

Pyros Interactive Viewer User Guide

Version T-2022.06-SP2, December 2022



Copyright and Proprietary Information Notice

© 2022 Synopsys, Inc. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>. All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

New in This Release	6
Related Products, Publications, and Trademarks	6
Conventions	6
Customer Support	7
Statement on Inclusivity and Diversity	8

1. Using the Pyros Viewer	9
Starting the Pyros Viewer and Loading a Design	9
Using the File > Open Menu	10
Using Command Line Options	10
Viewing a Design	11
Panning	12
Zooming	12
Querying Objects	12
Viewing Source Code that Built a PyCell Object	13
Selecting and Moving Objects	14
Selecting Objects	15
Modifying the Current Selection	15
Moving Selected Objects	15
Making Measurements	15

2. Using PyCell Explorer and the Python API	16
Exploring the Python API	16

3. Debugging PyCells	19
Starting the PyCell Studio IDE and Running the Training Designs	19
Specifying Breakpoints	20
Debugging With Breakpoints	20

4. Using Pyros Viewer Menus	23
--	-----------

Contents

File Menu	23
Edit Menu	24
View Menu	25
Tools Menu	26
Analyze Density Dialog Box	27
Draw Density Map Dialog Box	28
System Options Dialog Box	29
System Options: PyStudio > Behaviour	30
System Options: PyStudio > Font & Colors	31
System Options: Display > General	32
System Options: Display > Objects	34
System Options: Grid > Settings	35
System Options: Environment > Keyboard	36
Window Menu	37
PyCell Studio Menu (PyCell Studio IDE Only)	38
Help Menu	39
Pyros Viewer Toolbar Commands	39

5. Using Pyros Viewer Windows	40
Log Window	41
Layer/Purpose Window	42
Hierarchy Window	45
Information Window	46
User Grid Lines Window	47
PyCell Parameters Window	48
Debugging While Changing Parameters	49
Python Console Window	49
Drc Errors Window	50
Status Bar	51

6. Using PyCell Studio IDE	52
PyCell Studio IDE Window	53
PyCell Studio IDE Menus	54
File Menu	55
Edit Menu	55

Contents

Tools Menu	56
Find Menu	57
Breakpoints Menu	58
Debug Menu	59
PyCell Studio IDE Toolbar Commands	59
Debugger Output Window	61
<hr/>	
7. Using Stretch Handles and Auto-Abutment	62
Using Stretch Handles	62
Using Auto-Abutment	64
<hr/>	
A. Santana Technology Display File	67
Creating the Santana Technology Display File	67
Layer Purpose Pair Display Structure	68
Example	69
<hr/>	
B. References	71
Environment Variables	71
.cnitools Directory	72
Changing Start-Up Defaults	72
Using Replay Files	73

About This User Guide

Pyros is an interactive layout viewer for the PyCell Studio tool.

This preface includes the following sections:

- [New in This Release](#)
- [Related Products, Publications, and Trademarks](#)
- [Conventions](#)
- [Customer Support](#)
- [Statement on Inclusivity and Diversity](#)

New in This Release

Information about new features, enhancements, and changes, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the *PyCell Studio Release Notes* on the SolvNetPlus site.

Related Products, Publications, and Trademarks

For additional information about the PyCell Studio tool, see the documentation on the Synopsys SolvNetPlus support site at the following address:

<https://solvnetplus.synopsys.com>

You might also want to see the documentation for the following related Synopsys products:

- Custom Compiler™
- Laker®

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Courier	Indicates syntax, such as <code>write_file</code> .

Convention	Description
<i>Courier italic</i>	Indicates a user-defined value in syntax, such as <code>write_file design_list</code>
Courier bold	Indicates user input—text you type verbatim—in examples, such as <code>prompt> write_file top</code>
Purple	<ul style="list-style-type: none">• Within an example, indicates information of special interest.• Within a command-syntax section, indicates a default, such as <code>include_enclosing = true false</code>
[]	Denotes optional arguments in syntax, such as <code>write_file [-format fmt]</code>
...	Indicates that arguments can be repeated as many times as needed, such as <code>pin1 pin2 ... pinN.</code>
	Indicates a choice among alternatives, such as <code>low medium high</code>
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Bold	Indicates a graphical user interface (GUI) element that has an action associated with it.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy .
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.

Customer Support

Customer support is available through SolvNetPlus.

Accessing SolvNetPlus

The SolvNetPlus site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNetPlus site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNetPlus site, go to the following address:

<https://solvnetplus.synopsys.com>

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNetPlus site, click REGISTRATION HELP in the top-right menu bar.

Contacting Customer Support

To contact Customer Support, go to <https://solvnetplus.synopsys.com>.

Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1

Using the Pyros Viewer

Pyros is an interactive layout viewer. This chapter describes how to use the Pyros Viewer Graphical User Interface.

Using the Pyros viewer is discussed in the following sections:

- [Starting the Pyros Viewer and Loading a Design](#)
- [Viewing a Design](#)
- [Selecting and Moving Objects](#)

For more information see [Using Pyros Viewer Menus](#) and [Using Pyros Viewer Windows](#).

Starting the Pyros Viewer and Loading a Design

Type `pyros` in a terminal window to launch the Pyros viewer.

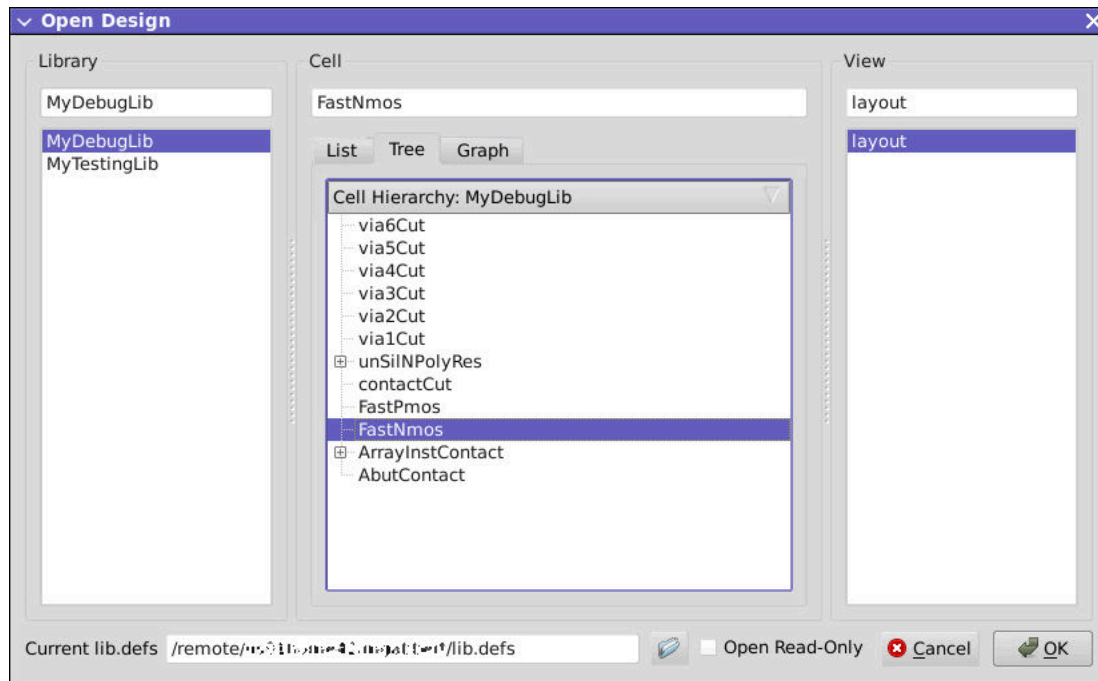
After the Pyros viewer is loaded, open a design using the menu or command line:

- [Using the File > Open Menu](#)
- [Using Command Line Options](#)

Using the File > Open Menu

Choose **File > Open** to display the **Open Design** dialog box. If there is a `lib.defs` file in the current directory, the dialog box opens and shows all of the libraries defined, as shown in [Figure 1](#).

Figure 1 Open Design Dialog Box



Note:

If there is no `lib.defs` file, then the `lib.defs` file in the user's home directory is used by default. If no `lib.defs` file is found, then an error dialog box is generated.

Each time the dialog box is brought up, the application makes a call to `OpenAccess` to load and refresh the `lib.defs` file. The **Tree** and **Graph** Cell tabs can be used to view different hierarchy relationships among the designs in the library.

Multiple designs can be opened each in its own display canvas, but each design must have the same technology library associated with it.

Using Command Line Options

The Pyros viewer provides a number of different command line options that can be displayed by using the `-h` help option. [Table 1](#) lists the command line options.

Table 1 *Pyros Command Line Options*

Option	Description
<code>--rp=REPLAY_FILE</code>	Run the replay file. Replay files are created in .cnitools/*.rp
<code>--rps=REPLAY_FILE</code>	Run the replay file in step mode. Type 'c' to step
<code>--cmd=COMMAND</code>	Run command <i>COMMAND</i>
<code>--commands</code>	List the currently registered commands.
<code>--commandHelp=CMD</code>	List help for command <i>CMD</i> .
<code>--py=PYTHON_FILE</code>	Run the Python file.
<code>--ld=LIB/CELL[/VIEW]</code>	Load the design or all views of the specified design. Syntax is 'lib/cell/view' or 'lib/cell'.
<code>--ldf=DESIGNS_FILE</code>	Load all designs listed in the file <i>DESIGNS_FILE</i> . Syntax of this file is 'lib cell view' per line for each design to be loaded.
<code>--no_cell_hier</code>	Do not open all designs to produce the cell hierarchy in the Open Design dialog.
<code>--bg</code>	Open in background mode.
<code>--opengl</code>	Use OpenGL rendering.
<code>--defstyle</code>	Use default style for widget rendering. Useful for older X11 rendering used by VNC-type.
<code>-h, --help</code>	Show this help message and exit.
<code>-V, --version</code>	Display the tool's version number and exit.

Viewing a Design

Viewing a design is discussed in the following sections:

- [Panning](#)
- [Zooming](#)
- [Querying Objects](#)
- [Viewing Source Code that Built a PyCell Object](#)

Panning

Panning is supported as follows:

- Use the arrow keys on the keyboard. Hold down on the key to do Dynamic Panning.
- Use the on-screen view widget in top right corner of the layout view by clicking on one of the four directional arrows. Hold down on the arrow to do Dynamic Panning.
- Use Dynamic Drag Pan by clicking down on the middle mouse button while dragging the cursor.
- Choose **View > Pan Up**, **View > Pan Down**, **View > Pan Right**, and **View > Pan Left**.

Zooming

Zooming is supported as follows:

- Use the middle mouse button scroll wheel (scroll up to zoom in, scroll down to zoom out).
- Use the right mouse button to select a zoom area.
- Use the on-screen view widget in top right corner of the layout view (Zoom In, Zoom Out, Zoom Home).
- Type 'Z' on the keyboard to zoom in, 'Shift+Z' to zoom out.
- Choose **View > Zoom Home**, **View > Zoom Selected**, **View > Zoom In**, and **View > Zoom Out**.

Querying Objects

Querying objects is supported as follows:

- Press the Spacebar while positioning the cursor over the object. The object's information shows up on the [Status Bar](#) and the [Information Window](#).
- With the **Dynamic Highlight** display option on (found under **Tools > Options, Display > General**), move the cursor over an object. Pressing the Spacebar cycles over all of the objects under the cursor from smallest to largest.

Viewing Source Code that Built a PyCell Object

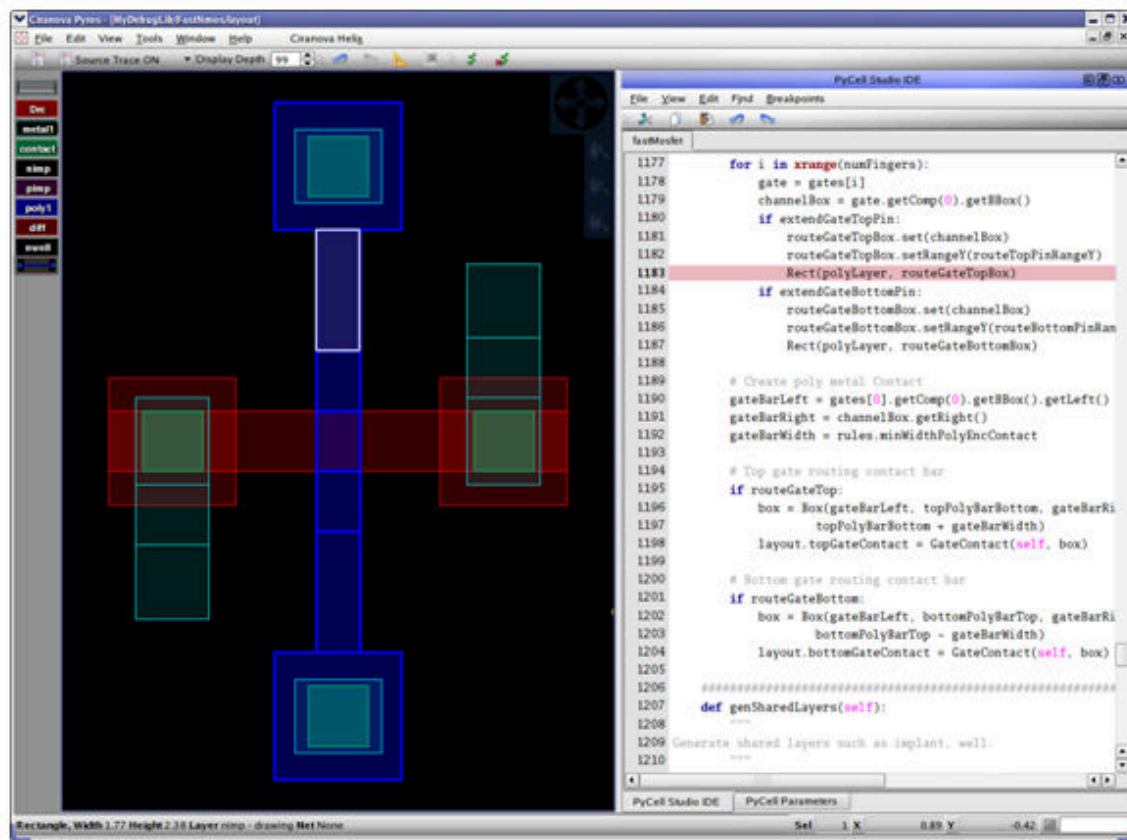
Use the following procedure to view source code for an object:

1. Turn on source tracing using the **Source Trace** toolbar command.
2. Reevaluate the PyCell by clicking the **Apply** button in the **PyCell Parameters** window.
3. Select an object, and then click the **View Source Code For Selected Object** button in the **Source Trace** toolbar or under the **View** menu. The **PyCell Studio IDE** window opens with the source code in a text editor, and the line is highlighted. Repeat this step to view source traces for other design objects.

Alternatively, for objects that cannot be selected (for example, 1x shapes in advanced technology nodes), highlight the shape by hovering the cursor over it, then press Ctrl +I (for the **View > View Source Code for Highlighted Object** command). If more than one shape is under the cursor, press the Spacebar to cycle highlighting among the shapes.

4. Because source tracing is implemented as a property on shapes in the layout, the PyCell library should be compiled with the source tracing property embedded within the PyCells; this is done using the `--srctrace` command-line option for the `cngenlib` utility (as described in the *Utilities Reference Manual*).
5. Note that source tracing should be enabled before designs are loaded; it has no effect on designs which are already loaded. In addition, open designs must be evaluated before source tracing is active (as in step 2 above); this is required, so that source tracing properties can be associated with shapes in the layout. For example, after opening a PyCell design, one or more parameter values should be changed and applied before source tracing is active.

