only mode Constraint Manager launches with a limited functionality set. You can view the constraint information that a .brd file contains. All constraints appear in native delay values (for example, not a length only Performance mode). You cannot modify or export these constraints as certain menu functionality is disabled: right mouse buttons will not allow you to create, modify, or delete objects. *Print* and *View* menu options are available.

Read-only mode Constraint Manager includes all worksheets. However, SigWave or simulation actuals data are unavailable. *Actual* and *Margin* information is available for the constraints based on the design's current state.

Although Allegro® Physical Viewer does not let you change DRC modes, as they are inherited from the board, you can run DRC from Allegro to display *actual* data in Constraint Manager, which changes the database; then save it in Allegro® Physical Viewer.

Domains, Workbooks, Worksheets, and Cells

The Constraint Manager workspace (see Figure 1-2, and Figure 1-3 on page 23) contains the following components.

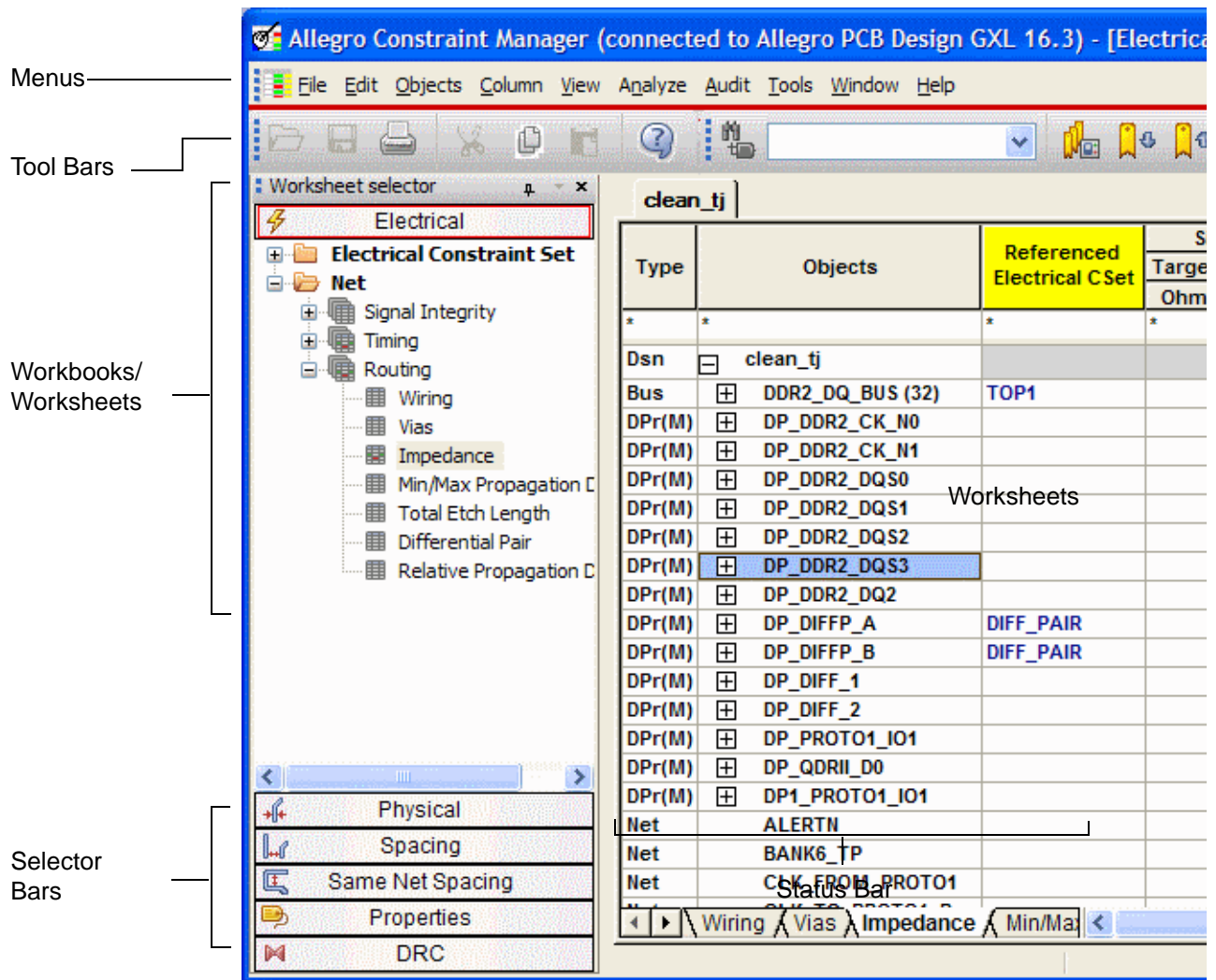
The:

- *Menus* for command access
- *Tool Bars* for quick command access
- *Selector Bar* for switching among domains and DRC and Properties Workbooks
- *Worksheet Selector* for selecting the appropriate worksheet
- *Type* column for identifying the type of object in the *Objects* column
- *Worksheets* for capturing, editing, and validating constraints
- *Status Bar* for feedback on object selection and constraint processing
- *DRC Status* indicator for checking the state of design rule checking

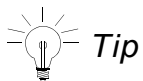
Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Figure 1-2 The Constraint Manager workspace



Note: When you select an object in Constraint Manager and right-click, you can also access commands from a context-sensitive, pop-up menu.



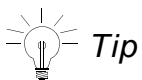
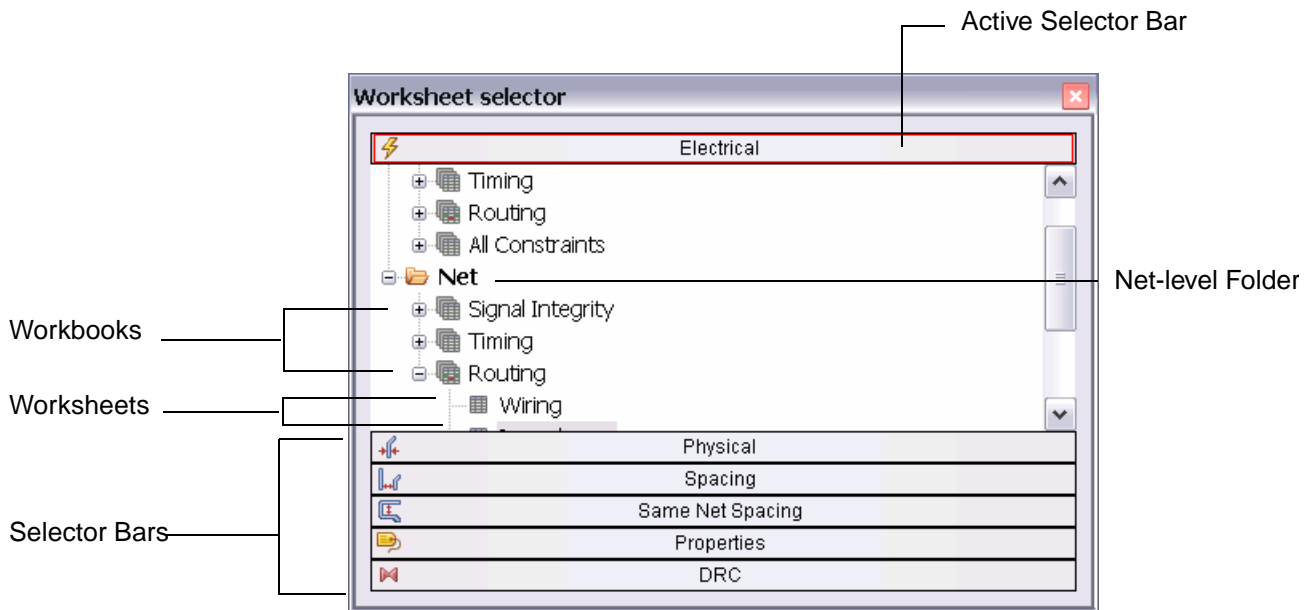
Tip

The *Status Bar* provides key information about cell contents, the state of objects, error conditions, and conditions and processes in your design. When in doubt, consult the status bar.

The Worksheet Selector

Use the *Worksheet Selector* to access the appropriate worksheet that you want to work in. Selector Bars let you access individual constraint worksheets, properties worksheets, and DRC worksheets, which you access by clicking on a *Selector Bar*. You can also undock and reposition the *Worksheet Selector*.

Figure 1-3 Worksheet Selector



Tip

Grab the border of the *Worksheet Selector* and reposition it to get a full view of workbook and worksheet selector nodes (as shown).

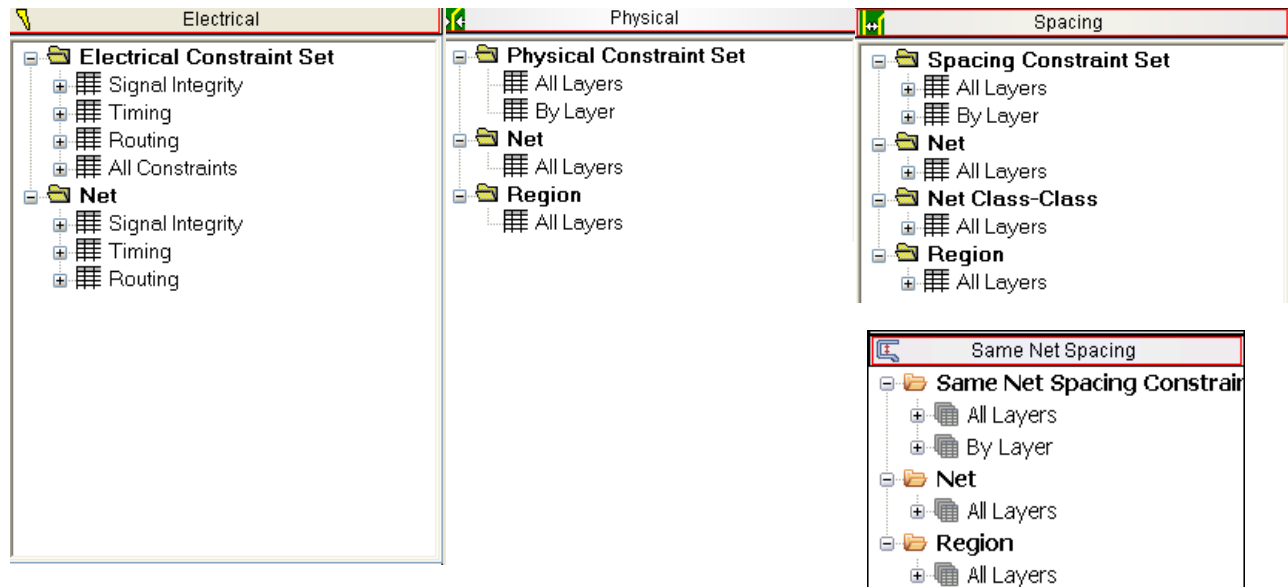
Domain Selector Bars

Constraint Manager organizes constraints, and constraint sets, by domain: *Electrical*, *Physical*, *Spacing*, and *Same Net Spacing*. You access each domain by clicking on the appropriate *Selector Bar*, which is located at the bottom of the *Worksheet Selector* (see Figure 1-3).

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Figure 1-4 Worksheet Hierarchy



- In the *Constraint Set Folders* for all domains, you define generic rules and you create generic object groupings. You can later assign these rules to the appropriate net-related objects in your design.
- In the *Net* folders for all domains, you can create net-specific object groupings, and you can define certain net properties. In the *Electrical* domain, you can also create a constraint set based on the characteristics of a net object.
- In the *Physical*, *Spacing* and *Same Net Spacing* constraint folders, worksheets based on layer, or by all layers, contain *Nets*, *Classes*, and *Regions*.

Important

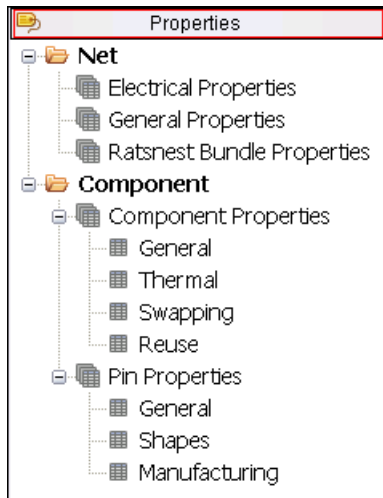
By Layer view of *Physical* and *Spacing* cells is not available in Constraint Manager, when launched from OrCAD PCB Editor or Allegro PCB Editor, Performance L option.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Properties Selector Bar

Use the *Properties* selector bar to manage net, component, and pin properties.



The *Net* folder provides you with a quick glance of electrical and general properties. Some cells in these worksheets cannot be edited.

The *Component* folder provides component coordinates, based on placement information, source data for third-party thermal analysis tools, and part definitions. Also included are electrical, thermal, and pin fabrication data. Some cells in these worksheets cannot be edited.

Important

System Connectivity Manager also provides component worksheets where you can define and edit these properties.

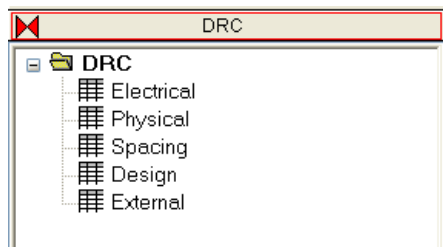
See the [Allegro Platform Properties Reference](#) for more information on component properties.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

DRC Selector Bar

Use the *DRC* selector bar to view and waive design rule violations on objects in PCB Editor or APD. See the Objects – Waive command in the *Constraint Manager Reference* for more information.



Workbooks

Once you expand a parent *Object Type* folder, workbooks organize objects by design discipline. For example, the *Electrical* domain contains the *Signal Integrity*, *Timing*, *Routing*, and *Custom Measurements* workbooks. Also in the *Electrical* domain, the *All Constraints* workbook in the *Electrical CSet* folder consolidates constraints from all worksheets to give you a global view. Subordinate to the *All Constraints* workbook is the *User Defined* folder, which contains constraints that you have defined in SigXplorer.

Note: The worksheet hierarchy is different if you launch Constraint Manager in exploration mode, or from Allegro® Design Entry HDL or Allegro® System Architect.

When you select a workbook, all worksheets that belong to that workbook appear in a shared worksheet window. You can use the *Worksheet Selector* to select a worksheet or you can select a worksheet by clicking on the appropriate tab in the shared workbook window. You may have to scroll horizontally to locate the desired worksheet tab.

Note: When you launch Constraint Manager from a physical layout editor, the cells that are in view are populated first. As you scroll other cells into view, the layout tool updates hidden cells as they become visible in Constraint Manager.

The Analyze – Analysis Modes command controls DRC checks. Also, refer to the *DRC State Bar*, located at the bottom of Constraint Manager, adjacent to the *Status Bar*, to

- learn the state of DRC updates
- determine if DRCs are up-to-date for all enabled checks

Physical and Spacing Workbook Views

Unlike *Electrical* worksheets, *Physical*, *Spacing*, and *Same Net Spacing* CSet worksheets include layers, which correspond to the cross-section view of your design. Furthermore, you can view these layers collectively (*All Layers*) or individually (*By Layer*). You can also view them at the CSet-level or at the Net-level. See “[All Layers / By Layer CSet Views](#)” on page 28. and “[All Layers Net View](#)” on page 29.

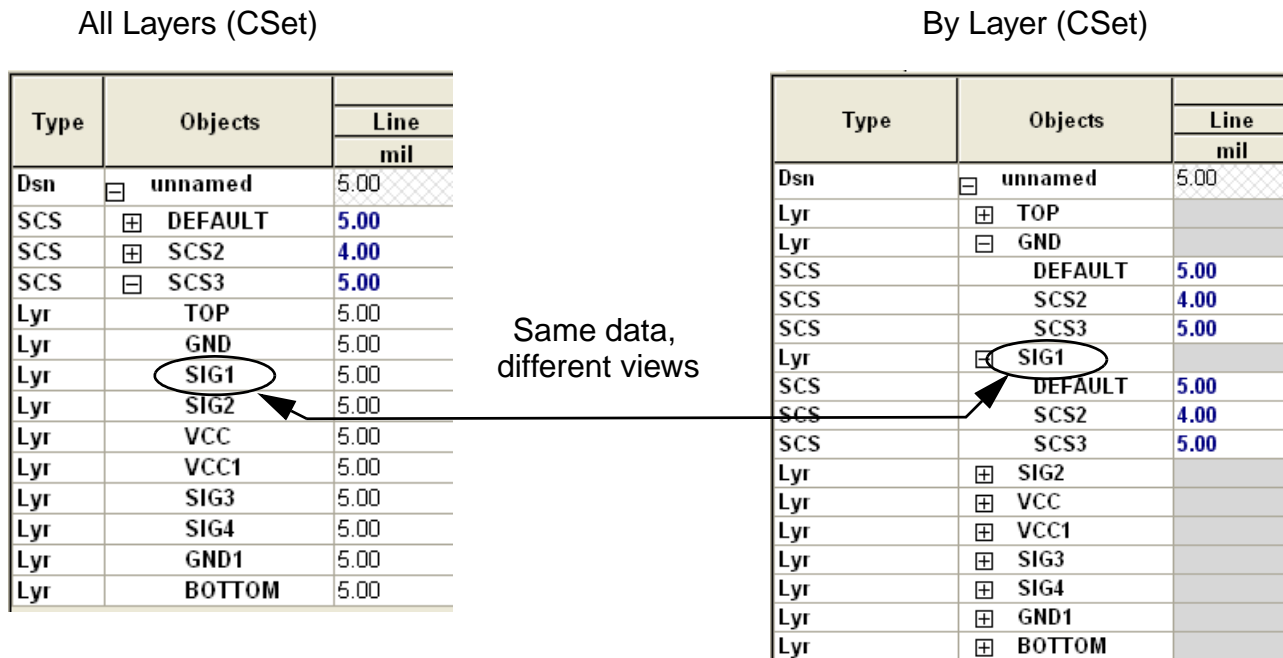
Important

By Layer view of *Physical* and *Spacing* cells is not available in Constraint Manager, when launched from OrCAD PCB Editor or Allegro PCB Editor, Performance L option.

Working in the CSet object folder lets you work in the abstract, defining CSets that will later be applied to net objects. The *All Layers* view shows CSets in collapsed form and layers associated with a CSet in expanded form. The *By Layer* view (in the CSet folder) shows layers in collapsed form and CSets associated with each layer in expanded form.

Note: The CSet view does *not* have a *Referenced CSet* column.

Figure 1-6 All Layers / By Layer CSet Views



Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Figure 1-7 All Layers Net View

Working in the *Net* object folder lets you define CSets based on existing CSets or based on constraints already on *Net* objects. The *All Layers* view shows *Container Net* objects in collapsed form and *Net* objects in expanded form.

Note: The *Net*, *Net Class-Class* (in the *Spacing* domain) and *Region* object types have a *Referenced CSet* column but do not have a *By Layer* view

All Layers (Net)

Type	Objects	Referenced Spacing CSet	Line
			mil
Dsn	<input type="checkbox"/> unnamed	DEFAULT	5.00
Bus	<input checked="" type="checkbox"/> ADDRESS_BUS	DEFAULT	5.00
DPr	<input checked="" type="checkbox"/> DP_MEM_CLOC	DEFAULT	5.00
DPr	<input checked="" type="checkbox"/> DP_PHASE_CLO	SCS2	4.00
DPr	<input type="checkbox"/> DP_SYNC_CLOC	SCS3	5.00
Net	SYNC_CLOCK_	SCS3	5.00
Net	SYNC_CLOCK_	SCS3	5.00
Net	BRANCH_0	DEFAULT	5.00

Same Net Spacing DRC Modes

As with other domains, you enable design rule checks for *all* layers through the *Analysis Modes* dialog box. However, in the *Same Net Spacing* domain, you can control design rule checks by layer.

You define a *Same Net Spacing CSet* in the CSet folder, and later assigning that CSet to a constraint object. In this way, you can enable or disable the CSet, effectively providing you with a granular level of control of setting constraint modes by layer. You do this by choosing TRUE or FALSE in the *Enable DRC By-Layer* column in the *Options* worksheet (see Figure 1-8).

Note: A by-layer constraint check is a slave to the mode for that constraint as set in the *Analysis Modes* dialog box (choose *Analyze – Analysis Modes*). That is, if the individual DRC is not enabled in the *Same Net Spacing Modes* tab, Constraint Manager ignores the state of the *Enable DRC By-Layer* column.

Figure 1-8 Layer-based DRC Modes

Type	Objects	Enable DRC By-Layer
*	*	*
Dsn	<input type="checkbox"/> dp3	TRUE
SNSC	<input checked="" type="checkbox"/> DEFAULT	TRUE
SNSC	<input type="checkbox"/> ERASE	TRUE:FALSE:TRUE:FA...
Lyr	TOP	TRUE
Lyr	GROUND1	FALSE
Lyr	INNER1	TRUE
Lyr	INNER2	FALSE
Lyr	GROUND2	FALSE
Lyr	INNER3	FALSE
Lyr	INNER4	FALSE
Lyr	POWER1	FALSE
Lyr	POWER2	TRUE
Lyr	POWER3	FALSE
Lyr	POWER4	FALSE
Lyr	GROUND3	TRUE
Lyr	INNER5	FALSE
Lyr	INNER6	FALSE
Lyr	GROUND4	FALSE
Lyr	BOTTOM	FALSE
SNSC	<input type="checkbox"/> POWER	FALSE
Lyr	TOP	FALSE
Lyr	GROUND1	FALSE
Lyr	INNER1	FALSE

Type	Objects	Referenced Same Net Spacing CSet	Enable DRC By-Layer
*	*	*	*
Dsn	<input type="checkbox"/> dp3	DEFAULT	TRUE
NClS	ERASE	ERASE	TRUE:FALSE:TRUE:FALSE
NClS	POWER	POWER	FALSE
DP	<input checked="" type="checkbox"/> DIFFPAIR0	DEFAULT	TRUE

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Cells

Cells hold data, results, or calculations. Constraint Manager uses different colors or shades of color in cells depending on the state of your design and on the scope of the data in the cell.

2000 MIL

Indicates a value inherited from a higher-level object, such as a CSet.

216.7 MIL

Indicates a Pass state — A value indicating that the cell falls within the set constraint limit.



Indicates a Pass state — A value indicating that all child cells of the parent object fall within the set constraint limit.

1975 MIL

Indicates a directly set value in a cell. Also called an override.



Indicates a value that cannot be computed. The reason appears when you hover your mouse over a yellow cell and observe the message displayed in the status line, located at the lower-left corner of Constraint Manager.

You may have not have . . .

- enabled the DRC mode for the cell
- completely placed the object
- completely routed the object
- correctly set up simulation parameters

Note: You can set certain constraints, such as differential pairs, in more than one domain. Constraint Manager indicates a constraint edit in one domain by coloring the cell of the same constraint in the opposite domain yellow.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

-245.3 MIL

Indicates a Fail state — A value indicating that the cell violates the set constraint limit. Constraint Manager rolls up worst-case Margins to higher-level objects.



Indicates a Fail state — A value indicating that any child cell of the parent object violates the set constraint limit.



Indicates a cell which is not applicable to the object. These cells never contain values.



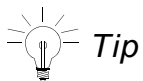
Indicates a cell that you cannot edit.

6354.06

Indicates a cell containing a formula (the red bar to the right of the cell).

AD3

Indicates a cell that is bookmarked.



Tip

To guide you in entering data into a cell, right-click in the cell and choose *Change* from the pop-up menu.

Constraint Manager's User Interface Controls

Constraint Manager employs the same conventional window and worksheet controls that are used in Microsoft Windows Explorer® and Microsoft Excel®. Constraint Manager also supports the Microsoft Intellimouse® and wheel mouse.

Table 1-2 User Interface Controls

Task	Feature	Usage
Command Access	■ Pull-down Menus	Click the pull-down menu at the top of Constraint Manager to access commands.
	■ Icons	Click an icon to execute a command. If you briefly hover the cursor above an icon, a tool tip displays in the status bar (located at the lower-left corner of Constraint Manager) describing the icon's function.
	■ Keyboard Shortcuts	Press Control and press: <p>p (to print) z (to undo) c (to cut) v (to paste) f (to find) d (to delete)</p> <p>Also, you can access many commands by pressing Alt along with the underlined character, and you can assign your own shortcuts (see the Tools – Customize Shortcut Keys command in the <i>Constraint Manager Reference</i>).</p>
Window and Worksheet Sizing and Placement	■ Right-Click (context sensitive)	Depending on the object selected, you can right-click to quickly access a command to act on that object.
	■ Drag and Drop	You can drag to reposition the Constraint Manager window, and individual worksheets, on your desktop.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Task	Feature	Usage
	■ Sizing Borders	You can resize the Constraint Manager window or an individual worksheet open within Constraint Manager by dragging the border.
	■ Maximize/Minimize	You can minimize an open worksheet to an icon or you can maximize it to focus only on that worksheet.
	■ Dismiss	You can click the dismiss [X] button (located at the top right-corner of the worksheet) to close a worksheet. Constraint data is not lost when you dismiss a worksheet.
Worksheet Viewing	■ Window Select	You can click and drag on an open worksheet to reposition it.
	■ Object Expand/Collapse	<p>You can use the worksheet selector to work at any object level in the hierarchy (from the system level to the pin pair level) by expanding [+] and collapsing [-] the object tree-structure.</p> <p>You can also choose <i>Objects – Expand</i> and <i>Objects – Collapse</i> from the pull-down menus to achieve the same effect.</p>
	■ Cascade	<p>You can view all open worksheets arranged one-behind-the-other by cascading (<i>Window – Cascade</i>).</p> <p>Constraint Manager orders Worksheets so that each is selectable with a click of the mouse. The active window is placed in the foreground and is identifiable by an active (selected) border.</p>
	■ Tile	You can view all open worksheets simultaneously by tiling (<i>Window – Tile</i>). Each open worksheet is automatically sized to accommodate the size of the Constraint Manager window.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Task	Feature	Usage
	■ New Window	You can duplicate the content of the active worksheet in a new window. This lets you to focus your view on different objects in the same worksheet.
	■ Worksheet Tab Select	When you expand a constraint discipline (signal integrity, timing, routing) from the worksheet selector, all objects within that discipline appear in a worksheet window. You then click a related tab to activate the desired worksheet. You may have to scroll horizontally until the desired worksheet tab is visible.
	■ Synchronize Rows	When you modify rows of the worksheet in focus (for example, scrolling down or expanding a bus), Constraint Manager promotes the same modification to all worksheets that have synchronization enabled.

Accelerator Keys

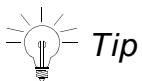
Constraint Manager provides function keys and modified function keys that provide quick access to common functions.

Function	Keys
View Options	Cntrl+F6
Rename	F2
Print	Cntrl+p
Analysis Modes	Cntrl+F9
Analysis Settings	Shift+F9
Analyze	F9
Find	Cntrl+f

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Function	Keys
Find Next	F3
Find Previous	Shift F3
Open a new window on the active worksheet	Cntrl+n
Move to the next open worksheet	Cntrl+Tab
Move to the previous open worksheet	Shift+Cntrl+Tab
Next worksheet tab	F6
Previous worksheet tab	Shift+F6
Close the active worksheet	Cntrl+F4
Select a contiguous range of cells	Shift+Click
Select a non-contiguous range of cells	Cntrl+Click
Expand object rows	Alt+ or Num+
Collapse object rows	Alt- or Num-
Cut	Cntrl+x or Shift+Del
Copy	Cntrl+c or Cntrl+Ins
Paste	Cntrl+v or Shift+Ins
Delete an object or cell content	Del
Enable object bookmarks	Cntrl+F5
Next object bookmark	F5
Previous object bookmark	Shift+F5



For information on how to assign your own accelerator keys to commands, or how to reassign Constraint Manager's default assignments, see the [Tools – Customize Shortcut Keys](#) command in the *Constraint Manager Reference*.

Enhancements Done in 16.3

- [Online Formula Support](#)
- [Differential Pair Naming](#)
- [Changes in the Edit Via List Dialog Box](#)
 - [Via List Viewer](#)
 - [Sorted Via List](#)
 - [New Options in the Draw options Dialog Box](#)
- [Advanced Tabbed View](#)
- [New Look Analysis Modes Dialog Box](#)
- [Goto Constraint Set Command](#)
- [Object Membership Count](#)
- [Filter Functionality](#)
- [Suffix Indicating Number of Members](#)
- [Miscellaneous Changes](#)

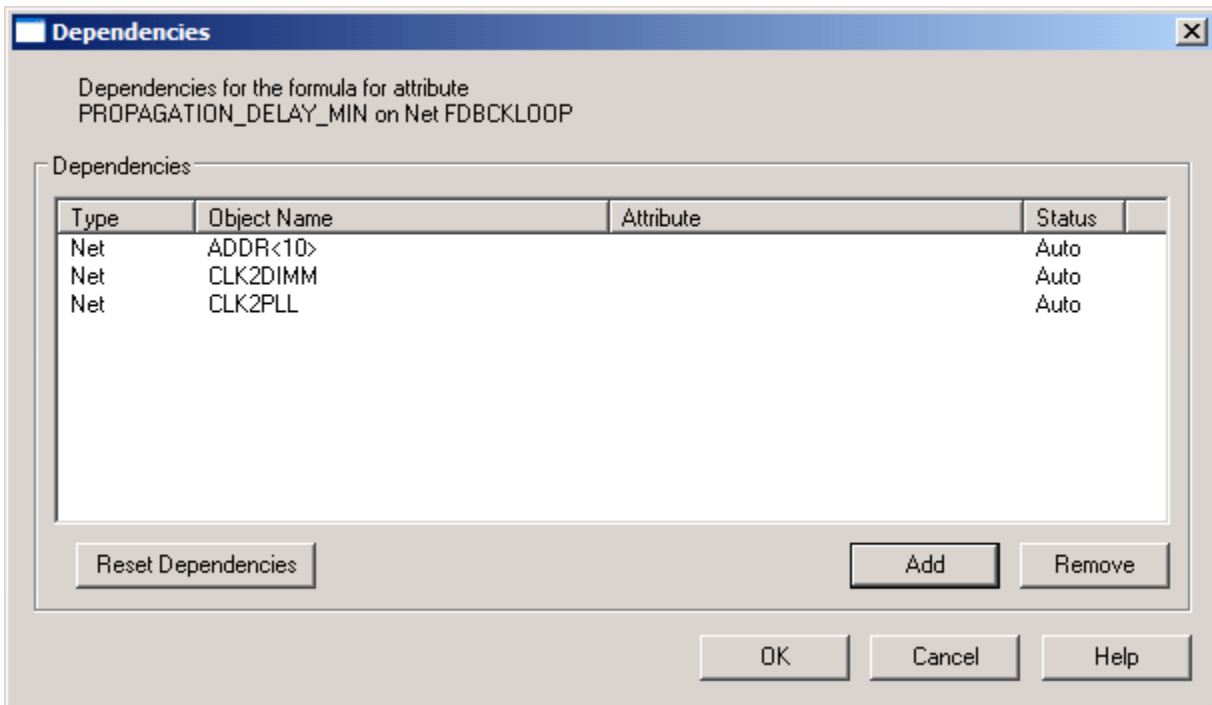
Online Formula Support

Since the initial release of Advanced Constraints, one of the recognized limitations was that formulas had to be recalculated manually. The Online Formulas feature addresses this limitation by tracking the dependencies that formulas have on design objects and optionally updating the formula automatically. These dependencies on other cells or database objects

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

(for example, the value in a cell, or the etch length of a net) are automatically created but can also be edited in a new dialog.

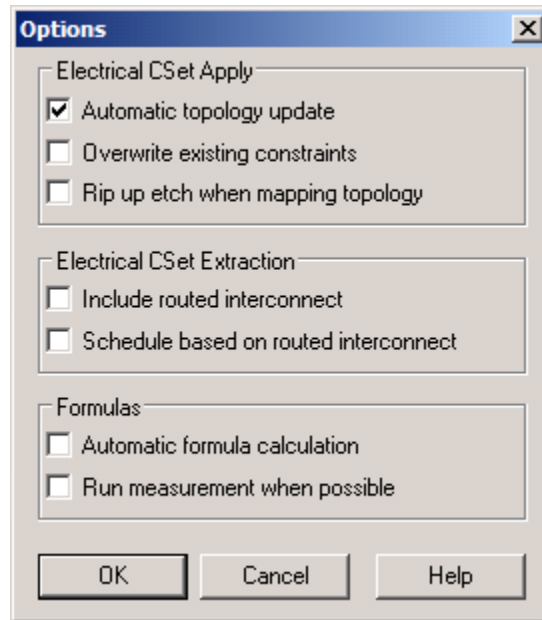


When one of those dependencies changes, the formula is marked as *out of date*. If automatic formula calculation is on, out of date formulas recalculate their new values as needed. If automatic formula calculation is set to off, out of date formulas are indicated by the background of the cell turning yellow.

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Choose the *Edit – Calculate* or *Edit – Calculate All* menu commands to update the formulas. You can turn on the *Automatic formula calculation* feature using the corresponding options in the Options dialog box (*Tools – Options*).



See [“Calculating a Formula”](#) on page 161. Also see [Allegro Constraint Manager Reference Guide](#) for more information.

Differential Pair Naming

You now have the ability to rename library- and model-defined differential pairs. Previously, the system-generated names could not be changed.

See [Renaming Differential Pairs](#) in *Allegro Design Entry HDL-Constraint Manager User Guide* for more information.

Changes in the Edit Via List Dialog Box

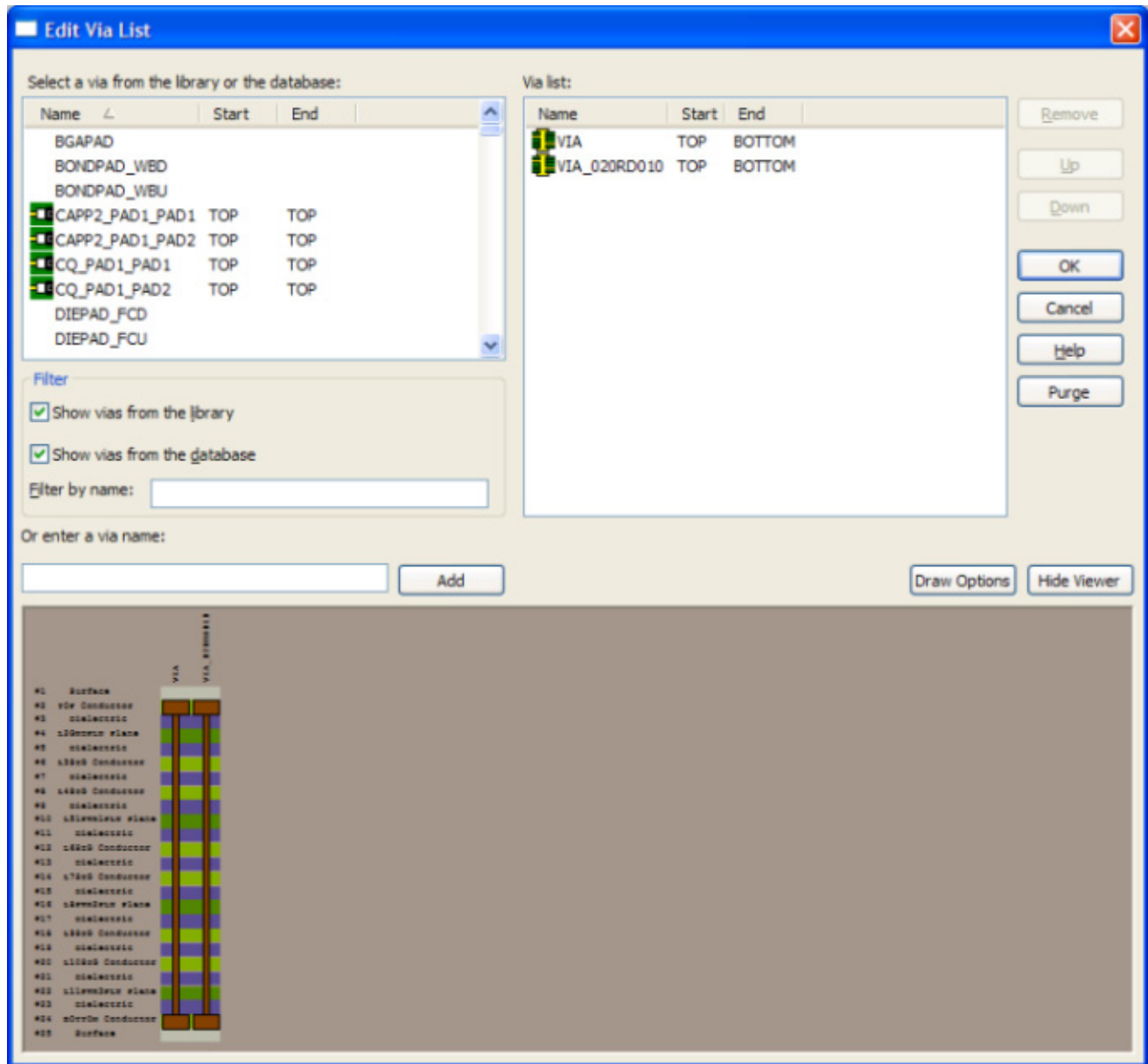
Via List Viewer

The Via List dialog now supports a graphical view of the vias associated with the respective Physical CSet. To access the via list viewer, right-click a Via List cell in the Physical domain of Constraint Manager and choose *Change* from the pop-up menu. The Via List Viewer appears at the bottom pane of the Edit Via List dialog box. You can toggle between showing

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

and hiding the Via list viewer using the *Show Viewer* (or *Hide Viewer*) button in the Edit Via List dialog box.

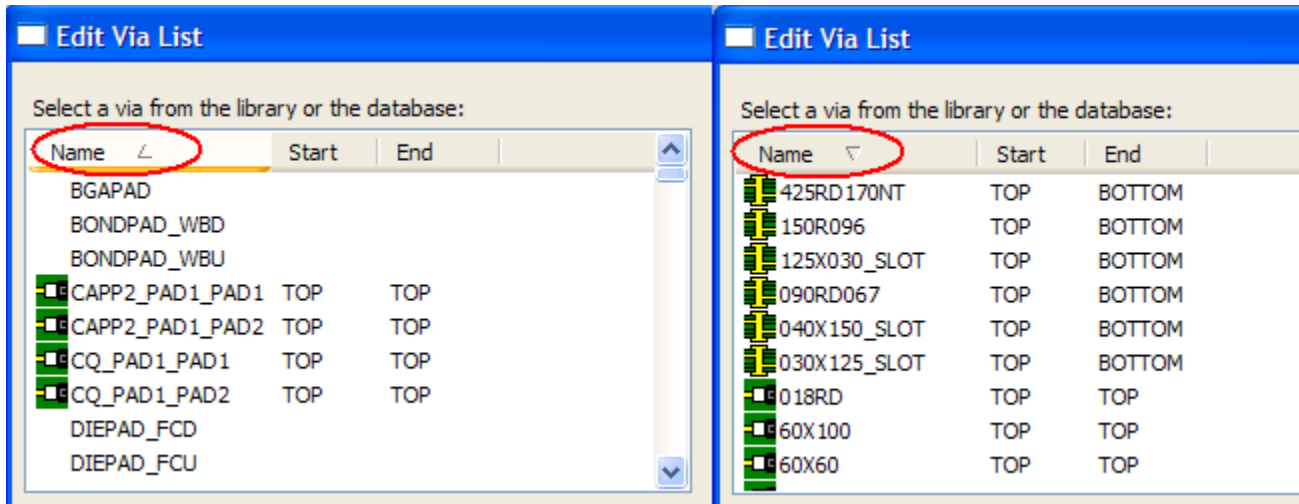


Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Sorted Via List

You can also sort the via list alphabetically. The arrow on the name field points downwards or upwards depending on whether the list is sorted in increasing or decreasing order.

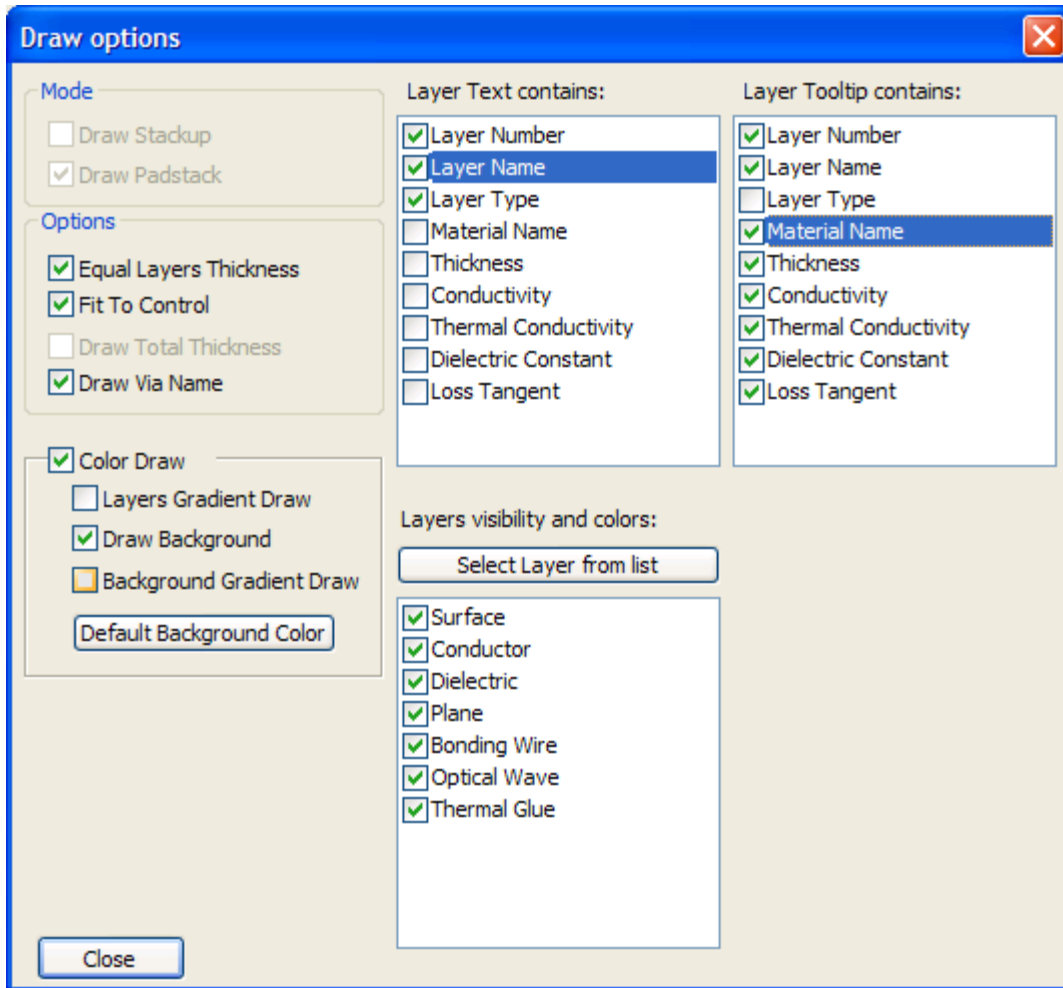


Allegro Constraint Manager User Guide

Welcome to Constraint Manager

New Options in the Draw options Dialog Box

There are new controls for color selection, layer visibility, and tooltips in the Draw options dialog box.



Note: The display of vias represents the ones which comprises a Via List and not necessarily the entire set used in the Design.

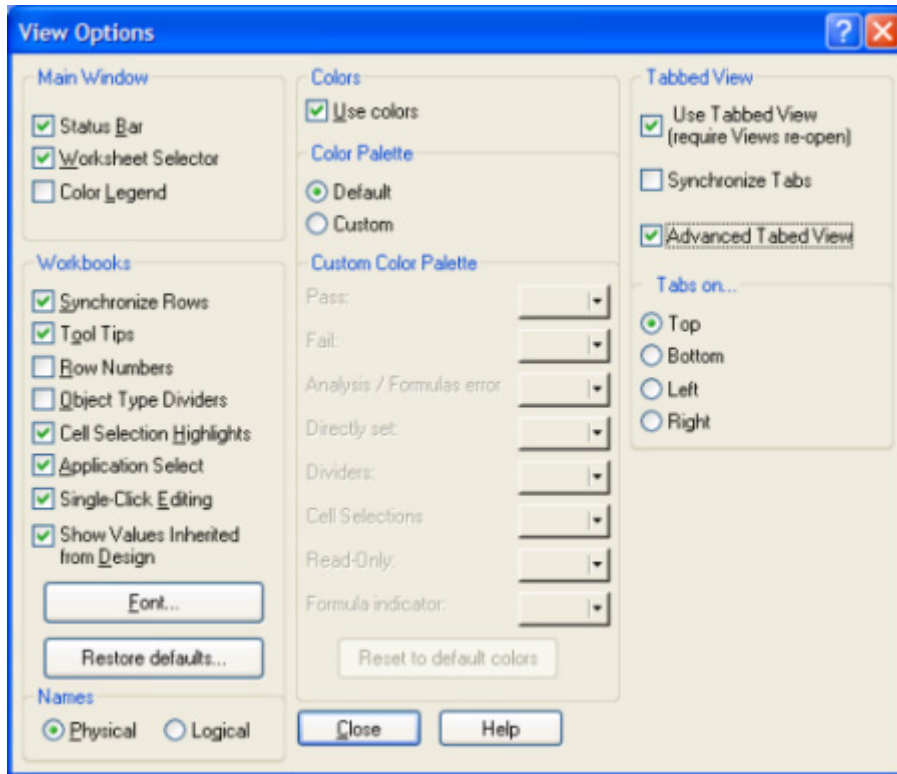
Advanced Tabbed View

With the new *Advanced Tabbed View* option, you now have better control over the Design tabs. This is especially useful in the front-end designs with hierarchical designs containing many unique designs, such that the tabs shrink so much that they are not readable. To enable

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

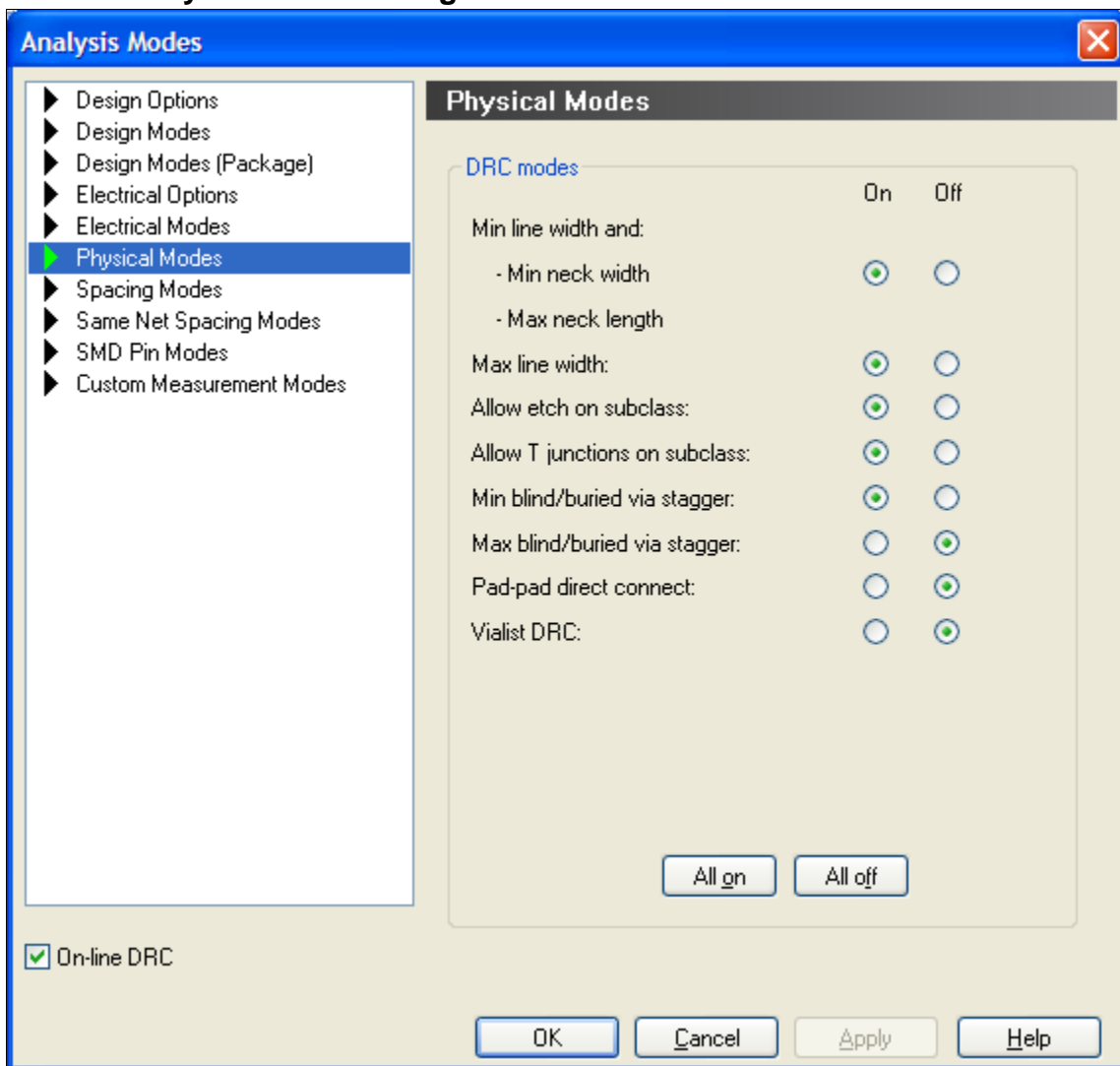
this option, choose *View – Options* and select the *Advanced Tabbed View* check box in the View Options dialog box.



New Look Analysis Modes Dialog Box

The tabbed Interface in Analysis Modes dialog box has been enhanced for a cleaner look and faster navigation.

Figure 1-9 Analysis Modes Dialog Box



Goto Constraint Set Command

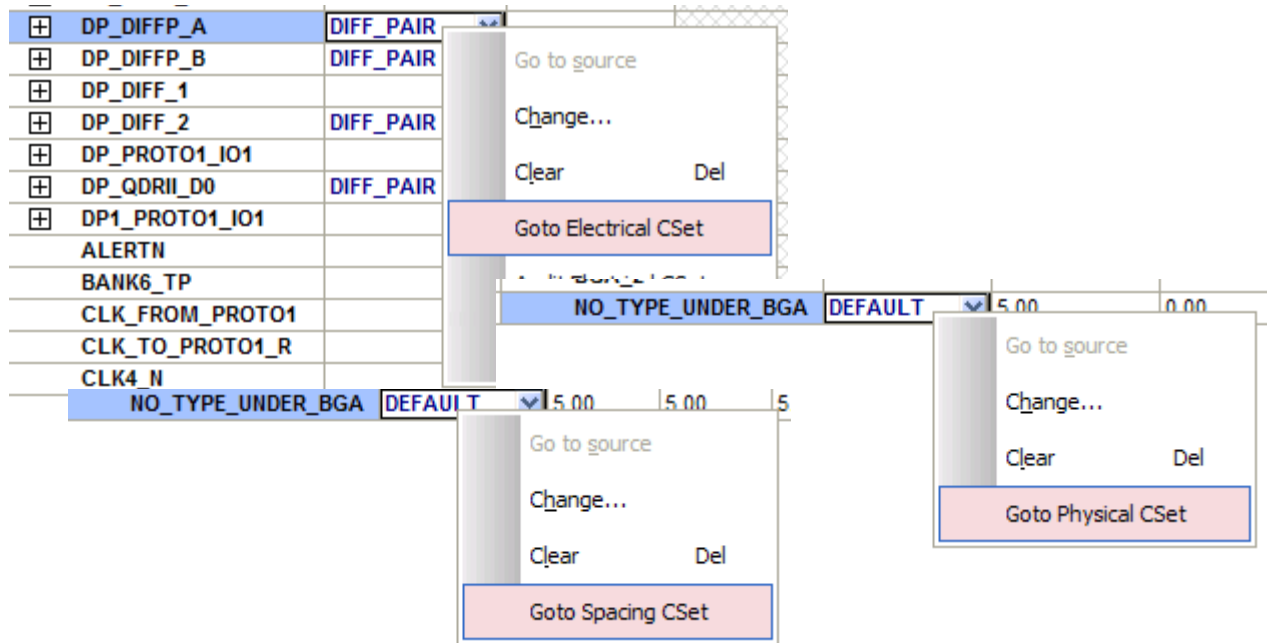
A new menu command, Goto Constraint Set, has been included in the RMB menu of the *Referenced Constraint Set* cell for all the domains:

- Electrical - *Goto Electrical CSet*
- Spacing - *Goto Spacing CSet*

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

■ Physical - *Goto Physical CSet*

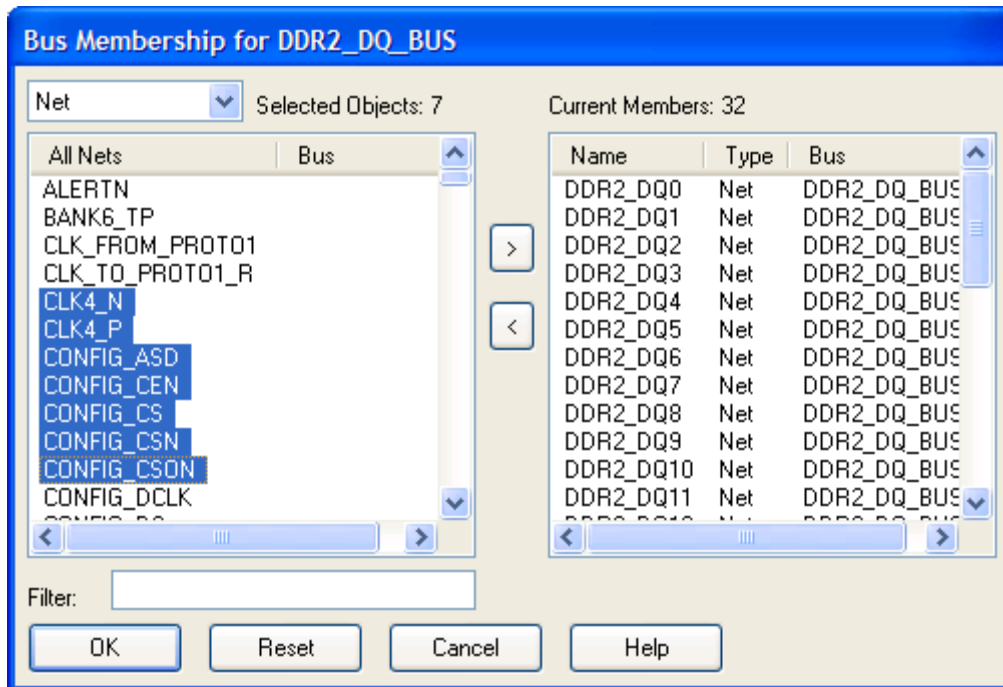


Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Object Membership Count

Two counters have been added to the Group Membership dialog box to show number of selected objects and total number of members in the group.

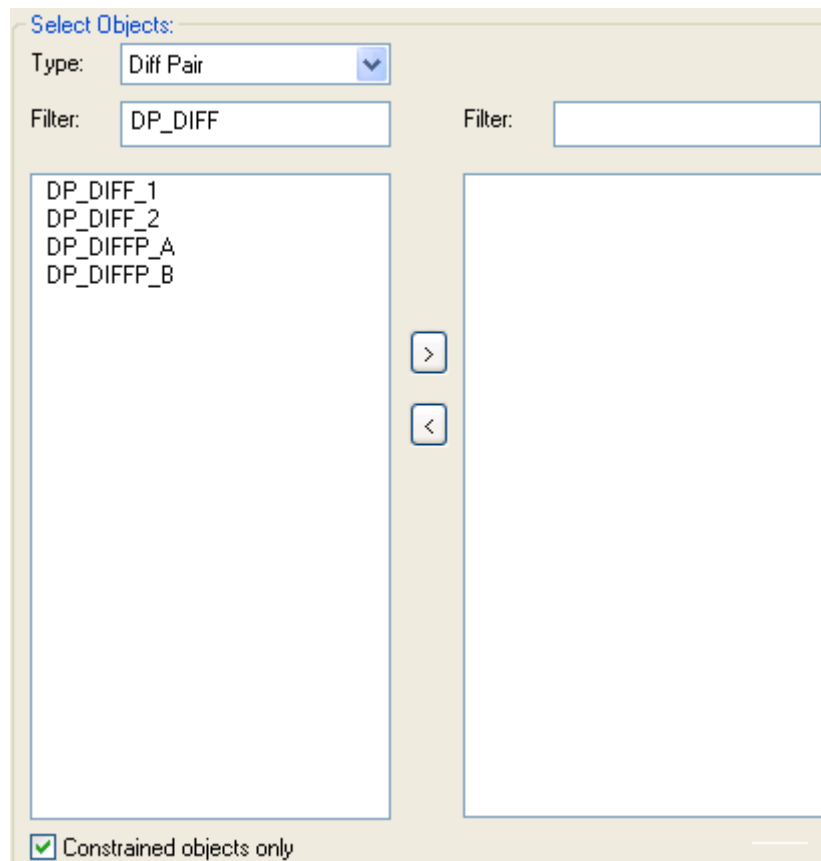


Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Filter Functionality

The *Filter* field functionality in various dialog boxes of Constraint Manager has been aligned with similar fields in the Allegro application dialogs. The Filter field automatically assumes an asterisk (*) at the end of a string and then prunes the list it applies to as you type.



Suffix Indicating Number of Members

Buses, match groups, and Net Class names in the Constraint Manager worksheets are suffixed with the number of members they contain. For example in the figure below The match group DDR_LANE1 (14) has 14 members and the bus DDR2_DQ_BUS (32) has 32 members. Similarly, the Net Class named, NET_CLS1 (6) contains six members.

MGrp	⊕	DDR2_LANE0 (9)
MGrp	⊕	DDR2_LANE1 (14)
MGrp	⊕	DDR2_LANE2 (12)
MGrp	⊕	DDR2_LANE3 (10)
Bus	⊕	DDR2_DQ_BUS (32)

NCIs	⊕	NET_CLS1 (6)
NCIs	⊕	NET_CLS2 (8)
NCIs	⊕	NET_CLS3 (4)

Allegro Constraint Manager User Guide

Welcome to Constraint Manager

Miscellaneous Changes

- [No Constraint Manager UI in Non-Graphic Mode](#)
- [Copying Complex Constraints](#)
- [Resizable Dialog Boxes](#)

No Constraint Manager UI in Non-Graphic Mode

The Constraint Manager UI is no longer displayed when you run Allegro in the *-nograph(ics)* mode.

Copying Complex Constraints

You can now copy and paste complex constraints such as `VIA_LIST`.

Resizable Dialog Boxes

Some of the dialog boxes, such as the Bus Membership dialog box and the Edit Via List dialog boxes can now be resized.

Working with Constraint Objects

Topics in this chapter include

- [About Constraint Object Hierarchy](#) on page 50
- [About Objects](#) on page 53
- [Designs and Systems](#) on page 53
- [Net Class](#) on page 54
- [Net Class-Class](#) on page 55
- [Differential Pairs](#) on page 56
- [Match Groups](#) on page 67
- [Buses](#) on page 84
- [Nets and Xnets](#) on page 85
- [Pin Pairs](#) on page 86
- [Ratsnest Bundle](#) on page 89
- [Region](#) on page 90
- [Region Class](#) on page 91
- [Region Class-Class](#) on page 91

About Constraint Object Hierarchy

This chapter presents information on how to use hierarchical constraint objects in Constraint Manager. See Chapter 3, [“Working With Reusable Constraint Objects — CSets”](#) on page 93 for information on reusable constraint objects — Constraint Sets.

Constraint Manager enforces a precedence on objects in your design. Constraints that you specify at the top of the object hierarchy become inherited by the next lower-level object in the hierarchy. Constraints that you define at the lower-levels of the object hierarchy take precedence over (override) the same constraints defined at the next higher-level in the object hierarchy.

This ordering of objects lets you define constraints at highest level possible, only setting overrides on lower-level objects where necessary. Refer to [“Constraint Object Hierarchy”](#) on page 51 for information on constraint objects and their precedence.



You should work at the highest level possible in the constraint object hierarchy. For example, you should group individual address signals into a *Bus*. In this way, you only have to constrain the single *Bus* object once rather than having to repeat this process for each signal. Through inheritance, each signal in the *Bus* receives the constraint value assigned at the *Bus* level.

In certain worksheets in the *Electrical* domain, the children of an object reflect the results of an analysis and are not used for the constraint precedence hierarchy. These result-objects are not differentiated from the normal constraint hierarchy but will be maintained for reading. You cannot edit these constraints.

Allegro Constraint Manager User Guide

Working with Constraint Objects

See Chapter 1, Constraint Architecture in the *Allegro® Platform Constraints Reference* for detailed information on the Allegro System Constraint Architecture and descriptions of individual constraints.

Table 2-1 Constraint Object Hierarchy

	Electrical	Physical	Spacing (net-to-net / same-net)	
	-----	Design	Design	
	Net Class	Net Class	Net Class	
	Bus	Bus	Bus	
	Differential Pair	Differential Pair	Differential Pair	
	Match/Relative Group	-----	-----	
	Xnet	Xnet	Xnet	
	Net	Net	Net	
	Pin Pair	Pin Pair	Pin Pair	
	-----	-----	Net Class-Class *	
	-----	Region	Region	
	-----	Region Class	Region Class	
	-----	-----	Region Class-Class *	

I
N
H
E
R
I
T
A
N
C
E



O
V
E
R
R
I
D
E



* Not available in the Same Net Spacing domain

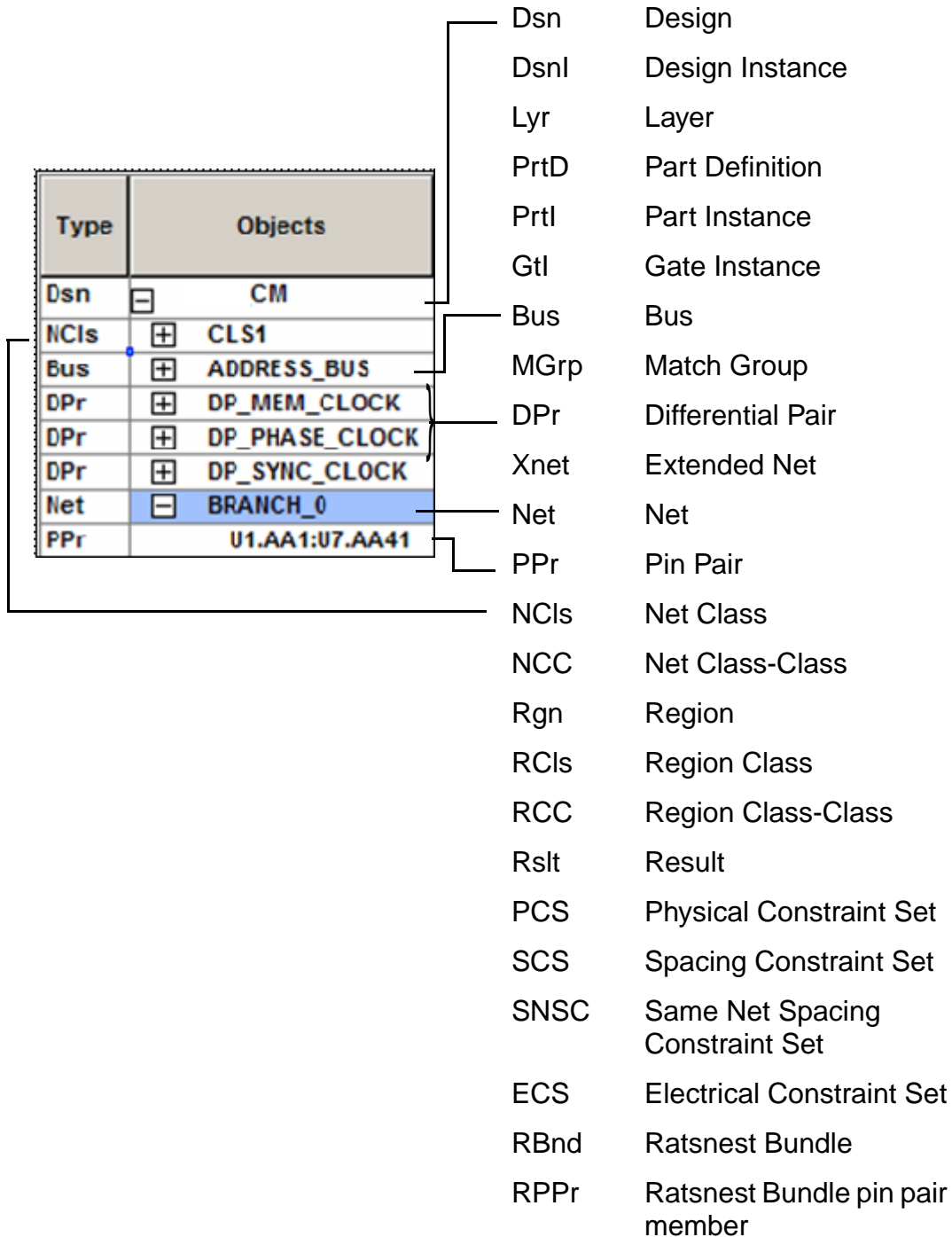
Types

The *Type* column indicates the object type for the selected row, as depicted in [Figure 2-1](#) on page 52.

Allegro Constraint Manager User Guide

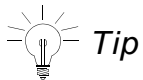
Working with Constraint Objects

Figure 2-1 Constraint Types



About Objects

This section describes objects and object groupings in Constraint Manager. See “[About Constraint Object Hierarchy](#)” on page 50 for more information. Refer to the *Allegro[®] Constraint Manager Reference* for detailed, step-by-step procedures.

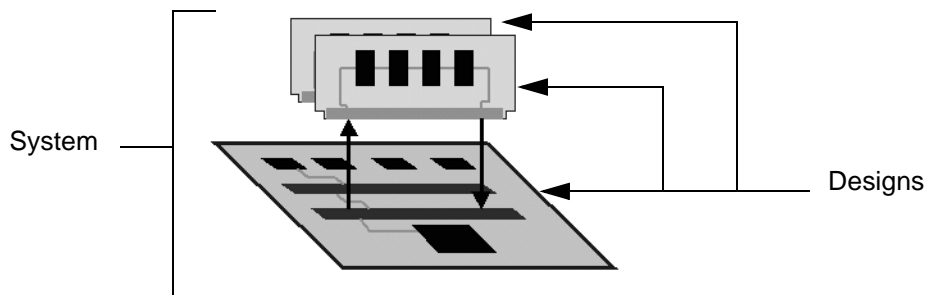


Tip

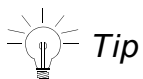
You can bookmark any design element by selecting it in the *Objects* column, then right-clicking and choosing *Bookmark – Object Bookmark* from the pop-up menu. A square appears to the left of the object to aid you in locating the object. The bookmark follows the object across worksheets. You can also cycle through defined bookmarks, and remove them as well.

Designs and Systems

A *Design* represents a stand-alone board or a board in a *System*, a schematic in Allegro Design Entry HDL, or behavioral logic in Allegro Design Architect. In a multi-board configuration, each board becomes a separate design in the system.



A *System* represents a configuration of designs (boards) including Xnets that traverse these designs and their interconnecting cables and connectors.



Tip

Constraint Manager’s [Tabbed View](#) aids you in quickly selecting from participating designs in a system.

Net Class

A *Net Class* constraint object lets you group net objects that share common characteristics and require a similar constraint requirement.

Type	Objects
Dsn	<input type="checkbox"/> unnamed
NClS	<input type="checkbox"/> CLS1
Net	S_ADDR2
Net	S_ADDR3
Net	S_ADDR4
Net	S_ADDR5

Allowable members of a *Net Class* include buses, differential pairs, Xnets, and nets.

See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Net Class](#) command in the *Constraint Manager Reference* for more information on the *Net Class* constraint object.

Net Class Rules

The following rules apply to creating a *Net Class*

- You can constrain a *Net Class* with a *CSet*
- You can override individual members of a *Net Class*
- You can constrain a *Net Class* directly (though we recommend using a *CSet*)
- As you create a *Net Class* in the *Physical* domain, you can specify that it also occurs in the in the *Spacing* domain. The converse is true.
- A *Net Class* in the *Electrical* domain must be unique to that domain
- A *Net Class* created in the *Spacing* domain carries over to the *Same Net Spacing* domain; the converse is also true
- A net can be a member of only one *Net Class* per domain

Net Class-Class

A *Net Class-Class* is a constraint object that you define and constrain to represent an *intra*-class spacing relationship among members within a *Net Class* or an *inter*-class spacing relationship among objects in the different *Net Classes*.

Type	Objects
Dsn	<input type="checkbox"/> unnamed
NClS	CLS2
NClS	<input type="checkbox"/> CLS3
NCC	CLS2

Allowable members of a *Net Class-Class* include *Net Classes*.

See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Net Class-Class](#) command in the *Constraint Manager Reference* for more information on the *Net Class-Class* constraint object.

Net Class-Class Rules

The following rules apply to creating a *Net Class-Class*. You can

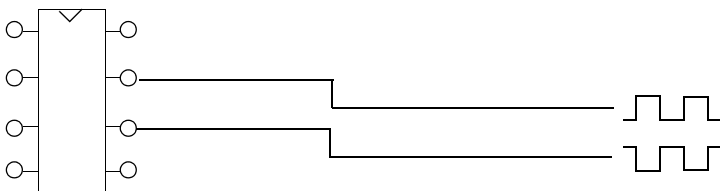
- constrain a *Net Class-Class* with a *CSet*
- override individual members of a *Net Class-Class*
- constrain a *Net Class-Class* directly (though we recommend using a *CSet*)
- create a *Net Class-Class* only in the *Spacing* domain

Note: *Net Class-Classes* are not supported in the *Same Net Spacing* domain, or in Design Entry HDL or Allegro System Architect.

Differential Pairs

A *differential pair* represents a pair of Xnets or nets that are routed differentially. Differential signaling is a method of sending the same information over two traces.

Differential data transfer employs two traces and (at least) one driver with a positive and negative output and a two terminal receiver. For digital applications, the driver terminals are sending out signals of opposite polarity. While the non-inverted (positive) output transmits a low-to-high transmission, the inverted (negative) output transmits a high-to-low transmission.



To learn more about working with Differential Pair constraints, see

- ❑ [Best Practices: Working with Differential Pairs](#)
- ❑ the [Objects – Create – Differential Pairs](#) command in the *Constraint Manager Reference* for more information on the *Differential Pair* constraint object.
- ❑ [Differential Pair Constraint Data Sheets](#) in the *Allegro Platform Constraints Reference*

Constraint Manager supports two types of differential pairs:

■ Model-defined Differential Pairs

You specify model-defined differential pairs in a device signal model by designating inverting and non-inverting signals of the differential pair. You can uniquely characterize the differential pair by specifying pin parasitics, launch delays, logic thresholds, and buffer delays.

You assign device signal models to components using the PCB editor, APD, or SigXplorer. Constraint Manager then recognizes the model-defined differential pair through its view of the board database.

Allegro Constraint Manager User Guide

Working with Constraint Objects

■ User-defined differential pairs

You can create user-defined differential pairs directly in Constraint Manager on a net-level object. This affords you more flexibility in renaming differential pair objects and changing differential pair membership, but you forgo the accuracy of model-defined differential pairs.

Note: Constraint Manager does not support system-level differential pairs.

Differential Pair Worksheets

In the *Electrical* domain, you specify differential pair constraints globally in the *Differential Pair* worksheet of the *Routing* workbook. In the *Physical* domain, you specify differential pair constraints in both the *Net* and *Region* worksheets. If layer variances are required, you can specify *By Layer* differential pair constraints in the *Physical Constraint Set* folder.

Important

The DIFFP_PRIMARY_GAP, MIN_LINE_WIDTH, DIFFP_NECK_GAP and MIN_NECK_WIDTH constraints are allowed on both *Electrical CSets* and *Physical CSets*; however, the *Electrical CSet* value takes precedence over the *Physical CSet* value. Furthermore, in the *Physical* domain, a *Region* takes precedence over an override. See [Constraint Objects and Hierarchy](#) in the *Allegro Platform Constraints Reference* for more information.