Figure 2 Viewing Source Code



Selecting and Moving Objects

Selecting and moving objects is discussed in the following sections:

- [Selecting Objects](#)
- [Modifying the Current Selection](#)
- [Moving Selected Objects](#)
- [Making Measurements](#)

Selecting Objects

Only objects at the current level can be selected, as follows:

- Click an object with the left mouse button.
- Use the left mouse button to select an area.

Modifying the Current Selection

Modifying the current selection is supported as follows:

- Holding down the 'Shift' key while selecting adds to the selected set.
- Holding down the 'Ctrl' key while selecting toggles the selection status of the object (so objects can be removed from the selected set).

Moving Selected Objects

Use the following procedure to move selected objects:

1. Choose **Edit > Move Selected**, or type 'm'. The [Status Bar](#) then shows:

Specify anchor point for move

2. Then click anywhere to define the anchor point. The selected objects are now placed on the cursor and the [Status Bar](#) shows:

Select location to move to...

3. Then click again to move the objects.

Making Measurements

Choose **Edit > Add Ruler** to create a ruler to measure the distance between two points in the layout. The first point is specified by clicking on the left mouse button, and the second point is specified by clicking again on the left mouse button. Note that if you want to pan or zoom during creation of a ruler, then the keyboard keys should be used for these pan or zoom operations (Z or Shift+Z for zoom, arrow keys for pan).

2

Using PyCell Explorer and the Python API

This chapter discusses using the PyCell Explorer to work with the Python API.

Using PyCell Explorer is discussed in the following section:

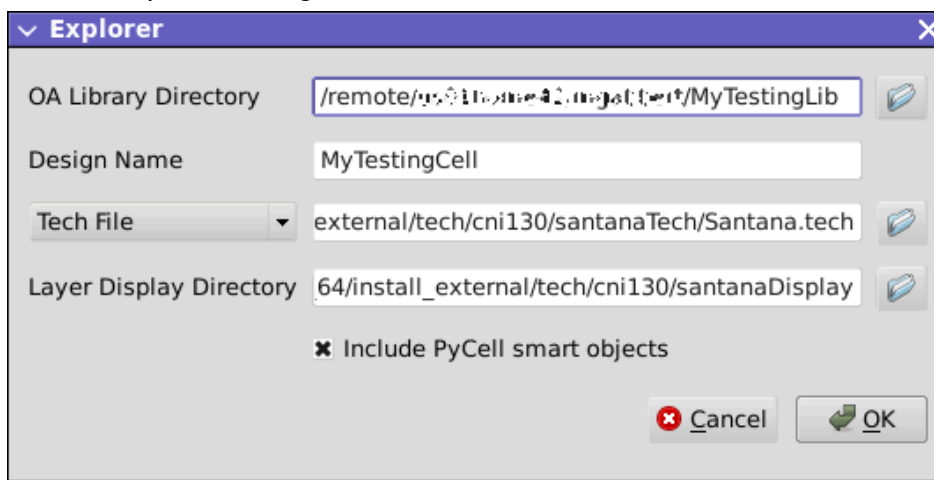
- [Exploring the Python API](#)

Exploring the Python API

Type `cnexp` at the terminal window to open the **PyCell Explorer** window.

Fill in the **Explorer** dialog box if you want to change the default values, or just click **OK** to use the defaults. An empty design is created for exercising the API.

Figure 3 Explorer Dialog Box



The **PyCell Explorer** window contains a Python interpreter console window with the PyCell Explorer module preloaded. The Python interpreter is built upon the standard Python interpreter, with the addition of specific classes and methods which are used for creating parameterized cells. The PyCell Explorer module can be used to interactively create parameterized cell designs. It is very useful for trying out different commands from

the Python API, to see how they work and see the resulting physical design objects get created or modified in the display window.

At this point, all of the various classes, methods and functions defined for the Python API can now be used and commands can be interactively entered. For example, if the following Python command is entered:

```
rect1 = Rect(Layer('metal1'), Box(0, 0, 1, 1))
```

then the display shows a rectangle with these coordinates created and displayed on the 'metal1' layer. As interactive commands from the Python API are entered, the display is updated with the results from executing each interactive command. This direct interactive graphical feedback makes it very easy to see how various classes and methods defined in the Python API operate.

Now type (without pressing Enter):

```
rect1.
```

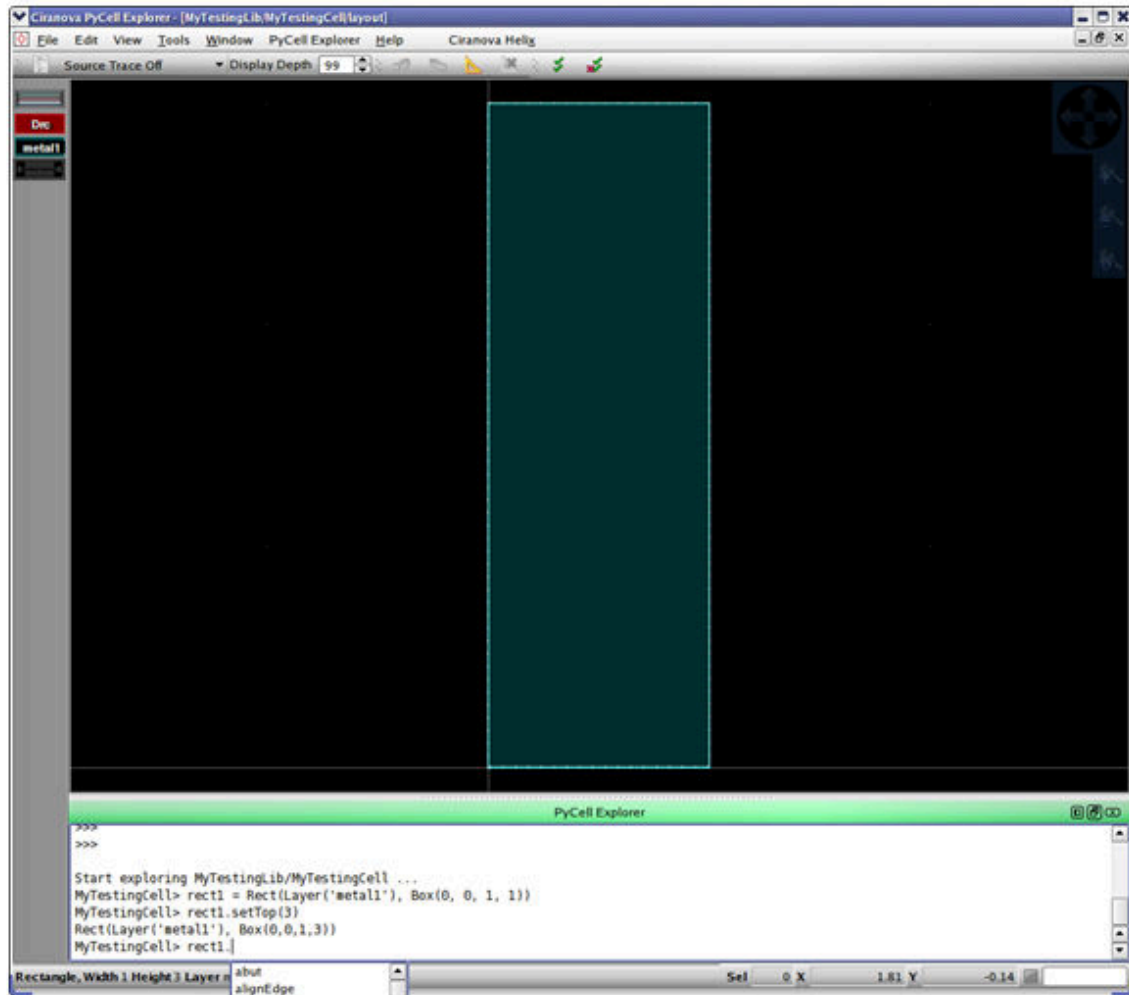
At this point, a list pops up showing all of the methods for the `Rect` object. You can continue typing or select one of the methods from the list. For example, you can continue to type and enter:

```
setTop(3)
```

Chapter 2: Using PyCell Explorer and the Python API Exploring the Python API

And then you see the following modification to the rectangle:

Figure 4 *Python API*



3

Debugging PyCells

This chapter describes using the PyCell Interactive Debugging Environment (IDE) to debug PyCells.

Debugging PyCells using the PyCell Studio IDE is discussed in the following sections:

- [Starting the PyCell Studio IDE and Running the Training Designs](#)
- [Specifying Breakpoints](#)
- [Debugging With Breakpoints](#)

For more information, see [Using PyCell Studio IDE](#).

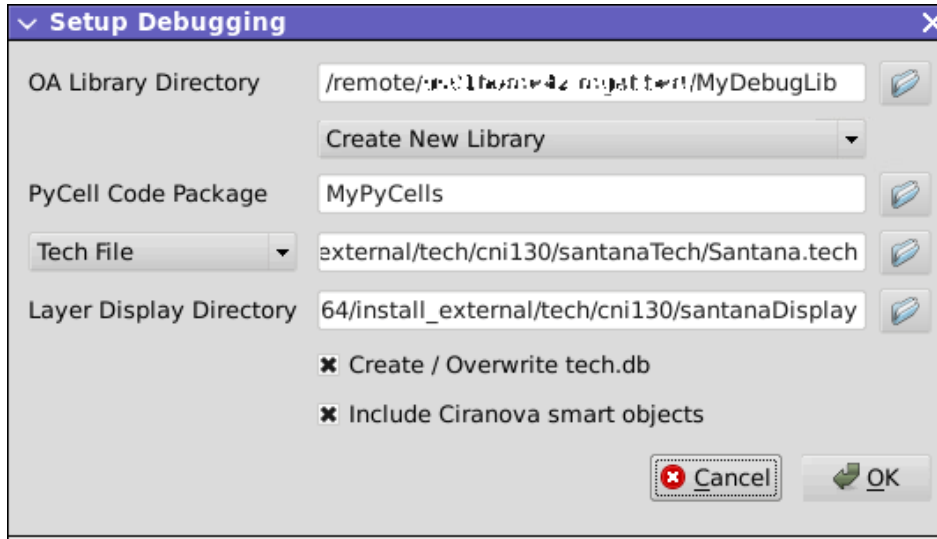
Starting the PyCell Studio IDE and Running the Training Designs

Type `cndbg` in the terminal window to open the PyCell Studio Interactive Debugging Environment (IDE).

The PyCell IDE can be used in a training mode by choosing **PyCell Studio > Training Example**. See [PyCell Studio Menu \(PyCell Studio IDE Only\)](#) for more information.

At this point, the **PyCell Studio IDE** and **Debugger Output** windows are displayed, along with the **Setup Debugging** dialog box.

Figure 5 Setup Debugging Dialog Box



Click **OK** to take the defaults. The training files are loaded and the debugger runs resulting in four different designs: `fastMosfet.py`, `resistorUnit.py`, `resistor.py`, and `techUtils.py`.

Specifying Breakpoints

Before you rerun the debugger, specify some breakpoints:

1. Select the **fastMosfet.py** tab in the **PyCell Studio IDE** window and type 'Ctrl-F'. This displays the **Find** dialog box at the bottom of the editor (see [Find Menu](#)).
2. In the **Find** field of the dialog box, type 'g' and the `genLayout` entry of the default search list is entered and this routine is scrolled to in the editor.
3. Place a breakpoint at line 824 '`self.createTransistor()`' by clicking to the left of the line number.

Debugging With Breakpoints

With breakpoints set, now rerun the debugger:

1. Click the **Run in the debugger** button on the toolbar.
2. The debugger starts and eventually stops at the breakpoint.

3. Click the **StepInto** button to step into the 'createTransistor' method.
4. Click the **Step** icon button to step over each line of code.

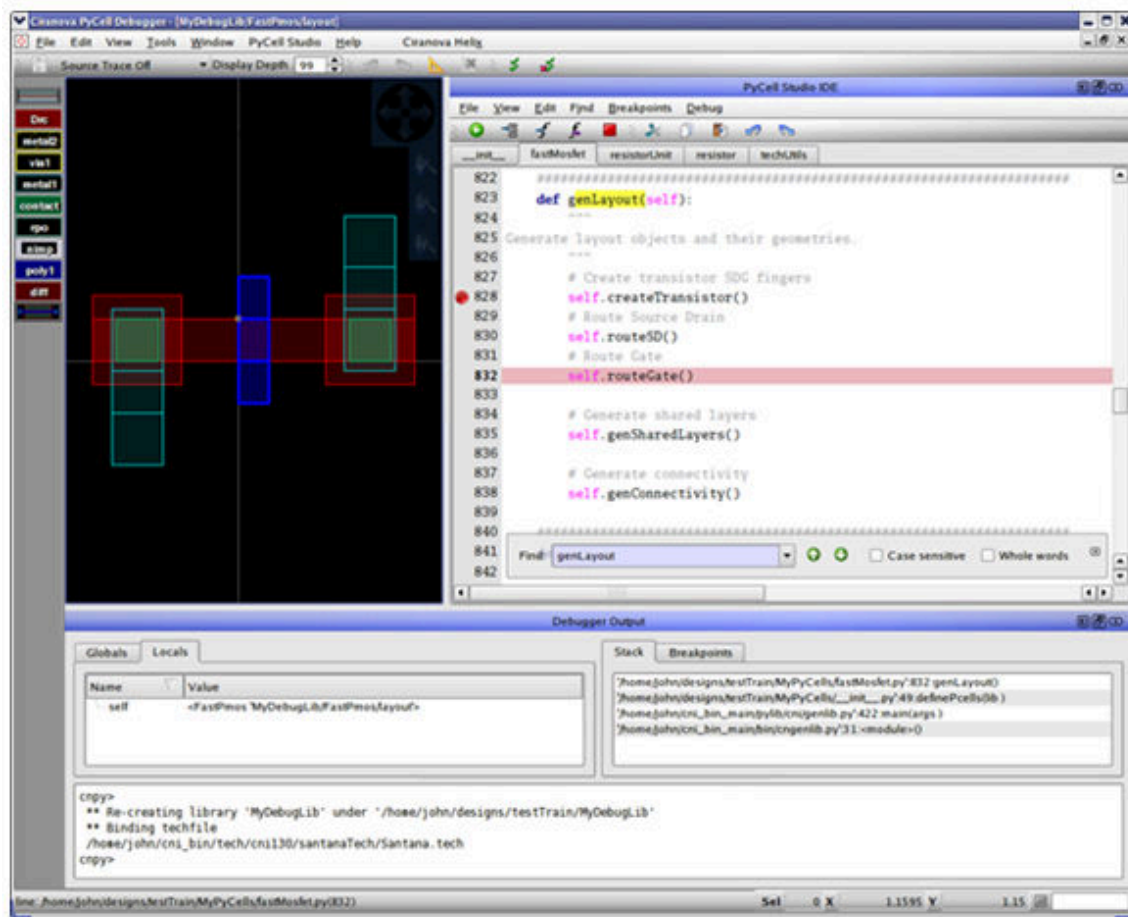
The display updates as each object is created and the **Globals**, **Locals**, and **Stack** views in the **Debugger Output** window are updated with the current information.

5. Click the **Run in the debugger** button again to continue running until the next breakpoint is hit.

You can query the objects in the display while debugging to see the results of the PyCell creation.

Figure 6 shows the breakpoint and the generated layout:

Figure 6 Debugging Layout



In addition, the "PyCell Studio™ Tutorial" shows how to use this "built-in" Interactive Debugging Environment (IDE) to view and debug Python PyCell™ parameterized cell designs.