Table 2-2 Differential Pair Constraints by Domain

Constraint/ Property	Domain	Column Super Header	Column Label
DIFFP_MIN_SPACE	Physical	Differential Pair	Min Line Spacing
DIFFP_MIN_SPACE	Electrical	Line Spacing	Min
DIFFP_PRIMARY_GAP	Physical	Differential Pair	Primary
DIFFP_PRIMARY_GAP	Electrical	Coupling Parameters	Primary Gap
DIFFP_NECK_GAP	Physical	Differential Pair	Neck
DIFFP_NECK_GAP	Electrical	Coupling Parameters	Neck Gap
DIFFP_COUPLED_PLUS	Physical	Differential Pair	(+) Tolerance
DIFFP_COUPLED_PLUS	Electrical	Coupling Parameters	(+) Tolerance
DIFFP_COUPLED_MINUS	Physical	Differential Pair	(-) Tolerance

Allegro Constraint Manager User Guide

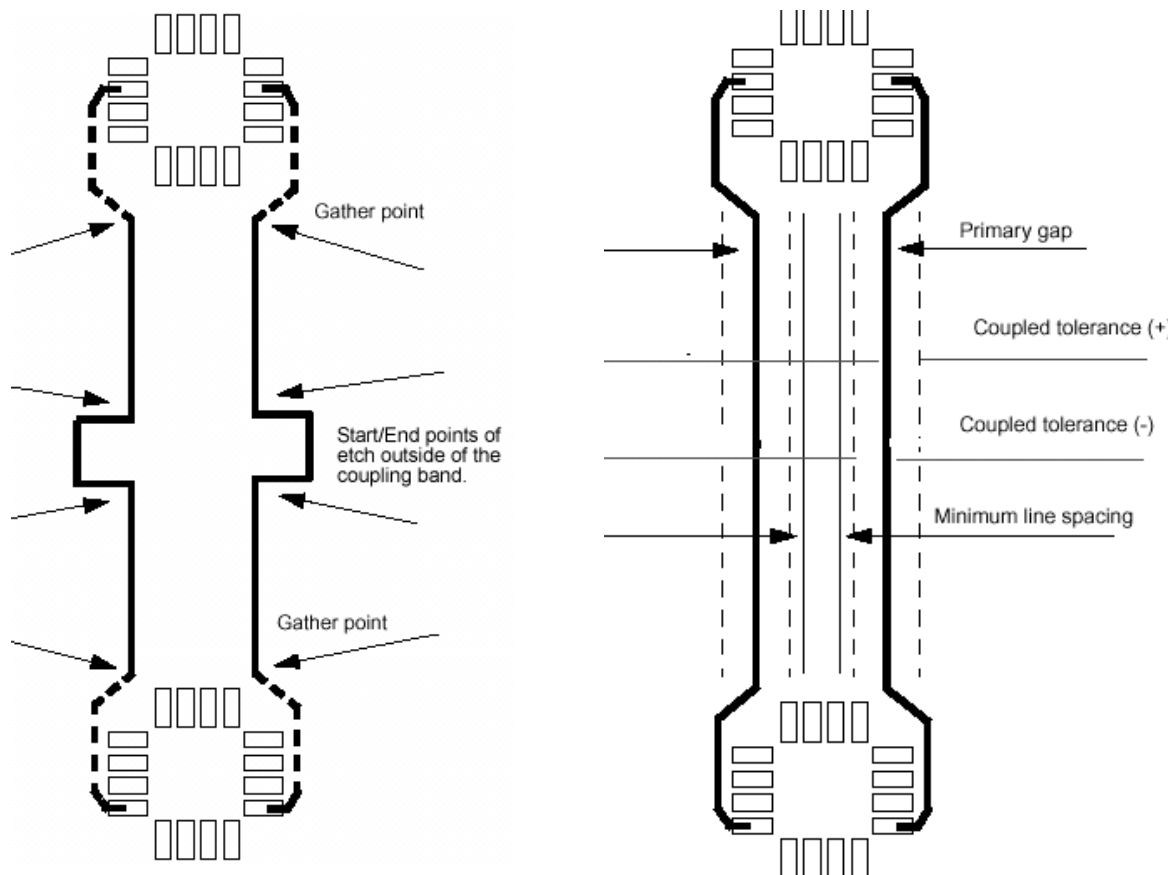
Working with Constraint Objects

Table 2-2 Differential Pair Constraints by Domain

Constraint/ Property	Domain	Column Super Header	Column Label
DIFFP_COUPLED_MINUS	Electrical	Coupling Parameters	(-) Tolerance
DIFFP_UNCOUPLED_LENGTH	Physical	Uncoupled Length	Max
DIFFP_UNCOUPLED_LENGTH	Electrical	Uncoupled Length	Max
MIN_LINE_WIDTH	Physical	Line Width	Min
MIN_LINE_WIDTH	Electrical	Coupling Parameters	Primary Width
MAX_LINE_WIDTH	Physical	Line Width	Max
MIN_NECK_WIDTH	Physical	Neck	Min Width
MIN_NECK_WIDTH	Electrical	Coupling Parameters	Neck Width
MAXIMUM_NECK_LENGTH	Physical	Neck	Max Length
DIFFP_GATHER_CONTROL	Physical	Uncoupled Length	Gather Control
DIFFP_GATHER_CONTROL	Electrical	Uncoupled Length	Gather Control
GATHER_LENGTH_IGNORED	Electrical	Uncoupled Length	Length Ignored
DIFFP_PHASE_TOL	Physical	Phase Tolerance	Tolerance
DIFFP_PHASE_TOL	Electrical	Phase Tolerance	Tolerance

Figure [2-2](#) shows the boundaries and events that trigger differential pair rule checking and analysis.

Figure 2-2 Differential Pair Gather Points and Coupling Bands



The *Differential Pair* worksheets contain four major constraint categories:

■ *Uncoupled Length*

Uncoupled length constraints limit the amount of coupling between differential pair members. When *gather control* is set to *ignore*, the *actual* uncoupled length is the cumulative etch that lies outside of the coupling band yet lies within the boundaries of the two gather points, or from driver to receiver. A violation results when the uncoupled length exceeds the value you specify in the *max* cell.

The *length ignored* cell contains the *actual* gather length that is ignored, which is reported on the member net or Xnet and does not bubble up to the differential pair object.

Allegro Constraint Manager User Guide

Working with Constraint Objects

■ *Phase Tolerance*

Phase tolerance constraints ensure that differential pair members are in synchronization and in phase as they switch. You enter a *tolerance* value as a function of time (in nano-seconds) or length (in mils). The *Actual* value reflects the difference in time or length between the members of the differential pair. A violation occurs when the actual value exceeds the tolerance value.

■ *Line Spacing*

The *minimum line spacing* constraint specifies the minimum distance allowed between any two segments of each member of the differential pair. After analysis, the actual cell contains the value containing the smallest gap spacing. A violation occurs when the *Actual* spacing is less than the *min* value.

Note: The minimum line spacing constraint value that you enter must be less than or equal to the *Primary Gap* minus the (-) *Tolerance*, and it must be equal to or less than the *Neck Gap* minus the (-) *Tolerance*.

■ *Coupling*

Coupling constraints determine the uncoupling events for a routed differential pair. Constraint Manager uses these events to determine the *uncoupled length* and *phase deviation*. Use the *differential calculator* to determine values to enter into the *primary gap*, *neck gap*, and *tolerance* cells. See [“Using the Differential Calculator”](#) on page 61 for more information.

- In the *Primary Width* cell, you enter a value for the ideal width of each member of the differential pair.
- In the *Primary Gap* cell, you enter a value for the ideal edge-to-edge spacing between the pair that should be maintained for the entire length of the pair. In the (+/-) *Tolerance* cells, you enter values to define two bands around the primary gap in which the lines of a pair can go beyond or closer than the primary gap value. When the lines of etch are within these bands, they are considered coupled.
- In the *Neck Width* cell, you enter a value for the minimum allowable width for a line in a differential pair as it goes through confined areas among densely placed components.
- In the *Neck Gap* cell, you enter a value for the edge-to-edge spacing between a pair as it goes through tight areas full of component pins and vias. The smallest allowable gap consists of the *Neck Gap* minus the (-) *Tolerance*.



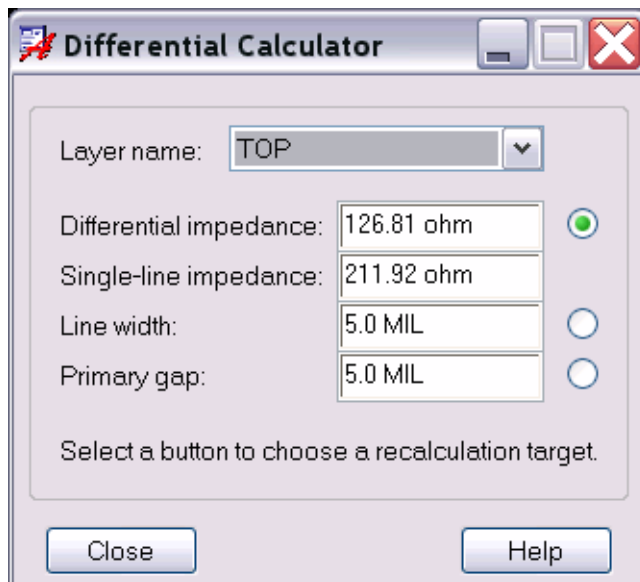
Tip

Neck Gap overrides any value in the *Primary Gap* when the differential pair's spacing collapses to or below the value of the *Min neck width* rule in an Electrical CSet assigned to the nets in the differential pair object. Therefore:

- Ensure that the neck gap does not go below any *Min* line spacing value you have set.
- You do not need to define a neck gap if you set (-) *Tolerance* with a value that accounts for the needed neck gap.

Using the Differential Calculator

Use the *Differential Calculator* to perform what-if scenarios to determine what combinations of line width and primary gap values can help you obtain a particular differential impedance. The calculator is available from the Electrical CSet- and Net worksheets.



To access the differential calculator, in the Primary Gap, Neck Gap, or +/- Tolerance cells, right-click and choose *Change* from the pop-up menu. Then click *Calculator*.

Allegro Constraint Manager User Guide

Working with Constraint Objects

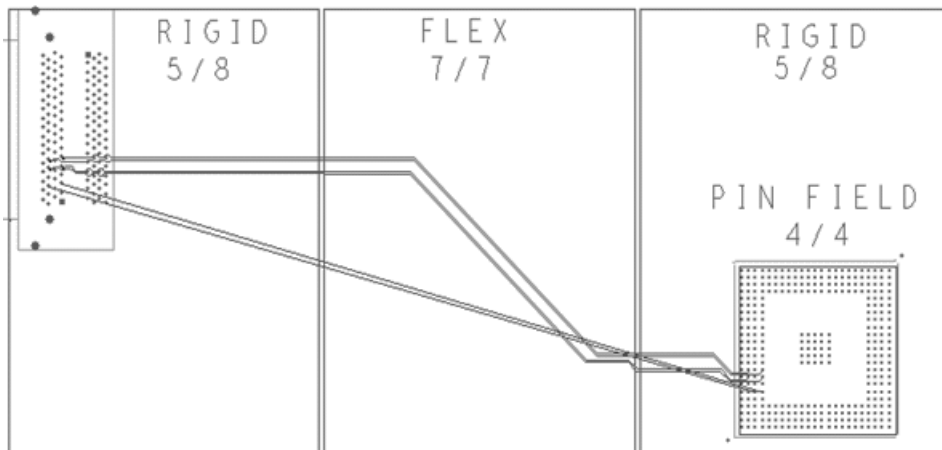
You can perform calculations only for edge-coupled differential pairs on a selected ETCH/ CONDUCTOR layer and account for material, thickness, electrical conductivity, dielectric constant, loss tangent, and shield.

<i>Layer name</i>	Indicates the ETCH/CONDUCTOR layer for which you are running the calculation. By default, the TOP/SURFACE layer appears.
<i>Differential impedance</i>	Specifies the impedance of the differential pair. Calculated for a pair of lines having the specified <i>Line width</i> and <i>Primary gap</i> on the layer.
<i>Single-line impedance</i>	Indicates the impedance of one line of etch on the selected layer. Changing this value automatically recalculates the <i>Line width</i> field. Each time the <i>Line width</i> field in this calculator changes—either when you directly modify it or when you select it to be recalculated—this value is recalculated, too.
<i>Line width</i>	Specifies the minimum width of each line of the differential pair. Changing this value automatically recalculates the <i>Single-line impedance</i> field. And changing the <i>Single-line impedance</i> field automatically recalculates this value.
<i>Primary gap</i>	Indicates the ideal edge-to-edge spacing between the pair that should be maintained for the entire length of the pair.

Differential Pairs by Constraint Region

A signal's impedance is affected by the dielectric constant of the material through which it passes. To maintain a constant impedance, line width and gap dimensions will differ for internal and outer layers. Because the electrical constraints apply globally, you must constrain the differential pairs in the *Physical* domain, which can vary by layer.

Rigid-flex designs require a change in material. Therefore, line widths and gaps may need to be narrower or wider, depending on the two materials. Because the physical editors allow only one material per layer, the only way to specify different constraints due to a change in material, is to use a constraint region to re-define a differential pair's line width and gap.

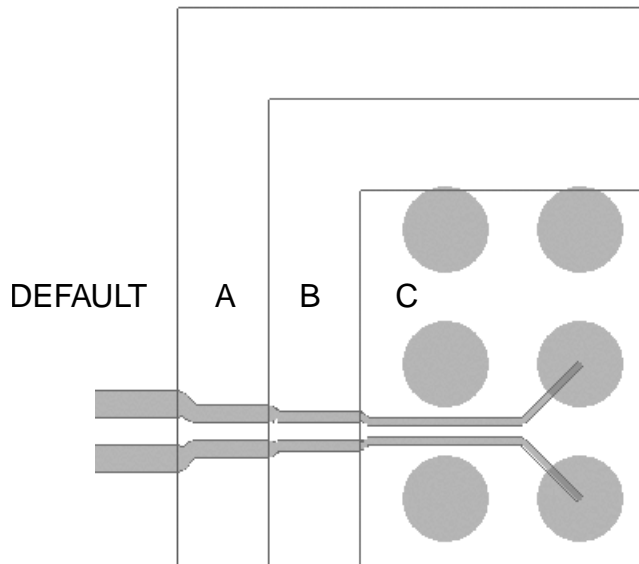


Note: To present the most flexible design options, additional differential pair physical constraints (Min Line Space and Tolerance +/-) appear in both domains (see [Differential Pair Constraints by Domain](#) on page 57). Although the *Electrical* domain lacks by-layer variance support, it does take precedence over a constraint of the same name in the *Physical* domain.

In [Figure 2-3](#) on page 64, differential pair line and gap constraints deviate from the values specified in the DEFAULT PCSet as the signal traverses the different materials and tight spacing confines upon entering a BGA.

These are represented by unique Region rules: A, B, and C.

Figure 2-3 Differential Pair Constraints by Region



To accommodate these differing line width and gap requirements, on the same or different layers, a good strategy is to:

1. In the *Physical* domain, create three *Region* constraint objects (in the *All Layers* worksheet in the *Regions* folder) for each region: A, B, and C.

Note: If the dimensions of the region's geometrical boundaries have not been defined, you must specify and name them in a physical editor before you can create a *Region* constraint object.

For information on how to create a *Region*, see the [Objects – Create – Region](#) command in the *Constraint Manager Reference*.

2. Create and define *PCSets* to reflect the new line width and gap constraints, as well as other unique differential pair parameters.

Note: Alternatively, use directly-set constraint values (overrides) instead of new *PCSets*.

For information on how to create a *PCSet*, see the [Objects – Create – Physical CSet](#) command in the *Constraint Manager Reference*.

3. In the *Physical* domain, associate each *Region* constraint object with the respective *PCSet* that contains the unique rules for that region.

For information on how to associate a *PCSet* with a *Differential Pair* constraint object, see the [Objects – Constraint Set References](#) command in the *Constraint Manager Reference*.

Differential Pair Rules

The following rules apply to differential pairs.

Model-defined Differential Pair	User-defined Differential Pair
<ul style="list-style-type: none">■ You create model-defined differential pairs in the PCB editor or APD using the <i>Analyze – SI/EMSim – Model</i> command.	<ul style="list-style-type: none">■ You create user-defined differential pairs in Constraint Manager using the <i>Objects – Create – Differential Pair</i> command or in the PCB editor or APD using the <i>Logic – Assign Differential Pair</i> command.
<ul style="list-style-type: none">■ Model-defined differential pairs are preferred in the high-speed flow because they uniquely characterize the differential pair members including pin parasitics, launch delays, logic thresholds, and buffer delays.	<ul style="list-style-type: none">■ User-defined differential pairs are not as accurate as model-defined differential pairs because they use default IBIS device values.
<ul style="list-style-type: none">■ A member in a model-defined differential pair cannot be a member of another differential pair object.	<ul style="list-style-type: none">■ A member in a user-defined differential pair cannot be a member of another differential pair object.
<ul style="list-style-type: none">■ You can extract both members of a model-defined differential pair object into SigXplorer with full coupling effects intact.	<ul style="list-style-type: none">■ You can extract only one member of a user-defined differential pair object into SigXplorer.
<ul style="list-style-type: none">■ Model-defined differential pairs take precedence. If you create a user-defined differential pair and later use the same members in a model-defined differential pair, the model-defined differential pair takes precedence.	<ul style="list-style-type: none">■ User-defined differential pairs lend precedence. If you create a user-defined differential pair and later use the same members in a model-defined differential pair, the model-defined differential pair takes precedence.

Allegro Constraint Manager User Guide

Working with Constraint Objects

Model-defined Differential Pair

- You cannot rename a model-defined differential pair object in Constraint Manager.
- You cannot change the membership of a model-defined differential pair in Constraint Manager. You must use the PCB editor, APD, or SigXplorer to do this by editing the device model.
- You cannot create a model-defined differential pair in the design entry editor when used in the Design Entry HDL – Constraint Manager high-speed flow.
- With a model-defined differential pair, any device that references the same device type will inherit the differential pair specified in that model. This reuse is analogous to creating an Electrical CSet and assigning it to many design objects in Constraint Manager.

User-defined Differential Pair

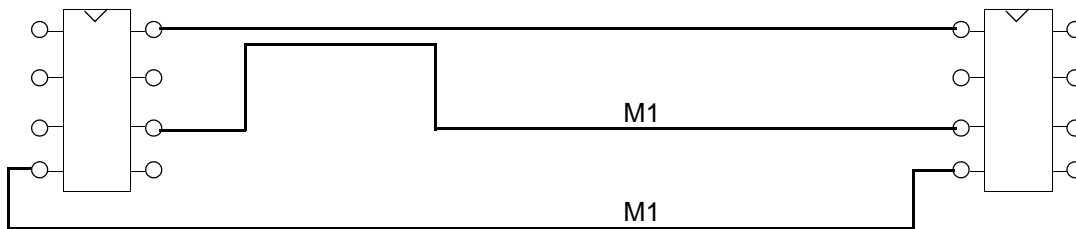
- You can rename a user-defined differential pair object in Constraint Manager.
- You can change the membership of a user-defined differential pair in Constraint Manager.
- You can create a user-defined differential pair in the design entry editor when used in the Design Entry HDL – Constraint Manager high-speed flow.

Analysis and DRC checking is not supported in this mode.
- You must create user-defined differential pairs individually in Constraint Manager. Although auto-setup simplifies the process, it is not the same as inheritance with model-defined differential pairs.

Match Groups

A *Match Group* is a collection of nets, Xnets, or pin pairs which must all match (in *delay* or *length*) or be relative to a specific *target* within the group.

In this illustration two of the three nets belong to a *Match Group*—M1. To maintain a match propagation delay, you must extend the middle net to match the delay of the bottom net.



See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Match Group](#) command in the *Constraint Manager Reference* for more information on the *Match Group* constraint object.

You can create *Match Groups* manually by adding properties directly to Xnets, or by using Constraint Manager to explicitly define pin pairs in an Electrical CSet. Regardless of the method that you choose, *Match Groups* are resolved down to specific pin pairs and are further defined by the following attributes: *Scope*, *Delta*, *Tolerance* and *Target*.

Generating pin pairs for the Match Group members

Match Groups can be populated directly with pin pairs manually or through explicit pin pairs defined in an Electrical CSet. If the *Match Group* was instead created by adding nets or Xnets directly, an additional attribute is needed to guide the resolution of pin pairs that will ultimately be compared in the *Match Group*. In the *Pin Pairs* column of the *Relative Propagation Delay* worksheet, you can guide the generation of pin pairs with the following options:

Choosing this option . . .	Generates pin pairs based on . . .
■ <i>All Drivers/All Receivers</i>	All combinations of drivers and receivers.
■ <i>Longest Driver/Receiver</i>	The longest driver-receiver pin pair. Uses the longest pin pair when there are not any drivers or receivers.
■ <i>Longest Pin Pair</i>	The longest pin pair.

When you analyze the *Match Group*, Constraint Manager automatically generates pin pairs based on the pinuse of the pins in the net or Xnet and the manhattan distances between the pins. Pin pairs appear hierarchically under the net or Xnet within the *Match Group*.



You create a *Match Group* in the *Objects* column of the *Relative Propagation Delay* worksheet by adding member nets and Xnets. If you cannot get the pin pairs that you want from the available settings, you should remove the net or Xnet from the *Match Group*, create the pin pairs that you want, and then add those pin pairs back to the *Match Group*.

Defining Match Group requirements

The two main requirements for a *Match Group* are *Delta* and *Tolerance*, which occupy the *Delta:Tolerance* column in the *Relative Propagation Delay* worksheet. Based on these settings, a Match Group is further classified as *Relative* or *Match*.

Match Delay

You specify only a *Tolerance* value; all pin pairs are compared against each other and must be within the specified *Tolerance*.

Relative Delay

You specify both *Delta* and *Tolerance* values, and select a *Target*. The *Target* may be implicit or explicit; each pin pair is compared to the *Target* pin pair by the specified *Delta* and within the specified *Tolerance*.

The following attributes characterize the requirements for a Match Group:

- **Tolerance** *Tolerance* is the allowable skew when matching member pin pairs. You specify *Tolerance* as either *length*, *delay*, or a *percentage*.

If you define only a *Tolerance* value for a member pin pair or for the *Match Group*, the member is compared to every other pin pair within the specified *Tolerance*.

If you define a *Delta* value for a member, the member is matched to the *Target*, plus or minus the *Delta* and within the specified *Tolerance*.

Allegro Constraint Manager User Guide

Working with Constraint Objects

- *Delta* *Delta* is the value added to, or subtracted from, the routed length of the *Target*. Constraint Manager uses the *Delta* to determine the required length of the pin pair before applying the *Tolerance*.

The *Delta* may be negative, in which case the value is subtracted from the routed length of the *Target*, or positive (or unsigned), in which case the value is added to the routed length of the *Target*. You can specify *Delta* as either *length* or *delay*.

If the *Delta* is unspecified, the pin pair must match all other pin pairs in the *Match Group* (within the *Tolerance*). A design rule violation results when the difference is greater than the *Tolerance*.

Note: If you specify zero for a *Delta* (which is different from leaving the field blank), the value is compared directly to the routed *length* or *delay* of the *Target*.

- *Target* *Target* is a pin pair that is referenced by all pin pairs in the *Match Group*. See [“Determining the target pin pair”](#) on page 71 for more information.

Target applies only when a *Delta* value exists; only one *Target* is specified for the entire *Match Group*.

When you explicitly set the *Target* pin pair (using the right mouse button), it remains the *Target*. When the *Target* is not user-defined, it may change from one net to another based on changes to the original *Target*, such as a change in its manhattan length.

If a pin pair is the *Target* in one *Match Group*, it does *not* have to be the *Target* in another *Match Group*, if that pin pair is a member of multiple *Match Groups*.

Examples

The following three scenarios cover *Match Groups* (*match* and *relative delay*) under varied conditions.



You can define a generic *Match Group* in an *Electrical CSet*. You can subsequently use the generic *Match Group* to define the net- or Xnet-specific groups when a net references the *Electrical CSet*.

Match Group (Scenario 1)

Pin Pair	Target	Delta	Tolerance	Comments
PP1	N/A	Unspecified	10 mils	In this scenario, you do not require a <i>Target</i> pin pair because the <i>Delta</i> is unspecified. Constraint Manager compares the routed length of each pin pair to every other pin pairs' routed length. If all pin pair differences are less than 10 mils, then the <i>Match Group</i> is within constraint limits. When you do not specify a <i>Delta</i> value, it is considered a match delay, not a relative match delay. Note: You can verify <i>Unrouted Length</i> . Choose <i>Analyze – Modes</i> and click the <i>Options</i> tab. Constraint Manager reports the worst-case result for each pin pair.
PP2	N/A	Unspecified	10 mils	
PP3	N/A	Unspecified	10 mils	

Allegro Constraint Manager User Guide

Working with Constraint Objects

Match Group (Scenario 2)

Pin Pair	Target	Delta	Tolerance	Comments
PP1		0 mils	10 mils	In this scenario, PP3 is the reference for the <i>Match Group</i> . You must route all pin pairs zero mils longer than PP3 with a <i>Tolerance</i> of 10 mils. If PP3 is 1000 mils, you must route PP1 and PP2 between 990 mils and 1010 mils. When you specify a <i>Delta</i> value, the Match Group is a <i>relative</i> match delay and each pin pair is compared to a single <i>Target</i> pin pair.
PP2		0 mils	10 mils	
PP3	X	0 mils	10 mils	

Match Group (Scenario 3)

Pin Pair	Target	Delta	Tolerance	Comments
PP1	X	100 mils	10 mils	In this scenario, PP1 is the reference for the <i>Match Group</i> . The routed length of all other pin pairs, plus (or minus) the <i>Delta</i> , must be within 10 mils of PP1. If PP1 is 1000 mils, PP2 must be no less than 1090 mils and no greater than 1110 mils; also, PP3 must be no less than 890 mils and no greater than 910 mils.
PP2		100 mils	10 mils	
PP3		- 100 mils	10 mils	

Determining the target pin pair

In a *Match Group*, Constraint Manager selects one of the pin pairs (or you specify one) as the *Target* and all of the other pin pairs are matched against this *Target* within the given *Delta* and *Tolerance*. When you manually specify a *Target*, there are no *Delta* and *Tolerance* values for that row.

If the TARGET keyword is displayed in the Delta/Tolerance cell for more than one member of a Match Group, the constraint information is invalid. You should *clear* or enter an explicit delta/tolerance value for each member that should not be considered the *Target*.

Allegro Constraint Manager User Guide

Working with Constraint Objects

Constraint Manager determines the *Target* pin pair as follows. The pin pair . . .

1. that you explicitly set (using the right mouse button in the *Delta:Tolerance* column).
2. with the smallest absolute *Delta* value (if all pin pairs have a *Delta* value).
3. with the longest manhattan length (if more than one pin pair has the same—smallest—*Delta* value).

Note: If none of the pin pairs has a *Delta* value, no *Target* is chosen; therefore, all pin pairs are compared to each other and the group is considered match delay, not relative match delay.

For example, if one pin pair has a *Delta* value of -300 and two pin pairs have a *Delta* value of zero, the pin pair with a zero *Delta* and the longest manhattan distance is chosen as the default *Target*. Although zero is larger than -300, the absolute value of -300 is 300, which is larger than zero. The two pin pairs with the zero value are compared by manhattan distance, and the larger is selected as the *Target*.

Note: When the *Delta* for all pin pairs is set to Null, the delay is considered match delay, not relative match delay.

Scope

Scope controls the validation of the Match Group (*Local* and *Global*), as well as the generation of Match Groups (*Bus* and *Class*). Once you define the members of a *Match Group*, you can specify the scope in the . . .

- *Scope* column of the *Relative Propagation Delay* worksheet (in the *Net* folder).
- *Scope* column of the *Relative Propagation Delay* worksheet (in the *Electrical Cset* folder).
- *Rel Prop Delay* tab in SigXplorer (choose *Set – Constraints*).

You can specify the following scope options:

- *Local* Validates only pin pairs within each net (or Xnet) against other pin pairs in the same net (or Xnet) for each member of the *Match Group*.
- *Global* Validates all pin pairs against all other pin pairs in the *Match Group*.

■ *Bus*

Bus scope is useful in situations where groups of signals are replicated and constraining them would require multiple *Electrical CSets* that only differ by *Match Group* name. If the nets that reference the *Electrical CSet* are not grouped into buses, a single *Match Group* with a *Global* scope is created.

Note: *Bus* scope can only be set in an *Electrical CSet* and only applies during the *Electrical CSet* mapping process.

If you subsequently apply the *Electrical CSet* to . . .

- a bus member, Constraint Manager maps the *Match Group* constraints to the bus member.
- the parent *Bus* object, Constraint Manager maps the *Match Group* constraints to all members of the *Bus*.
- a non-*Bus* member, Constraint Manager applies a *Global* scope and retains the original name of the *Match Group*.

When an *Electrical CSet* is applied to nets, all generated pin pairs are added to unique *Match Groups* based upon their *Bus* membership. The unique *Match Group* name is based upon the *Electrical CSet Match Group* name suffixed with the *Bus* name.

The generated *Match Group* is created with a scope of *Global* so that they are validated appropriately, as defined above in the *Global* setting. The *Bus* scope setting allows you to reference the same *Electrical CSet* from multiple *Buses*, resulting in unique *Match Groups* for the members/pin-pairs of each *Bus* (as required). This reduces the number of *ECSets* needed to constrain a design.

Example:

An *Electrical CSet* defines a *Match Group* (*mymatchgroup*), with a *Bus* scope, which is referenced to two buses: *BusA* and *BusB*.

After net-level *Electrical CSet* mapping, Constraint Manager creates the *Match Groups* with a *Global* scope

mymatchgroup_BusA

mymatchgroup_BusB

■ *Class*

Class scope is useful in situations where groups of signals are replicated and constraining them would require multiple *Electrical CSets* that only differ by *Match Group* name. If the nets that reference the *Electrical CSet* are not grouped into *Classes*, a single *Match Group* with a *Global* scope is created.

Note: You can set *Class* scope only in an *Electrical CSet* and it only applies during the *Electrical CSet* mapping process.

When an *Electrical CSet* is applied to nets, all generated pin pairs are added to unique *Match Groups* based upon their *Class* membership. The unique *Match Group* name is based upon the *Electrical CSet Match Group* name suffixed with the *Class* name.

The generated *Match Group* is created with a scope of *Global* so that they are validated appropriately, as defined above in the *Global* setting. The *Class* scope setting allows you to reference the same *Electrical CSet* from multiple *Classes*, resulting in unique *Match Groups* for the members/pin-pairs of each *Class* (as required). This reduces the number of ECsets needed to constrain a design.

Class scope is more flexible than *Bus* scope because a *Class* can include more disparate signals than a *Bus* can. A *Bus* is typically limited to vectored nets or nets that share a common topology; a *Class* often includes control signals, too.

You can also use *Class* and *Bus* scope to ensure that unique *Match Groups* are created in the top level of a hierarchical design in Allegro Design Entry HDL or Allegro System Architect, where replicated blocks exist.

Important

The default configuration for *Electrical Classes* is *Local*, which allows for electrical constraints and *Match Groups* to be created in a hierarchical block, and remain specific to that block at a higher level, and also in PCB Editor.

■ *Class*
(Continued)

Example:

An *Electrical CSet* defines the following *Match Group* (mymatchgroup) with a *Class Scope*, which is referenced to two *Net Classes*: ClassA and ClassB.

After net-level *Electrical CSet* mapping, Constraint Manager creates the *Match Groups* with a *Global* scope

```
mymatchgroup_ClassA
```

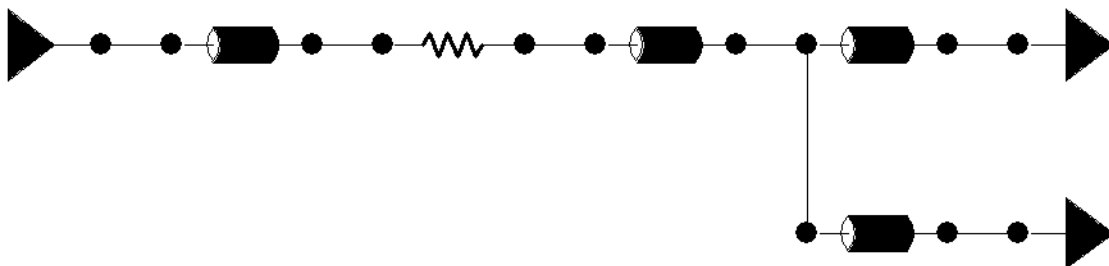
```
mymatchgroup_ClassB
```

Scope Scenarios

The following scenarios illustrate various scope settings and how they are used to create different *Match Groups*, depending on the desired constraints.

Assume you have two buses, each with three Xnets, and you want to match the two signal paths in each Xnet by matching the driver to the top receiver and the driver to the bottom receiver (two pin pairs per Xnet, six pin pairs in each bus). Depending on the constraint requirements, you can create several different *Match Groups*, each with a relative propagation delay constraint. In this scenario, we will be creating the constraints by applying an *Electrical CSet* with an associated topology.

Figure 2-4 Match Group Topology



Allegro Constraint Manager User Guide

Working with Constraint Objects

Scenario 1: Local Scope

Each driver to receiver path must match within each Xnet, and can match independent of the paths in any other Xnet.

You want to use a *Local* scope so that the topology in [Figure 2-4](#) on page 75 results in a single *Match Group* with eight pin pairs, two per Xnet.

Note: With a *Local* scope, Constraint Manager validates pin pairs independently in each Xnet.

When you apply the *Electrical CSet* to a *Bus*, the *Match Group* spans both buses, adding members of each bus to the *Match Group* (as shown in [Figure 2-5](#)).

Figure 2-5 Net-level Match Group with Local Scope

Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Scope	Delta:Toler ns
			Pin 1	Pin 2		
			mil	mil		
<input type="checkbox"/> System						
<input type="checkbox"/> my_db						
<input type="checkbox"/> MG1						
U17.7:U10.12 [ADDR5]					Local	:0.5 ns
U17.7:U11.12 [ADDR5]					Local	:0.5 ns
U17.9:U10.11 [ADDR4]					Local	:0.5 ns
U17.9:U11.11 [ADDR4]					Local	:0.5 ns
U18.7:U10.9 [ADDR3]					Local	:0.5 ns
U18.7:U11.9 [ADDR3]					Local	:0.5 ns
U18.9:U10.8 [ADDR2]					Local	:0.5 ns
U18.9:U11.8 [ADDR2]					Local	:0.5 ns
U19.7:U10.7 [ADDR1]					Local	:0.5 ns
U19.7:U11.7 [ADDR1]					Local	:0.5 ns
U19.9:U10.6 [ADDR0]					Local	:0.5 ns
U19.9:U11.6 [ADDR0]					Local	:0.5 ns
<input type="checkbox"/> BUS1	EC1					
<input type="checkbox"/> ADDR0	EC1					
<input type="checkbox"/> ADDR1	EC1					
<input type="checkbox"/> ADDR2	EC1					
<input type="checkbox"/> BUS2	EC1					
<input type="checkbox"/> ADDR3	EC1					
<input type="checkbox"/> ADDR4	EC1					
<input type="checkbox"/> ADDR5	EC1					
W_R						

Note: With a *Local* scope, all pin pairs occupy a single *Match Group*; Constraint Manager compares pin pairs to only pin pairs in the same Xnet. In [Figure 2-5](#), Constraint Manager compares pin pair U17.7:U10.12 only with pin pair U17.7:U11.12 because they come from the same Xnet, ADDR5.

Allegro Constraint Manager User Guide

Working with Constraint Objects

Scenario 2: Global Scope

Each driver to receiver path must match every other driver to receiver path, regardless of the path of the Xnet or *Bus*. This scenario is common to the majority of *Match Groups*, especially where it is only one pin pair in a particular Xnet.

Use a *Global* scope so the topology results in a single *Match Group* with eight pin pairs, two per Xnet, as shown in [Figure 2-4](#) on page 75.

Figure 2-6 Net-level Match Group with Global Scope

Objects	Reference d Electrical CSet	Pin Pairs	Pin Delay		Scope	Delta:Tol ns
			Pin 1	Pin 2		
			mil	mil		
<input type="checkbox"/> System						
<input type="checkbox"/> my_db						
<input type="checkbox"/> MG1						
U17.7:U10.12 [ADDR5]					Global	:0.5 ns
U17.7:U11.12 [ADDR5]					Global	:0.5 ns
U17.9:U10.11 [ADDR4]					Global	:0.5 ns
U17.9:U11.11 [ADDR4]					Global	:0.5 ns
U18.7:U10.9 [ADDR3]					Global	:0.5 ns
U18.7:U11.9 [ADDR3]					Global	:0.5 ns
U18.9:U10.8 [ADDR2]					Global	:0.5 ns
U18.9:U11.8 [ADDR2]					Global	:0.5 ns
U19.7:U10.7 [ADDR1]					Global	:0.5 ns
U19.7:U11.7 [ADDR1]					Global	:0.5 ns
U19.9:U10.6 [ADDR0]					Global	:0.5 ns
U19.9:U11.6 [ADDR0]					Global	:0.5 ns
<input type="checkbox"/> BUS1	EC1					
<input type="checkbox"/> ADDR0	EC1					
<input type="checkbox"/> ADDR1	EC1					
<input type="checkbox"/> ADDR2	EC1					
<input type="checkbox"/> BUS2	EC1					
<input type="checkbox"/> ADDR3	EC1					
<input type="checkbox"/> ADDR4	EC1					
<input type="checkbox"/> ADDR5	EC1					
W_R						

Note: With a *Global* scope, Constraint Manager validates pin pairs collectively in the (single) *Match Group*, as shown in [Figure 2-6](#). In this scenario, Constraint Manager compares pin pair U17.7:U10.12 to all other pin pairs in the *Match Group*.