4

Using Pyros Viewer Menus

This chapter describes using the Pyros viewer menus.

The Pyros viewer is controlled through the following menus and commands:

- [File Menu](#)
- [Edit Menu](#)
- [View Menu](#)
- [Tools Menu](#)
- [Window Menu](#)
- [PyCell Studio Menu \(PyCell Studio IDE Only\)](#)
- [Help Menu](#)
- [Pyros Viewer Toolbar Commands](#)

File Menu

Table 2 *File Menu*

Name	Key	Description
Open	F5	Display the Open Design dialog box. See Using the File > Open Menu .
Reload		Re-load the current design, discarding recent changes.
Run Script		Run a Python script file.
Save	F2	Save the design to disk. Does nothing if the design was opened read only.
Save As	F3	Save the design with a new library cell view name. Can be used with read only designs.

Table 2 File Menu (Continued)

Name	Key	Description
Close	Ctrl+W	Close the active design.
Close All		Close all designs.
Export Design		Creates an XML version of the active design in the local directory named <library>_<cell>_<view>.xml
Export Image		Save contents of main viewing window as an image file. You can select the type of image file (BMP, JPEG, PNG, and so on) to be generated. SVG (Scalable Vector Graphics) is also an option.
Exit	Ctrl+Q	Exit the application.

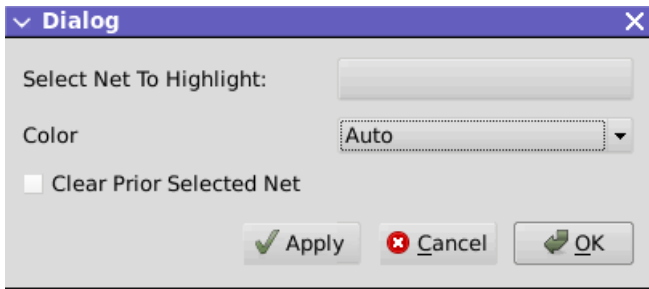
Edit Menu

Table 3 Edit Menu

Name	Key	Description
Undo	U	Undo the previous editing operation. See also Pyros Viewer Toolbar Commands .
Redo	Shift+U	Redo the previous editing operation. See also Pyros Viewer Toolbar Commands .
Un-Select All	Ctrl+D	Deselects all selected design objects.
Move Selected	M	Move the selected design objects. The Status Bar prompts for an anchor point then the point to move to.
Delete Selected	Del	Delete the selected design objects.
Add Ruler	K	Select two points to add a ruler. See also Pyros Viewer Toolbar Commands .
Cancel Current Function	Esc	Cancel the current editing command.
Delete Rulers	Shift+K	Delete all rulers. See also Pyros Viewer Toolbar Commands .

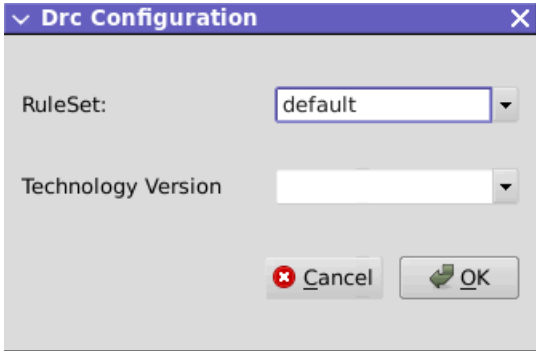
View Menu

Table 4 View Menu

Name	Key	Description
Zoom Home	Ctrl+H	Zoom out to entire view of the design. See also Pyros Viewer Toolbar Commands .
Zoom Selected	Ctrl+Shift+Z	Zoom to the currently selected objects.
Zoom In	Z	Zoom in. See also Pyros Viewer Toolbar Commands .
Zoom Out	Shift+Z	Zoom out by 2x viewing factor. See also Pyros Viewer Toolbar Commands .
Pan Up	Up	Move the view of the design up by 10%.
Pan Down	Down	Move the view of the design down by 10%.
Pan Left	Left	Move the view of the design left by 10%.
Pan Right	Right	Move the view of the design right by 10%.
Highlight Selected Nets		Highlight the nets of the currently selected objects by dimming all other nets. Only nets at the top level are highlighted.
Highlight Net By Name		Highlight a net by selecting its name in the dialog box. Only nets at the top level are highlighted. This command displays the following dialog box.
		
Un-Highlight Nets		Restore the intensity of the display for all nets.
View Source Code For Selected Object		Go to the PyCell™ source code which generated the selected geometry.
View Source Code For Highlighted Object	Ctrl+I	Go to the PyCell™ source code which generated the highlighted geometry.

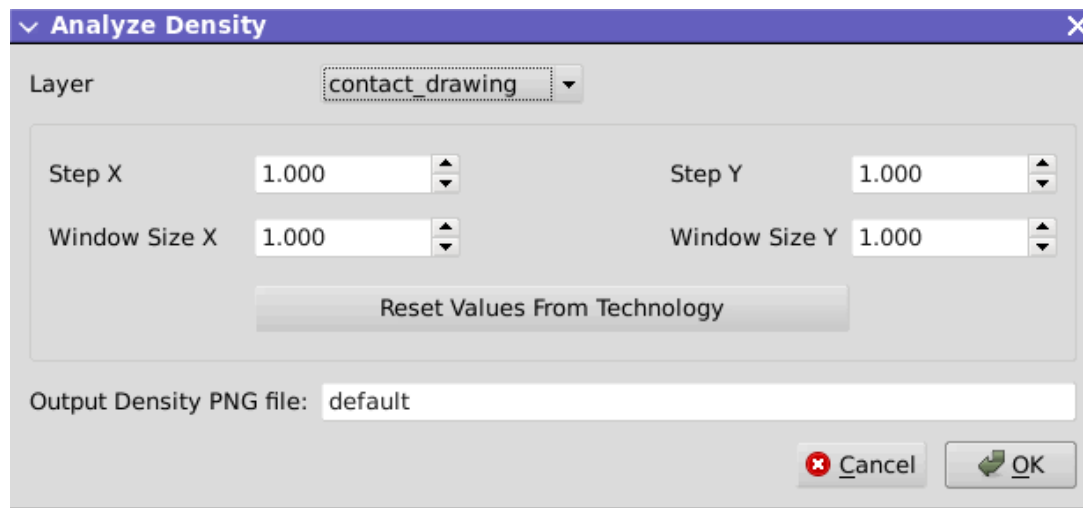
Tools Menu

Table 5 Tools Menu

Name	Description
Drc > Run Drc	Run the internal DRC tool. See also Pyros Viewer Toolbar Commands .
Drc > Clear Drc Markers	Clear all DRC error marker geometries. See also Pyros Viewer Toolbar Commands .
Drc > Drc Configuration	Select the technology file rule set to be used when running internal DRC tool. This command displays the Drc Configuration dialog box. 
Density > Analyze Density	Display the Analyze Density Dialog Box .
Density > Draw Density Map	Display the Draw Density Map Dialog Box .
Density > Clear Density Map	Erase the current density map from the display.
Options	Display the System Options Dialog Box .

Analyze Density Dialog Box

Choose **Tools > Density > Analyze Density** to open the **Analyze Density** dialog box.



Relative density on a layer can be analyzed using this dialog. Density is computed by defining a square window of a specified size and placing it in different locations of the layout. The first location of the window is in the lower-left corner of the layout. The subsequent locations are obtained by shifting the window by the specified step in the vertical and/or horizontal directions, resulting in a square grid with the size of the specified step.

Table 6 *Analyze Density Dialog Box Options*

Option	Description
Layer:	Select the layer to analyze. The current rule set (see Tools > Drc > Drc Configuration) is searched for any density rules, and if found, the list of layers presented is restricted to those with density rules defined for them.
Step X, Step Y	The step size.
Window Size X, Window Size Y	The window size.
Reset Values From Technology	The current rule set (see Tools > Drc > Drc Configuration) is searched for any density rules, and if found, the step and window size are set to those values. The Density Display Range (%) Min value (see Draw Density Map Dialog Box) is also set to the minimum density value for that rule.
Output Density PNG file	The results of the Density Analysis are stored in this file. They can be viewed using the Draw Density Map Dialog Box .

When the analysis is complete, the [Draw Density Map Dialog Box](#) appears.

Draw Density Map Dialog Box

Choose **Tools > Density > Draw Density Map** to open the **Draw Density Map** dialog box.

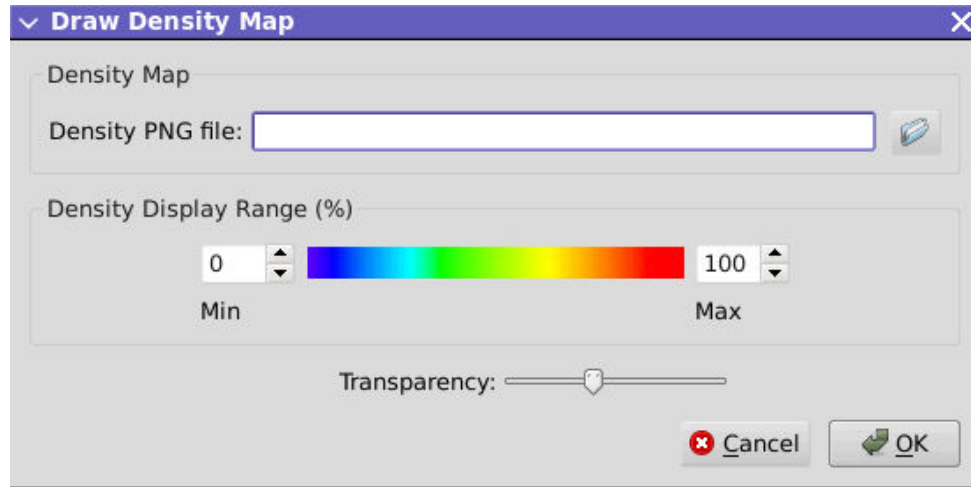
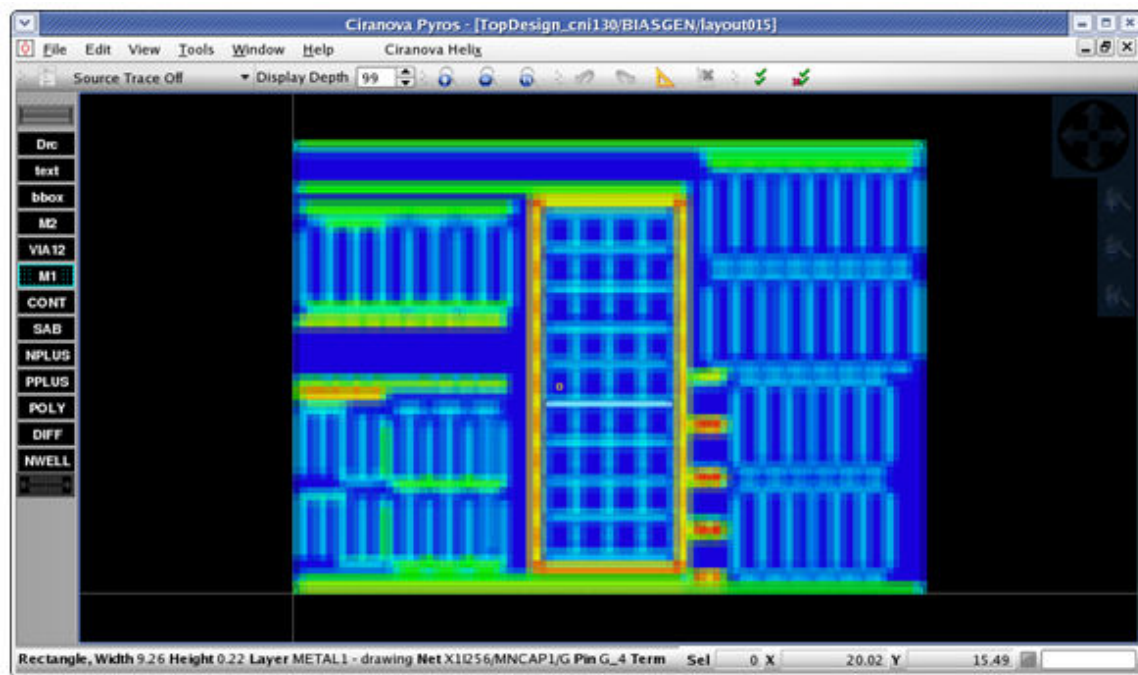


Table 7 Draw Density Map Dialog Box Options

Option	Description
Density PNG file	The density map file produced by the Density Analyzer.
Density Display Range (%)	The results can be filtered by setting the minimum (Min) or maximum (Max) density values to be shown. For example, if a density rule specifies that any value greater than or equal to 80% is an error, set the Min value to 80 to show only the problem areas.
Transparency	The density map transparency can be adjusted using this slider.

Figure 7 Density Map



System Options Dialog Box

Choose **Tools > Options** to display the **System Options** dialog box.

The left-side panel contains the list of options categories that can be modified:

- [System Options: PyStudio > Behaviour](#)
- [System Options: PyStudio > Font & Colors](#)
- [System Options: Display > General](#)
- [System Options: Display > Objects](#)
- [System Options: Grid > Settings](#)
- [System Options: Environment > Keyboard](#)

You must click **OK** or **Apply** to have the current options changed.

System Options: PyStudio > Behaviour

Use the options on this page to specify how the PyCell Studio IDE text editor handles tabs, indentation, and line wrapping.

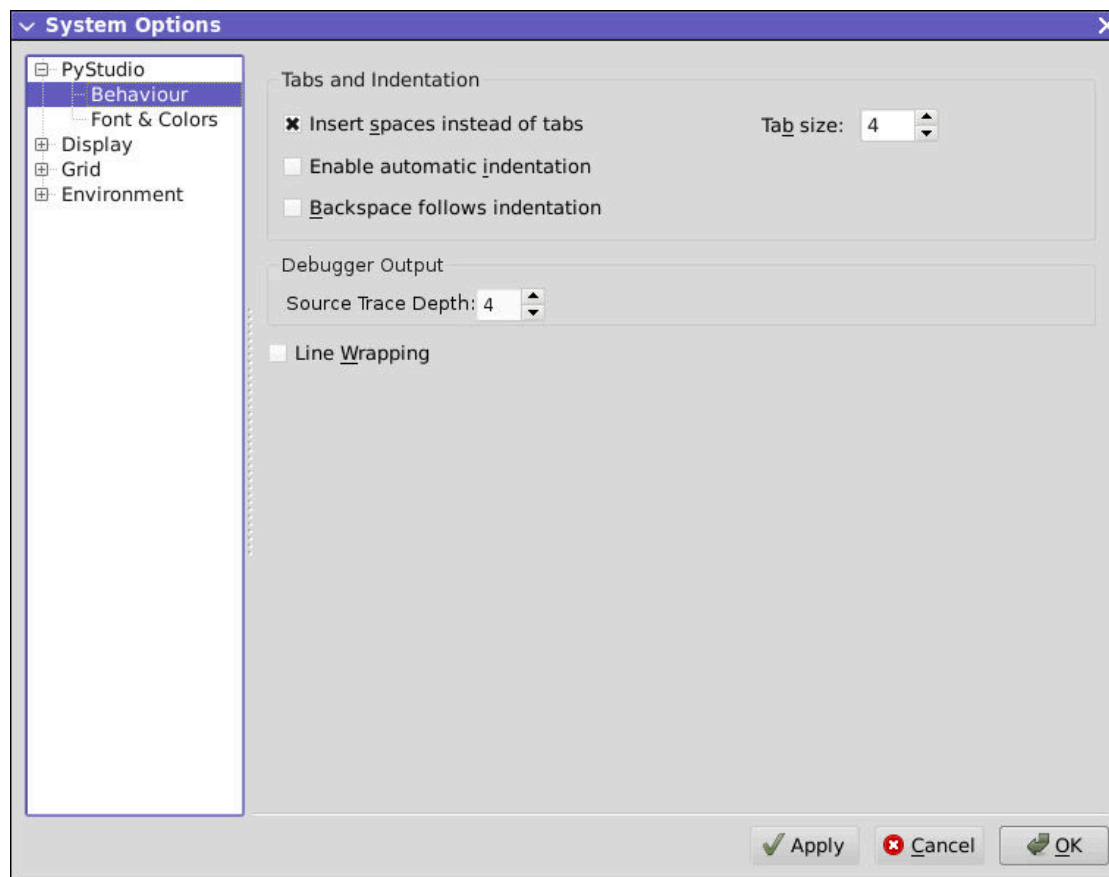


Table 8 System Options Dialog Box: PyStudio > Behaviour

Option	Description
Insert spaces instead of tabs	When enabled, the text editor converts each Tab you type into a number of spaces specified by the Tab size field. The default is 4.
Enable automatic indentation	When enabled, the text editor starts the cursor on a new line with the same indentation as the previous line. Otherwise, the text editor starts the cursor on a new line without any indentation.
Backspace follows indentation	When enabled, pressing Backspace at indentation deletes the number of spaces specified by the Tab size field with a single keystroke. Otherwise, pressing Backspace deletes one space.

Table 8 System Options Dialog Box: PyStudio > Behaviour (Continued)

Option	Description
Source Trace Depth	Specify the depth of the code stacks when the Source Trace toolbar command is ON. The default is 4.
Line Wrapping	When enabled, long lines of PyCell code which do not fit in the editor window are wrapped.

System Options: PyStudio > Font & Colors

Use the options on this page to specify how the PyCell Studio IDE text editor displays different categories of text.

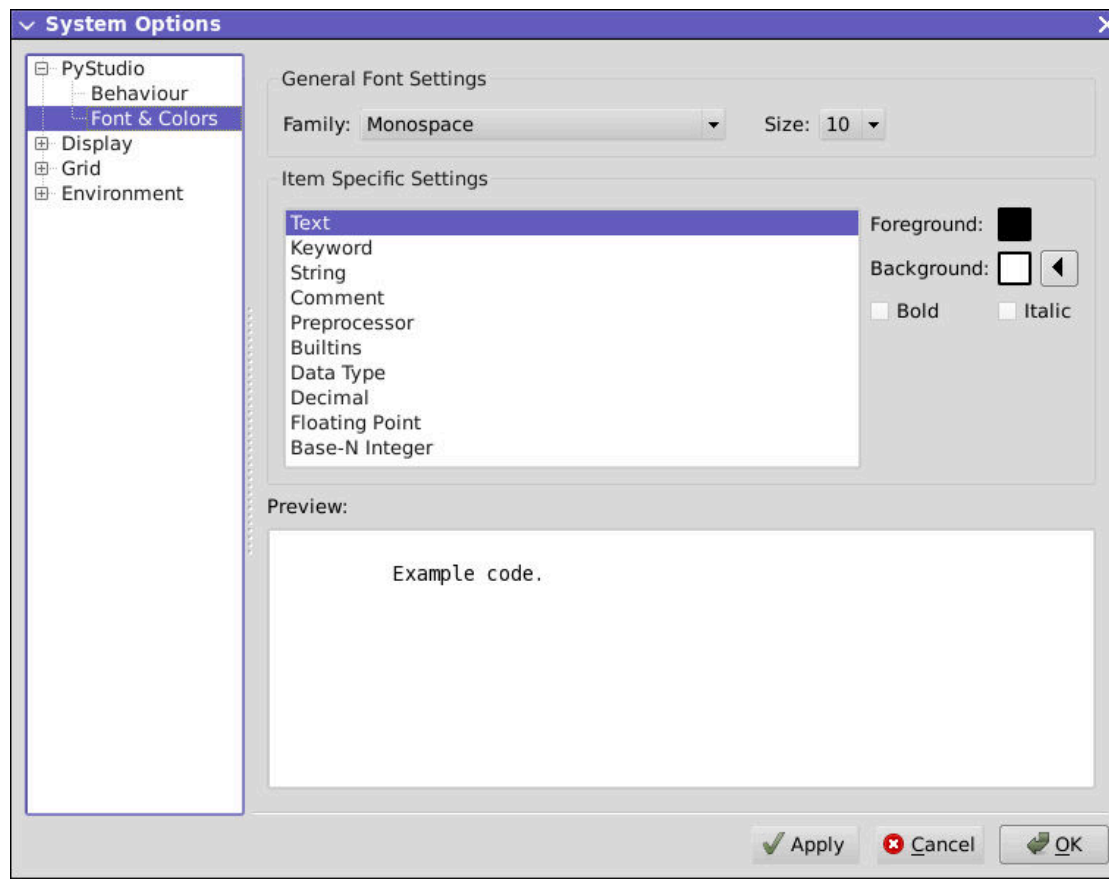


Table 9 System Options Dialog Box: PyStudio > Font & Colors

Option	Description
General Font Settings	Specify the font Family and Size for text displayed in the editor.
Item Specific Settings	For each category of text displayed in the editor, specify the Foreground and Background colors to use, as well as the Bold and Italic characteristics.

System Options: Display > General

Use the options on this page to configure general aspects of the Pyros viewer GUI.

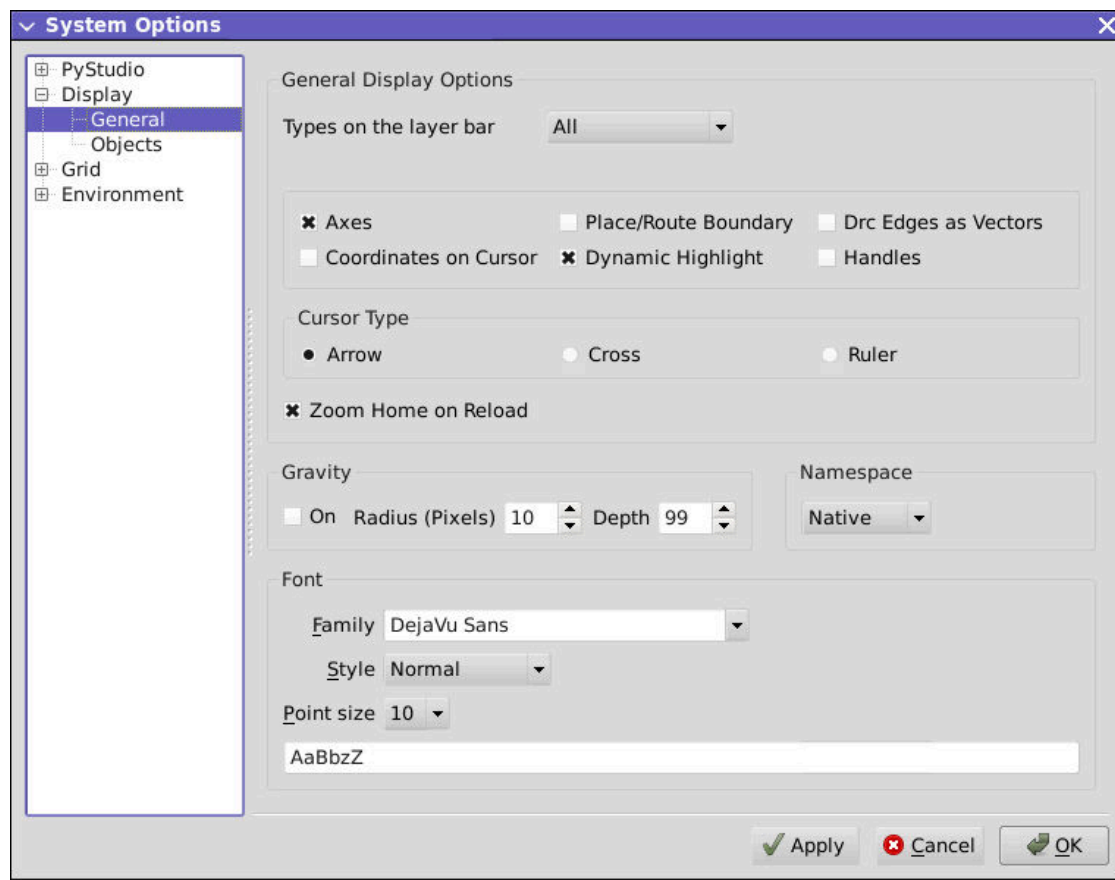


Table 10 System Options Dialog Box: Display > General Options

Option	Description
Types on the layer bar	This can be switched to put All used layers, only used Physical layers, or only used Metal layers.

Table 10 System Options Dialog Box: Display > General Options (Continued)

Option	Description
Axes	Show X and Y axes on the display.
Coordinates on Cursor	The current X/Y location is shown to the upper-right of the cursor.
Place/Route Boundary	Draw the oaPRBoundary object (if defined) for the current design. The color chosen is the color defined for the 'prBoundary' layer in the technology file.
Dynamic Highlight	As the cursor hovers over an object, it is highlighted on the display and its information is shown in the Status Bar . Press the spacebar to cycle through all of the objects under the cursor and to have the object information shown in the Information Window .
Drc Edges as Vectors	Specify how to display DRC error markers which are polygon edges. If this option is ON, a small arrow at the end of the edge is displayed in order to indicate the direction of the edge which helps to distinguish between the "inner" and "outer" side of the edge. For example, when you move the cursor from the beginning point of an edge to its end point, the "inner" side is on the left (that is, the original polygon to which the edge belonged was on this side).
Handles	Show or Hide stretch or abutment handles.
Cursor Type	Change the cursor type from an Arrow , to Cross-hairs , to a cross-hair Ruler with ruled dimensions.
Zoom Home on Reload	When ON, the Pyros viewer auto-zooms when the design is reloaded either by File > Reload or when debugging PyCell code and hitting a breakpoint. When debugging code, you can turn this option OFF so that the view does not change at each breakpoint.
Gravity	When On , when hovering over an object, the cursor snaps to an object's nearest edge. When outside an object, the cursor snaps to an object's edge if within the Radius (Pixels) value. Set the Depth to snap to objects at the specified hierarchy depth level
Namespace	Change the OpenAccess Namespace for displaying certain names, such as Pin or Instance names.
Font	Change the font (Family , Style , and Point size) used for all menus and dialog boxes.

System Options: Display > Objects

Use the options on this page to change how or whether certain objects are displayed.

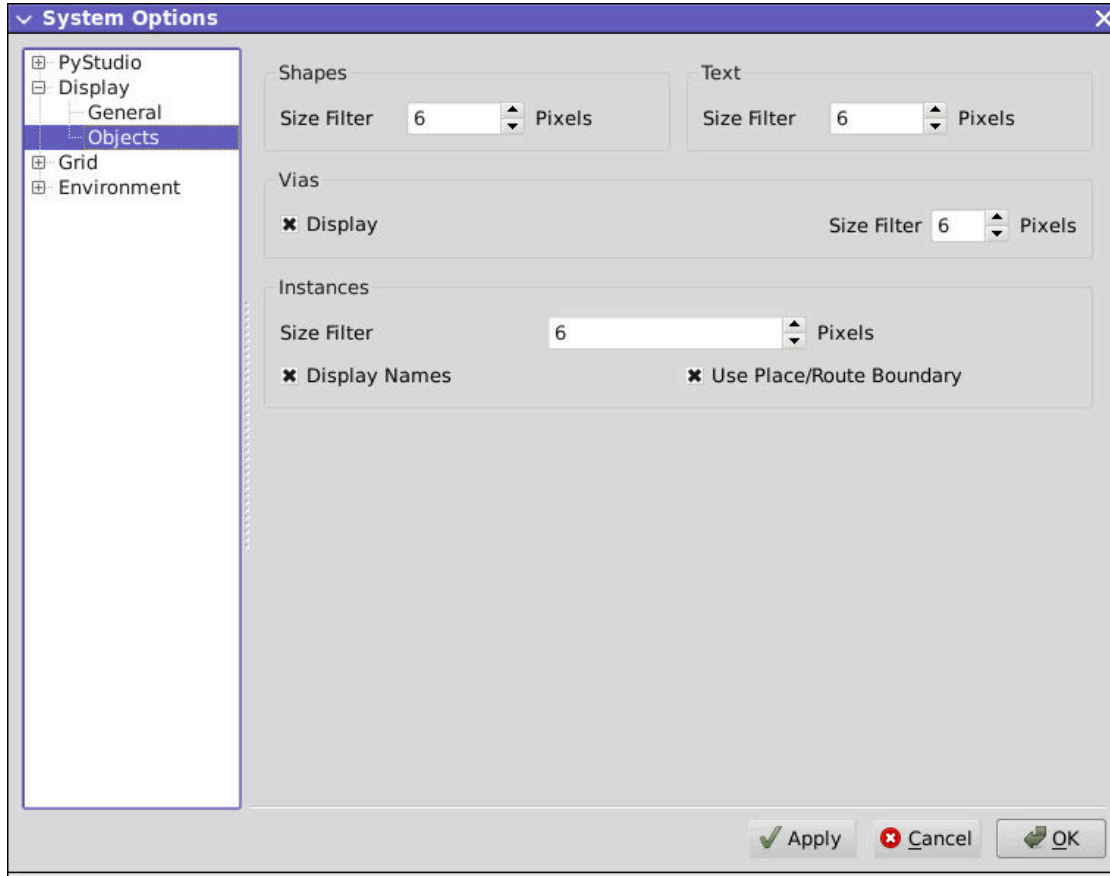


Table 11 System Options Dialog Box: Display > Objects

Option	Description
Shapes	Do not display shapes if their size in pixels falls below the Size Filter value.
Text	Do not display text if its size in pixels falls below the Size Filter value.
Vias	Specify whether to Display vias at all, and if so, do not display them if their size in pixels falls below the Size Filter value.
Instances	Do not display instances if their size in pixels falls below the Size Filter value. Also specify whether to Display Names and Use Place/Route Boundary .

System Options: Grid > Settings

Use the options on this page to specify how the Pyros viewer grid is displayed.