Scenario 3: Bus Scope

Each driver to receiver path must match every other driver to receiver path, but only within the same bus. In this example, we have the Xnets grouped into two buses; however, your interface may have the same bus structure replicated many times.

One option is to create a separate *Electrical CSet* for each bus using a *Global* scope; the resulting *Electrical CSet*s would be identical except for the name of the *Match Group*. With a *Bus* scope, the topology in [Figure 2-4](#) on page 75 results in two *Match Groups* (derived from a common *Match Group*), with six pin pairs each.

Note: As with a *Global* scope, Constraint Manager validates pin pairs collectively between both *Buses* when the **Electrical CSet** is applied to either *Bus*.

You must specify a *Bus* scope, for a *Match Group*, within an *Electrical CSet* (as shown in [Figure 2-7](#)) in either the *Electrical CSet* folder or in SigXplorer. You can then apply the *Electrical CSet* to a *Bus* member in the *Relative Propagation Delay* worksheet. Although you defined a *Bus* scope at the *Electrical CSet* level, when the *Electrical CSet* is applied to a *Bus* member at the net level, the *Scope* column indicates *Global*.

Figure 2-7 Electrical CSet-level Match Group with Bus Scope

Objects	Pin Pairs	Scope	Delta:Tolerance
			ns
<input type="checkbox"/> System			
<input type="checkbox"/> my_db			
<input type="checkbox"/> EC1			
<input type="checkbox"/> MG1			
U19.9:U10.6		Bus	:0.5 ns
U19.9:U11.6		Bus	:0.5 ns

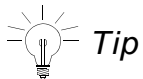
When you apply the *Electrical CSet* to a *Bus*, each *Bus* inherits constraints from the same *Match Group*. In this way, the same *Match Group* can span many *Buses*, yet you can modify the *Match Group* on a per bus basis. Constraint Manager prepends the common *Match Group* name to the independent *Bus* name (as shown in [Figure 2-8](#) on page 79), in effect, uniquely qualifying the *Match Group*. In this way, the same *Match Group* (and the same *Electrical CSet*) is common to many buses, yet you can modify each independently.

Allegro Constraint Manager User Guide

Working with Constraint Objects

Figure 2-8 Net-level Match Group with Bus Scope

Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Scope	Delta:Tolerance ns
			Pin 1	Pin 2		
			mil	mil		
<input type="checkbox"/> System						
<input type="checkbox"/> my_db						
<input type="checkbox"/> MG1_BUS1						
U18.9:U10.8 [ADDR2]					Global	:0.5 ns
U18.9:U11.8 [ADDR2]					Global	:0.5 ns
U19.7:U10.7 [ADDR1]					Global	:0.5 ns
U19.7:U11.7 [ADDR1]					Global	:0.5 ns
U19.9:U10.6 [ADDR0]					Global	:0.5 ns
U19.9:U11.6 [ADDR0]					Global	:0.5 ns
<input type="checkbox"/> MG1_BUS2						
U17.7:U10.12 [ADDR5]					Global	:0.5 ns
U17.7:U11.12 [ADDR5]					Global	:0.5 ns
U17.9:U10.11 [ADDR4]					Global	:0.5 ns
U17.9:U11.11 [ADDR4]					Global	:0.5 ns
U18.7:U10.9 [ADDR3]					Global	:0.5 ns
U18.7:U11.9 [ADDR3]					Global	:0.5 ns
<input type="checkbox"/> BUS1	EC1					
<input checked="" type="checkbox"/> ADDR0	EC1					
<input checked="" type="checkbox"/> ADDR1	EC1					
<input checked="" type="checkbox"/> ADDR2	EC1					
<input type="checkbox"/> BUS2	EC1					
<input checked="" type="checkbox"/> ADDR3	EC1					
<input checked="" type="checkbox"/> ADDR4	EC1					
<input checked="" type="checkbox"/> ADDR5	EC1					
W_R						



Tip

You can also influence the mapping process by specifying *optional pins* in the topology. See [Optional Pins](#) in the *SigXplorer Command Reference* for more information.

Allegro Constraint Manager User Guide

Working with Constraint Objects

Scenario 4: Class Scope

In this scenario, an *Electrical CSet* is defined in the *Electrical Constraint Set* folder and it contains a *Match Group* with a *Scope of Class*.

Type	Objects	Pin Pairs	Scope	Delta:Tolerance
				mil
Dsn	[-] class_scope2			
ECS	[-] DDR2_DQ0			
ECSM	MATCH_POINT	Longest Pin Pair	Class	0 ns:5 %

ECSet
Match Group

In the *Relative Propagation Delay* worksheet of the *Routing* workbook in the *Net* folder, the DDR2_DQ bus contains 16 data signals and is assigned the DDR2_DQ0 *Electrical CSet*.

Type	Objects	Referenced Electrical CSet
Bus	[-] DDR2_DQ	DDR2_DQ0
Net	DDR2_DQ0	DDR2_DQ0
Net	DDR2_DQ1	DDR2_DQ0
Net	DDR2_DQ2	DDR2_DQ0
Net	DDR2_DQ3	DDR2_DQ0
Net	DDR2_DQ4	DDR2_DQ0
Net	DDR2_DQ5	DDR2_DQ0
Net	DDR2_DQ6	DDR2_DQ0
Net	DDR2_DQ7	DDR2_DQ0
Net	DDR2_DQ8	DDR2_DQ0
Net	DDR2_DQ9	DDR2_DQ0
Net	DDR2_DQ10	DDR2_DQ0
Net	DDR2_DQ11	DDR2_DQ0
Net	DDR2_DQ12	DDR2_DQ0
Net	DDR2_DQ13	DDR2_DQ0
Net	DDR2_DQ14	DDR2_DQ0
Net	DDR2_DQ15	DDR2_DQ0

Allegro Constraint Manager User Guide

Working with Constraint Objects

Next, the *Bus* is divided equally into separate *Net Classes* (DDR2_0 and DDR2_1). Control signals are also added to each *Net Class*.

Type	Objects	Referenced Electrical CSet
NCIs	<input type="checkbox"/> DDR0_LANE0	
Net	DDR2_DM0	CONTROL_SIGNAL
Net	DDR2_DQS0	CONTROL_SIGNAL
Net	DDR2_DQ0	DDR2_DQ0
Net	DDR2_DQ1	DDR2_DQ0
Net	DDR2_DQ2	DDR2_DQ0
Net	DDR2_DQ3	DDR2_DQ0
Net	DDR2_DQ4	DDR2_DQ0
Net	DDR2_DQ5	DDR2_DQ0
Net	DDR2_DQ6	DDR2_DQ0
Net	DDR2_DQ7	DDR2_DQ0
NCIs	<input type="checkbox"/> DDR2_LANE1	
Net	DDR2_DM1	CONTROL_SIGNAL
Net	DDR2_DQS1	DDR2_DQ0
Net	DDR2_DQ8	DDR2_DQ0
Net	DDR2_DQ9	DDR2_DQ0
Net	DDR2_DQ10	DDR2_DQ0
Net	DDR2_DQ11	DDR2_DQ0
Net	DDR2_DQ12	DDR2_DQ0
Net	DDR2_DQ13	DDR2_DQ0
Net	DDR2_DQ14	DDR2_DQ0
Net	DDR2_DQ15	DDR2_DQ0

Next, you can see the *Match Groups*. When merged to the top-level of a hierarchical block, the *Match Groups* are qualified with unique names, consisting of the concatenation of the *Electrical CSet Match Group* name and the *Class* name.

Type	Objects	Referenced Electrical CSet	Pin Pairs	Scope
Dsn	<input type="checkbox"/> class_scope2			
MGrp	<input type="checkbox"/> MATCH_POINT_DDR0_LANE0			
Net	DDR2_DQ0	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ1	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ2	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ3	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ4	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ5	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ6	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ7	DDR2_DQ0	Longest Pin Pair	Global
MGrp	<input type="checkbox"/> MATCH_POINT_DDR2_LANE1			
Net	DDR2_DQS1	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ8	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ9	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ10	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ11	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ12	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ13	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ14	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ15	DDR2_DQ0	Longest Pin Pair	Global

Match Group Rules

The following rules apply to *Match Groups*:

- You *must* specify a *Match Group* in only the *Relative Propagation Delay* worksheet of the *Routing* workbook.
 - You can set *Match Group* constraints for the entire group and override individual members of the group as desired, offering differing levels of *delta* and *tolerance*.
 - You specify *Match Group* delays at the *design-* or the *system-*level.
 - You can include a *Match Group* member in multiple *Match Groups*.
 - A match delay constraint from a pre-14.0 board database is upreved with a *delta* value of zero. This implies that all group members will match a specified target pin pair.
- Note:** Constraint Manager, when launched from Allegro® PCB Series L, supports *Match Groups* only on net-related objects, not on *Electrical CSets*.
- If you are specifying a *Bus-* or *Class-*scope, you must create the *Match Group* in a *CSet* in the *Electrical CSet* folder.

See “[Match Groups](#)” on page 67 for more information on *Match Group* objects.

Multi-group Membership

You can include a member of a *Match Group* in another *Match Group*. In this way, you can constrain the same member differently in each *Match Group*. In Figure 2-9, members *CAS0L* and *CAS1L* are members of two *Match Groups*. Furthermore, *CAS1L* is constrained differently in each *Match Group*.

Figure 2-9 Multi-group Membership

Objects	Scope	Rel.
		Delta:Tolerance
ns		
System		
lesson5b		
MATCH_GROUP_B	Global	100 ns:5 %
BHEL	Global	100 ns:5 %
BLEL	Global	100 ns:5 %
CAS0L	Global	100 ns:5 %
CAS1L	Global	100 ns:5 %
MATCH_GROUP_C	Global	100 ns:5 %
CAS0L	Global	100 ns:5 %
CAS1L	Global	150 ns:10 %

Allegro Constraint Manager User Guide

Working with Constraint Objects

Note: Refer to the *Objects – Create – Match Group* and *Objects – Membership – Match Group* commands in the *Constraint Manager Reference* for information on including nets, Xnets, and pin pairs in multiple *Match Groups*.

Analyzed values appear in the *Actual* and *Margin* columns of a net, Xnet, or pin pair. If you include the net, Xnet, or pin pair in a *Match Group*, the analyzed values appear on that member in each *Match Group* to which it belongs, and not on the object's row outside of the *Match Groups*. To obtain the value of a net, Xnet, or pin pair that is a *Match Group* member, you must expand that *Match Group*.

With multi-group membership, use the following techniques to locate *Match Group* member objects:

■ *Status Bar*

Highlight a net, Xnet, or pin pair in the Object's column and observe the *status bar*. Constraint Manager reports all parent objects in which the member belongs.

PinPair Result: J1A.1:J1B.1 [Match Group MG1, Net BANKJ1_10F50]

In the example above, the *status bar* (located at the lower-left corner of Constraint Manager) identifies the pin pair, its parent net, and the parent *Match Groups*.

■ *Find*

Use the *Find* command (choose *Edit – Find*) to locate the selected object. Use the *Find Next* command (choose *Edit – Find Next*) to find the next occurrence of the object in your design. One at a time, Constraint Manager locates the next occurrence of the selected object in all parent objects to which it belongs, including multiple *Match Groups*.

■ *Select*

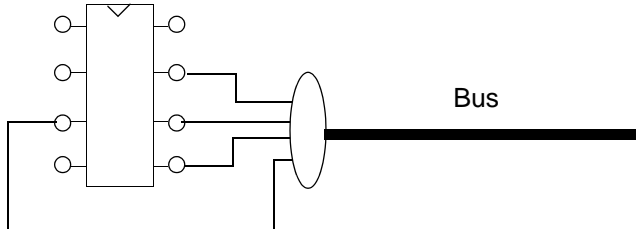
Use the *Select* command to highlight all occurrences of the selected object, including instances of the object in multiple *Match Groups*.

■ *Filter*

Use the *Filter* command (choose *Objects – Filter*) to locate only the occurrences of a selected net or Xnet, including instances of the object in multiple *Match Groups*. Where *Select* locates an instance of an object in multiple groups, *Filter* lets you filter out members of the *Match Group*.

Buses

A *bus* represents a named collection of differential-pairs, Xnets, or Nets.



See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Bus](#) command in the *Constraint Manager Reference* for more information on the *Bus* constraint object.

You can use a bus to group functionally similar nets, Xnets, and differential pairs. Constraints captured on a bus are inherited by all members of the bus.

You can create an Electrical CSet based on the characteristics of a bus. You could then use SigXplorer to define pin scheduling of bus members and to augment constraint information.

When you associate the Electrical CSet with a bus, all members (bits) of the bus inherit the constraints defined in the Electrical CSet. In [Figure 2-10](#), the VIDEO_DATA_CSET Electrical Cset is referenced to BUS (NETS 3, 4, 5). Each member of the bus inherits the properties defined in this Electrical CSet.

Figure 2-10 Bus Inheritance

Objects	REFERenced Electrical CSet	Pin
[-] A2B		
NET5	PCI_CSET	
[-] A1 (A)		
[+] B1		
[+] BUS (NETS 3,4,5)	VIDEO_DATA_CSET	
[+] B1 (B)		

Collapsed Bus

Objects	REFERenced Electrical CSet	Pin
[-] A2B		
NET5	PCI_CSET	
[-] A1 (A)		
[+] B1		
[-] BUS (NETS 3,4,5)	VIDEO_DATA_CSET	
NET3	VIDEO_DATA_CSET	
NET4	VIDEO_DATA_CSET	
NET5	VIDEO_DATA_CSET	
[+] B1 (B)		

Expanded Bus

See [Chapter 3, “Working With Reusable Constraint Objects — CSets”](#) for information about creating CSets and associating them with objects

See [Chapter 6, “Using Constraint Manager with Other Tools Across the Allegro Platform”](#) for information on using SigXplorer.

Bus Rules

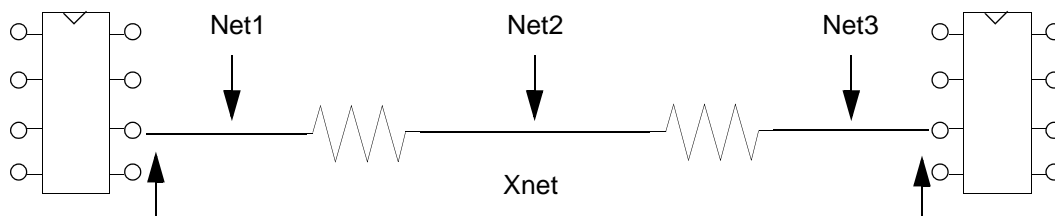
The following rules apply to creating a bus

- You can create a bus in all net worksheets
- When used in conjunction with Allegro[®] Design Entry HDL, Constraint Manager cannot create a bus. A bus is only realized with the appropriate property in the schematic
- A bus must be at the design-level (not the system-level).

Nets and Xnets

A *net* represents an electrical connection from one pin to another pin (or pins) on the same device or on a different device.

See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* for more information on Xnets and Nets.



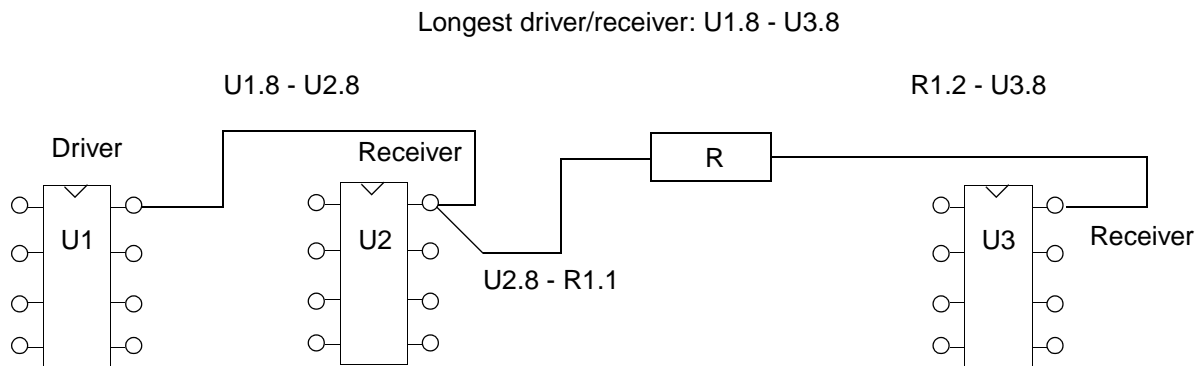
If the path of a net traverses a passive, discrete device (resistor, inductor or capacitor), then each net segment is represented by an individual net entity in the board database. Constraint Manager, however, interprets these net segments as a contiguous extended net, or an *Xnet*. An Xnet can also traverse connectors and cables in a multi-board configuration.

You can associate a net or Xnet with an CSet. See [Chapter 3, “Working With Reusable Constraint Objects — CSets”](#) for information about creating CSets and associating them with objects.

Pin Pairs

A *pin pair* represents a pair of logically connected pins, often a driver-receiver connection. Pin Pairs may not be directly connected but they must exist on the same net or Xnet.

See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Pin Pairs](#) command in the *Constraint Manager Reference* for more information on Pin Pairs.

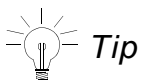


You use pin pairs to capture specific pin-to-pin constraints for a net or an Xnet. You can also use pin pairs to capture generic pin-to-pin constraints for CSets. Generic pin pairs are used to automatically define net- or Xnet-specific pin pairs when the CSet is referenced.

You may specify pin pairs explicitly (for example, U1 . 8 U3 . 8), or they can be derived based on the following criteria:

- longest pin pair
- longest driver-receiver pair
- all driver-receiver pairs

Note: Physical and Spacing pin pairs cannot be derived; you must create them explicitly.



Tip

When you import a topology file from SigXplorer and apply the *Electrical CSet* to a net object, Constraint Manager creates pin pairs on the net or Xnet based on those defined in the original topology file.

Once established, you associate a pin pair with a CSet. See [Chapter 3, “Working With Reusable Constraint Objects — CSets”](#) for information about creating CSets and associating them with objects.

Allegro Constraint Manager User Guide

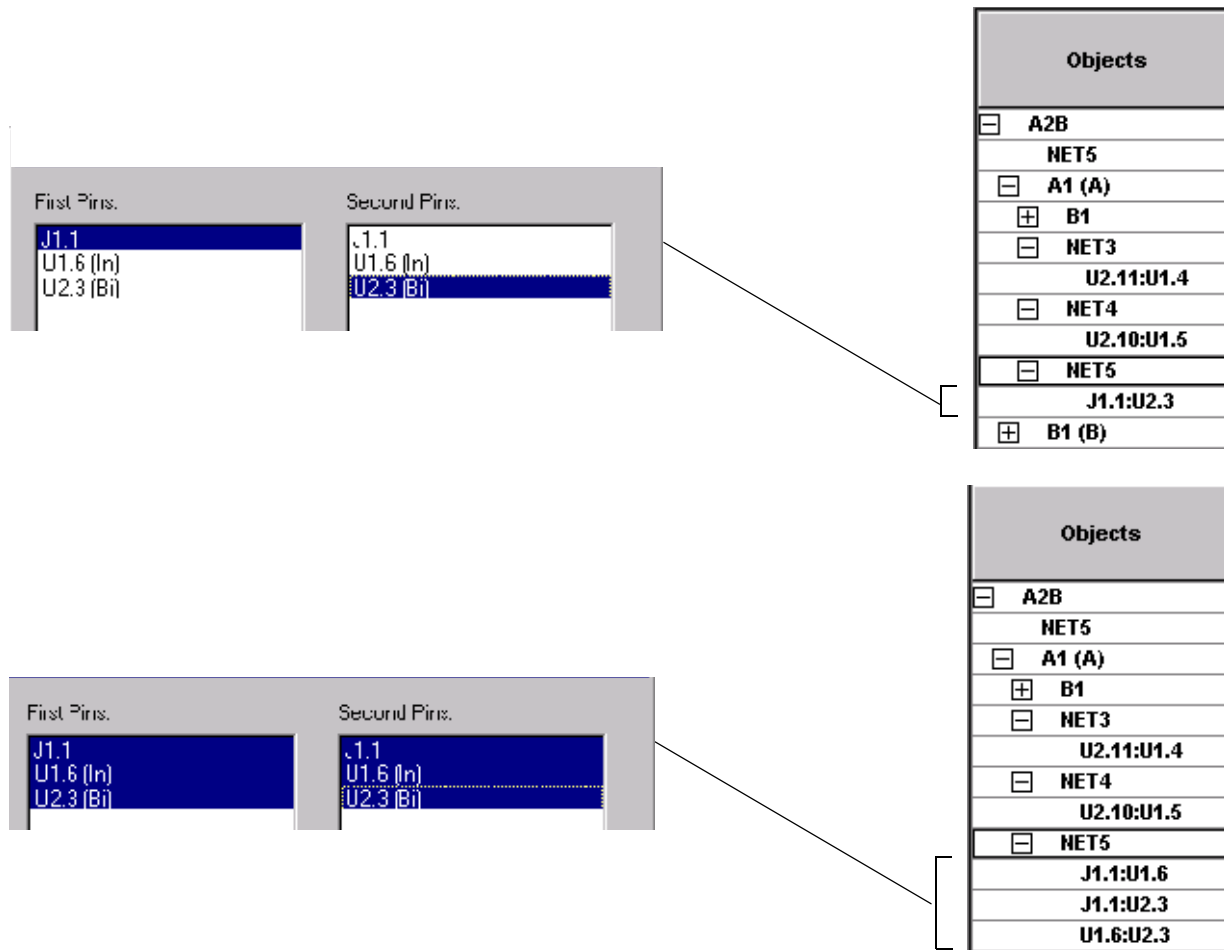
Working with Constraint Objects

Note: The Allegro® Design Entry HDL database does not directly support CSet, Match Group, differential pair, pin pair, and Xnet objects. Constraint Manager does, however, update and validate constraints on these objects in the schematic.

Examples of pin pairs

Figure 2-11 shows two examples of creating pin pairs. In the top example, a single, one-to-one pin map is specified on Net 5 (J1, Pin 1 to U2, Pin 3). In the bottom example, all permutations of pin mappings is realized. If all *first* pins and all *second* pins are selected, Constraint Manager maps all source pins to all target pins while excluding each pin combination that appears in both lists.

Figure 2-11 Example Pin Pairs



 **Important**

Use `Shift-click` to select a range of pins in the pin list or `Control-click` to selectively highlight pins.

Pin Pair Rules

The following rules apply to creating pin pairs

Physical, Spacing, and Electrical domains

Pins must exist in the object from which you create the pin pair.

Electrical domain

- Pin Pairs can only be defined in the following worksheets:

Electrical Workbook

All Constraints

Timing

Routing

Electrical Worksheet

- *Signal Integrity/Timing/Routing*

- *Switch/Settle Delays*

- *Setup/Hold*

- *Impedance*


- *Min/Max Propagation Delay*

- *Relative Propagation Delay*

- Objects in the *All Constraints* and *Timing* worksheets must have a driver and a receiver as pin pairs.
- Pin Pair length is the length of the etch path between the two pins, if the pins are routed. If not routed, the total manhattan distance of the ratsnest lines connecting the pins is used.
- Constraint Manager determines longest/shortest pin pair length based on drivers and receivers. If there are not any drivers or receivers, all pins in an Xnet are considered.
- For a relative propagation delay constraint, only the longest pin pair is determined.

Ratsnest Bundle

A *Ratsnest Bundle* constraint object lets you organize and manipulate ratsnest connections as a named group. This makes it more efficient to work with the *Interconnect Flooplanner* and the *Global Route Environment*. You edit ratsnest bundle attributes in the *Ratsnest Bundle Properties* worksheet in the *Properties* domain.

RBnd	 BNDL_65 (24)
RPPr	U1000.AA1:U999.T4 [26]
RPPr	U1000.AA3:U999.AA5 [25]
RPPr	U1000.AA4:U999.AA3 [6]

Note: Constraint Manager recognizes ratsnest bundles, and their attributes, as defined in a layout editor.

The following applies to a ratsnest bundle:

- Bundles display *RBnd* in the *Objects* column.
- Bundle pin pair members display *RPPr* in the *Objects* column.
- You can pre-populate a ratsnest bundle by selecting pin pairs before creating the bundle, or you can add members later.
- You can crossprobe to highlight a ratsnest bundle in a physical editor.

For more information about ratsnest bundles, see

- [Working with Bundles](#) in the *Allegro PCB Editor User Guide: Working with Global Route Environment* for overview information.
- the [B commands](#) bundle operations in the *Allegro PCB and Package Physical Layout Command Reference* for procedural information.
- the [Objects – Create – Ratsnest Bundle](#) command in the *Constraint Manager Reference* for procedural information.
- the [Objects – Membership – Ratsnest Bundle](#) command in the *Constraint Manager Reference* for procedural information.

Region

A *Region* constraint object adds or modifies constraints on all nets that cross the boundaries of the region's shape.

Type	Objects
Dsn	unnamed
Rgn	BGA_1MM

See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Region](#) command in the *Constraint Manager Reference* for more information on the *Region* constraint object.

Region Rules

The following rules apply to creating a *Region*. You

- specify a *Region* in the *Region* workbook in either the *Physical* or *Spacing* domain
- delimit a *Region* with a geometric shape, or a group of shapes, that you draw on a subclass layer in PCB Editor
- constrain a *Region* with a *Physical-* or *Spacing-CSet*
- constrain a *Region* directly with explicit constraint values (though Cadence recommends using a *CSet*)

Note: *Regions* are not supported in Design Entry HDL or Allegro System Architect.

Region Class

A *Region Class* constraint object lets you constrain the members of a *Net Class*—within a *Region*—differently than the original constraints on that *Region*.

Type	Objects
Dsn	<input type="checkbox"/> unnamed
Rgn	<input type="checkbox"/> BGA_1MM
RClS	CLS1

Allowable members of a *Region Class* include *Net Classes*.

See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Region Class](#) command in the *Constraint Manager Reference* for more information on the *Net Class-Class* constraint object.

Region Class Rules

The following rules apply to creating a *Region Class*. You can

- constrain a *Region Class* with a *CSet*
- override individual members of a *Region Class*
- constrain a *Region Class* directly (though we recommend using a *CSet*)
- create a *Region Class* only in the *Physical* or *Spacing* domains

Note: *Region Class* are not supported in Design Entry HDL or Allegro System Architect.

Region Class-Class

A *Region Class-Class* constraint object lets you specify the minimum spacing between members of *Net Classes* within a *Region*.

Type	Objects
Dsn	<input type="checkbox"/> unnamed
Rgn	BGA_1MM
Rgn	<input type="checkbox"/> RGN1
RCC	CLS1:CLS2
Rgn	RGN2

See [Constraint Architecture](#) in the *Allegro Platform Constraints Reference* and the [Objects – Create – Region Class-Class](#) command in the *Constraint Manager Reference* for more information on the *Region Class-Class* constraint object.

Region Class-Class Rules

The following rules apply to creating a *Region Class-Class*. You can

- constrain a *Region Class-Class* with a *CSet*
- override individual members of a *Region Class-Class*
- constrain a *Region Class-Class* directly (though we recommend using a *CSet*)
- create a *Region Class-Class* only in the *Spacing* domain

Note: *Region Class-Classes* are not supported in the *Same Net Spacing* domain, or in Design Entry HDL or Allegro System Architect.

Working With Reusable Constraint Objects — CSets

Topics in this chapter include

- [Reusable Constraints](#) on page 94
- [Methods of Constraining Nets](#) on page 95

Reusable Constraints

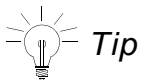
This chapter presents information on how to use reusable constraint objects in Constraint Manager. See Chapter 2, “[Working with Constraint Objects](#)” on page 49 for information on hierarchical constraint objects.

Constraining Objects

You can constrain an object by referencing a CSet or setting a constraint value directly on the object. An object that is not constrained will inherit constraint values through the precedence rules of the [Constraint Object Hierarchy](#) on page 51.

Constraint Sets (CSets)

The *Electrical*, *Physical*, *Spacing* and *Same Net Spacing* domains support Constraint Sets (CSets). A CSet is a named, reusable collection of constraint values. CSets are not supported in the *Design* domain.

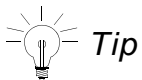


Tip

You define generic rules in the *Constraint Set* object folder. These generic rules can subsequently be applied to objects in the *Net*, *Class*, and *Region* folders.

As design requirements change, you can

- edit the CSet constraints. All objects that reference the CSet will automatically inherit these changes.
- assign a different CSet, one that reflects a different rule-set, to the object.
- specify override properties on individual objects. Cells with overrides are colored blue.



Tip

CSets can be referenced by any number of objects (*Bus*, *Differential Pair*, *Xnet*, *Net*, *Net Class*, *Net Class-Class*, *Region*, *Region Class*, *Region Class-Class*) but an object may reference only one CSet per domain.

Physical, Spacing, and Same Net Spacing CSets

A *Physical CSet* consists of one value per layer for each physical constraint. A *Spacing CSet* consists of one value per layer for each spacing constraint. *Spacing CSets* are further classified into net-to-net and same net domains. In all designs, Constraint Manager provides one *Physical*, one *Spacing*, and one *Same Net Spacing CSet*, named *DEFAULT*, which you cannot delete or rename; however, you can modify the *DEFAULT* CSet constraints to suit your design requirements.

Electrical CSets

With an *Electrical CSet*, you define the constraints in the set. There is no pre-defined configuration, nor any pre-defined values. You can delete *Electrical CSets*.

Methods of Constraining Nets

This section covers different methods of constraining nets in your design, including:

- Inheritance
- Overrides
- Container objects
- CSet references

Inheritance

Each *Net* depicted below inherits the 5-mil *Min Line Width* from the *DEFAULT* Physical CSet.

1 2	Type	Objects	Referenced Physical CSet	Line Width	
				Min mil	Max mil
102	Net	USE3	DEFAULT	5.00	0.00
103	Net	USE4	DEFAULT	5.00	0.00
104	Net	USE5	DEFAULT	5.00	0.00
105	Net	USE6	DEFAULT	5.00	0.00
106	Net	VCC	DEFAULT	5.00	0.00
107	Net	VSR_RTN	DEFAULT	5.00	0.00
108	Net	24V	DEFAULT	5.00	0.00
109	Net	48V	DEFAULT	5.00	0.00

Allegro Constraint Manager User Guide

Working With Reusable Constraint Objects — CSets

Overrides

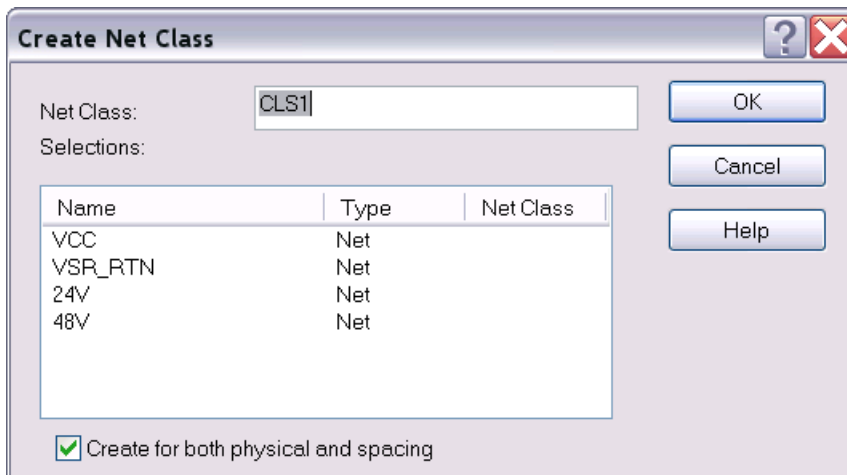
We need to increase the line width of the voltage rails (Rows 106 - 109). As these are contiguous cells, simply dragging through them and specifying 7 mils is the most-direct method of constraining these nets

1	Type	Objects	Referenced Physical CSet	Line Width	
				Min	Max
2				mil	mil
102	Net	USE3	DEFAULT	5.00	0.00
103	Net	USE4	DEFAULT	5.00	0.00
104	Net	USE5	DEFAULT	5.00	0.00
105	Net	USE6	DEFAULT	5.00	0.00
106	Net	VCC	DEFAULT	7.00	0.00
107	Net	VSR_RTN	DEFAULT	7.00	0.00
108	Net	24V	DEFAULT	7.00	0.00
109	Net	48V	DEFAULT	7.00	0.00

These directly-set constraint values are called *overrides* and appear in a blue tint. The advantage of constraining a container object is that it is quick and direct and it follows a constraint precedence; the disadvantage is that members of the container are fixed (though membership can be redefined) so constraints are not transferable to other container objects.

Container Objects

If we create a container object, such as a *Net Class*, we can constrain the container directly instead of constraining member nets individually. If we change the constraint value on the the container, members of the container automatically inherit the change in constraint value.



1	Type	Objects	Referenced Physical CSet	Line Width	
				Min	Max
2				mil	mil
3	Dsn	unnamed	DEFAULT	5.00	0.00
4	NCIs	CLS1	DEFAULT	7.00	0.00

Allegro Constraint Manager User Guide

Working With Reusable Constraint Objects — CSets

Constraint Manager presents *Net Classes* before *Nets*. The Nets from Rows 106 - 109 now appear, in collapsed form, in the *Net Class* container on Row 4.

If we expand the *Net Class*, we can see that the members inherit the new line width value of 7 mils.

1	Type	Objects	Referenced Physical CSet	Line Width	
				Min	Max
2				mil	mil
3	Dsn	<input type="checkbox"/> unnamed	DEFAULT	5.00	0.00
4	NCLs	<input type="checkbox"/> CLS1	DEFAULT	7.00	0.00
5	Net	VCC	DEFAULT	7.00	0.00
6	Net	VSR_RTN	DEFAULT	7.00	0.00
7	Net	24V	DEFAULT	7.00	0.00
8	Net	48V	DEFAULT	7.00	0.00
9	Bus	<input checked="" type="checkbox"/> ADDRESS_BUS	DEFAULT	5.00	0.00
10	DPr	<input checked="" type="checkbox"/> DP_MEM_CLOC	DEFAULT	5.00	0.00



Tip

Use a net container object to group and constrain a small, focused collection of similar nets.

Referencing a CSet

Let's create a Physical CSet and define a 7-mil *Minimum Line Width* constraint.

1	Type	Objects	Line Width	
			Min	Max
2			mil	mil
3	Dsn	<input type="checkbox"/> unnamed	5.00	0.00
4	PCS	<input checked="" type="checkbox"/> DEFAULT	5.00	0.00
5	PCS	<input checked="" type="checkbox"/> PCS1	7	0.00

Next, let's associate the *Physical CSet* with the *Net Class*, whose members are the voltage rails that we want to constrain.

1	Type	Objects	Referenced Physical CSet	Line
				Min
2				mil
3	Dsn	<input type="checkbox"/> unnamed	DEFAULT	5.00
4	NCLs	<input type="checkbox"/> CLS1	DEFAULT	5.00
5	Net	VCC	DEFAULT	5.00
6	Net	VSR_RTN	PCS1	5.00
7	Net	24V	(Clear)	5.00

Allegro Constraint Manager User Guide

Working With Reusable Constraint Objects — CSets

The advantage of referencing a CSet to impart constraints on a *Net Class* is that you can reuse the CSet to constrain similar *Net Classes*, whose members are different.

1 2	Type	Objects	Referenced Physical CSet	Line Width	
				Min mil	Max mil
3	Dsn	<input type="checkbox"/> unnamed	DEFAULT	5.00	0.00
4	NCIs	<input type="checkbox"/> CLS1	PCS1 <input checked="" type="checkbox"/>	7.00	0.00
5	Net	VCC	PCS1	7.00	0.00
6	Net	VSR_RTN	PCS1	7.00	0.00
7	Net	24V	PCS1	7.00	0.00
8	Net	48V	PCS1	7.00	0.00



Tip

Use a net container object to group and constrain a small, focused collection of similar nets. Use a reusable CSet, referenced to one or more container objects, to apply a broad brush approach to constraining nets.

ECSets and Topology Templates

Topics in this chapter include

- [“What is a Topology Template?”](#) on page 100
- [“Importing ECSets”](#) on page 102
- [“Mapping Templates and ECSets to Net-related Objects”](#) on page 102
- [“Audits”](#) on page 106
- [“Exporting ECSets”](#) on page 111
- [“Migrating Legacy Electrical Rule Sets”](#) on page 111

What is a Topology Template?

A topology template file (.top) is an on-disk image of the SigXplorer database. A topology template file contains the same data as an ECSet, including electrical constraints, but it also contains information to support the graphical representation of a circuit topology in SigXplorer.