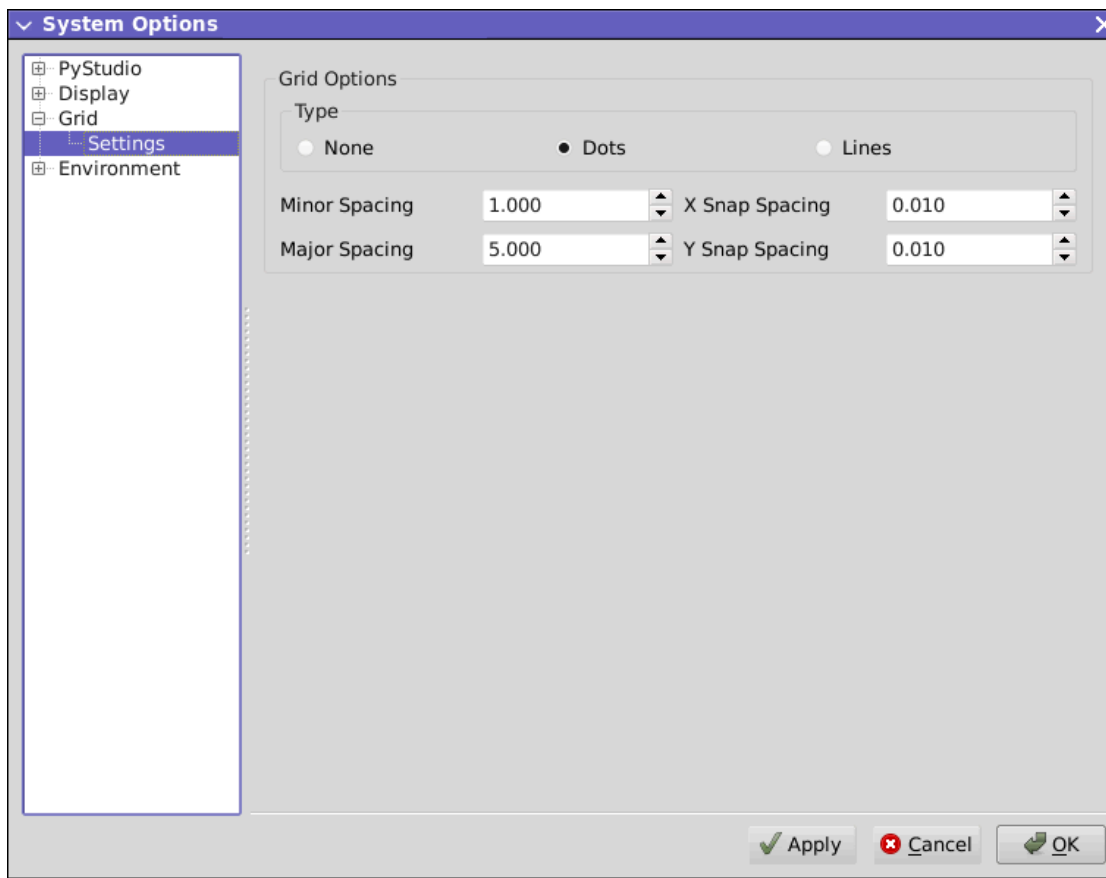


Table 12 System Options Dialog Box: Grid > Settings

Option	Description
Type	Switch from showing None (no grid), to Dots (a dotted grid) or Lines (a lined grid).
Minor Spacing	Set the minor display grid spacing. Minor dot grids are smaller than major grids. Minor line grids have lesser intensity.
Major Spacing	Set the major display grid spacing.
X Snap Spacing	Set the invisible cursor snap grid in the X direction.
Y Snap Spacing	Set the invisible cursor snap grid in the Y direction.

System Options: Environment > Keyboard

Use this page to add or edit keyboard shortcuts for Pyros viewer commands.

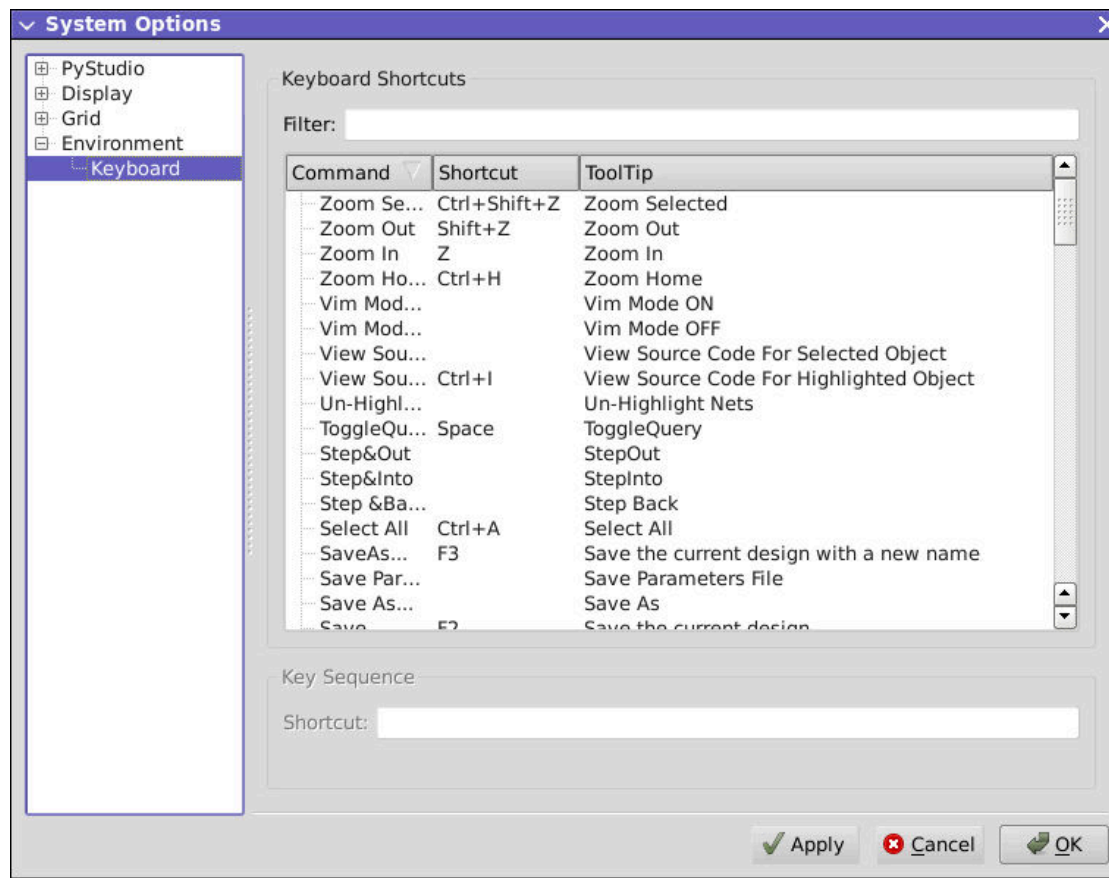
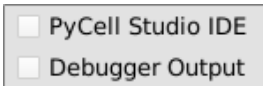
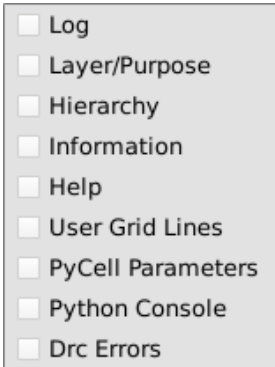


Table 13 System Options Dialog Box: Environment > Keyboard

Option	Description
Keyboard Shortcuts	Filter the list of commands displayed. Click the column headings, Command , Shortcut , or ToolTip to sort the list by the column.
Key Sequence	After selecting a command from the list, add or edit the Shortcut .

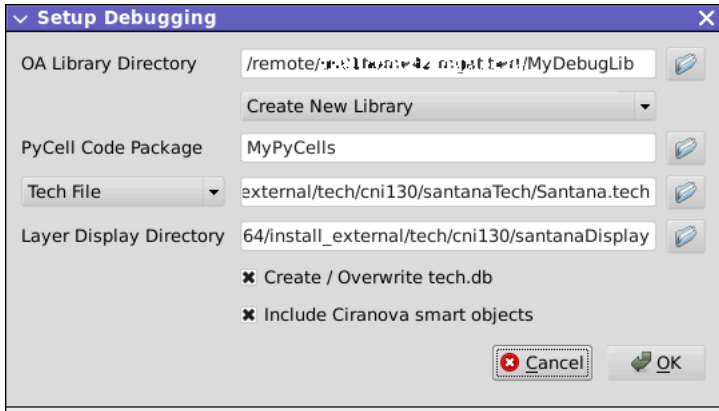
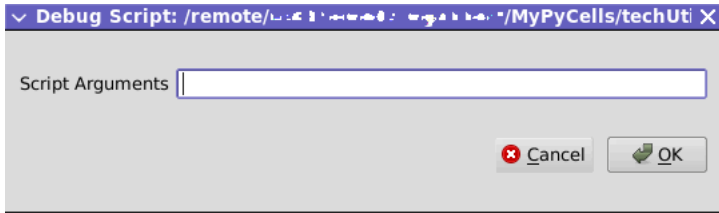
Window Menu

Table 14 Window Menu

Name	Key	Description
Tile		Tile all the canvas windows
Cascade		Cascade all the canvas windows
Next	Shift+W	Activate the next canvas window
Previous	W	Activate the previous canvas window
Descend	Shift+E	Descends one level in the hierarchical design.
Ascend	Ctrl+E	Ascends one level in the hierarchical design.
Debugger Windows		<p>Submenu with a list of debugger windows that can be hidden or displayed. See Using PyCell Studio IDE.</p>  <p>Note: This menu option is only visible when running PyCell Studio IDE (cndbg).</p>
Dock Windows		<p>Submenu with a list of windows that can be hidden or shown. See Using Pyros Viewer Windows.</p> 

PyCell Studio Menu (PyCell Studio IDE Only)

Table 15 PyCell Studio Menu

Name	Description
Training Example	Creates a training example using a placeholder 130nm library (cni130) for you in your current working directory. This helps you to learn and explore PyCell Studio IDE features and how to code PyCells. See Debugging PyCells for more information.
Debug Existing Code	Allows you to set up your PyCell library for debugging by defining which OA library to debug using what PyCell code. This command displays the Setup Debugging dialog box.
	
Debug Python Script	Debug a single Python module open in the editor window. This command displays the Debug Script dialog box.
	
Debug Drc Script	Debug a sequence of DRC operations implementing a design rule in the technology file. You can see step-by-step how local derived layers are created, and how DRC error markers are generated. Note: Global derived layers defined in the technology file are available to use in the DRC script.

Help Menu

Table 16 Help Menu

Name	Key	Description
Help Contents	F1	Open the PyCell Studio online help menu in a compatible HTML browser. Most of the PyCell Studio manuals are delivered as PDF files, so you must also configure a compatible PDF reader.
About		List the version.
Frequently Asked Questions		Display the Help window with the list of frequently asked questions.

Pyros Viewer Toolbar Commands

The Pyros viewer supports two commands that appear only on the toolbar, as well as others that replicate menu commands.



Table 17 Pyros Viewer Toolbar Commands

Name	Description
Source Trace OFF ON	Disable or enable source tracing when loading or changing parameters on a design or instance.
Display Depth	Set the hierarchy depth to display.
Zoom In	Zoom in. Same as View > Zoom In .
Zoom Out	Zoom out by 2x viewing factor. Same as View > Zoom Out .
Zoom Home	Zoom out to entire view of the design. Same as View > Zoom Home .
Undo	Undo the previous editing operation. Same as Edit > Undo .
Redo	Redo the previous editing operation. Same as Edit > Redo .
Add Ruler	Select two points to add a ruler. Same as Edit > Add Ruler .
Delete Rulers	Delete all rulers. Same as Edit > Delete Rulers .
Run Drc	Run the internal DRC tool. Same as Tools > Drc > Run Drc .
Clear Drc Markers	Clear all DRC error marker geometries. Same as Tools > Drc > Clear Drc Markers .

5

Using Pyros Viewer Windows

This chapter describes the Pyros viewer windows (and status bar).

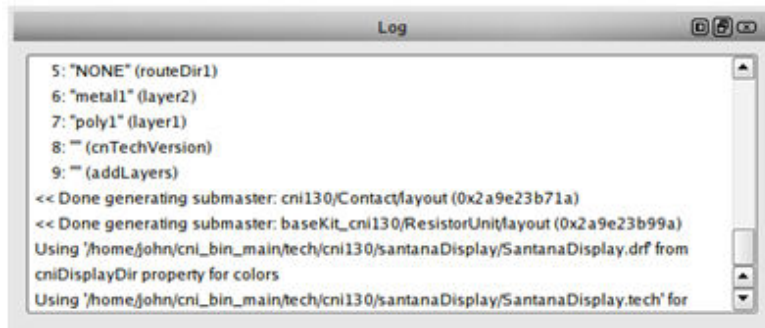
From the Pyros viewer, choose **Window > Docking Windows** to display or hide the following windows:

- [Log Window](#)
- [Layer/Purpose Window](#)
- [Hierarchy Window](#)
- [Information Window](#)
- [User Grid Lines Window](#)
- [PyCell Parameters Window](#)
- [Python Console Window](#)
- [Drc Errors Window](#)
- [Status Bar](#)

Log Window

All messages sent to the `cni` log buffers can be viewed in the **Log** window.

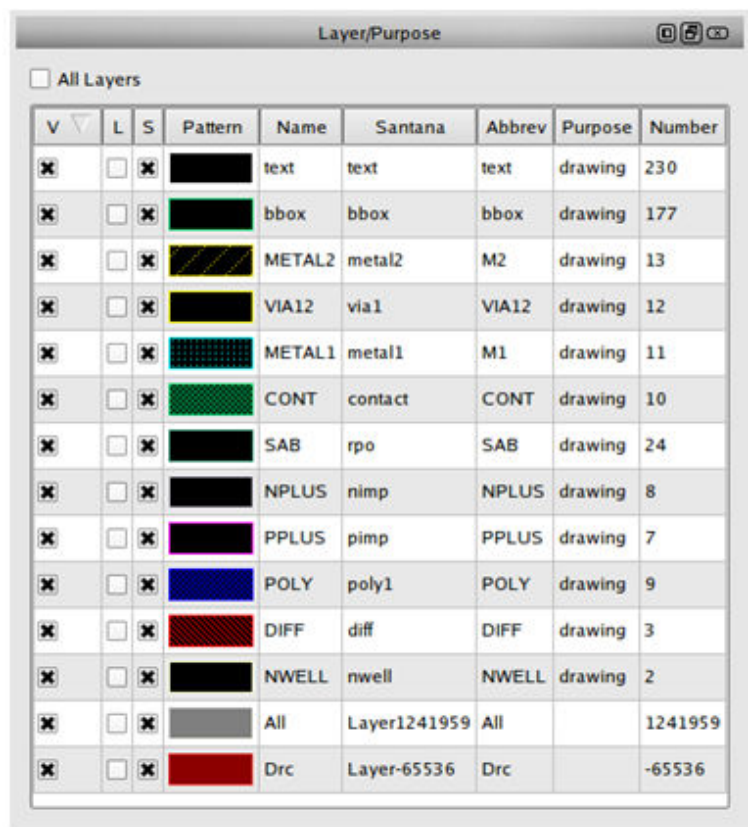
Figure 8 Log Window



Layer/Purpose Window

The **Layer/Purpose** window shows the active layer purpose pairs (LPPs) or all defined layers by selecting the **All Layers** option.

Figure 9 *Layer/Purpose Window*



The screenshot shows the 'Layer/Purpose' window with the 'All Layers' checkbox selected. The window contains a table with the following columns: V, L, S, Pattern, Name, Santana, Abbrev, Purpose, and Number. The table lists 14 layers, including text, bbox, METAL2, VIA12, METAL1, CONT, SAB, NPLUS, PPLUS, POLY, DIFF, NWELL, All, and Drc.

V	L	S	Pattern	Name	Santana	Abbrev	Purpose	Number
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		text	text	text	drawing	230
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		bbox	bbox	bbox	drawing	177
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		METAL2	metal2	M2	drawing	13
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		VIA12	via1	VIA12	drawing	12
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		METAL1	metal1	M1	drawing	11
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		CONT	contact	CONT	drawing	10
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		SAB	rpo	SAB	drawing	24
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NPLUS	nimp	NPLUS	drawing	8
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		PPLUS	pimp	PPLUS	drawing	7
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		POLY	poly1	POLY	drawing	9
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		DIFF	diff	DIFF	drawing	3
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NWELL	nwell	NWELL	drawing	2
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		All	Layer1241959	All		1241959
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		Drc	Layer-65536	Drc		-65536

Right-clicking brings up a menu for showing or hiding columns.

Figure 10 *Show/Hide Columns*

- 
- The screenshot shows a context menu with the following items, each preceded by a checked checkbox:
- ☒ Visible
 - ☒ Locked
 - ☒ Selectable
 - ☒ Pattern
 - ☒ Name
 - ☒ SantanaName
 - ☒ Abbreviation
 - ☒ Purpose
 - ☒ Number

The entries can be sorted by a particular column by clicking the column heading.

The **Layer** toolbar shows all currently 'active' layers (layers with data).

Figure 11 Layer Toolbar



Clicking an icon brings up a dialog box for changing the current view setting for that layer:

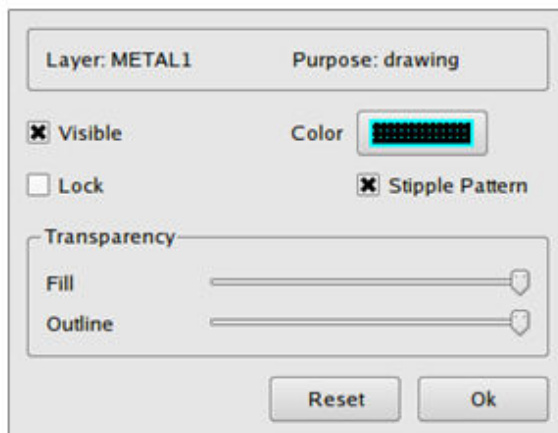


Table 18 View Setting Dialog Box Options

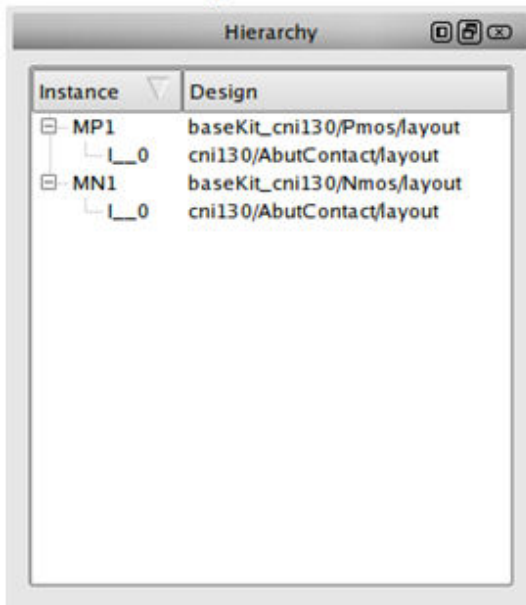
Option	Description
Visible	Turn visibility ON or OFF.
Color	Display a color chart to change the layer color.
Lock	Lock the current settings for this layer. Any changes made using the All Layers toolbar button do not affect this layer.
Stipple Pattern	Switch between solid and stipple fill pattern.
Fill Transparency Slider	Adjust the fill transparency from 0% (invisible) to 100% (fully opaque).
Outline Transparency Slider	Adjust the outline transparency from 0% (invisible) to 100% (fully opaque).

Right-clicking a layer button in the toolbar makes that layer visible and all other layers invisible. Right-clicking again makes all layers visible. Clicking with the middle mouse button toggles the visibility of that layer ON and OFF. Rotating the mouse wheel changes the transparency of the layer fill and outline colors.

Hierarchy Window

The **Hierarchy** window shows the instance hierarchy of the current design.

Figure 12 Hierarchy Window



Selected instances in the canvas window are highlighted in the list.

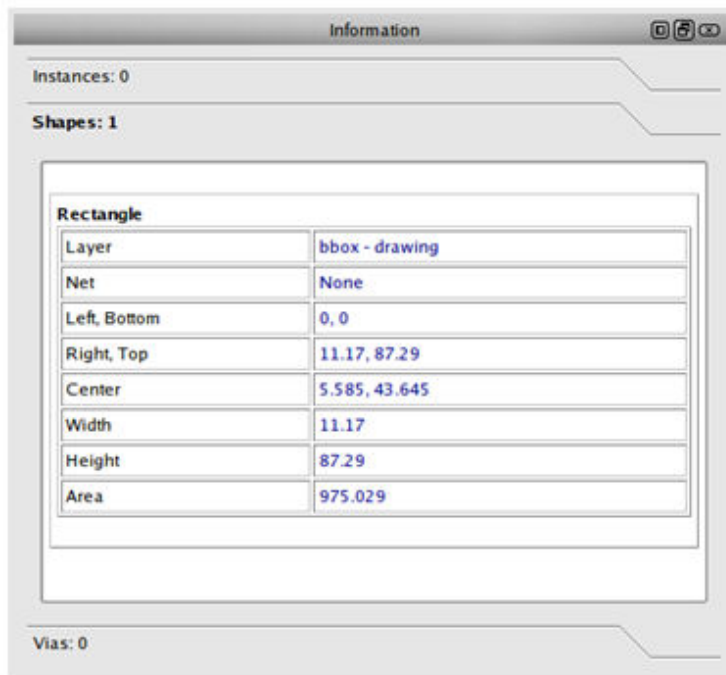
Clicking an entry in the list selects the appropriate instance in the canvas window.

Double-clicking an entry opens the master design of that instance inside a new canvas window.

Information Window

The **Information** window shows information about the currently selected or highlighted object.

Figure 13 Information Window

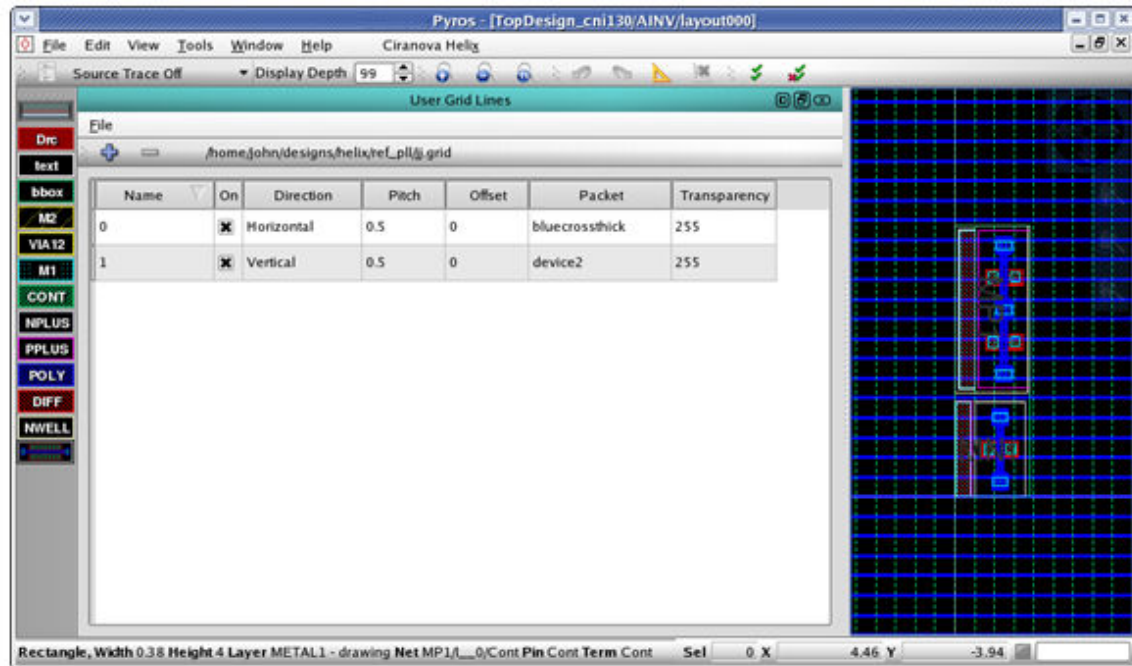




Only objects at the current level can be selected, but objects at any level can be highlighted by positioning the cursor over the object and pressing the Spacebar. In order to cycle through shapes, smallest to largest, press the Spacebar repeatedly.

User Grid Lines Window

The **User Grid Lines** window is used to define and display user-definable line grids.

Figure 14 User Grids Window



Select **Add User Grid**  to add a new user grid, or **Delete User Grid**  to remove the currently selected user grid.

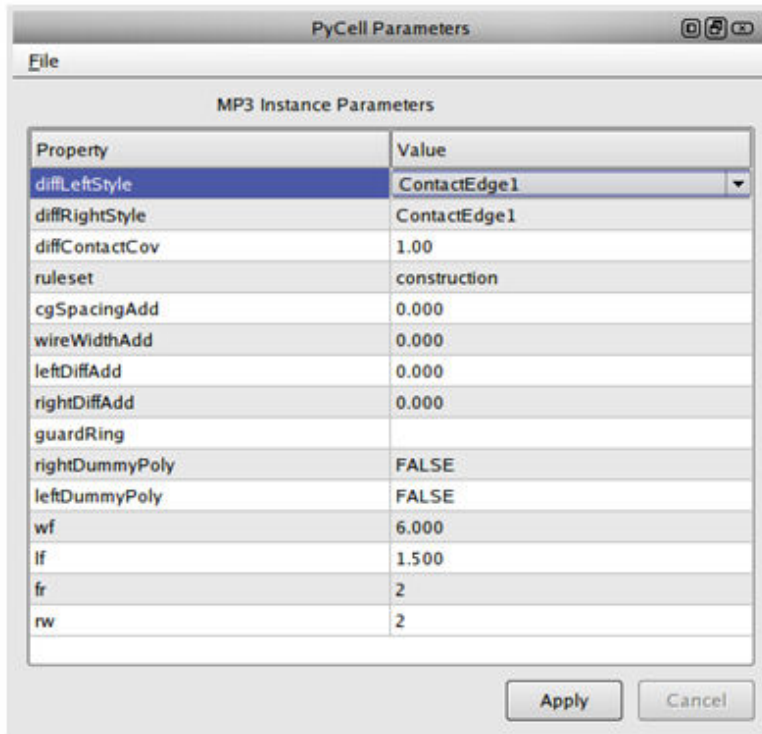
The user grids must have unique names.

Direction (Horizontal or Vertical), **Pitch**, **Offset**, **Packet**, and **Transparency** can all be defined for each grid. The set of grids can be saved to a file and loaded from a file by using the **File** menu.

PyCell Parameters Window

When a selected instance has parameters, the **PyCell Parameters** window is filled with the current parameter values for that instance. Clicking **Apply** causes OpenAccess to generate a new submaster for the instance.

Figure 15 PyCell Parameters Windows



The **PyCell Parameters** window contains one menu, **File**, with the following options:

Table 19 PyCell Parameters Window — File Menu Options

Name	Key	Description
Restore Default Parameters	F8	Retrieve the default parameter values from the master of the instance but do not apply them.
Load Parameters File		Load a previously saved set of parameter values.
Save Parameters File		Save the current set of parameter values to a file.

Note that as a convenience, parameters are shown and can be changed for library cells or any other design supermasters (such as contained in the "cnPyBasicDlo" library).

Parameter values can then be modified and applied to generate different submaster designs for this supermaster design cell. The updated layout is shown in the main viewing window.

The PyCell author should use constraints as much as possible when parameters are defined, as this provides a more user-friendly **PyCell Parameters** window. For example, if a parameter is defined using a *choice* constraint to specify the allowed values, then the **PyCell Parameters** window generates a pull-down list having only these elements specified by the *choice* constraint as the allowed values.

Debugging While Changing Parameters

When running the PyCell Studio IDE (`cndbg`) the **PyCell Parameters** window contains an additional menu: **Debug**.

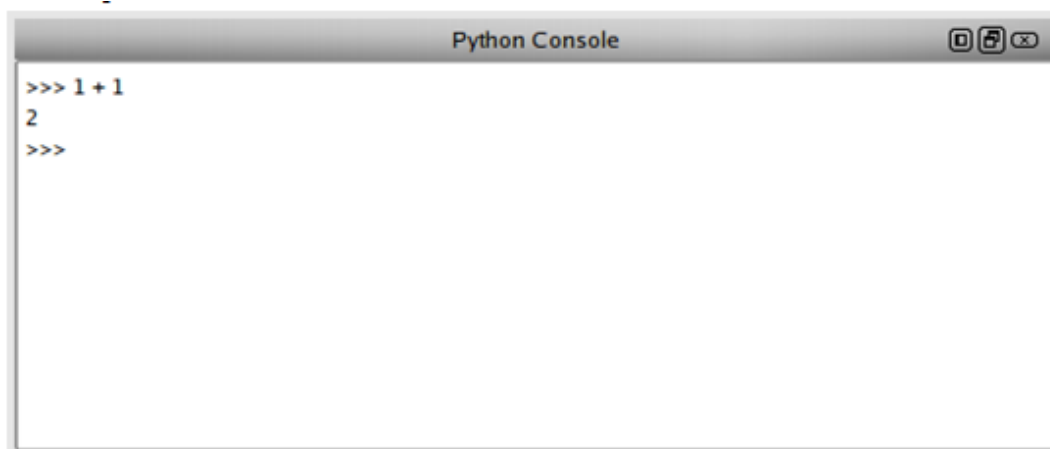
See the section on [Viewing Source Code that Built a PyCell Object](#) to load the Python source code for the PyCell design into the editor and set breakpoints where you want the debugger to stop.

Set the parameter values you want to change, and instead of clicking the **Apply** button, click the entry under the **Debug** menu to start debugging the PyCell evaluation with the new parameter values.

Python Console Window

This is a general-purpose standalone **Python Console** window running the Python interpreter.

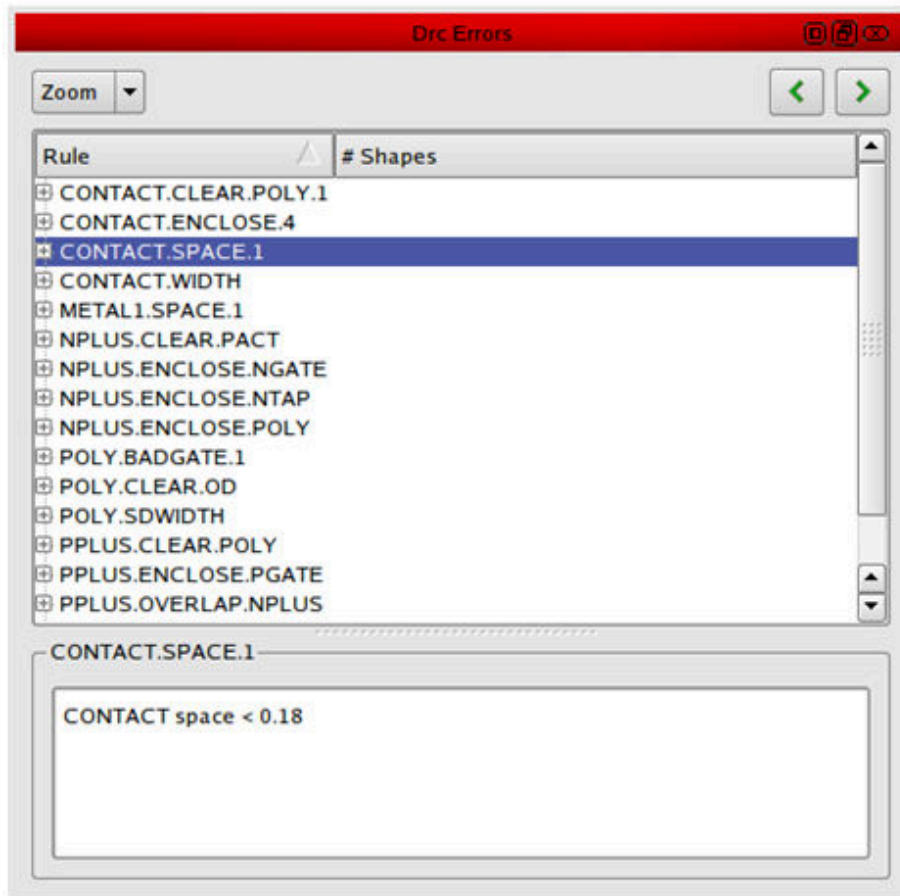
Figure 16 Python Console Window



Drc Errors Window

Results from a DRC run are listed in the **Drc Errors** window.