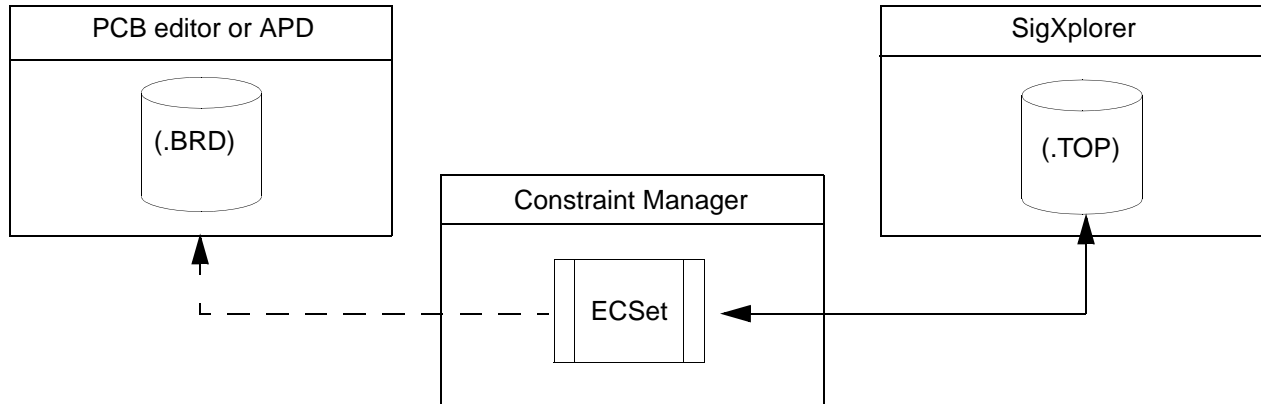
Note: Constraint Manager, when launched from an L Series PCB editor, does not support topology exploration. If you are using an L Series PCB editor, you can ignore the descriptions of the following commands in this chapter:

- ❑ *File – Import Electrical CSets*
- ❑ *File – Import Analysis Results*
- ❑ *File – Export Electrical CSets*
- ❑ *File – Export Analysis Results*
- ❑ *Audit – Topology Templates*

In pre-14.0 designs, a topology template was applied to a group of target nets. In the process, constraint information was extracted from the topology file and flattened; constraints were copied to individual target nets as properties. With Constraint Manager, these properties are not flattened; they remain intact, collectively, as an ECSet. Net-related objects in the design reference the ECSet rather than a collection of individual properties.

The topology template can be imported to, and exported from, Constraint Manager. When imported, the topology template information will be instantiated within Constraint Manager as an ECSet where it can be manipulated separate from the topology template. The ECSet is saved with the database of the host application from which Constraint Manager was invoked: a board file (.brd) or a schematic view.

Figure 4-1 Topology Template/ECSet constraint flow



With such a close alignment between a topology template and an ECSet, you can access SigXplorer directly from Constraint Manager. In fact, you can define your constraints in SigXplorer—as a topology template—and then import this information into Constraint Manager as an ECSet. Conversely, you can define your constraints in Constraint Manager—as an ECSet—and then export this information to SigXplorer as a topology template. See the [Design Exploration Phase \(with SigXplorer\)](#) on page 137 for information on the Constraint Manager-SigXplorer design flow.

The only constraint that you cannot define in Constraint Manager is user-defined pin scheduling. This must be defined in SigXplorer. You can, however, select from a list or pre-defined pin schedules in Constraint Manager.

Importing ECSets

Constraint Manager promotes design reuse through the following command:

Use this command	To
-------------------------	-----------

<i>File – Import – Electrical CSets</i>	Import a selected on-disk topology template into Constraint Manager. The imported template will become an ECSet which can be referenced by net-related objects that share the same electrical characteristics. See “Mapping Templates and ECSets to Net-related Objects” on page 102.
---	---

If the imported template was previously assigned as an ECset, the import will overwrite existing constraint values.

Note: If the *Automatic Topology Update* checkbox is enabled (*Tools – Options Figure 4-2* on page 103), the refreshed template information is immediately applied to the net-related objects; otherwise, you must choose *Tools – Update Topology* to apply the changes.

Mapping Templates and ECSets to Net-related Objects

Constraint Manager intelligently maps the constraint information, imported from a topology template or defined in an ECSet, to a candidate net that matches the topological characteristics of the referenced ECSet. If the candidate net does not match these topological characteristics, the mapping will fail and the constraints will not be applied.

In the Mapping Mode column in the *All Constraints* worksheet (in the ECSet folder), you can specify a mapping mode. Constraint Manager makes several *passes* based on the following mapping modes:

See the following commands in the *Allegro Constraint Manager Reference* for more information:

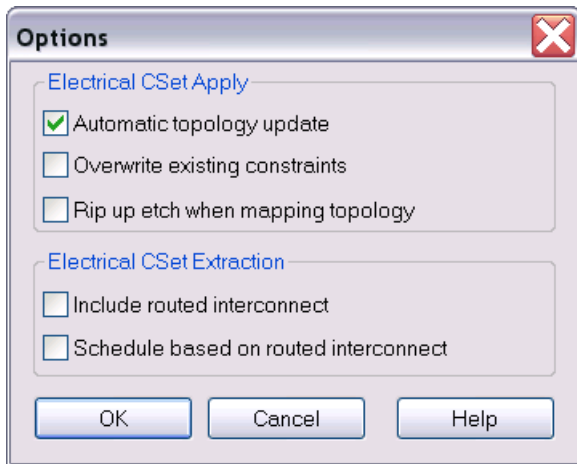
- [*Objects – Electrical CSet References*](#)
- [*File – Import – Electrical CSet*](#)

All other electrical constraints will be inherited regardless of mapping since the other constraints are not topology specific.

- Choose *Tools — Options* to control how objects in Constraint Manager inherit constraint information.

Figure 4-2 Options Dialog Box with Default Settings

The following describes how to use the *Electrical CSet Apply* fields of the *Options* dialog box (see [Figure 4-2](#) on page 103).



Allegro Constraint Manager User Guide

ECSets and Topology Templates

Checkbox Option	Function
<i>Automatic topology update</i> (Defaults to enabled)	<p>Controls how topology-related constraints are re-applied</p> <ul style="list-style-type: none">■ When the design changes (component placement, signal model updates) <p>– or –</p> <ul style="list-style-type: none">■ When an ECSet is initially referenced <p>When <i>enabled</i>, Constraint Manager applies changes on-the-fly as the design changes.</p> <p>When <i>disabled</i>, you can apply changes by choosing <i>Tools — Update Topology</i>.</p> <p>If you change state from <i>disabled</i> to <i>enabled</i>, Constraint Manager presents you with a confirming message stating that it will refresh stale nets and Xnets with updated topology data.</p>
<i>Overwrite existing constraints</i> (Defaults to disabled)	<p>Controls whether constraint values in the ECSet will overwrite any existing net-related constraints when an ECSet is re-applied. See “Methods of Constraining Nets” on page 95 for information about overriding inherited constraint values.</p>



Tip

Disabling automatic topology update may be necessary when design changes are frequent and complex ECSets are referenced.



Important

Enabling *overwrite existing constraints* is necessary when migrating pre-14.0 designs using the *Audit – Topology Properties* command. This will ensure that all net-related overrides— created by the pre-14.0 topology template mapping software—are removed.

Allegro Constraint Manager User Guide

ECSets and Topology Templates

Checkbox Option	Function
<i>Rip up etch when mapping topology</i> (Defaults to disabled)	Controls whether etch (clines and vias) is removed when an ECSet is re-applied and the schedule of the net changes.

Audits

Constraint Manager provides audits to give you feedback, in report form, about the constraints and their references in the design. Audit commands are available when Constraint Manager is invoked from a PCB- or design entry-editor, or APD. The following audits (accessible from the *Audit* menu) relate to ECSets.

Run this audit	To
<i>Constraints</i>	List net-related overrides and constraint violations.
<i>Obsolete Objects</i>	List objects that should, but no longer exist in in the board or schematic database, yet they are still being referenced in Constraint Manager.
<i>Electrical CSets</i>	List the mapping status of all objects which reference ECSets. Note: Constraint Manager, when launched from a Series L editor, does not support pin pairs. Any pin pairs that exist will appear as 'not supported' in the audit report.
<i>Topology Properties</i>	Migrate pre-14.0 ASSIGN_TOPOLOGY and TOPOLOGY_TEMPLATE properties to an ECSet reference. Note: Constraint Manager, when launched from an L Series PCB editor, does not support topology exploration.

The sections that follow describe these audits.

Constraint Audit

The Constraints audit (*Audit – Constraints*) generates a report listing constraint errors. This report aids you in troubleshooting constraint violations. The audit includes the following checks:

- Min values that exceed Max values
- Values less than zero
- Completeness violations

- Group membership violations
- Relative group violations
- Paired parallelism lengths and gap
- Setup and hold relative to clock period
- Differential pair member mismatches
- Net-related overrides

Note: A net override lets you replace the inherited value from an ECSet with a value that you specify on a net-by-net basis. Cells with net overrides are colored blue.

Obsolete Objects Audit

The *Obsolete Objects* audit (*Audit – Obsolete Objects*) generates a report that lists objects that must be reconciled between Constraint Manager and the PCB or schematic databases. Constraint Manager displays a *No Obsolete Objects* message as appropriate.

For example, if you use Constraint Manager to constrain an object in the schematic, that object will be stored in the schematic editor's constraint view of the HDL library. If you later delete that object in the schematic, that constraint will still be in Constraint Manager until it is reconciled with the obsolete objects audit.

This command is used subsequent to importing a dictionary and constraint file (*File – Import – Constraints*) or when the connectivity is disjoint between the component or net in schematic and the corresponding Constraint Manager object.

Note: The *Audit – Obsolete Objects* command is *not* available when running Constraint Manager in stand-alone mode.

The Audit Obsolete Objects dialog box contains the following fields:

Table 4-1 Obsolete Objects Dialog Box Options

Use this field	To
<i>Type</i>	Filter on an object type (bus, net, Xnet)
<i>Obsolete Objects</i>	List all objects that no longer exist in the PCB or schematic database, yet exist in Constraint Manager
<i>Existing Objects</i>	List all objects that exist in the PCB or schematic database, and in Constraint Manager

<i>Delete</i>	Remove objects, listed in the <i>obsolete objects</i> list, from the PCB or schematic database.
<i>Merge</i>	Assign all properties and constraints from the object selected in the <i>obsolete object</i> list to the object selected in the <i>existing object</i> list (properties and constraints on the existing object are not overwritten).

Electrical CSets Audit

The *Electrical CSets* audit (*Audit – Electrical CSets*) generates a report listing the current ECSets in the design and the status of all net-related objects that reference them. The head of the report summarizes the number of ECSet references and any errors.

The status reports the inheritance for each constraint defined in the ECSet including:

- Any mismatch of the topological characteristics between a net-related object and the ECSet (in which case, the constraint from the ECSet is not inherited).
- The net-related object that inherits the ECSet constraint.



When the head of the *Referenced Electrical CSet* column is yellow, this indicates there is a stale ECSet reference. You run the *Electrical ECSet* audit to clear this setting and to resolve any discrepancies.

Topology Templates Audit

The *Topology Templates* audit (*Audit – Topology Templates*) migrates deprecated properties to ECSet references used by Constraint Manager.

Note: Constraint Manager, when launched from an L Series PCB editor, does not support topology templates.

In pre-14.0 designs, electrical constraints were captured using three entities: the topology template (specified with the `TOPOLOGY_TEMPLATE` property), the topology assignment (specified with the `ASSIGN_TOPOLOGY` property), and the constraint set (specified with the `ELECTRICAL_CONSTRAINT_SET` property).

Allegro Constraint Manager User Guide

ECSets and Topology Templates

In 14.0 designs, only the ECSet association is supported; the `TOPOLOGY_TEMPLATE` and the `ASSIGN_TOPOLOGY` properties are no longer required. All topology information is now contained in the ECSet (specified with the `ELECTRICAL_CONSTRAINT_SET` property).

The topology templates audit removes the `TOPOLOGY_TEMPLATE`, `TOPOLOGY_TEMPLATE_REVISION`, and `ASSIGN_TOPOLOGY` references from net-related objects.

The *Audit Topology Templates* dialog box contains the following fields:

Table 4-2 Audit Old Template Dialog Box Options

Use this field	To
<i>Topology template values</i> (drop-down menu)	List all <code>TOPOLOGY_TEMPLATE</code> and <code>ASSIGN_TOPOLOGY</code> property values in the design. Once a property value is selected from the drop-down menu, all nets which have the same template value are listed.
<i>Update to use Electrical Cset</i>	
<i>Import</i> (radio button)	Import a new template. The field beside the radio button will be populated with a Topology Template (.top file) name if one exists on disk. Constraint Manager uses the <code>TOPOLOGY_TEMPLATE_PATH</code> environment variable to search for a template file with the same name as the property value
<i>Browse</i>	Find the appropriate template file if Constraint Manager cannot locate it.
<i>Existing</i> (radio button)	Use an existing ECSet. This option will be the default if the design contains an ECSet with the same name as the property value

Allegro Constraint Manager User Guide

ECSets and Topology Templates

Use this field

Overwrite existing constraints
(check box)

To

Controls whether constraint values in the ECSet overwrite any existing net-related constraints when an ECSet is re-applied.

Important

Enabling *overwrite existing constraints* is necessary when migrating pre-14.0 designs. This ensures that all net-related overrides—created by the pre-14.0 topology template mapping software—are removed.

Apply

Migrate all listed nets to reference the imported or existing ECSet. Once migrated, nets will have their `TOPOLOGY_TEMPLATE` and `ASSIGN_TOPOLOGY` properties deleted.

Constraint Manager displays a *properties up to date* message as appropriate.

Exporting ECSets

Constraint Manager promotes design reuse through the following commands:

Use this command	To
<i>File – Export – Electrical CSets</i>	Save selected ECSets, from a design, or from a system, to a topology template (.top) file on disk.
<i>File – Export – Constraints</i>	Export a dictionary and constraints file (.dcf) to disk. The dictionary and constraints file contains a complete snapshot of all electrical constraint information. This includes any user-defined properties, all ECSets and their constraints, and all net-related objects and their constraints (including ECSet references). The dictionary and constraints file is proprietary to Cadence Design Systems and, as such, is not available for editing.

You would typically save to a dictionary and constraint file prior to making extensive constraint modifications within Constraint Manager. The dictionary and constraints file is then considered an archive from which you could revert back.

Exporting a dictionary and constraints file results in overwriting constraint data saved from a previous archive.

Migrating Legacy Electrical Rule Sets

Designs created prior to release 14.0 may contain Electrical Rule Sets. As part of the process of upreving a design to release 14.0, these Electrical Rule Sets are migrated to Constraint Manager as Electrical Constraint Sets (ECSets).

Allegro Constraint Manager User Guide

ECSets and Topology Templates

Constraint Analysis

Topics in this chapter include

- [“Viewing Worksheet Cells and Objects”](#) on page 115
- [“Analyzing for DRC-based Constraints”](#) on page 119
- [“Analyzing for Simulation-based Constraints”](#) on page 121
- [“Simulation-based Custom Stimulus”](#) on page 122
- [“Analysis Results”](#) on page 128
- [“Interpreting Analysis Results Returned to a Worksheet”](#) on page 130
- [“Constraints Across the System”](#) on page 133

How Allegro[®] Constraint Manager Performs Analysis

Allegro[®] Constraint Manager analyzes the constraints in your design using two methods:

■ Design Rule Checks

Real-time design rule checks are made on objects constrained in the *Routing* worksheets. Results are returned to the worksheet cells in focus by comparing changes in the layout, such as moving a part, against the constraint limits that you specified for these objects.

As design rule violations are encountered, Constraint Manager colors the corresponding worksheets cells in red. Additionally, bow tie markers appear on offending objects in the layout.

See [“Analyzing for DRC-based Constraints”](#) on page 119 for information about interactive, online design rule checking.

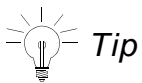
■ Simulated Analysis

Simulated analysis is made on objects constrained in the *Signal Integrity* and *Timing* worksheets in the Electrical domain.

Note: Constraint Manager, when launched from an L Series PCB editor, does not support the *Signal Integrity* and *Timing* workbooks.

Analyzed results are returned to the worksheet cells in focus by comparing computations (the actual) against the constraint limits that you specified for these objects. The actual, and the difference between the actual and the set constraint limit (the margin) are returned.

See [“Analyzing for Simulation-based Constraints”](#) on page 121 for information about analyzed constraints.



The analysis engine computes a value (*actual*) and compares this to the value specified in the CSet. The difference between the analysis value and the specified constraint value is the *margin*. Both actuals and margins are returned to the cells in the appropriate worksheets.

Note: Constraint Manager uses the same color scheme for both analyzed results and design rule checks. See [“Customizing Visibility”](#) on page 150 for more information.



You can click on an object in a Constraint Manager worksheet and choose *Object* – *Select* to highlight that object in the layout.

Viewing Worksheet Cells and Objects

As the complexity of your design increases, the number of objects in your design increases; and, correspondingly, the number of CSets associated with those objects increases. This can lead to a high-level of congestion in your worksheets. Fortunately, Constraint Manager lets you easily change your view of constraints, letting you change your focus as you work.

Allegro Constraint Manager User Guide

Constraint Analysis

Allegro Constraint Manager User Guide

Constraint Analysis

Table 5-1 Common worksheet commands

Task	Related Commands	Actions
Locating an object, a result, or a CSet	<i>Edit – Find</i>	<p>Finds the specified object.</p> <p>You can filter on the following:</p> <ul style="list-style-type: none"> ■ Match whole word only ■ Expand hierarchy <p>You can click <i>Find Next</i> (or F3) to locate the next occurrence.</p>
	<i>Edit – Go to source</i>	<p>Locates the parent object that owns the inherited ECSet of the selected child object.</p> <p>For example, if you select a child object, such as a bit in a bus, its constraints are most-likely inherited from a CSet that is associated with the parent object, in this case the bus.</p>
	<i>View – Options – Row Numbers</i>	Enable row numbering in the worksheets.
	<i>Objects – Filter</i>	<p>Selectively display (or hide) the following objects in the worksheets:</p> <ul style="list-style-type: none"> ■ net ■ Xnet ■ pin pair ■ results ■ differential pair ■ bus ■ match group ■ net class ■ net class-class ■ region ■ region class ■ region class-class

Allegro Constraint Manager User Guide
Constraint Analysis

Table 5-1 Common worksheet commands

Task	Related Commands	Actions
Controlling the worksheet or object hierarchy	<i>Objects – Expand/Collapse</i> (or use the [+] and [-] controls)	Expand or collapse the worksheet hierarchy in the worksheet selector or the object hierarchy in the worksheets. Worst-case analysis results on collapsed (hidden) objects are rolled up to the expanded object. This notification lets you work at any level in the object hierarchy.
	<i>View – Show All Rows</i>	Expand or collapse all rows in all worksheets.
Working in columns	<i>Column – Sort</i> (or double-click the column head)	Reverse the ordering of objects or constraint values in a column.
	<i>View – Hide/Show Column</i>	Hide columns in a worksheet so you can focus on a single, or a few, columns. Otherwise, you may have to scroll horizontally to access an out-of-view column.
	Resize	Resize column width. Grab a column border and drag.
Comparing cells	<i>Window – Tile</i>	Compare the cells of two or more different worksheets. You may have to scroll to view the desired cells.
	<i>Window – New Window</i>	Compare cells in the same worksheet. Constraint Manager opens the same worksheet in a different window. This lets you scroll to different cell views while allowing you to make concurrent edits in the same worksheet.

Analyzing for DRC-based Constraints

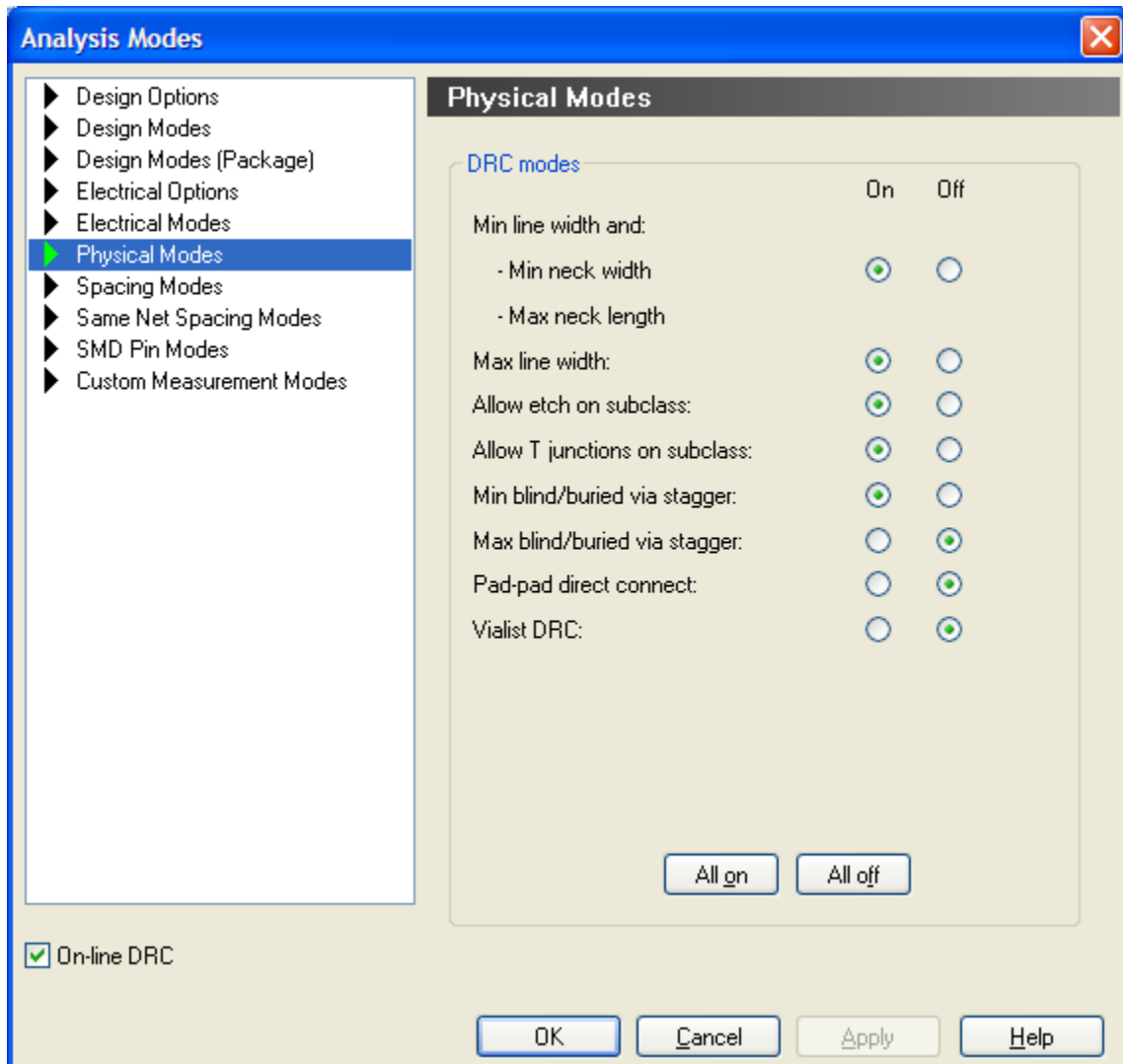
You control design rule checks with the domain tabs of the *Analysis Modes* dialog box (choose Analyze – Analysis Modes). Alternatively, you can specify analysis settings, DRC modes, and desired reports from a single dialog box (choose Objects – Report).

Note: Constraint Manager, when launched from an L Series PCB editor, does not support custom measurements or custom stimulus; therefore, these tabs are not visible in the *Analysis Modes* dialog box. Furthermore, the *Max xtalk* and *Max peak xtalk* DRC fields in the Electrical Mode are hidden.

DRC Constraint Modes

Use the *Analysis – Modes* dialog box to control which design rule checks (DRC) to run. When the layout changes, an enabled design rule check is triggered.

Figure 5-1 Analysis Modes dialog box: Electrical Modes View



Refer to the [Analyze – Analysis Modes](#) in the Constraint Manager Reference for information about how to use DRC constraint modes.

Analyzing for Simulation-based Constraints

Certain constraints in the Electrical Domain (*Signal Integrity* and *Timing*) require simulation to compute *actual* values. When the actual value is analyzed and returned to a worksheet cell, it is compared with the specified constraint value that is associated with the object being analyzed. The difference is calculated and displayed in the *Margin* column.

Important

To analyze for simulation-based constraints, Constraint Manager must be run with a PCB editor or APD.

Before you initiate an analysis (*Analyze – Analyze*) on an object, you should configure the analysis engine (*Analyze – Settings*). Alternatively, you can specify analysis settings, DRC modes, and desired reports from a single dialog box (*Objects – Report*).

In the *Analysis Settings* dialog box (see “[Analysis Settings](#)” on page 123, you specify the type of simulation, whether to use crosstalk timing windows, and the type of stimulus.

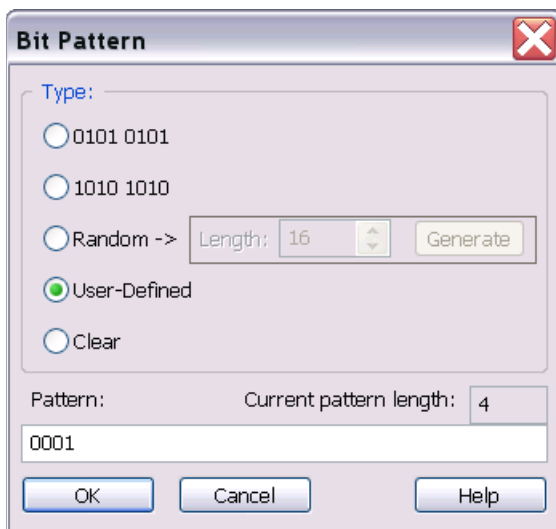
You can click *Preferences* to specify buffer information (in the *Analysis Preferences* dialog box). You can also choose to save a waveform for each analysis; waveforms can subsequently be viewed in SigWave (choose *Tools – SigWave*).

See the [Allegro® SI Simulation and Analysis Reference](#) for detailed information on analysis settings and preferences.

Simulation-based Custom Stimulus

In addition to capturing custom stimulus in a SigXplorer topology file and importing it into Constraint Manager, you can create your own stimulus patterns directly in the *Electrical Properties* worksheet in the *Signal Integrity* workbook (Electrical Domain). Custom Stimulus is associated with Custom Measurements (see [Customizing Simulations](#) on page 154).

Type	Objects	Duty Cycle	Jitter	Cycle to Measure	Offset	Bit Pattern
		%	ps		ns	
Dsn	<input type="checkbox"/> unnamed					
Bus	<input type="checkbox"/> ADDRESS_BUS					0001
Net	AD0					0001
Net	AD1					0001



Double-click in the Bit Pattern cell to invoke the editor.

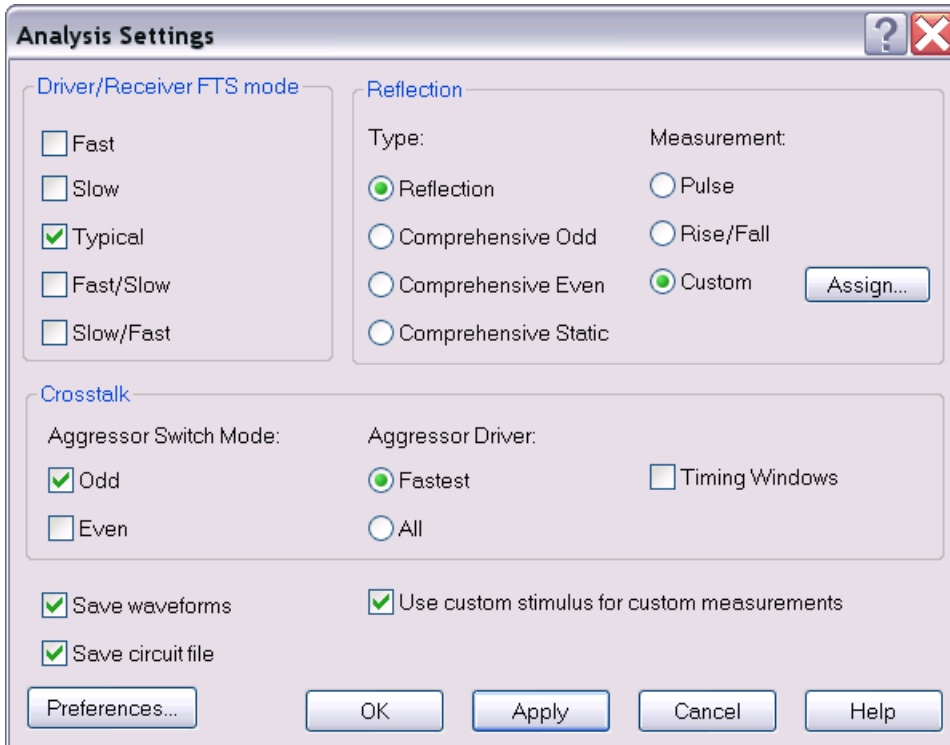
You can choose from 8-bit repeating patterns that begin with either high or low. You can also specify a randomly-generated pattern or a pattern of your choice, up to 512 bits.

Note: Constraint Manager captures clock measurements in the PULSE_PARAM property. The Offset column converts all cell entries to nanoseconds.

To enable custom stimulus

Custom Stimulus exercises Custom Measurements. You enable Custom Stimulus through the ([Analyze – Settings](#)) dialog box. Enable the *Reflection*, *Custom*, and *Use custom stimulus* for custom measurements radio buttons, as shown in [Figure 5-2](#) on page 123.

Figure 5-2 Analysis Settings



The Analysis Process

The following steps serve as a guideline (a checklist) of the steps involved in performing analysis in Constraint Manager. You may not need to perform all the steps all the time; it depends on where you use Constraint Manager in the design flow. For example, once you set DRC modes and analysis settings, you may decide to retain these settings for subsequent analysis.

Allegro Constraint Manager User Guide

Constraint Analysis

Step 1 Creating Design Objects

You want to combine objects, where appropriate, into easily-managed object groupings. In this way, constraints can be set at different levels in the object hierarchy.

- Where appropriate, combine designs into systems.

A system configuration database is advisable for maintaining system-level constraints and design objects. See the [Allegro[®] SI Simulation and Analysis Reference](#) for more information on system-level design.

- Where appropriate, combine nets and Xnets into buses and net classes.
- Where appropriate, combine nets or Xnets into differential pairs.

For modeled-defined differential pairs, each member of the differential pair must have the appropriate signal model assignment.

- Where appropriate, combine nets, Xnets, and pin pairs into match groups when specifying relative propagation delays.
- Where appropriate, specify pin pair connections.

See [“Working with Constraint Objects”](#) on page 49 for information about how to organize objects and [“Working With Reusable Constraint Objects — CSets”](#) on page 93 for information about creating and assigning CSets.

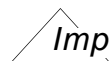
Step 2
Setting Constraints

Next, you create CSets based on your design requirements.

- Create an CSet in the appropriate worksheet.

This can be done (1) from scratch in the CSets object folder; (2), based on an existing net-related object in the Nets object folder; (3), by cloning an existing CSet; or (4), by importing a CSet.

Note: You can also select a net, extract it to SigXplorer, set constraints, update the topology in Constraint Manager (as an Electrical CSet), and apply the it to other objects.

 *Important*

When specifying constraint parameters in a worksheet cell, it may be helpful to right-click and choose *Change* from the pop-up menu. This will guide you through the appropriate parameters and syntax that applies for the specific constraint type that the cell represents.

See [“Working With Reusable Constraint Objects — CSets”](#) on page 93 for information about creating and assigning CSets.

Allegro Constraint Manager User Guide

Constraint Analysis

Step 3 Assigning Constraints

Next, you assign CSets to appropriate objects in your design. Child objects inherit the constraints from an CSet assigned to a parent object.

- Assign the CSet to a net-related object

-or-

Set a constraint directly on a net-related object.

If a CSet is already assigned to that object, the constraint change that you make will override the constraint value inherited from the CSet.

Assignments can be from the CSet or from a net-level object.

See [“Working With Reusable Constraint Objects — CSets”](#) on page 93 for information about creating and assigning CSets.

Step 4 Setting DRC Modes

Next, you specify how Constraint Manager performs design rule checks. You may want to make a trade-off between completeness and performance.

- Set the appropriate mode for design rule checking as described in [Analyzing for DRC-based Constraints](#) on page 119.

Step 5 Setting View Options

Next, you may want to change the way Constraint Manager presents data.

- Ensure that the *use color* checkbox is enabled (choose *View – Options*).
- Set the desired colors to use for results returned from analysis as described in [“Viewing Worksheet Cells and Objects”](#) on page 115

Allegro Constraint Manager User Guide

Constraint Analysis

Step 6

Setting Analysis Parameters (Electrical)

Next, you set up simulation parameters for reflection and crosstalk analysis.

- Specify parameters (*Analyze – Settings*) that govern the analysis engine as described in [“Analyzing for Simulation-based Constraints”](#) on page 121.

For an in-depth discussion of analysis parameters, see the [Allegro® SI Simulation and Analysis Reference](#).

Step 7

Setting Report Parameters (Optional)

Next, you specify report types and what objects to include in the report. You typically will want a report when you want to analyze *many* objects; otherwise, it is more practical to interpret results returned to worksheet cells when you are concerned with only a *handful* of objects.

- Specify the reports that you want generated from simulation-based analysis (choose *Objects – Report*).

In the *Report* dialog box, you identify the CSets, and net-related worksheets, to be included in the analysis results.

You can limit the report to specific object types (bus, differential pair, Xnet, net, net class, Cset), and to a specific condition (any condition, only violations, only failures, only objects that are constrained).

Note: Incidentally, from the *Report* dialog box, you can also specify DRC modes and analysis settings, and you can initiate simulation-based analysis.

For an in-depth discussion of reports, measurements, and computations, see the [Allegro® SI Simulation and Analysis Reference](#).

Allegro Constraint Manager User Guide

Constraint Analysis

Step 8

Selecting an object

Next, you choose which objects to analyze. At this stage, some analysis is complete based on DRC settings. Worst-case results of child objects roll up to the respective parent object.

- Specify a net, Xnet, or object grouping to be analyzed.
- Once in view, click in a cell in the object column.

You can select a range of cells using `Shift-Click` and non-contiguous cells by using `Cntrl-click`.

Step 9

Analyzing

Finally, you initiate the simulation(s).

- Choose *Analyze – Analyze* (or right-click and choose *Analyze* from the pop-up menu).



As the analysis progresses, you can receive feedback by monitoring the status bar (located at the lower-left corner of Constraint Manager).

Analysis Results

Results returned from Analysis take four forms:

- Generated DRCs in the layout
- Waveforms
- Reports
- Calculated *actuals* and *margins* populated in the worksheets

Each is discussed in the sections that follow.

Generated DRC Output

Updated constraint information is communicated to the PCB editor or APD. If a violation exists, a DRC bow tie marker is attached to the offending object in the layout.

Waveforms

Analysis results returned for certain constraints in the *Signal Integrity* and *Timing* worksheets yield waveform files. In Constraint Manager, choose *Tools – SigWave* to view these waveforms.

Reports

For each *enabled* net-related worksheet or CSet, a report is produced, `consmgr.rpt`, that lists constraint parameters, object assignments, and analysis results.

Worksheet cells

Analysis results returned to worksheet cells exhibit the following behavior.

- Cells give graphical feedback to reflect their status. See [Viewing Worksheet Cells and Objects](#) on page 115 for more information on default and user-defined colors. By default, the following color scheme is used for analysis:
 - Pass = green
 - Fail = red
 - Analysis error = yellow
 - Directly set = blue
- Cells that are grayed-out reflect that the cell is not applicable for the selected object.
- Cells will be colored blue if the cell contains a value which has been explicitly entered. This could happen when you override, for example, one bit of a bus object or when you specify a constraint directly on a net-related object rather than having that object inherit its constraint value from a referenced CSet.
- Cells which are populated and colored black reflect that the value is inherited from a higher-level cell or an CSet reference on the object. When you select an inherited cell, the status bar will indicate the source of the value. The source (the owner of the object) is reported as the object type and its name.

Interpreting Analysis Results Returned to a Worksheet

Figure 5-3 and Table 5-2 take you through a typical scenario of analyzing for propagation delay. Together, they explain how to interpret the analyzed results fed back to the worksheet.