Figure 17 DRC Errors Window



Selected DRC violation objects in the canvas highlight their appropriate entry in this window. Clicking a DRC entry in the window selects the appropriate DRC violation geometries in the canvas.

The Left and Right arrow buttons can be used to step from one DRC violation to the next. This selects the DRC violation in both the canvas window and in the **Drc Errors** window. The **Zoom** and **Pan** combo box controls the zooming and panning operations in the canvas window.

Clearing the DRC markers clears the contents of this window. The **Drc Errors** window is also updated each time the interactive DRC program is run.

Note that running DRC also supports technology file versioning. If DRC is run for a submaster or supermaster design which supports technology versioning, then the value of the `cnTechVersion` parameter is used automatically when running DRC. When DRC is run on non-PCell designs, and multiple technology versions are available in the current technology file, then a dialog box is displayed for you to select the technology version which should be used to run the DRC program.

Status Bar

In addition to the many supported Pyros viewer windows, the status bar shows objects that are under the cursor if the **Dynamic Highlight** option (under **Tools > Options, Display > General**) is on.

Other messages also appear here, such as when editing objects.

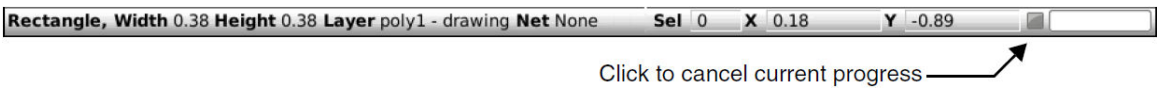


Table 20 Status Bar

Key	Description
Sel	Number of objects currently selected.
X,Y	Coordinates of the cursor in the current design.

6

Using PyCell Studio IDE

This chapter describes the PyCell Studio Interactive Debugging Environment (IDE).

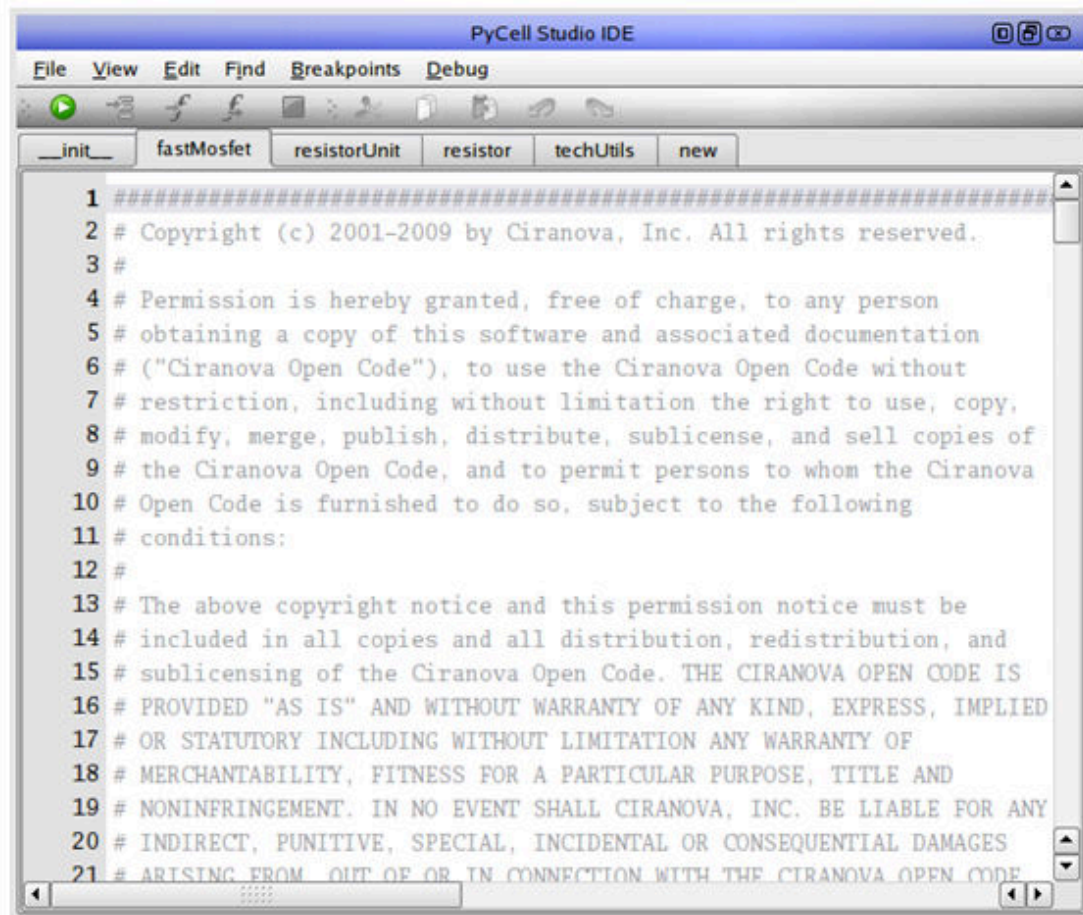
In addition to the Pyros viewers menus and windows described in this manual, the PyCell Studio IDE (`cndbg`) supports the following:

- [PyCell Studio IDE Window](#)
- [PyCell Studio IDE Menus](#)
- [PyCell Studio IDE Toolbar Commands](#)
- [Debugger Output Window](#)

PyCell Studio IDE Window

As noted in [Debugging PyCells](#), the **PyCell Studio IDE** window (cndbg) is used for viewing or editing Python source code files.

Figure 18 *PyCell Studio IDE Window*



It also contains all of the functions for the integrated debugger. Breakpoints can be set (or cleared) from the menu commands or by clicking to the left of the source code line numbers.

You can go directly to the definition of a class, function, or variable by selecting its name and choosing **Find > Go To Definition** (shortcut Ctrl+D) or right-clicking the selected name and choosing **Go To Definition** from the menu.

During a debug session, autocompletion of variable names and object attributes is enabled when you make code changes in the editor window.

Figure 19 *Text Editor Autocompletion*

```
758     contactLayer = layers.contact
759     metallLayer = layers.metall
760     metal2layer = l|
761
762     techGetP(lambdas, n.getPhysicalRule
763     minArea    layers
764     minClear   len
765     minClear   license
766     minClear   license
767     minClear   license
768     minClear   license
769     minClear   license
770     minClear   license
771     minClear   license
772     minClear   license
773     minClear   license
774     minClear   license
775     minClear   license
776     minClear   license
777     minClear   license
778     minClear   license
779     minClear   license
780     minClear   license
781     minClear   license
782     minClear   license
783     minClear   license
784     minClear   license
785     minClear   license
786     minClear   license
787     minClear   license
788     minClear   license
789     minClear   license
790     minClear   license
791     minClear   license
792     minClear   license
793     minClear   license
794     minClear   license
795     minClear   license
796     minClear   license
797     minClear   license
798     minClear   license
799     minClear   license
800     minClear   license
801     minClear   license
802     minClear   license
803     minClear   license
804     minClear   license
805     minClear   license
806     minClear   license
807     minClear   license
808     minClear   license
809     minClear   license
810     minClear   license
811     minClear   license
812     minClear   license
813     minClear   license
814     minClear   license
815     minClear   license
816     minClear   license
817     minClear   license
818     minClear   license
819     minClear   license
820     minClear   license
821     minClear   license
822     minClear   license
823     minClear   license
824     minClear   license
825     minClear   license
826     minClear   license
827     minClear   license
828     minClear   license
829     minClear   license
830     minClear   license
831     minClear   license
832     minClear   license
833     minClear   license
834     minClear   license
835     minClear   license
836     minClear   license
837     minClear   license
838     minClear   license
839     minClear   license
840     minClear   license
841     minClear   license
842     minClear   license
843     minClear   license
844     minClear   license
845     minClear   license
846     minClear   license
847     minClear   license
848     minClear   license
849     minClear   license
850     minClear   license
851     minClear   license
852     minClear   license
853     minClear   license
854     minClear   license
855     minClear   license
856     minClear   license
857     minClear   license
858     minClear   license
859     minClear   license
860     minClear   license
861     minClear   license
862     minClear   license
863     minClear   license
864     minClear   license
865     minClear   license
866     minClear   license
867     minClear   license
868     minClear   license
869     minClear   license
870     minClear   license
871     minClear   license
872     minClear   license
873     minClear   license
874     minClear   license
875     minClear   license
876     minClear   license
877     minClear   license
878     minClear   license
879     minClear   license
880     minClear   license
881     minClear   license
882     minClear   license
883     minClear   license
884     minClear   license
885     minClear   license
886     minClear   license
887     minClear   license
888     minClear   license
889     minClear   license
890     minClear   license
891     minClear   license
892     minClear   license
893     minClear   license
894     minClear   license
895     minClear   license
896     minClear   license
897     minClear   license
898     minClear   license
899     minClear   license
900     minClear   license
901     minClear   license
902     minClear   license
903     minClear   license
904     minClear   license
905     minClear   license
906     minClear   license
907     minClear   license
908     minClear   license
909     minClear   license
910     minClear   license
911     minClear   license
912     minClear   license
913     minClear   license
914     minClear   license
915     minClear   license
916     minClear   license
917     minClear   license
918     minClear   license
919     minClear   license
920     minClear   license
921     minClear   license
922     minClear   license
923     minClear   license
924     minClear   license
925     minClear   license
926     minClear   license
927     minClear   license
928     minClear   license
929     minClear   license
930     minClear   license
931     minClear   license
932     minClear   license
933     minClear   license
934     minClear   license
935     minClear   license
936     minClear   license
937     minClear   license
938     minClear   license
939     minClear   license
940     minClear   license
941     minClear   license
942     minClear   license
943     minClear   license
944     minClear   license
945     minClear   license
946     minClear   license
947     minClear   license
948     minClear   license
949     minClear   license
950     minClear   license
951     minClear   license
952     minClear   license
953     minClear   license
954     minClear   license
955     minClear   license
956     minClear   license
957     minClear   license
958     minClear   license
959     minClear   license
960     minClear   license
961     minClear   license
962     minClear   license
963     minClear   license
964     minClear   license
965     minClear   license
966     minClear   license
967     minClear   license
968     minClear   license
969     minClear   license
970     minClear   license
971     minClear   license
972     minClear   license
973     minClear   license
974     minClear   license
975     minClear   license
976     minClear   license
977     minClear   license
978     minClear   license
979     minClear   license
980     minClear   license
981     minClear   license
982     minClear   license
983     minClear   license
984     minClear   license
985     minClear   license
986     minClear   license
987     minClear   license
988     minClear   license
989     minClear   license
990     minClear   license
991     minClear   license
992     minClear   license
993     minClear   license
994     minClear   license
995     minClear   license
996     minClear   license
997     minClear   license
998     minClear   license
999     minClear   license
1000    minClear   license
```

PyCell Studio IDE Menus

The [PyCell Studio IDE Window](#) is controlled through the following menus and commands:

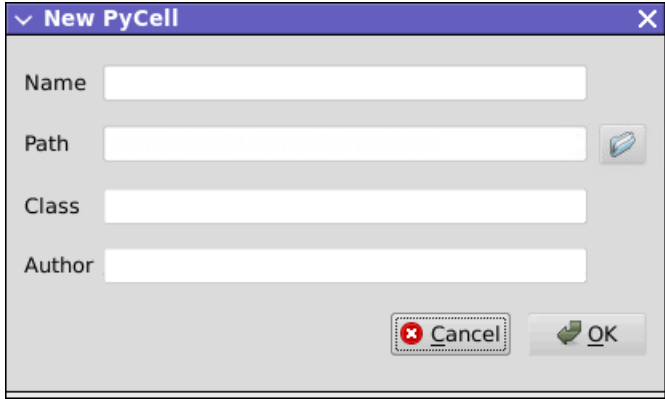
- [File Menu](#)
- [Edit Menu](#)
- [Tools Menu](#)
- [Find Menu](#)
- [Breakpoints Menu](#)
- [Debug Menu](#)

Note:

When using the PyCell Studio IDE, the Pyros viewer menus also include the [PyCell Studio Menu \(PyCell Studio IDE Only\)](#).

File Menu

Table 21 File Menu

Name	Key	Description
New File	Ctrl+N	Create a new PyCell file. This command displays the New PyCell dialog box.
		
Open File	Ctrl+O	Open a PyCell file.
Save File	Ctrl+S	Save the active PyCell file.
Save As		Save a copy of the active PyCell file with a new filename.
Save All Files	Ctrl+Shift+S	Save all open PyCell files.
Close File	Ctrl+Shift+W	Close the active PyCell file.
Close All Files		Close all open PyCell files.

Edit Menu

Table 22 Edit Menu

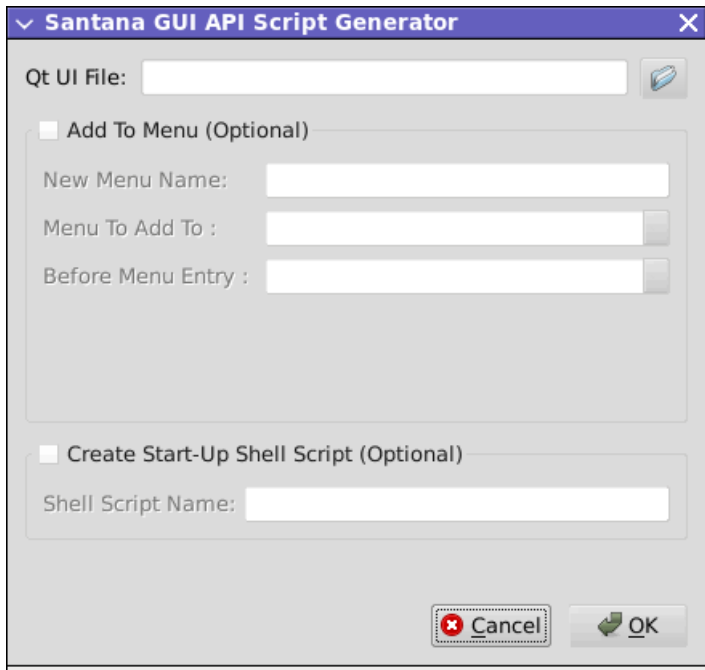
Name	Key	Description
Undo	Ctrl+Z	Undo the previous action.
Redo	Ctrl+Y	Repeat the previous action.
Cut	Ctrl+X	Copy the selected portion of text from the editor to a temporary buffer and delete the original text.

Table 22 Edit Menu (Continued)

Name	Key	Description
Copy	Ctrl+C	Copy the selected portion of text from the editor to a temporary buffer while retaining the original text.
Paste	Ctrl+V	Paste the copied text from a temporary buffer to the editor.
Delete		Delete the selected portion of text.
Select All	Ctrl+A	Select all text in the editor.

Tools Menu

Table 23 Tools Menu

Name	Description
Qt Designer	Invokes Qt Designer, a graphical user interface designer for Qt applications.
Generate Script Template	Generate a script template. This command displays the Santana GUI API Script Generator dialog box.
	
Run Script	Run a script.