Figure 5-3 Analyzing for Propagation Delay

Objects	Referenced Electrical CSet	Pin Pairs	Min Delay			Max Delay		
			Min	Actual	Margin	Max	Actual	Margin
			ns			ns		
⊕ LMD_BUS	LMDATA_NEW	All Drivers/All Receivers	0 MIL			3400 MIL		
⊕ MAA_BUS	SRAS_A							
⊖ MAB_BUS	MA_B13-NEW	All Drivers/All Receivers	2000 MIL	-245.3 MIL		3600 MIL		1328.1 MIL
⊕ MAB_0	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		176 MIL	3600 MIL		1424 MIL
⊖ MAB_1	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		216.7 MIL	3600 MIL		1383.3 MIL
U18.AC16:U25.117			2000 MIL	2216.7 MIL	216.7 MIL	3600 MIL	2216.7 MIL	1383.3 MIL
⊕ MAB_2	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		271.9 MIL	3600 MIL		1328.1 MIL
⊕ MAB_3	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		106.5 MIL	3600 MIL		1493.5 MIL
⊕ MAB_4	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		121.7 MIL	3600 MIL		1478.3 MIL
⊕ MAB_5	MA_B13-NEW	All Drivers/All Receivers	1975 MIL		13.6 MIL	3600 MIL		1611.4 MIL
⊕ MAB_8	MA_B13-NEW	All Drivers/All Receivers	1975 MIL		10.5 MIL	3600 MIL		1614.5 MIL
⊖ MAB_13	MA_B13-NEW	All Drivers/All Receivers	2000 MIL	-245.3 MIL		3600 MIL		1845.3 MIL
U18.AF22:U25.123			2000 MIL	1754.7 MIL	-245.3 MIL	3600 MIL	1754.7 MIL	1845.3 MIL
⊕ MCDQA_BUS	MDXX							
⊕ MCKE_BUS	CKE	All Drivers/All Receivers	1000 MIL			3250 MIL		
⊕ MD_BUS	MDXX-IO							

Table 5-2 Dissecting the Analyzing for Propagation Delay figure on page 130

Object	Cell Column	Comments
LMD_BUS	Referenced CSet	CSets are set directly on the bus-level object. The cell is rendered blue.
MAA_BUS		
MAB_BUS		
MCDQA_BUS		
MCKE_BUS		
MD_BUS		

Allegro Constraint Manager User Guide

Constraint Analysis


Table 5-2 Dissecting the Analyzing for Propagation Delay figure on page 130

Object	Cell Column	Comments
LMD_BUS	Min Delay (Actual/Margins) Max Delay (Actual/Margins)	<p>Cells are rendered green and do not contain values. This indicates that the last time the object (LMD_BUS) was analyzed, it was within the specified constraint values.</p> <p>For example, if the board was analyzed in an earlier design session, the analyzed values would not be saved with the board database. However, the last analyzed state of the object (<i>pass</i>, in this case) is communicated back to the cell in the form of a solid color.</p> <p>To populate the cells with integral values, you must re-run analysis (choose <i>Analyze – Analyze</i>). You could also import saved analysis results (choose <i>File – Import – Analysis results</i>).</p>
MAB_5 MAB_8	Min Delay (Min)	<p>Nets 5 and 8 of MAB_BUS have overrides. Because the overrides were specified explicitly in each cell, the cell is rendered blue.</p> <p>Notice that all other members of MAB_BUS inherit their values from the constraint specified at the bus level. Therefore, these cells are rendered black.</p>
MCKE_BUS	Min Delay/ Max Delay	Analysis failed, rendering the cells yellow. This was caused by an unplaced component attached to a net member of this bus.
MAB_4	Min Delay (Actual/Margin) Max Delay (Actual/Margin)	Analysis passes, rendering the cells green. Notice that only the <i>Margin</i> column contains an integral value; the <i>Actual</i> is solid. This is because the net has several hidden pin pairs. Since the cell can contain only one value, the cell is rendered a solid color to represent a pass/fail condition.
MAB_1	Min Delay (Actual/Margin) Max Delay (Actual/Margin)	Analysis passes, rendering the cells green. Notice that both the <i>Margin</i> and the <i>Actual</i> columns contain integral values. This is because the net has been completely expanded (bus, to net, to pin pair).

Allegro Constraint Manager User Guide

Constraint Analysis

Table 5-2 Dissecting the Analyzing for Propagation Delay figure on page 130

Object	Cell Column	Comments
MAB_BUS MAB_13	Min Delay (Actual/Margin)	<p>Analysis is in violation, rendering the cells red.</p> <p>Notice that both the <i>Margin</i> and the <i>Actual</i> columns contain integral values at the pin pair level of the MAB_13 net. This is because the net has been completely expanded (bus, to net, to pin pair).</p> <p>Also, notice that the net object that owns the pin pair displays a solid red in the actual column and an integral value in the margin column. This is because the worst-case violation is rolled up to the object that owns it. In this example, there is only one pin pair so it was rolled up.</p> <p>Again, if there were more than one pin pair in violation, since the <i>Actual</i> cell can contain only one value, the cell would be rendered a solid color to represent a pass/fail condition.</p> <p> Important</p> <p>Worst-case constraint violations on child objects are rolled up the object hierarchy to the parent object. That is, pin pairs roll up to the parent net or Xnet, nets or Xnets roll up to the parent bus, and buses roll up to the parent design. In this way, you can work at any level in the object hierarchy and still be informed of a constraint violation on a lower-level object that is hidden.</p> <p>Finally, the same worst-case pin pair violation on the MAB_13 net is rolled up to the parent bus, MAB_BUS.</p>

Constraints Across the System

A system configuration represents the electrical characterization of a system including all the participating designs, including interconnecting cables and connectors, as well as Xnets and pin pairs and their assigned CSets

Note: Constraint Manager, when launched from an L Series PCB editor, does not support board-to-board constraints.

You can set a constraint directly on a system-Xnet in your layout or you can define the constraint in Constraint Manager as a CSet and then reference it to a system-Xnet.

See the [Allegro SI Simulation and Analysis Reference](#) for a thorough discussion of system-level designs.

Allegro Constraint Manager User Guide

Constraint Analysis

Using Constraint Manager with Other Tools Across the Allegro Platform

Topics in this chapter include

- [Phases in the Design Flow](#) on page 136
- [Design Exploration Phase \(with SigXplorer\)](#) on page 137
- [Design Capture Phase](#) on page 139
- [Design Capture Phase \(with System Connectivity Manager\)](#) on page 139
- [Design Floorplanning and Implementation Phases](#) on page 144

Phases in the Design Flow

A typical PCB design flow contains the following phases:

- Exploration
 SigXplorer
- Capture
 Allegro Design Entry HDL, Allegro System Architect, System Connectivity Manager
- Floorplanning
 Allegro PCB SI
- Implementation
 PCB Editor, APD, SiP

Each phase in the design flow requires different tools. Constraint Manager provides a *common* environment for managing constraints across all tools in the design flow.

Not all phases in the design flow are mandatory. For example, a new design may be a derivative of a prior design. In this case, the *exploration* and *floorplanning* phases may not be needed.

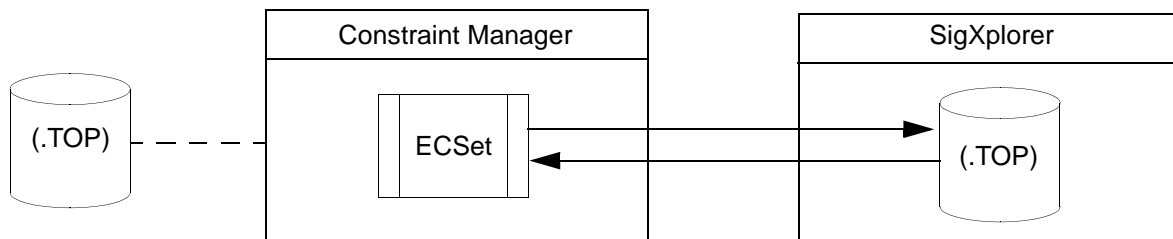
Constraint information in the board and in the schematic databases are synchronized using Design Sync. With Design Sync, you can specify that all constraints be synchronized or only those that have changed.

The sections that follow describe how to use Constraint Manager with other tools at each phase in the design flow.

Design Exploration Phase (with SigXplorer)

In the exploration phase, you focus on up-front exploration before the board is placed and routed. Board cross-section and material type are usually *not* known, although you can make assumptions from past designs. A netlist is *not* available in this phase of the design flow.

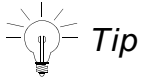
Use SigXplorer to perform simulations based upon the Electrical CSets characteristics (pins, scheduling, models). A unique topology template can be saved for each point explored in the solution space. The end result of the exploration phase is to create a library of Electrical CSets (.top files on disk) which would then be imported back into Constraint Manager where they could be swapped with other Electrical CSets, or where individual constraints could be moved between Electrical CSets.



In the exploration phase, you have a choice. Constraints can be proven in SigXplorer and saved as topology templates or they can be defined in Constraint Manager. The primary difference is presentation: SigXplorer is form-based; Constraint Manager is worksheet-based and perhaps it is easier to use for viewing and manipulating multiple constraint definitions.

Note: You will notice that only the *Electrical Constraint Set* folder is shown in Constraint Manager's worksheet selector. Absent of a board database, and a netlist, Constraint Manager does not show the *Nets* folder. For the same reason, *Physical*, *Spacing*, and *Same Net Spacing* worksheets do not appear. See the [Workbooks and Worksheets](#) figure on page 27 for information about the Nets folder.

Also, without a database with which to save constraint data, before exiting SigXplorer, or Constraint Manager, you must ensure that you save to a topology template to preserve your work.



You also use SigXplorer in the Constraint Manager flow to define custom measurements and custom stimulus. See [Analyzing for DRC-based Constraints](#) on page 119 for more information.

In SigXplorer, you simulate and analyze the topology. The following can be captured in a topology template:

- pin ordering (topology scheduling)
- termination strategy (and location on net)
- electrical constraints
- custom measurements and constrained custom measurements
- custom stimulus

Once exploration is complete, you save this information as a topology template (a `.top` file). This file represents the SigXplorer database. Constraint Manager is later used to import this information as an Electrical CSet.

Pin Scheduling

In Constraint Manager, you can select from the following pre-defined pin scheduling topologies:

- *minimum spanning tree*
- *star*
- *daisy chain*
- *source load daisy chain*
- *far-end cluster*

You select these from the *Wiring* worksheet of the *Routing* workbook. If you want to define your own pin schedules, you must manually wire the connections in SigXplorer and then export this information back to Constraint Manager (choose *File – Update Constraint Manager*).

Design Capture Phase

For information on using Constraint Manager in the Design Capture Phase, refer to the

- [Allegro Design Entry HDL – Constraint Manager User Guide](#)
- [System Connectivity Manager – Constraint Manager User Guide](#)
- [Capturing Design Constraints](#) chapter in the *Allegro Front-to-Back User Guide*.

Design Capture Phase (with System Connectivity Manager)

In the design capture phase, System Connectivity Manager provides a spreadsheet-based design environment. The spreadsheet-based interface is very effective while creating connectivity for designs with large pin count devices. While working with System Connectivity Manager, you use Constraint Manager to capture design constraints.



Tip

Enabling the *Transfer to/from Physical* button in the *Create Attribute Definition* dialog box ensures that user-defined properties flow between the logical and physical tools.

For more information, refer to . . .

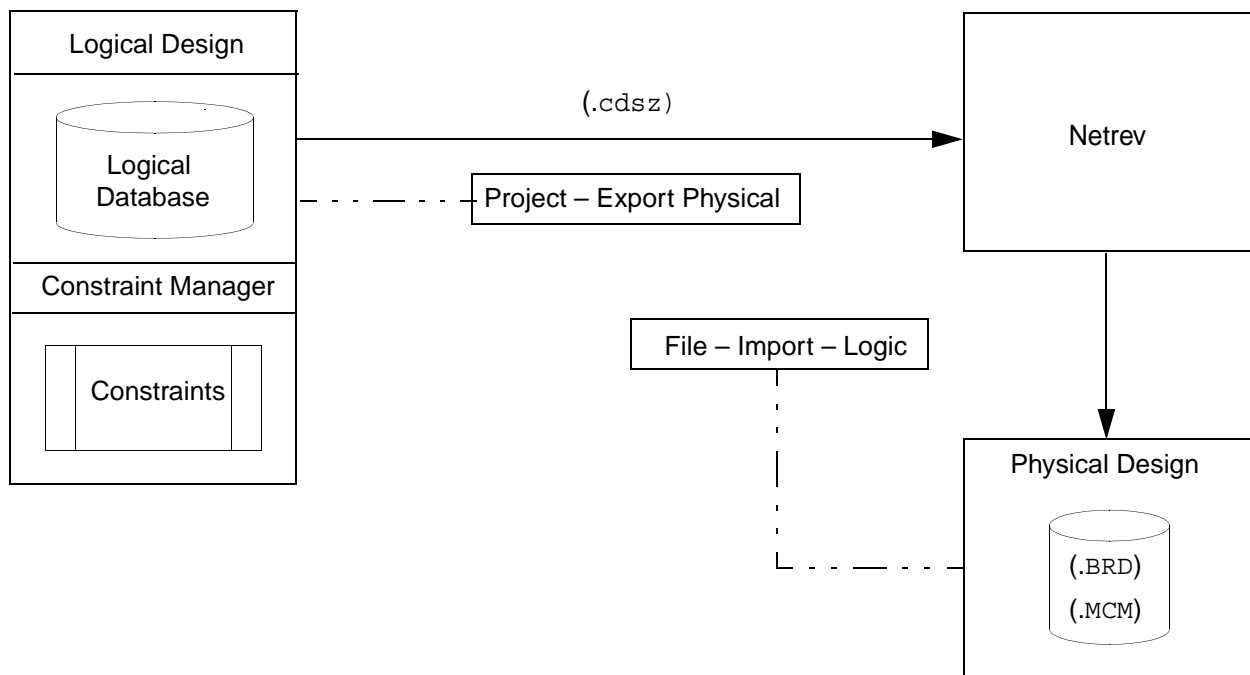
- [Front to Back Constraint Flow](#) on page 140.
- [Back to Front Constraint Flow](#) on page 142.
- [Working with Properties and Electrical Constraints](#) in the *System Connectivity Manager User Guide* for detailed information about how to capture design constraints while creating a design in System Connectivity Manager.
- [Working with Properties](#) in the *System Connectivity Manager User Guide* if you are using Constraint Manager as the property editor for System Connectivity Manager.

Front to Back Constraint Flow

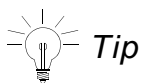
To create a physical layout for the design in System Connectivity Manager, the design data and constraint information is exported to the physical database of PCB Editor, using *Project – Export Physical*.

Export Physical extracts five package files — which communicate logic, part, pin, reference designator, and constraint information — and writes to the (.cdsz) file. This file is then used by Netrev for back-end processing, as illustrated in Figure 6-1. Refer to [Table 6-1](#) on page 141 for an explanation of package files (pst* .dat) used in the front to back flow.

Figure 6-1 Constraint Manager in Feed Forward Mode



The logical tools pass electrical, physical, and spacing constraint modifications (and classes) to the physical design tools (based on mode) for layers that exist in the the physical database, or a new, empty physical database seeded from information passed from a logical database.



Tip

See the [Allegro Design Entry HDL - Constraint Manager User Guide](#) for more information on *Overwrite Mode* and *Change Only* mode.

Allegro Constraint Manager User Guide

Using Constraint Manager with Other Tools Across the Allegro Platform

In the front-to-back flow, Design Editor generates the composite file, `pstdedb.cdsz`, which contains the following package files:

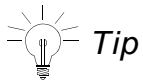
Table 6-1 Package Files

This file . . .	Contains . . .
<code>pstchip.dat</code>	Physical information for each type of symbol read from the <code>chips.prt</code> files and the physical parts tables, including electrical characteristics, such as pin direction and loading, logical to physical pin mapping, and voltage requirements. It defines the number of gates in each device, including gate and pin swapping information. This file also contains the name of the package and part symbol used to represent the device type in the physical layout (<code>JEDEC_TYPE</code>). Note: Device files are the third party equivalent of this file.
<code>pstxnet.dat</code>	A netlist that uses keywords (<code>net_name</code> , <code>node_name</code>) to specify the reference designators and pin numbers associated with each net. Constraints added to nets using Constraint Manager are written to the <code>pstcmdb.dat</code> file.
<code>pstxprt.dat</code>	Contains each physical package or part in the logic design along with its reference designator and device type. For packages or parts composed of multiple logic gates, the file identifies which gate was placed in which section of the package or part. Also contains attributes for parts and functions, and pin attributes specifically used for packaging. All other Pin attributes (constraints) are written to the <code>pstcmdb.dat</code> file.
<code>pstcmdb.dat</code>	Constraint and property information for the design.
<code>pstcmdb.dat</code>	The baseline used in determining changes to non-package related constraints and properties in the front-to-back flow.

Back to Front Constraint Flow

Import Physical is a prescribed set of processes that reconciles design data and constraints between physical- and logical databases.

The physical tools pass constraints to the logical database (based on mode). Physical and Spacing constraints, objects, and layers also make the transition to reconcile both databases.



Tip

See the [Allegro Design Entry HDL - Constraint Manager User Guide](#) for more information on *Overwrite Mode* and *Change Only mode*.

Import Physical calls Genfeed to extract six view files — which communicate component, part, function, pin, and constraint information and passes this file for front-end processing, as illustrated in Figure 6-2. Refer to Table 6-2 for an explanation of view files (**view.dat*) used in the back-to-front flow.

Figure 6-2 Constraint Manager in Feedback Mode

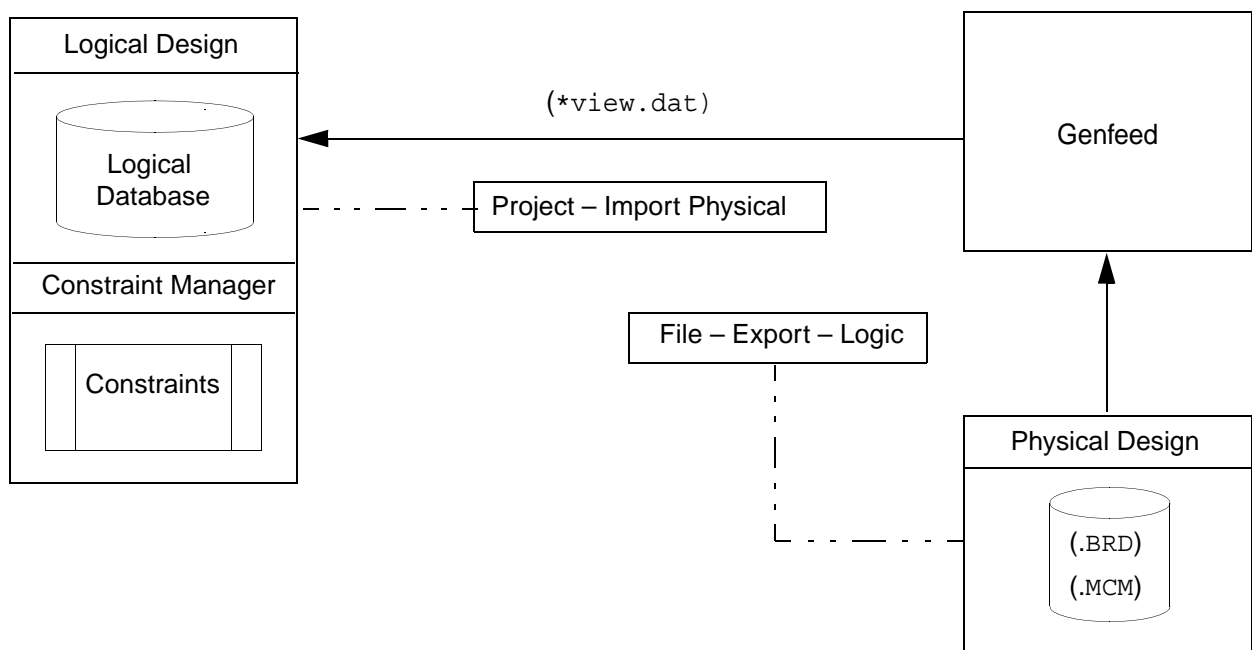


Table 6-2 View Files

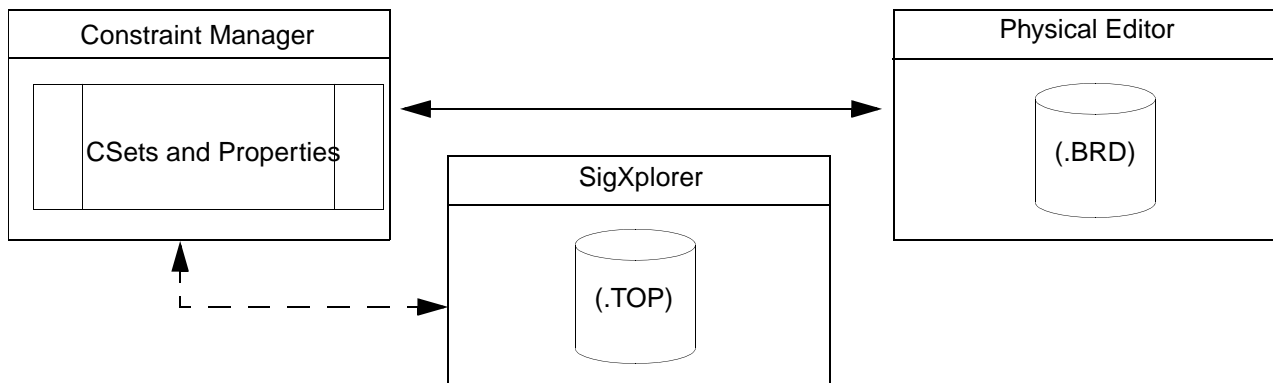
This file . . .	Contains . . .
compview.dat	Component information and properties, as defined in the physical tools.
funcview.dat	Function information and properties, as defined in the physical tools.
netview.dat	Connectivity information, as defined in the physical tools.
pinview.dat	Pin information and package-related properties, as defined in the physical tools
cmdbview.dat	Non-package related constraint and property information, as defined in the physical tools.
cmbcview.dat	The baseline used in determining changes to non-package related constraints and properties in the back-to-front flow.

Design Floorplanning and Implementation Phases

In the floorplanning and implementation phases, you focus on placement, routing, and manufacturing output. This section focuses on using Constraint Manager with back-end tools. For information on the front-to-back flow, you should also refer to [Design Capture Phase](#) on page 139.

Note: Constraint Manager, when launched from a Series L PCB editor, does not support topology exploration with SigXplorer.

Constraint Manager is used along with your physical editor to manage constraints. Constraint creation or modifications in Constraint Manager will automatically be synchronized with the board (.brd) database.



You can also use SigXplorer to perform simulations based upon the Electrical CSet's characteristics (pins, scheduling, models) of the net-related objects in your design. See [Using SigXplorer in the capture, floorplanning and implementation phases](#) on page 146 for information on using SigXplorer.

- Launch Constraint Manager from your physical editor (choose *Setup – Constraints – Constraint Manager*).
- Launch SigXplorer from Constraint Manager by selecting a net-related object (or an Electrical CSet) and choosing *Tools – SigXplorer*. You can also right-click and choose *SigXplorer* from the pop-up menu.

Allegro Constraint Manager User Guide

Using Constraint Manager with Other Tools Across the Allegro Platform

In the floorplanning and implementation phases, you use Constraint Manager to:

- Import reusable topology templates from SigXplorer. These map to Electrical CSets in Constraint Manager.

See [Importing ECSets](#) on page 102 for information on using reusing topology template files from SigXplorer.
- Consolidate individual Nets and Xnets into more easily-managed units such as buses and match groups.

See [Working with Constraint Objects](#) on page 49 for information on buses and match groups.
- Define bus, differential pair, net, Xnet, or pin pair constraints.

See [Working with Constraint Objects](#) on page 49 for information on objects in Constraint Manager.
- Define differential pairs.

See [Differential Pairs](#) on page 56.
- Define net-related constraint overrides, as appropriate.

See [Methods of Constraining Nets](#) on page 95 for information on overriding a constraint.
- Create CSets based on net-related objects such as buses, differential pairs, nets, and Xnets.
- Explore net topologies and schedule pins.

See [Pin Scheduling](#) on page 138 for information on pre-defined pin schedules.
- Audit CSets to resolve inconsistencies.

See [Audits](#) on page 106 for information on constraints and their assignments.
- Validate the design through design rule checks and analysis.

See [Chapter 5, "Topics in this chapter include"](#) for more information on validating constraints.
- Communicate layout changes to logical tools.

See [Design Capture Phase](#) on page 139 for more information on the front-to-back constraint flow.

- Open a partitioned design. Sections of the design that are partitioned are not editable, and open in Constraint Manager in read-only mode as indicated by cross-hatch shaded cells. You can analyze a partitioned design in Constraint Manager, but you cannot import constraints.

See [Objects Filter](#) in the *Constraint Manager Reference* for information on filtering partitioned designs.

See [Design Partitioning](#) in the *Allegro PCB and Package User Guide* for information on setting design partitions.

- Migrate constraint sets, from older, pre-14.0, databases into Electrical CSets used in Constraint Manager.

See [Topology Templates Audit](#) on page 108 for instructions.

Using SigXplorer in the capture, floorplanning and implementation phases

Note: Constraint Manager, when launched from a Series L PCB editor, or a logical editor, does not support topology exploration with SigXplorer and database synchronization.

Unlike in the exploration phase, where there is no board or netlist available, SigXplorer employs a different use model in the floorplanning and implementation phases.

Use SigXplorer to extract a net (or a net-related object such as a bus, differential pair, or match group) for topology exploration and constraint modification. The extraction can be routed (a trace in the PCB) or unrouted (a ratsnest). Used in this way, SigXplorer is aware of the electrical and physical characteristics of the net.

You also use SigXplorer in the Constraint Manager flow to define custom measurements and custom stimulus. See [Analyzing for DRC-based Constraints](#) on page 119 for more information.

- To extract a net-related object, select a net in the worksheet, then right-click and choose *SigXplorer* from the pop-up menu

Allegro Constraint Manager User Guide

Using Constraint Manager with Other Tools Across the Allegro Platform

SigXplorer will:

- Extract all electrical constraint and topology information from the selected object.

If the

- *Use Include Routed Interconnect* box is checked (choose *Tools – Options* in Constraint Manager), interconnect details (clines and vias) will be included.

Important

You cannot update Constraint Manager with a topology that contains traces or vias. You must condition the topology for Constraint Manager by choosing *Edit – Transform – for Constraint Manager* in SigXplorer.

- *Schedule Based on Routed Interconnect* box is checked (*Tools – Options* in Constraint Manager), the extraction derives connections from the user-defined net schedule (or from the default net schedule if none is specified). SigXplorer derives propagation delay and impedance from traces. Any unrouted segment derives its impedance and propagation velocity from the default settings.
- Selected object is a Bus or Differential Pair, the template will include information from the *first* Xnet or Net.

Refer to the [Tools – Options](#) command in the *Constraint Manager Reference* for more information on topology extraction.

- Display the appropriate ratsnest based upon the chosen topology schedule:
 - for pre-defined scheduling, this choice is the value of the RATSNEST_SCHEDULE property. See [Pin Scheduling](#) on page 138 for information on pre-defined pin schedules.
 - for user-defined scheduling, this choice is the value of the TEMPLATE property. See the online help for instructions on scheduling topologies.

Allegro Constraint Manager User Guide
Using Constraint Manager with Other Tools Across the Allegro Platform

Constraint Manager Your Way

Topics in this chapter include

- [Customizing the User Interface](#) on page 150
- [Customizing Worksheets](#) on page 151
- [Customizing Simulations](#) on page 154
- [Customizing Design Rule Checks](#) on page 159

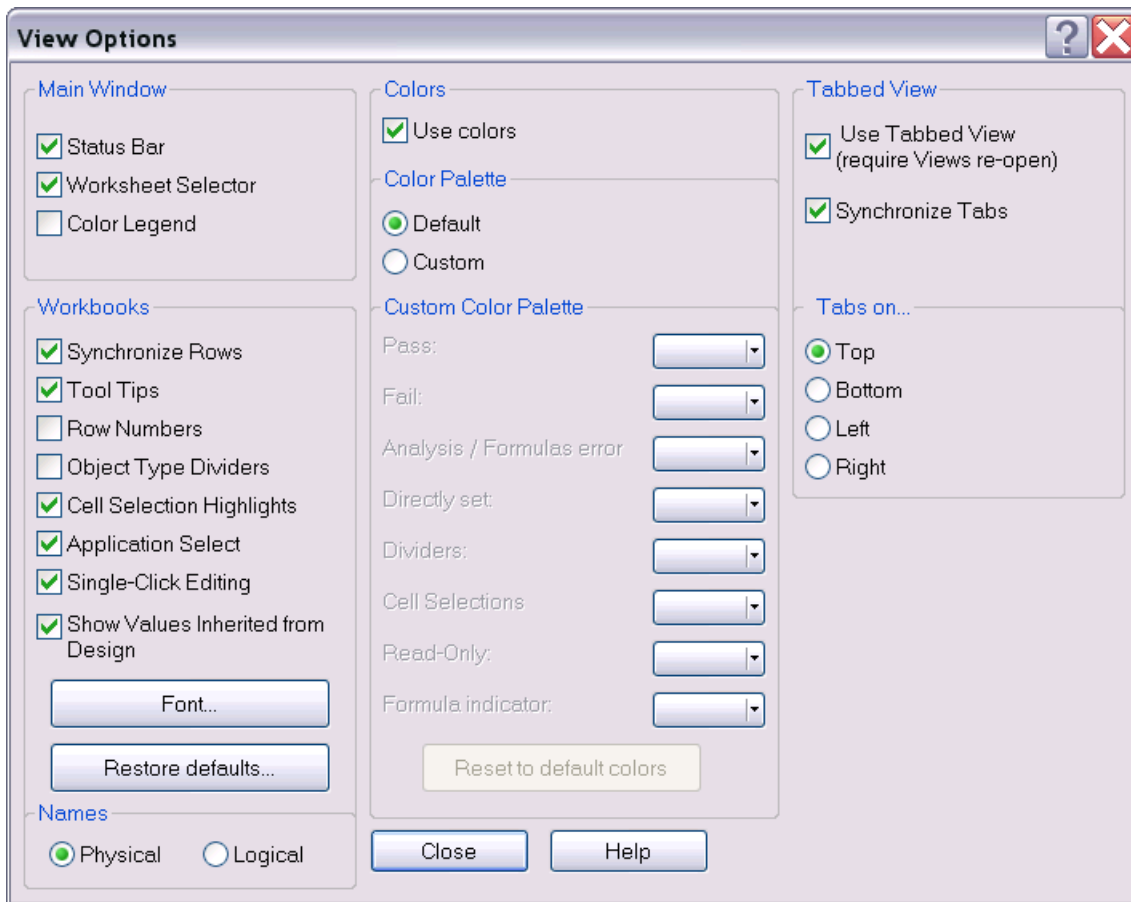
Customizing the User Interface

Constraint Manager gives you control of its user interface, including visibility options, user-definable keyboard shortcuts and user-definable toolbars.

Customizing Visibility

Constraint Manager affords you many options (choose View – Options.) to personalize your view of the user interface.

Figure 7-1 Setting View Options



For information on field descriptions, see the View – Options command in the *Allegro® Constraint Manager Reference*.

Customizing Keyboard Shortcuts

For information on how to assign your own accelerator keys to commands, or how to reassign Constraint Manager's default assignments, see the [Tools – Customize Shortcut Keys](#) command in the *Constraint Manager Reference*.

Customizing Toolbars

You can undock the toolbar (or a section of the toolbar) and reposition it in your workspace. You can also rearrange command icons within the toolbar strip and you can create your own toolbar, drawing from existing *File*, *Edit*, *View*, and *Filter* commands.

See the [Tools – Customize Toolbar](#) command in the *Constraint Manager Reference* for more information.

Customizing Worksheets

You can add user-defined or pre-defined attributes to Constraint Manager's default worksheets, and you can create your own customized workbooks and worksheets.

Important

Historically, PCB Editor uses the term *Property*, Constraint Manager uses the term *Constraint*, and Design Entry HDL uses the term *Attribute*. This chapter uses the term *Attribute* throughout to describe a *Constraint*, *Property*, or *Attribute*, which is validated or not.

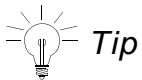
Each net-level customized worksheet that you add has an *Objects* column and a *Referenced CSet* column (you can hide the latter). Customized worksheets do not contain *Actual* and *Margin* columns.

Each attribute that you add to a worksheet requires a new column. New columns appear to the right of the active worksheet. As with overrides, Constraint Manager renders customized workbooks, worksheets, and columns with a blue tint in the *Worksheet Selector*.

You can add, rename, and delete workbooks, worksheets, column headers and columns, and you can control their visibility. You can also export worksheets that you define, and you can import customized worksheets from a different design. Although you can hide columns of default worksheets, you cannot delete them. Also, you cannot delete predefined worksheets or predefined attributes.

Customize Mode

In addition to a *Browse* mode, Constraint Manager has a *Customize* mode, which uses special icons in the *Worksheet Selector* to assist you in differentiating among predefined workbooks, worksheets, and columns from those that you add in *Customize* mode.



Tip

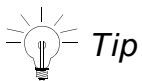
See the [Tools – Customize Worksheet](#) command in the *Constraint Manager Reference* for detailed information about *Customize* mode, including adding columns, adding workbooks, adding worksheets, and drag-and-drop support for column positioning.

Note: You can also access *Customize* mode by right-clicking in the *Worksheet Selector* and choosing *Customize Worksheet*.

[Figure 7-2](#) on page 153 depicts the expanded *Impedance* worksheet of the *Routing* workbook. Notice that default columns have a neutral, grayish tint except for the hidden *Referenced CSet* column, which is represented by the silhouette of a circle. Also note the column's superheader label (*Single-line Impedance*) depicted with a rectangle over a circle.

In the same figure, notice the *My Impedance* worksheet of the *My Routing* workbook. The workbook, worksheet, and all columns have a bluish tint. Also note the column's superheader (*My Single Line Impedance*).

The columns in the *My Single Line Impedance* worksheet are the same as those in the *Single Line Impedance* worksheet except for the difference in column labels: *Which Net* substitutes for *Target* and *Allowable Deviation* substitutes for *Tolerance*. Also, the *Electromotive Force* column label identifies the added column for the *Voltage* attribute.



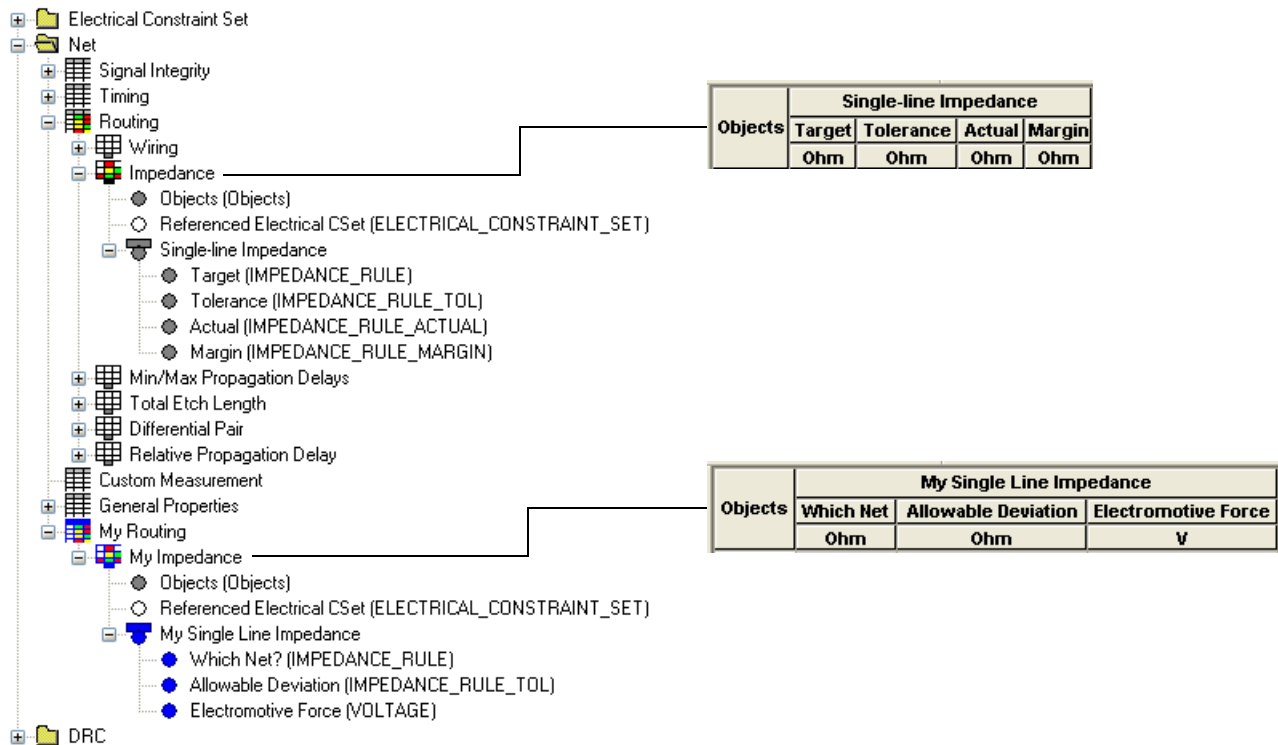
Tip

A column superheader icon appears as a rectangle over a circle; a column icon appears as a circle. A predefined column icon is gray; a customized column icon is blue; a hidden column icon is a silhouette of the column icon.

Allegro Constraint Manager User Guide

Customizing Worksheets

Figure 7-2 Worksheet Selector icons in Customize mode



User-defined Properties

You use a user-defined property to capture a characteristic of an object. Constraint Manager does not perform design rule checks or analysis on this property; it facilitates communication of the design intent to down-stream tools with which you may want to use to manipulate the objects associated with these properties.

You create user-defined properties in the PCB- or Package-Editor using the *Setup – Property Definitions* command, or in SigXplorer using the *Set – Constraints – User-Defined* command.

Note: You can also create a validated user-defined property directly in Constraint Manager. See the [Constrained Custom Measurements](#) on page 155.

Allegro Constraint Manager User Guide

Customizing Simulations

For Constraint Manager to report or display a user-defined property, you must add a column in any net-related worksheet and choose the desired property from a dialog box. Constraint Manager then adds a column to the far right of that worksheet with the property name as its column label. There are no *Actual* and *Margin* cells associated with user-defined properties. Additionally, the property appears under the *Electrical CSet* folder, in the *User-Defined* worksheet (and in the *Signal Integrity/Timing/Routing* worksheet) of the *All Constraints* workbook (see [Figure 7-3](#) on page 157).

To add a column, enable customization with the [Tools – Customize Worksheet](#) command, then Right-click and choose *Add Column* from the pop-up menu.

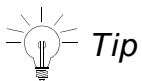
Customizing Simulations

Constraint Manager supports custom constraints, custom measurements, and custom stimulus when used with the SigXplorer legacy flow.

Note: Constraint Manager, when launched from a Series L PCB editor or OrCAD PCB Editor, does not support custom constraints, custom measurements, or custom stimulus. The *Custom Measurements* tab in the *Analysis Modes* dialog box (choose *Analyze – Analysis Modes*) as well as the *Custom Measurements* workbook is hidden.

Without Constraint Manager, you would have to extract a net from a board layout, define any custom constraints, custom measurements, or custom stimulus in SigXplorer, then re-apply the changes back to the net in the board layout. Because you have to do this one net at a time, this can be error prone and tedious. Because Constraint Manager has a global view of all nets in a board layout, application of custom constraints, custom measurements and custom stimulus is simplified.

You do not define custom constraints, custom measurements or custom stimulus in Constraint Manager, you only assign, manage, and analyze them. You define them in SigXplorer, save them as a topology file, and import them into Constraint Manager as an *Electrical CSet* (Choose *File – Import Electrical CSet*) or refresh the current *Electrical CSet* reference (Choose *Tools – Update Topology*). Any net-related object that references that *Electrical CSet* will inherit any custom constraint, custom measurement or custom stimulus data that was captured in that *Electrical CSet*.



By default, custom measurements are not included with an imported *Electrical CSet*. To override this behavior, you must enable the *Update existing or create new Custom Measurement worksheet* option. See the [File – Import – Electrical CSet](#) command in the *Constraint Manager Reference*.

In this way, an *Electrical CSet* performs triple-duty. Not only does an *Electrical CSet* contain pre-defined and custom constraints, it can also contain custom measurements and custom stimulus.



Because you can assign only a single *Electrical CSet* to a net-level object, you must define any constraint data along with custom measurement and custom stimulus in that same *Electrical CSet*.

Working with Custom Constraints

In addition to its broad set of pre-defined constraints, Constraint Manager supports constrained custom measurements.

Constrained Custom Measurements

You use constrained custom measurements (and custom stimulus) to specify your own constraints. These constraints differ from user-defined properties in that Constraint Manager can validate them through design rule checks and analysis.

You create constrained custom measurements in SigXplorer in the same way as you define unconstrained custom measurements, using the measurements expression editor. When you choose *None* as the constraint type, SigXplorer creates an unconstrained custom measurement. When you choose *minimum*, *maximum*, *min-max*, or *target: tolerance* as the constraint type, SigXplorer creates a constrained custom measurement, which is, in effect, a user-defined constraint.

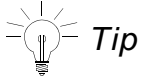


To invoke SigXplorer's *Measurement Expression Editor*: (1) click the *Measurements* tab; (2) right-click on the *Custom* tab; and (3) select *Add*.

Constrained custom measurements from SigXplorer appear under the *Electrical CSet* folder, in the *User-Defined* worksheet of the *All Constraints* workbook, as well as under the *Net* folder in the *Custom Measurements* workbook (see [Figure 7-3](#) on page 157). *Actual* and *Margin* cells associated with constraint also appear.

Use Model

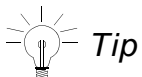
Typically, you will select a net object in Constraint Manager, right-click, then choose *SigXplorer* from the pop-up menu. The net's topology will then appear in SigXplorer where you can define the necessary custom constraints, or custom measurements and custom stimulus.



Consult the *Allegro® SI SigXplorer User Guide* for information about how to define custom constraints, custom measurements, and custom stimulus.

Next, you choose *File – Update Constraint Manager* to export the topology file from SigXplorer. The corresponding *Electrical CSet* is refreshed in Constraint Manager and all net-related objects that reference the *Electrical CSet* will inherit the custom constraints or custom measurements and custom stimulus that you just defined, along with any electrical constraints that were captured in the *Electrical CSet* before they were exported from Constraint Manager to SigXplorer.

Note: If the object that you extracted from Constraint Manager into SigXplorer does not reference an *Electrical CSet*, when you choose *File – Update Constraint Manager* from SigXplorer, the topology file is imported into Constraint Manager as an *Electrical CSet* with the same name as the extracted object, unless you chose *File – Save As*. You can then associate that *Electrical CSet* with other net objects (choose *Objects – Electrical CSet Reference*), and rename the *Electrical CSet* if you want to (choose *Objects – Rename*).



Constraint Manager retains the analysis modes settings that you define in the *Electrical CSet* when you export the topology to SigXplorer. See [Analyzing with Custom Measurements and Custom Stimulus](#) on page 158 for information about analysis modes.

Managing Custom Measurements

Once defined and imported from SigXplorer, custom measurements and constrained custom measurements populate an *Electrical CSet* under the *Custom Measurements* workbook in the *Net* folder. Each set of custom measurements or constrained custom measurements (an *Electrical CSet*) appears as an individual worksheet. Each custom measurement appears as a column in the worksheet (see [Figure 7-3](#) on page 157).

Allegro Constraint Manager User Guide

Customizing Simulations

Figure 7-3 User-defined Constraints and Custom Measurements

The screenshot displays the Allegro Constraint Manager interface. On the left is a tree view of the project structure, including 'Electrical Constraint Set', 'Net', and 'Custom Measurement'. The 'Custom Measurement' folder is expanded, showing items like 'PRDP_IN_ECSET (tommytechwriter)', 'CUST_MEZ (tommytechwriter)', 'CONSTRAINED_CUSTOM_MEZ (tommytechwriter)', and 'ANOTHER_CUST_MEZ (tommytechwriter)'. A red arrow points from the label 'Constrained Custom Measurement' to this folder.

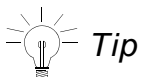
On the right, three data tables are shown:

- Electrical Constraint Sets: All Constraints**: A table with columns 'Objects', 'MY_PROP', and 'CONSTRAINED_CUST_MEZ_MAX'. It lists constraints for the 'tommytechwriter' system, including 'ANOTHER_CUST_MEZ', 'CONSTRAINED_CUSTOM_MEZ' (with a value of 6.000), 'CUST_MEZ', and 'MY_PROP_IN_ECSET'.
- Nets: Routing**: A table with columns 'Objects', 'MY_PROP', and 'CONSTRAINED_CUST_MEZ_MAX'. It lists constraints for 'tommytechwriter' on nets NET1, NET2, NET3, and NET4. NET3 has a value of 6.000.
- Nets: Custom Measurement**: A table with columns 'Objects', 'Referenced Electrical CSet', 'DieNoiseMarginHigh', 'DieNoiseMarginLow', 'DieOvershootHigh', and 'DieOvershootLow'. It lists constraints for 'tommytechwriter' on nets NET1 through NET4, each with a referenced electrical CSet.

At the bottom, three labels with red arrows point to specific elements:

- 'Custom Measurement Workbook' points to the tree view.
- 'Custom Measurement Worksheets' points to the 'Nets: Custom Measurement' table.
- 'User-defined Constraint (property)' points to the 'CONSTRAINED_CUSTOM_MEZ' property in the 'Electrical Constraint Sets' table.

Constraint Manager maintains custom measurement and custom stimulus associations—*Electrical CSet* to object—as well as analyzed results from one session to the next.



Tip

When you select an *Electrical CSet* in the *Referenced Electrical CSet* column of the *Custom Measurements* worksheet, and then right-click and choose *SigXplorer* from the pop-up menu, that *Electrical CSet* is exported from Constraint Manager to SigXplorer as a topology with custom measurements and custom stimulus intact. Choosing *File – Update Constraint Manager* from SigXplorer then refreshes the custom measurements in Constraint Manager with any changes.

Analyzing with Custom Measurements and Custom Stimulus

Analyzing a net-related object with a custom constraint or custom measurement involves the same steps as described in [The Analysis Process](#) on page 123. Additionally, you must follow these steps:

➤ **Specify Analysis Settings**

Custom measurements apply only to *reflection* simulations. You specify the simulation type in the *Analysis Settings* dialog box. If you have custom stimulus defined in the *Electrical CSet*, it too must be enabled for analysis. See the [Analysis Settings](#) on page 123 for more information on how to specify the simulation type and how to enable custom stimulus.

➤ **Enable Custom Measurements**

You enable custom measurements through the *custom measurements* tab of the *Analysis Modes* dialog box (choose *Analyze – Analysis Modes*). Measurements appear as children in a tree structure with the parent object representing the *Electrical CSet* that contains the set of custom measurements.

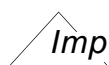
Note: Constraint Manager, when launched from Allegro PCB Board L Series or OrCAD PCB Editor, do not support custom measurements or custom stimulus.

The checkbox adjacent to the parent object also serves as a toggle switch for all measurements in the *Electrical CSet*: *all on* (when checked) or *all off* (when unchecked). Only checked measurements appear in analysis results (see [Analysis Results](#) on page 128 for more information).

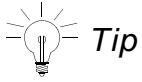
➤ **Analyze Custom Measurements**

You do not analyze a custom constraint or custom measurement; rather, you analyze the object that references an *Electrical CSet* which contains custom constraints or custom measurements.

The scope of a net object that contains a custom measurement can range from a pin-pair to a bit of a bus to an entire bus. Once you have (1) imported an *Electrical CSet* that contains custom measurements, (2) assigned the *Electrical CSet* to a net object, (3) specified analysis settings, and (4) enabled custom measurements, you then select the net object and choose *Analyze – Analyze* (or right-click and choose *Analyze* from the pop-up menu).

 **Important**

As many net objects can have the same custom measurement, you must click the appropriate tab to access the object that you want to analyze.



You can specify analysis settings, DRC modes, custom measurements, and desired reports from a single dialog box (choose Objects – Report).

Customizing Design Rule Checks

This section presents different methods used to customize DRCs, including

- Formulas
- User-defined Constraints
- User-defined Predicates
- User-defined Measurements

You can choose from pre-defined predicates and pre-defined measurements or you can create your own to address your unique design requirements. This section complements the procedures and reference information of the Edit – Formula command in the *Constraint Manager Reference*.

Formulas

To meet your unique constraint requirements, you can extend constraints and properties (pre-defined and user-defined) with formulas. A formula is required when a static value is not sufficient and a relatively simple calculation is required to compute the desired constraint or property value.

You can build a formula using the following methods, alone or in combination:

- Cell selection, mixed with operands
- Pre-defined and user-defined predicates
- Programmatically, with the Cadence SKILL scripting language

Each method of defining a formula is progressively more complex, yet yields more power and flexibility in using them to constrain your design.

Inserting a formula in a cell

Click in any writable cell, and then right-click and choose *Formula* from the pop-up menu (or choose *Edit – Formula*). This invokes the *Single-line Editor*, as shown in Figure 7-4.

If you defined a formula using the *Multi-line Editor*, it will be called if you need to subsequently edit a formula. See [Working with the Multi-line Editor](#) in the *Constraint Manager Reference*. Constraint Manager remembers which editor that you last used in building a formula. For easy identification, Constraint Manager renders the far end of a cell that contains a formula with a thin, red, vertical stripe.

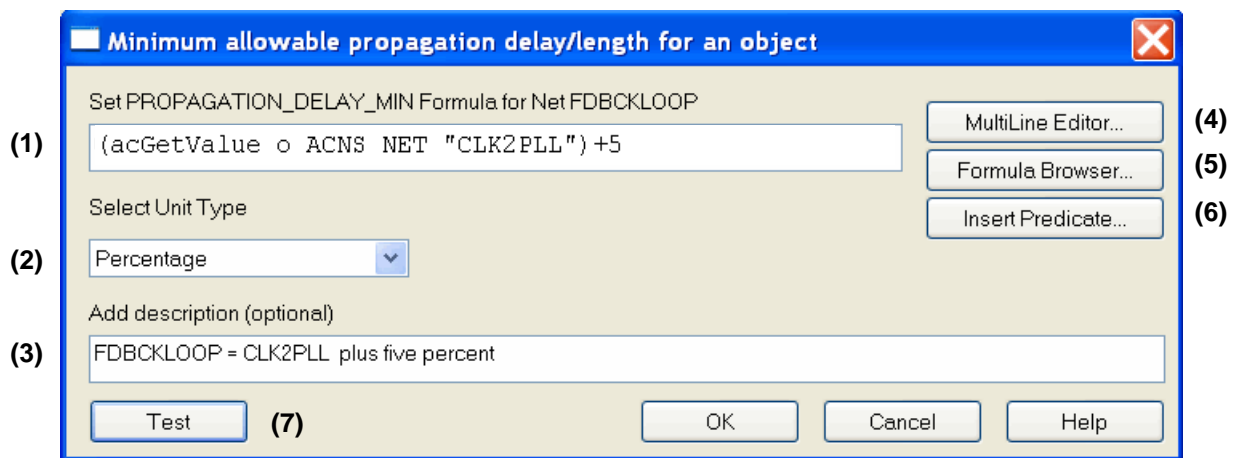
The Single-line Editor

You use the *Single-line Editor* to build simple formulas, which may contain user-defined constraints, properties, and predicates.

In Figure 7-4, Item (1) is where you type in your formula. You can use point-and-click to add cell addresses to the formula. Item (2) is the *Unit Type*, which varies depending on the cell being modified by the formula. You enter a description of the formula in Item (3). Item (7) exercises the formula. You should always test your expressions for proper syntax and function. Item (4) launches the *Multi-line Editor* used to create formulas programmatically (see [Programmatic Formulas](#) on page 164).

Once you create formulas, Item (5), they are available in the formula browser for use elsewhere in the design. Item (6) is for adding predicates to the formula (see [Pre-defined Predicates](#) on page 163).

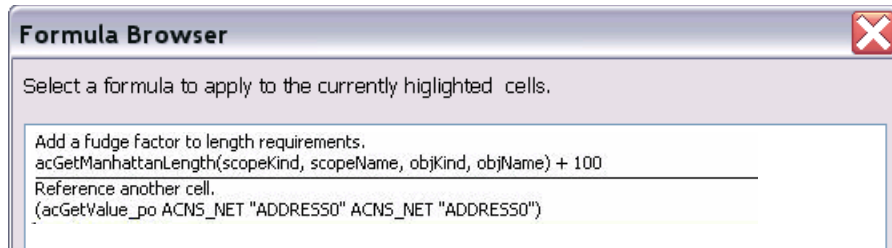
Figure 7-4 Single-line Editor



A Library of Formulas

The *Formula Browser* lets you view all the formulas in the design and assign them to specific cells.

Figure 7-5 The Formula Browser



Cell selection and operands

The cell selection and operand method provides for point-and-click cell selection, mixed with operands to build a formula. Clicking within a cell adds the code required to access that cell in the formula. For example, `(acGetValue_o ACNS_NET "CLK2PLL")+5` results from selecting a cell for an another object in the worksheet, in the same column as the one selected for Formula editing, and then typing `+5`.

Cell selection automatically inserts the appropriate predefined predicate to get the value for the selected cell. The inserted predicates will differ based upon what information is needed to retrieve the value. For example, the same object in a different cell only needs the name of the cell's attribute because the object being edited is considered the default object.

Calculating a Formula

Constraint Manager keeps track of a formula's dependencies on other cells or database objects. For example, the value in a cell or the etch length of a net. When one of those dependencies changes, the formula is known to be out of date or stale. If automatic formula calculation is on, out of date formulas recalculate their new values as needed. If automatic formula calculation is off, out of date formulas are indicated by the background of the cell turning yellow.

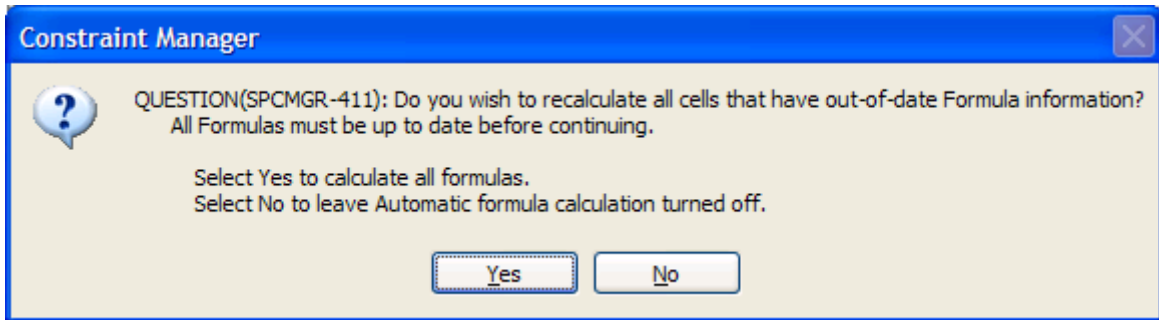
You can use the *Edit – Calculate* or *Edit – Calculate All* menu commands to bring these formulas back up to date.

To turn the automatic formula calculation on or off, select or deselect the *Automatic formula calculation checkbox* in the Options dialog box (choose *Tools – Options*).

Allegro Constraint Manager User Guide

Formulas

When the automatic formula calculation function is set from Off to On, the following message is displayed



You can also review or modify the dependencies using the Dependencies dialog box (choose *Edit – Dependencies*).

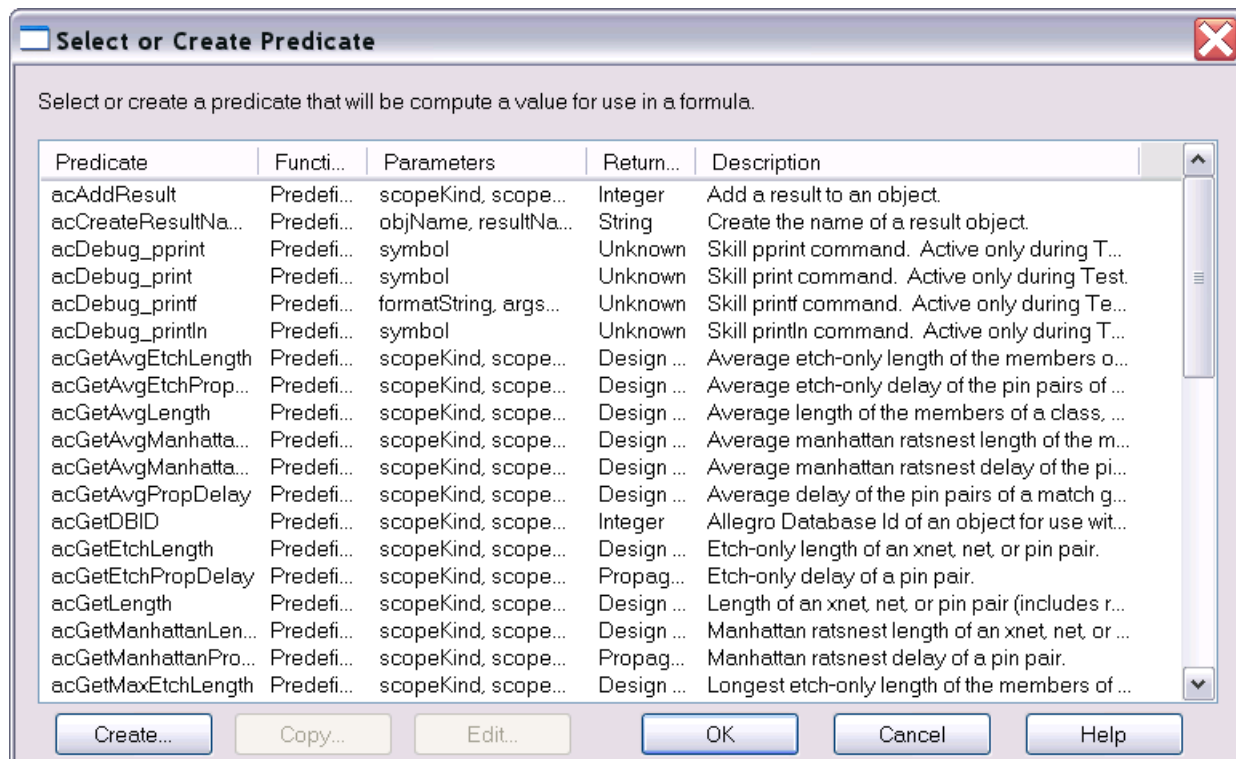
See [Automatic Formula Calculation](#) in Allegro Constraint Manager Reference for more information.

Pre-defined Predicates

Predicates are functions that return data (usually a single value). They work to access design information to enhance the functionality of formulas, other predicates, and measurements (see [User-defined Predicates and Measurements](#) on page 170). You can choose from pre-defined predicates and you can create your own. See [Edit – Formula](#) in the *Constraint Manager Reference* for a list of pre-defined predicates, their supported parameters, and their function.

You access the *Select or Create Predicate* dialog box (see [Figure 7-6](#)) by clicking *Insert Predicate* (see Item 6 in [Figure 7-4](#) on page 160).

Figure 7-6 Select or Create Predicate dialog box



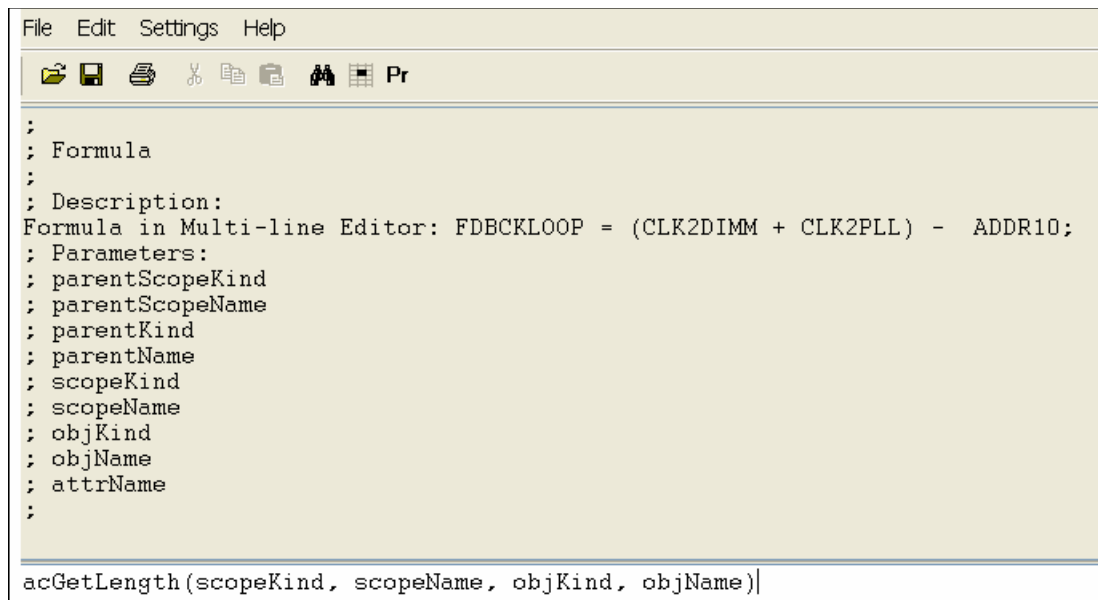
Note: See [Predicate Parameters](#) on page 173 for column definitions.

In addition to a broad collection of pre-defined predicates, you can create your own (see [User-defined Predicates and Measurements](#) on page 170).

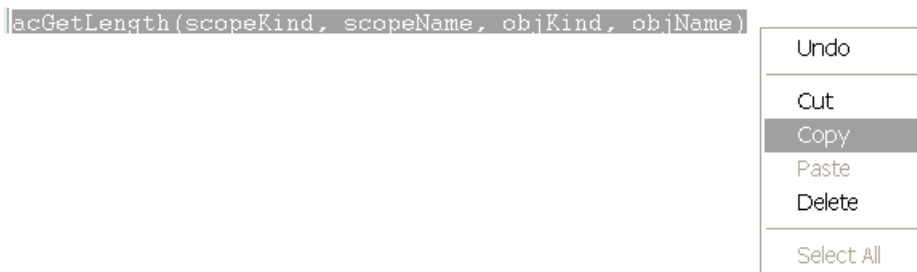
Programmatic Formulas

The *Multi-line Editor* (see Item 6 in [Figure 7-4](#) on page 160) allows for the most flexibility when adding programming constructs to formulas. For usage information, see [Using the Multi-line Editor](#) in the *Constraint Manager Reference*.

The top of the *Multi-line Editor* shows the formula's description and the list of parameters; the bottom shows the formula that you are building.



You can type any text in the *Multi-line Editor* to build your formula. Common editor functions, such as *Copy* and *Paste*, are selectable from the context menu (right-click).



Allegro Constraint Manager User Guide

Formulas

You can improve the formula by assigning each predicate to a variable and wrapping the formula in a `let` statement. The `let` statement also protects the variables from being influenced by global access.

```
let( (a b c)
  ;
  ; Set up lengths to be used in the calculation
  ;
  a = acGetLength(scopeKind, scopeName, objKind, "CLK2PLL")
  b = acGetLength(scopeKind, scopeName, objKind, "CLK2DIMM")
  c = acGetLength(scopeKind, scopeName, objKind, "ADDR<10>")
  ;
  ; Calculate the value to be returned by the formula
  ;
  (a + b) - c
)
```

Support for Online Formula

Constraint Manager determines and stores dependency information for all the Advanced Constraints formulas as they are created or loaded for the first time. The dependency information stored with the formulas such that a formula is not recreated each time the design is closed and reopened.

Dependency information is used to update the status of calculated values and indicates when they have become stale.

Some cases where calculated values can become stale (out of date) are:

- A formula on a cell that references the value in another cell.
- A formula that calls a predicate that returns some information about a database object.

When values are changed in Constraint Manager, or if database objects are updated in PCB Editor, Constraint Manager determines which values are out of date, and recalculates them when necessary.

Note: Launching Constraint Manager marks all formula's stale. This is indicated by cell with formula's colored yellow. When you run *Calculate* on a stale cell, the cell value is updated and cell color is cleared. If you run *Calculate All*, all the cells in all worksheets are updated and the yellow color is cleared.

See [Options Dialog Box](#) in Allegro Constraint Manager Reference for more information.

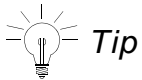
Testing and Debugging Formulas

You can use Constraint Manager's test environment to debug formulas. In a *debug* session, Constraint Manager temporarily makes changes to the database; however, these changes will not be saved once you exit the test environment.

You enter the test environment from the

- *Multi-line Editor* (choose *File – Test*)
- *Single-line Editor* (click *Test*). See Item 7 in [Figure 7-4](#) on page 160.

Formula testing produces a log file similar to the log file used for testing user-defined predicates and user-defined measurements (see [Figure 7-12](#) on page 177). See the [Multi-line Editor](#) commands of the *Edit – Formula* command in the *Constraint Manager Reference* for information on how to interpret these log files.



Constraint Manager supplies many pre-defined predicates (look for `acDebug_print*`), which you can use to debug your formulas. See [Pre-defined Predicates](#) in the *Constraint Manager Reference*.

User-defined Constraints

In addition to its broad set of pre-defined constraints, Constraint Manager lets you define constraints to meet your unique design requirements. A user-defined constraint is required to capture a unique design requirement, which is not represented in the default constraint system. User-defined Constraints also specify how to validate a new column.

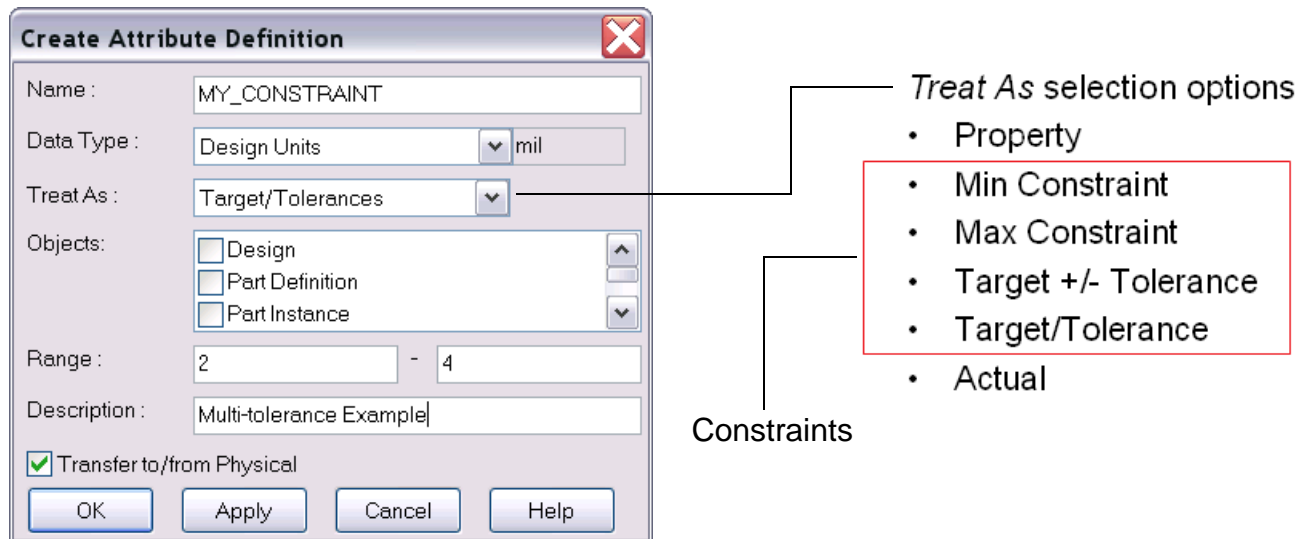
Important

Unlike pre-defined constraints, user-defined constraints analyze only within Constraint Manager and not in a PCB or package editor.

A user-defined constraint requires its own column. For information on adding custom workbooks, worksheets, columns, and attributes, see

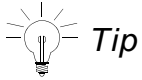
- [Customizing Worksheets](#) on page 151.
- The [Tools – Customize Worksheet](#) command in the *Constraint Manager Reference*.

Figure 7-7 Adding a column and a constraint



Allegro Constraint Manager User Guide

User-defined Constraints



Tip

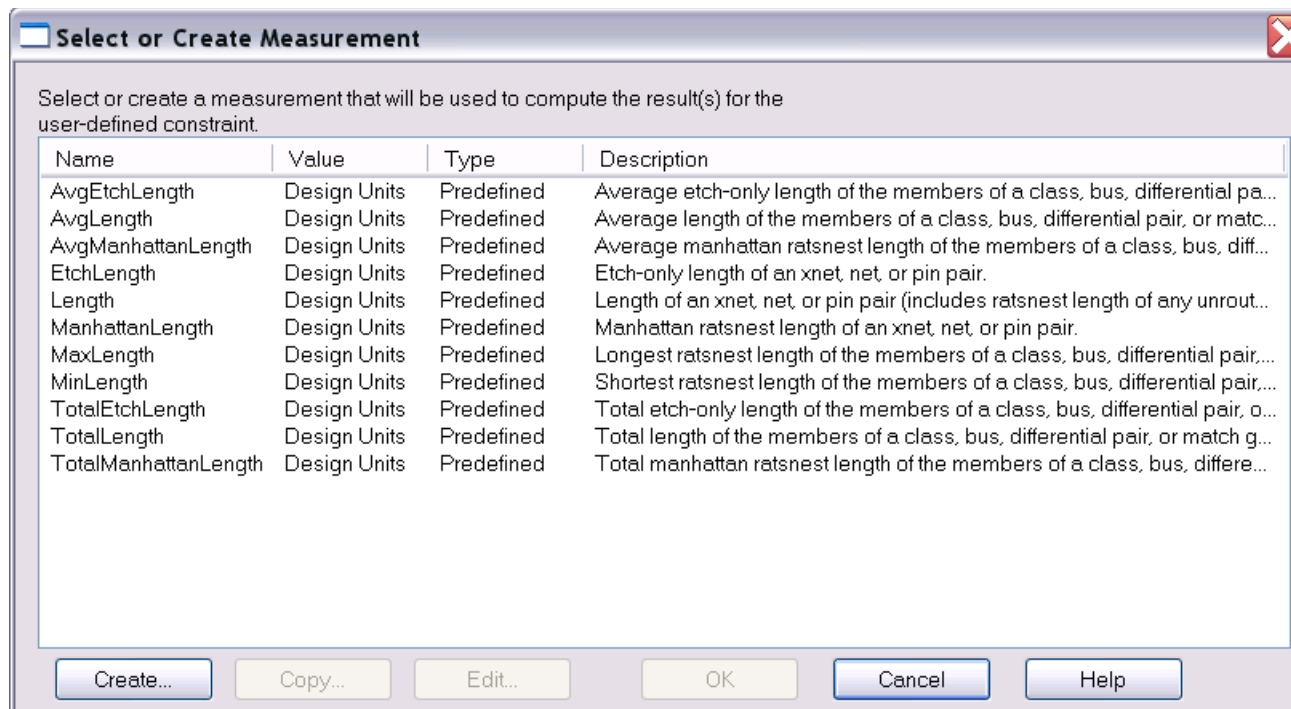
The *Treat As* drop-down menu in the *Create Attribute Definition* dialog box (see [Figure 7-7](#) on page 168) is where you specify the new user-defined attribute's type. This can be a property, a constraint, or an actual.

- ❑ *Property* specifies a user-defined property
- ❑ *Min Constraint, Max Constraint, Target +/- Tolerance, Target/Tolerance* specifies a user-defined constraint, and a corresponding *Actual* and *Margin*

Target/Tolerance allows for separate plus and minus tolerance values; *Target +/- Tolerance* allows for a single tolerance value

- ❑ *Actual* specifies a single column that is linked to a measurement and used to display the measurement results. An *Actual* can be part of a user-defined constraint.

When you click *OK* to accept one of the above constraint types, the system automatically creates a related *Actual* and *Margin* for the constraint. The related *Actual* also needs an associated measurement to provide results when the constraint is analyzed. As such, the system presents the *Select or Create Measurement* dialog box.



Note: The list of measurements is based on the *Data Type* field in the *Create Attribute Definition* dialog box (see [Figure 7-7](#) on page 168). If the pre-defined measurements are not what you need, you can create your own (see [User-defined Actuals](#) on page 178).

User-defined Predicates and Measurements

Constraint Manager has many pre-defined, readily-usable, predicates and measurements and it provides the capability to augment them with predicates and measurements that you build to meet your unique design requirements. Predicates function as the building blocks of Formulas, Measurements, and other Predicates.

User-defined Predicates

Predicates are functions that return data (usually a single value). They work to access design information to enhance the functionality of formulas, other predicates, and measurements.

Predicates are more flexible than formulas or measurements. You can define any parameters in any order and have any return type that you want. When you use a predicate you have to match its use to its definition—all parameters have to match the definition and its return value must be appropriate for the object type.

You can write a simple predicates to

- sort a list of numbers
- find the smallest number
- open, close, or write to a log file
- perform an algebraic calculation

You can write a complex predicate to

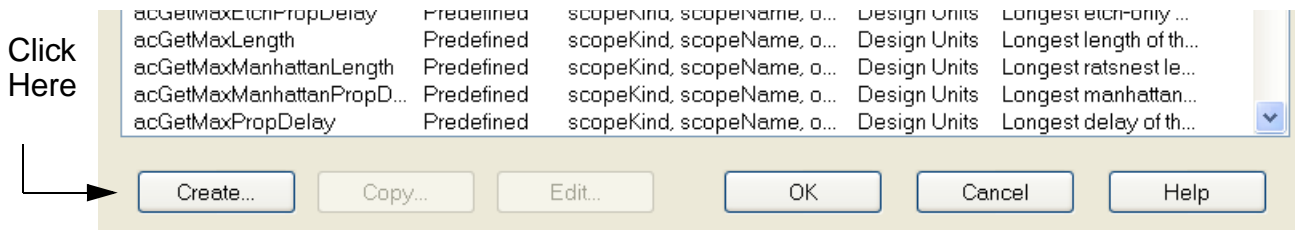
- determine the overlap of two physical objects
- return the distance of an object to a plane
- return a list of database objects that match a specific criteria
- find the closest via to a cline

The road to a predicate is through both formulas and user-defined measurements. Once you add a formula (see [Inserting a formula in a cell](#) on page 160), choose *Insert Predicate* (see Item 6 in [Figure 7-4](#) on page 160).

Allegro Constraint Manager User Guide

User-defined Predicates

- In the *Select or Create Predicate* dialog box, choose *Create*.

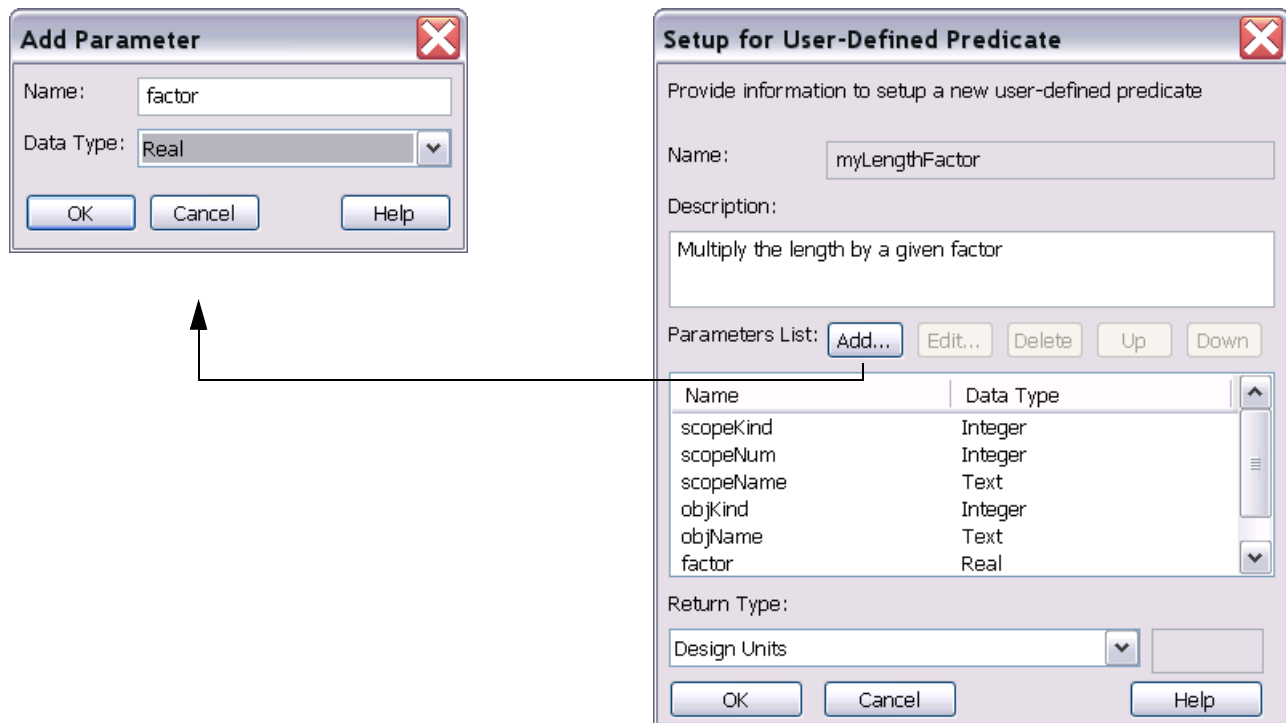


This invokes the *Setup for User-Defined Predicate* dialog box, as shown in Figure 7-8.

Defining a predicate

You supply a name, a description, a list of parameters, and a return type (boolean, integer, real, or text). Parameters are the conduit between a predicate and a formula (see [Predicate Parameters](#) on page 173).

Figure 7-8 Defining a predicate

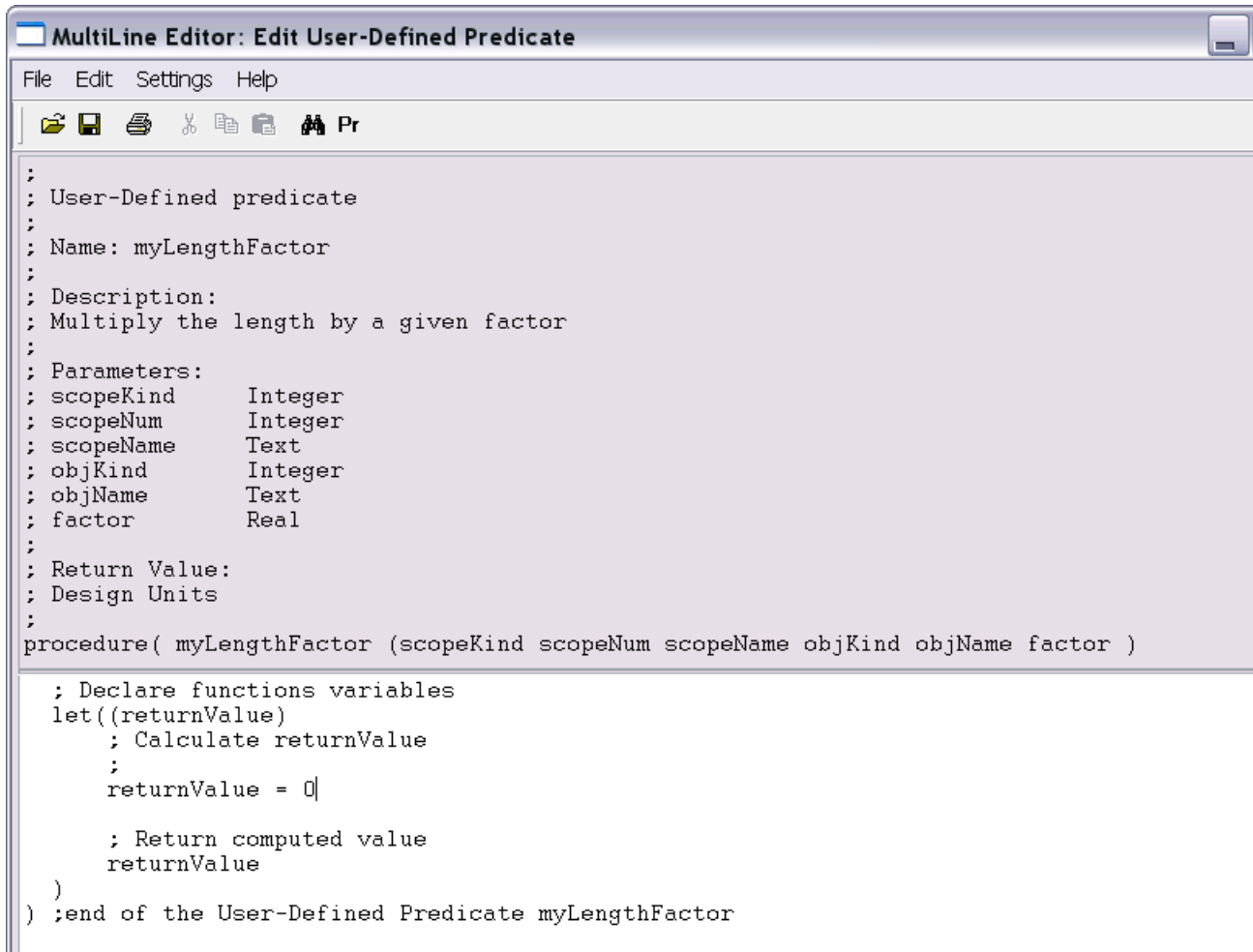


- Once you complete all fields and click *OK*, the *Multi-line Editor* appears (see [Figure 7-9](#) on page 172).

Allegro Constraint Manager User Guide

User-defined Predicates

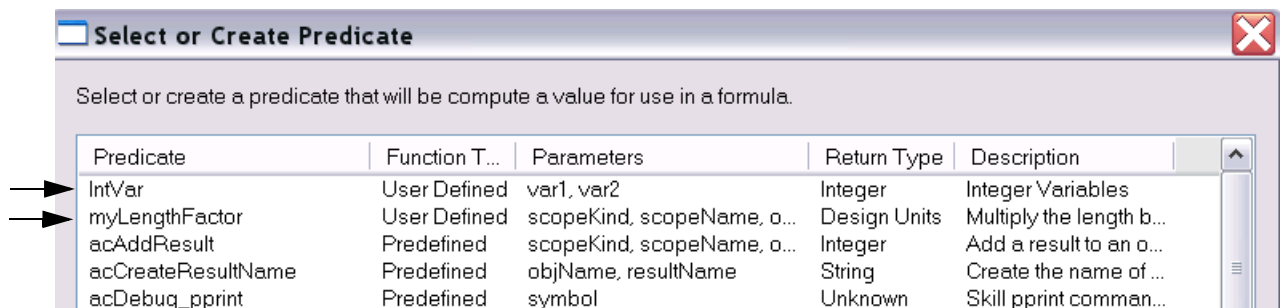
Figure 7-9 Editing a predicate



```
MultiLine Editor: Edit User-Defined Predicate
File Edit Settings Help
[Icons] Pr
;
; User-Defined predicate
;
; Name: myLengthFactor
;
; Description:
; Multiply the length by a given factor
;
; Parameters:
; scopeKind      Integer
; scopeNum       Integer
; scopeName      Text
; objKind        Integer
; objName        Text
; factor         Real
;
; Return Value:
; Design Units
;
procedure( myLengthFactor (scopeKind scopeNum scopeName objKind objName factor )
)
; Declare functions variables
let((returnValue)
  ; Calculate returnValue
  ;
  returnValue = 0|

  ; Return computed value
  returnValue
)
) ;end of the User-Defined Predicate myLengthFactor
```

Once you code and save your predicate, the *Select or Create Predicate* dialog box appears with your new predicate, ready for future use.



Select or create a predicate that will be compute a value for use in a formula.

Predicate	Function T...	Parameters	Return Type	Description
IntVar	User Defined	var1, var2	Integer	Integer Variables
myLengthFactor	User Defined	scopeKind, scopeName, o...	Design Units	Multiply the length b...
acAddResult	Predefined	scopeKind, scopeName, o...	Integer	Add a result to an o...
acCreateResultName	Predefined	objName, resultName	String	Create the name of ...
acDebug_pprint	Predefined	symbol	Unknown	Skill pprint comman...

Predicate Parameters

A formula contains parameters that are automatically filled in with information about the cell that contains the formula (see [Edit – Formula](#) in the *Constraint Manager Reference* for a list of formula parameters). A predicate also has parameters, often of the same name and function as those in formulas.

If you want to use a predicate to get information about the current cell, you can just use these parameters as is in the predicate call:

```
acGetLength(scopeKind, scopeName, objKind, objName)
```

If you want to use a predicate to get information about other objects, you can substitute the predicate parameters with other values.

- `scopeKind` parameters include `ACNS_SYSTEM` and `ACNS_DESIGN`
- `scopeName` parameters are strings containing the name of your system or one of the designs in it

Substitute `scopeKind` and `scopeName` together to indicate which *design* or *system* you want to reference to find your object. If you are not using Design Links, you do not need to define these objects.

Substitute the `objKind` parameters to specify the type of the object you are referencing. These include:

- `ACNS_NULL`
- `ACNS_SYSTEM`
- `ACNS_DESIGN`
- `ACNS_XNET`
- `ACNS_NET`
- `ACNS_REGION`
- `ACNS_ECSET`
- `ACNS_CLASS`
- `ACNS_DIFFPAIR`
- `ACNS_BUS`
- `ACNS_PINPAIR`

Allegro Constraint Manager User Guide

User-defined Predicates

- ACNS_MATCHGROUP
- ACNS_RESULT
- ACNS_PCSET
- ACNS_SCSET
- ACNS_CLASS_CLASS
- ACNS_REGION_CLASS
- ACNS_REGION_CLASS_CLASS

Note: Use `ACNS_NULL` when you do not want any objects, such as a substitute for the parent object.

Substitute the `objName` parameter to specify a string containing the name of the object. Substitute the `attrName` parameter to specify a string containing the name of an attribute.

Examples of predicate parameters

Requirement

To find the length of a system-level Xnet in your formula

Predicate Call

```
acGetLength(ACNS_SYSTEM, "MY_SYSTEM",  
ACNS_XNET, "SOME_XNET")
```

To find the length of the current object

```
acGetLength(scopeKind, scopeName,  
objKind, objName)
```

Note: You do not have to substitute the parameters, as they are substituted automatically based on the current object row

If you have a formula on *NET1* and you want the length of *NET2*

```
acGetLength(scopeKind, scopeName,  
objKind, "NET2")
```

If you have a formula on *NET1* in *DESIGN1* and you want the length of *NET1* in *DESIGN2* of the active System

```
acGetLength(scopeKind, "DESIGN2",  
objKind, objName)
```

Allegro Constraint Manager User Guide

User-defined Predicates

Requirement

If you have a formula on *NET1* in *DESIGN1* and you want the length of *XNET2* in *MYSYSTEM*

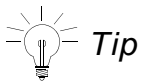
Predicate Call

```
acGetLength(ACNS_SYSTEM, "MYSYSTEM",  
ACNS_XNET, "XNET2")
```

Testing and Debugging User-defined Predicates

You can use Constraint Manager's test environment to debug user-defined predicates (and user-defined measurements, as described in [Testing and Debugging User-defined Measurements](#) on page 182). In a *debug* session, Constraint Manager temporarily makes changes to the database, based on the parameters that you supply. These changes will not be saved once you exit the test environment.

You enter the *Test Environment* from the *Multi-line Editor* (*Choose File – Test*). Before you can run a test, you must supply parameter *VALUES*. The *Test Select Parameters* dialog box (see [Figure 7-10](#) on page 176) lists all the parameters that you defined when you set up the predicate. When you click on a parameter *NAME*, a pop-up window appears, and is based on the parameter *TYPE* (Boolean, Integer, Real, Text).

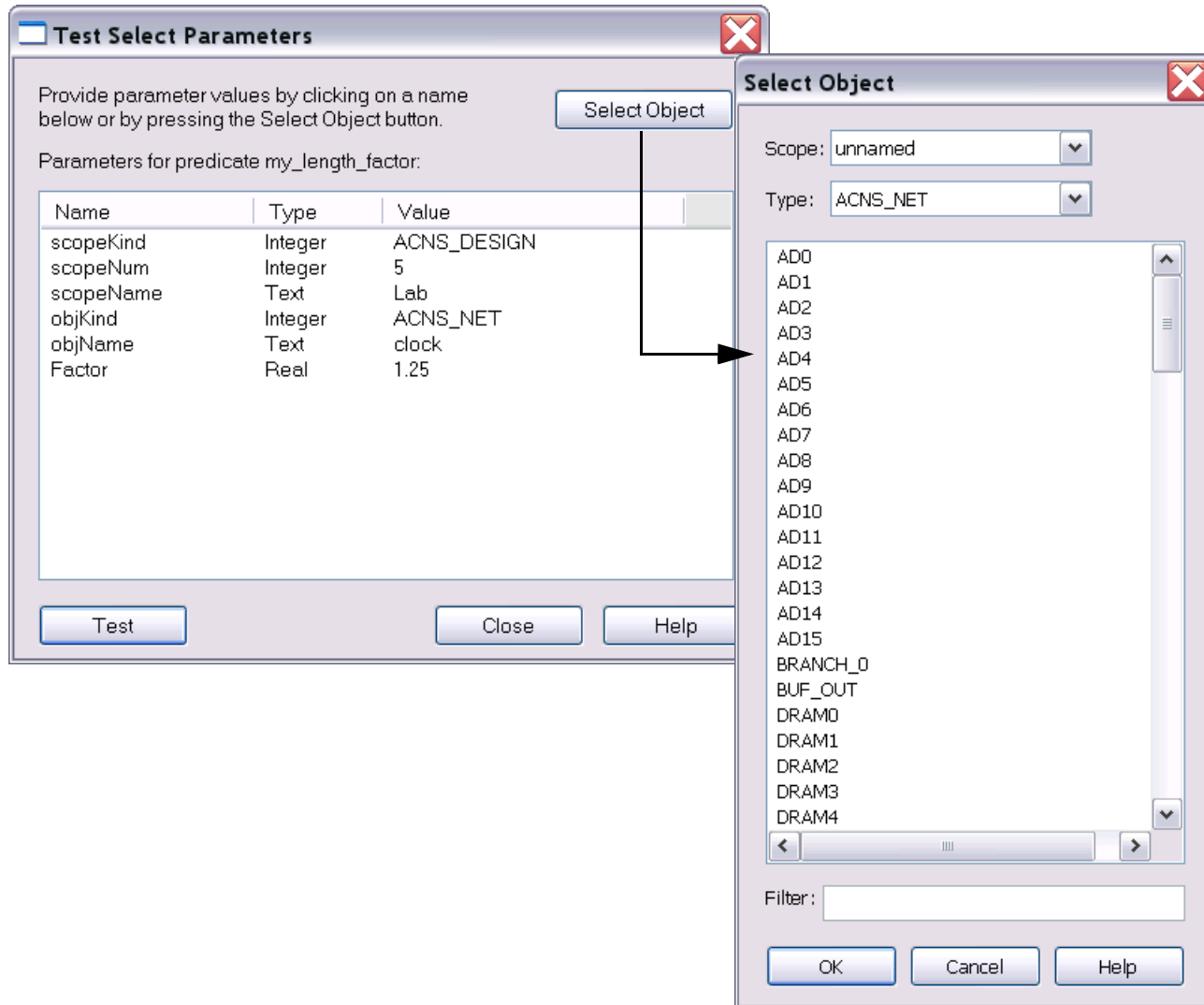


Constraint Manager supplies many pre-defined predicates (look for `acDebug_print*`), which you can use to debug your user-defined predicates. See [Pre-defined Predicates](#) in the *Constraint Manager Reference*.

Allegro Constraint Manager User Guide

User-defined Predicates

Figure 7-10 Test Select Parameters dialog box



In Figure 7-10, the *Select Object* window lets you select any object, and information about this object is used to automatically populate the corresponding parameters.

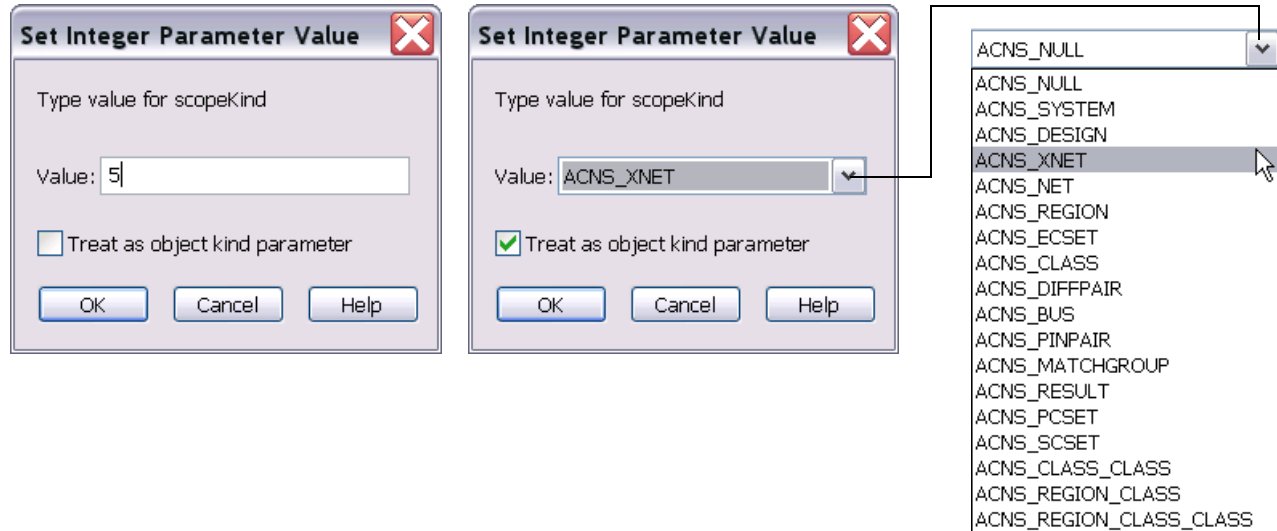
Note: For measurements, this will populate all visible parameters. For predicates, any parameters with the same name as the measurement parameters are populated. When you press *OK*, the *Test Select Parameters* dialog box returns with as many values as possible, updated automatically.

Boolean-, Real-, and Text-parameter *TYPEs* accept a *VALUE* that you enter directly; Integer parameter *TYPEs* accept a direct *VALUE* when the *Treat as object kind parameter* is unchecked. If checked, you are presented with a list of all parameter *TYPEs* in the design.

Allegro Constraint Manager User Guide

User-defined Predicates

Figure 7-11 Parameter values by integer and by object kind



Once all parameters are entered in the *Test Select Parameters* dialog box, click *Test*. A debug window appears with metrics of the test, including the function's name, calculated results, return value, and LINT parsing.

Figure 7-12 Functional test output

```
**** DEBUG OUTPUT ****
Begin function
Length 12425.8, Calculated Length 15532.3

*** RETURN VALUE ***
15532.29

**** SKILL LINT ****
INFO (REP008): Program SKILL Lint started on Oct 15 09:42:35 2007.
INFO (PREFIXES): Using prefixes: "none"
INFO (STRICT): Using strict checking of global variables.
INFO (IQ): IQ score is 100 (best is 100).
INFO (IQ1): IQ score is based on 0 error messages, 0 general warning messages.
INFO (REP110): Total enhancement      : 0.
INFO (REP110): Total external global  : 0.
INFO (REP110): Total package global   : 0.
INFO (REP110): Total warning global    : 0.
INFO (REP110): Total error global     : 0.
INFO (REP110): Total unused vars      : 0.
INFO (REP110): Total next release    : 0.
INFO (REP110): Total alert          : 0.
```

You can continue to test the predicate by supplying new or different parameters, and values, in the *Test Select Parameters* dialog box (see [Figure 7-10](#) on page 176).

When you click *Close* in the *Function Test* dialog box (see Figure 7-12), then the *Test Select Parameters* dialog box, you exit the *Test Environment* and return to the *Multi-line Editor* (see Figure 7-13).

Figure 7-13 A user-defined Predicate with debugging statements

```
; Declare functions variables
let((tempValue returnValue)
  ; Calculate returnValue
  ;
  acDebug_printf("Begin function\n")
  tempValue = acGetEtchLength(scopeKind scopeName objKind objName)
  returnValue = tempValue * factor
  acDebug_printf("Length %g, Calculated Length %g\n" tempValue returnValue)

  ; Return computed value
  returnValue
)
;end of the User-Defined Predicate myLengthFactor
```

User-defined Actuals

A user-defined *Actual* is an attribute that is associated with a measurement. When Constraint Manager analyzes the *Actual*, the measurement is triggered and the cell that contains the *Actual* is populated with results.

A user-defined *Actual*

- is a term used interchangeably with the term *Measurement*
- displays the results of a measurement
- can occupy a user-defined column to facilitate custom reports
- can be used in defining a user-defined constraint

You must add a column to a worksheet to accommodate an *Actual* cell. For information on adding custom workbooks, worksheets, columns, and attributes, see

- [Customizing Worksheets](#) on page 151.
- The [Tools – Customize Worksheet](#) command in the *Constraint Manager Reference*.

The measurement is coupled with the *Actual* and executes when the *Actual's* associated constraint is analyzed.

Note: The measurement executes for the object being analyzed and typically returns results (*Actual* values) for the same.

Once the new column is in place, the *Create Attribute Definition* dialog box appears (see [Figure 7-14](#) on page 179). Note that *Actual* is selected from the *Treat As* drop-down menu.

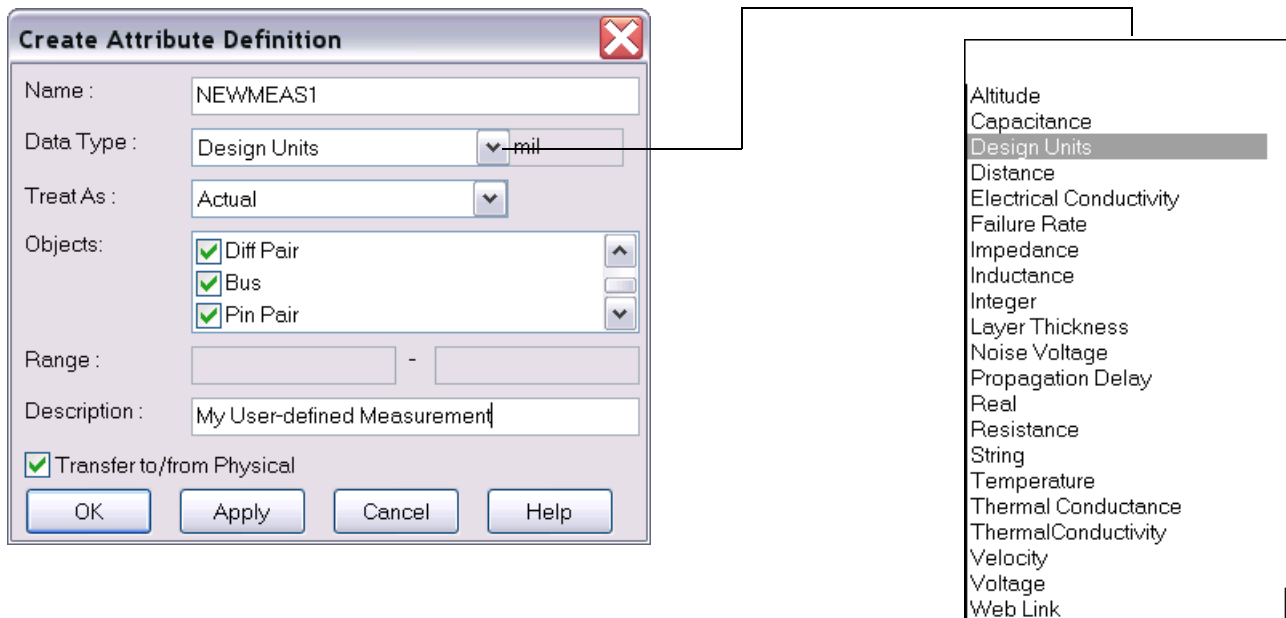
- Complete the *Setup for User-defined Measurement* dialog box as shown in [Figure 7-15](#) on page 180. Specify a name, a description, and supported objects.

Note: The *Parameter List* is fixed and the buttons are inactive for user-defined measurements. This dialog box is also used to set up user-defined predicates (see [User-defined Predicates](#) on page 170).

User-defined Measurements

A user-defined measurement is required to compute a unique result, which is needed to validate the design intent. It requires a new column and it governs how to calculate the results of the new column.

Figure 7-14 Measurement attribute definition (Treat As equals Actual)



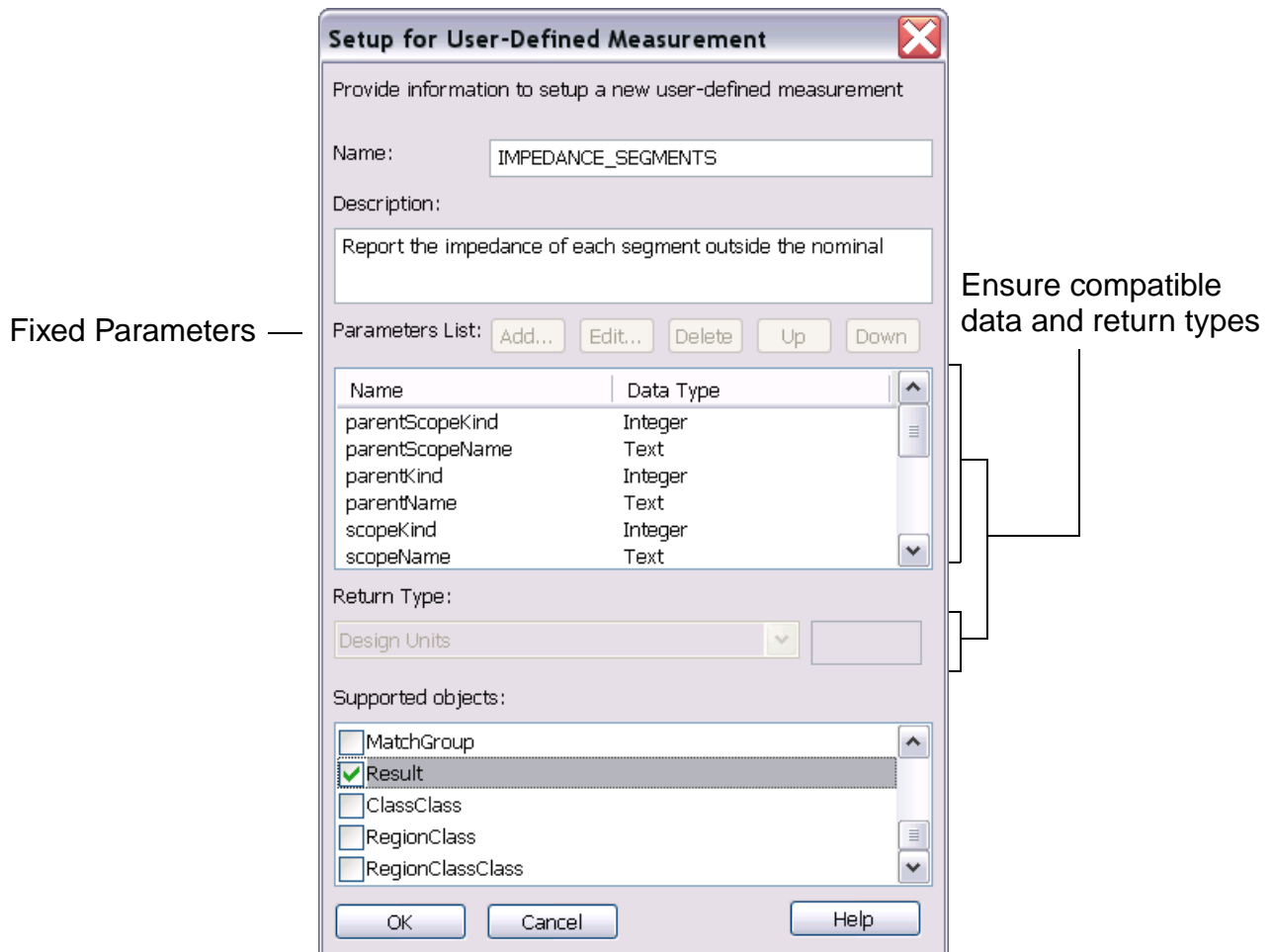
Allegro Constraint Manager User Guide

User-defined Predicates

User-defined Measurements . . .

- have a fixed set of parameters, which are populated by Constraint Manager upon analysis.
- are based on data types, such as design units, as defined in the *Data Type* drop-down menu
- are based on supported object. *Actuals* calculate for rows that contain supported objects.
- return ACNS_OK to indicate successful completion; ACNS_FAIL to indicate errors.

Figure 7-15 Setting up your measurement



Allegro Constraint Manager User Guide

User-defined Predicates

Once completed, click *OK*. The *Multi-line Editor* appears. For information on using the *Multi-line Editor* (see the [Edit – Formula](#) command in the *Constraint Manager Reference*).

Figure 7-16 A User-defined measurement

```
: User-Defined measurement
:
: Name: newMeas1
:
: Description:
:
: Parameters:
: parentScopeKind      Integer
: parentScopeName     Text
: parentKind           Integer
: parentName           Text
: scopeKind            Integer
: scopeName            Text
: objKind              Integer
: objName              Text
: constraintName       Text
: actualName           Text
: vstatusName          Text
: verifyMode           Integer
:
: Return Value:
: Design Units
:
: procedure( newMeas1 (parentScopeKind parentScopeName parentKind parentName scopeKind scopeName objKin
:
: ; Declare functions variables and return code
: let((done tempValue)
:   ; Perform calculation and populate <actualName> using
:   ; predicate acPutValue or acPutValue_p
:   ;
:   ; e.g. done = acPutValue(scopeKind scopeName objKind objName actualName <type> <calculatedValue
:   ;
:   acDebug_printf("Start Measurement\n")
:   tempValue = myLengthFactor(scopeKind scopeName objKind objName 1 25)
:   done = acPutValue(scopeKind scopeName objKind objName actualName ACNS_DOUBLE_TYPE tempValue)
:
:   ; If calculation done, return ACNS_OK, otherwise ACNS_FAIL
:   ;
:   if(done then
:     ACNS_OK
:   else
:     ACNS_FAIL
:   )
: )
: ;end of the User-Defined Measurement newMeas1
```

When defining a measurement:

- Use one of the `acPutValue()` predicates to populate the *Actual*. The *Actual* is passed into the measurement through the `actualName` parameter.
- If the measurement cannot compute a result, populate the verification status with the reason using `acPutValue()` predicate, with an `ACNS_STRING_TYPE`. The verification status is passed into the measurement through the `vstatusName` parameter.

Allegro Constraint Manager User Guide

User-defined Predicates

- If the measurement needs to create a new result object to display under the object being measured, the following predicates are required:
 - `acCreateResultName` to create the name of the new object
 - `acAddResult` to associate the new object with the object being analyzed
 - `acPutValue` to populate the `actualName` on the new object
- When you are finished with defining the measurement, *Save* and *Close* the *Multi-line Editor*. The new measurement is available for selection.



Tip

A measurement returns a status to the calling function, not a calculated value: `ACNS_OK` upon success; `ACNS_FAIL` upon encountering an error.

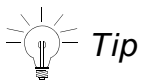
Calculated values populate an *Actual* cell using the `acPutValue` predicate in the body of the measurement.

Actual cells are typically populated for the current object or for result objects created by the measurement using the `acCreateResultName` and `acAddResult` predicates.

- Once you dismiss the remaining dialog boxes, right-click on the new column head and choose *Analyze* from the pop-up menu. This yields results for the new *Actual* cell with the measurement that you just defined.

Testing and Debugging User-defined Measurements

Most parameters for user-defined measurements are determined from the object row that is being measured; however, because a measurement can be applied to more than one row, this information is not known when the measurement is being created or edited. A set of values for the parameters must be provided before the measurement can be tested.



Tip

Constraint Manager supplies many pre-defined predicates (look for `acDebug_print*`), which you can use to debug your user-defined measurements. See Pre-defined [Predicates](#) in the *Constraint Manager Reference*.

Allegro Constraint Manager User Guide

User-defined Predicates

All measurements require the following, fixed set of parameters:

- `parentScopeKind`
- `parentScopeName`
- `parentKind`
- `parentName`
- `scopeKind`
- `scopeName`
- `objKind`
- `objName`
- `constraintName`
- `actualName`
- `vstatusName`
- `verifyMode`

When running the measurement, the last four parameters are always the same; therefore, you do not need to specify them.

Some of these parameters can be determined by Constraint Manager, or they are always the same when running the measurement. These parameters do not need to be specified and will not appear in the parameter list in the *Test Select Parameters* dialog box. Usually, only the `scopeKind`, `scopeName`, `objKind`, and `objName` parameters need to be specified

All measurements include one of the following `TYPE` parameters:

- `ACNS_DOUBLE_TYPE`
- `ACNS_STRING_TYPE`
- `ACNS_ENUM_TYPE`
- `ACNS_INTEGER_TYPE`
- `ACNS_LONG_TYPE`
- `ACNS_DOUBLE_ARRAY_TYPE`
- `ACNS_STRING_ARRAY_TYPE`
- `ACNS_ENUM_ARRAY_TYPE`

Allegro Constraint Manager User Guide

User-defined Predicates

- ACNS_INTEGER_ARRAY_TYPE
- ACNS_LONG_ARRAY_TYPE

The `TYPE` parameter indicates how the `calculatedValue` parameter is stored in the attribute named by the `actualName` parameter. Enum types are treated as strings and must be a legal value for the `actualName`. Array types are lists of values used to populate attributes that contain multiple values (colon separated strings).

Important

The *Test Environment* for debugging user-defined measurements is identical to that of user-defined predicates.

See

- [User-defined Measurements](#) on page 179
- [Testing and Debugging User-defined Predicates](#) on page 175