Find Menu

Table 24 Find Menu

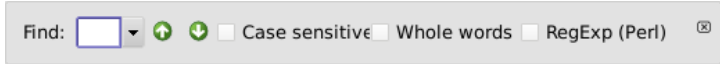
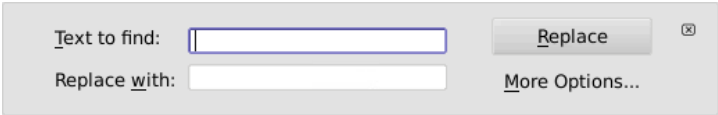
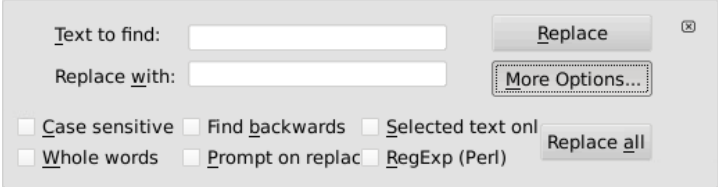
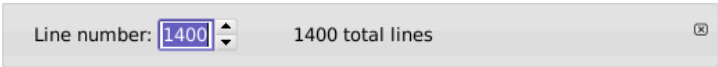
Name	Key	Description
Find	Ctrl+F	<p>Display the Find dialog box at the bottom of the editor to search the text for the specified string.</p>  <ul style="list-style-type: none"> • Use the arrow icons to Find Previous and Find Next. • The following options are supported to refine your search: <ul style="list-style-type: none"> • Case sensitive: Search distinguishes between uppercase and lowercase letters. • Whole words: Search ignores partial matches. • RegExp (Perl): Search using Perl language style regular expressions.
Find Next	F3	Find the next instance of the search term.
Find Previous	Shift+F3	Fine the previous instance of the search term.
Clear Search Highlight		Clear any highlighting of text resulting from a successful search.

Table 24 Find Menu (Continued)

Name	Key	Description
Replace	Ctrl+R	<p>Display the Replace dialog box at the bottom of the editor to search the text for the specified string and replace it with another.</p>  <p>Click More Options to expand the available Replace options.</p>  <ul style="list-style-type: none"> • The following options are supported to refine your search: • Case sensitive: Search distinguishes between uppercase and lowercase letters. • Whole words: Search ignores partial matches. • Find backwards: Search as with Find Previous. • Prompt on replace: After finding a match, ask for confirmation from you before replacing it. • Selected text only: Search only the highlighted text rather than the entire file. • RegExp (Perl): Search using Perl language style regular expressions. <p>Click Replace All to replace every instance of the search string with the replacement string.</p>
Go To Line	Ctrl+G	<p>Use the Go To Line dialog box to display the specified line number.</p> 
Go To Definition	Ctrl+D	<p>Go directly to the definition of a class, function, or variable by selecting its name and choosing this option.</p>

Breakpoints Menu

Table 25 Breakpoints Menu

Name	Key	Description
Toggle Breakpoint	F9	Set or remove a breakpoint at the current line.

Table 25 Breakpoints Menu (Continued)

Name	Key	Description
Clear All Breakpoints		Clear all breakpoints set in the code.

Debug Menu

Table 26 Debug Menu

Name	Description
Debug	Run the active PyCell file in the debugger. The debugger stops if it hits a breakpoint.
Step	Execute the current line of code, then move to the next line and stop. If the current line is a call to a user-defined function, execute the entire user-defined function as a unit, then move to the next line of the current procedure and stop. Note: In some debuggers, this is called <i>StepOver</i> .
StepInto	Execute the current line of code, then move to the next line and stop. If the current line is a call to a user-defined function, move to the first line of the user-defined function and stop.
StepOut	If the current line of code is in a user-defined function called by another procedure, execute the remaining lines of codes in the user-defined function, then move to the next line of the calling procedure and stop.
StopDebugger	Stop running the debugger.

PyCell Studio IDE Toolbar Commands

The PyCell Studio IDE Window features the following toolbar commands that replicate some menu commands.



Table 27 Pyros Viewer Toolbar Commands

Name	Description
Run in the debugger	Same as Debug > Debug . Run the active PyCell file in the debugger. The debugger stops if it hits a breakpoint.

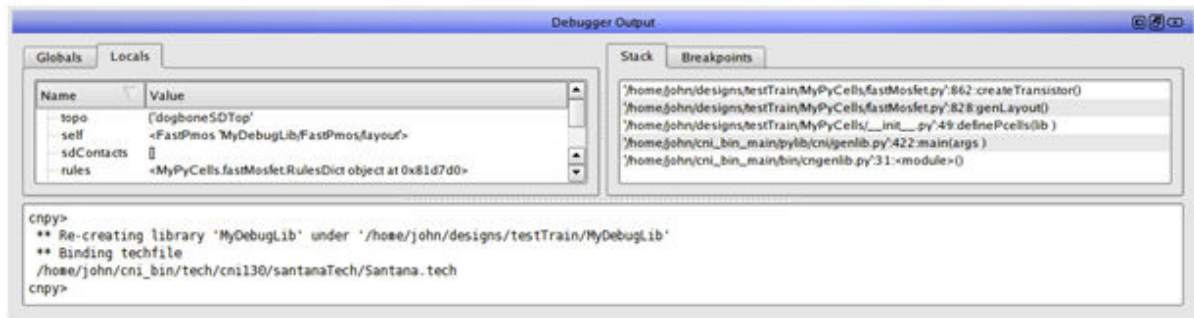
Table 27 Pyros Viewer Toolbar Commands (Continued)

Name	Description
Step	<p>Same as Debug > Step.</p> <p>Execute the current line of code, then move to the next line and stop.</p> <p>If the current line is a call to a user-defined function, execute the entire user-defined function as a unit, then move to the next line of the current procedure and stop.</p> <p>Note: In some debuggers, this is called <i>StepOver</i>.</p>
StepInto	<p>Same as Debug > StepInto.</p> <p>Execute the current line of code, then move to the next line and stop.</p> <p>If the current line is a call to a user-defined function, move to the first line of the user-defined function and stop.</p>
StepOut	<p>Same as Debug > StepOut.</p> <p>If the current line of code is in a user-defined function called by another procedure, execute the remaining lines of codes in the user-defined function, then move to the next line of the calling procedure and stop.</p>
StopDebugger	<p>Same as Debug > StopDebugger.</p> <p>Stop running the debugger.</p>
Cut	<p>Same as Edit > Cut.</p> <p>Copy the selected portion of text from the editor to a temporary buffer and delete the original text.</p>
Copy	<p>Same as Edit > Copy.</p> <p>Copy the selected portion of text from the editor to a temporary buffer while retaining the original text.</p>
Paste	<p>Same as Edit > Paste.</p> <p>Paste the copied text from a temporary buffer to the editor.</p>
Undo	<p>Same as Edit > Undo.</p> <p>Undo the previous action.</p>
Redo	<p>Same as Edit > Redo.</p> <p>Repeat the previous action.</p>
New File	<p>Same as File > New File.</p> <p>Create a new PyCell file.</p>
Open File	<p>Same as File > Open File.</p> <p>Open a PyCell file.</p>
Save File	<p>Same as File > Save File.</p> <p>Save the active PyCell file.</p>

Debugger Output Window

The **Debugger Output** window opens when running the PyCell Studio IDE and contains all of the outputs from the debugging process.

Figure 20 Debugger Output Window



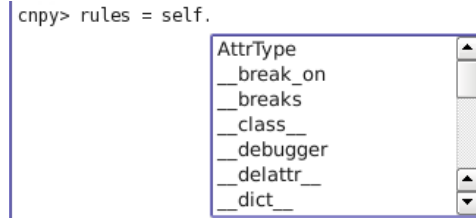
The upper left section contains the current **Global** or **Local** variables and their values.

The upper right section contains the current **Stack** position and the list of defined **Breakpoints**. Clicking a stack line positions the editor on that line of code. Clicking a breakpoint entry positions the editor on the line of code containing that breakpoint.

The bottom section contains the Python console for the process currently being debugged. Output from the process is listed here. You can also enter any Python statements, for example, to print the values of variables.

During a debug session, autocompletion of object attributes is enabled when you type an object name followed by a period.

Figure 21 Debugger Output Autocompletion



7

Using Stretch Handles and Auto-Abutment

The Pyros viewer also supports the advanced editing features of stretch handles and auto-abutment, which are available in many layout editing tools. These features allow you to graphically modify the layout generated by a PyCell, and then have the relevant PyCell parameters automatically updated.

This chapter discusses the following Pyros viewer features:

- [Using Stretch Handles](#)
- [Using Auto-Abutment](#)

Using Stretch Handles

The Pyros viewer supports graphical stretch handles which can be attached to different shapes in the generated layout for a PyCell. These graphical stretch handles are represented by small diamond shapes attached to different locations of the shape, which can be selected using the mouse and then moved to expand or contract the shape.

The graphical stretch handle can either be moved in a NORTH_SOUTH direction or in an EAST_WEST direction. When the graphical stretch handle is de-selected, then the corresponding PyCell parameter is automatically updated to reflect the new size for the shape.

For example, the following example screen shots show how the length of the poly gate rectangle for an NmosH transistor can be changed using these stretch handles.

In [Figure 22](#), stretch NmosH transistor poly gate rectangle to the right, to increase the length parameter.

Figure 22 Stretch Length (Before)

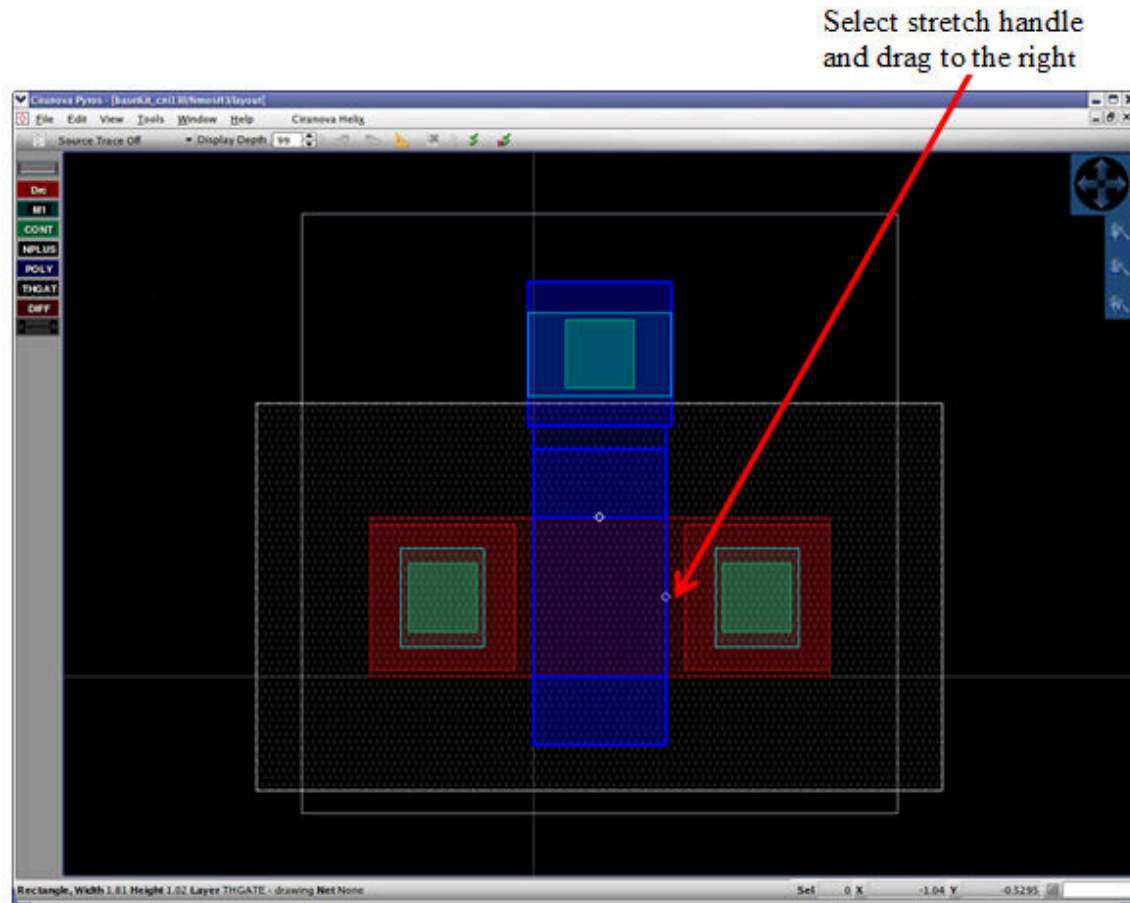
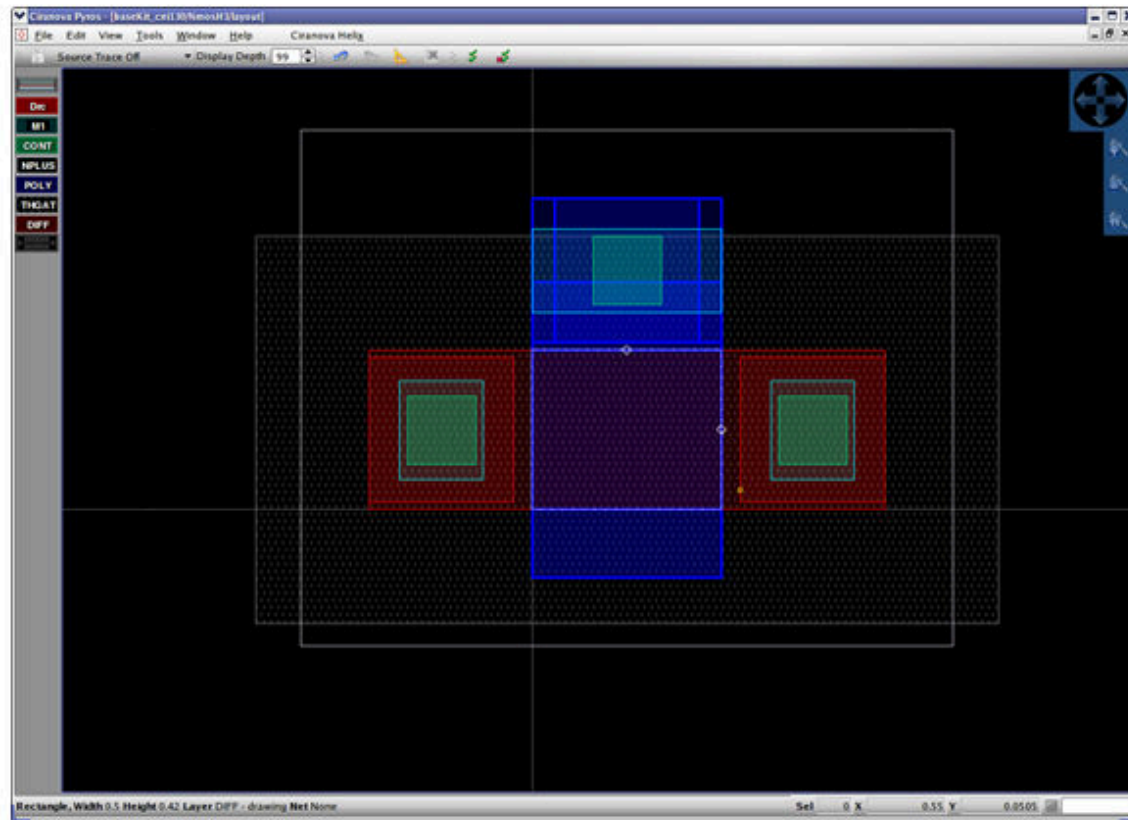


Figure 23 shows the NmosH transistor with stretched poly gate rectangle and length PyCell parameter value updated.

Figure 23 *Stretch Length (After)*



Using Auto-Abutment

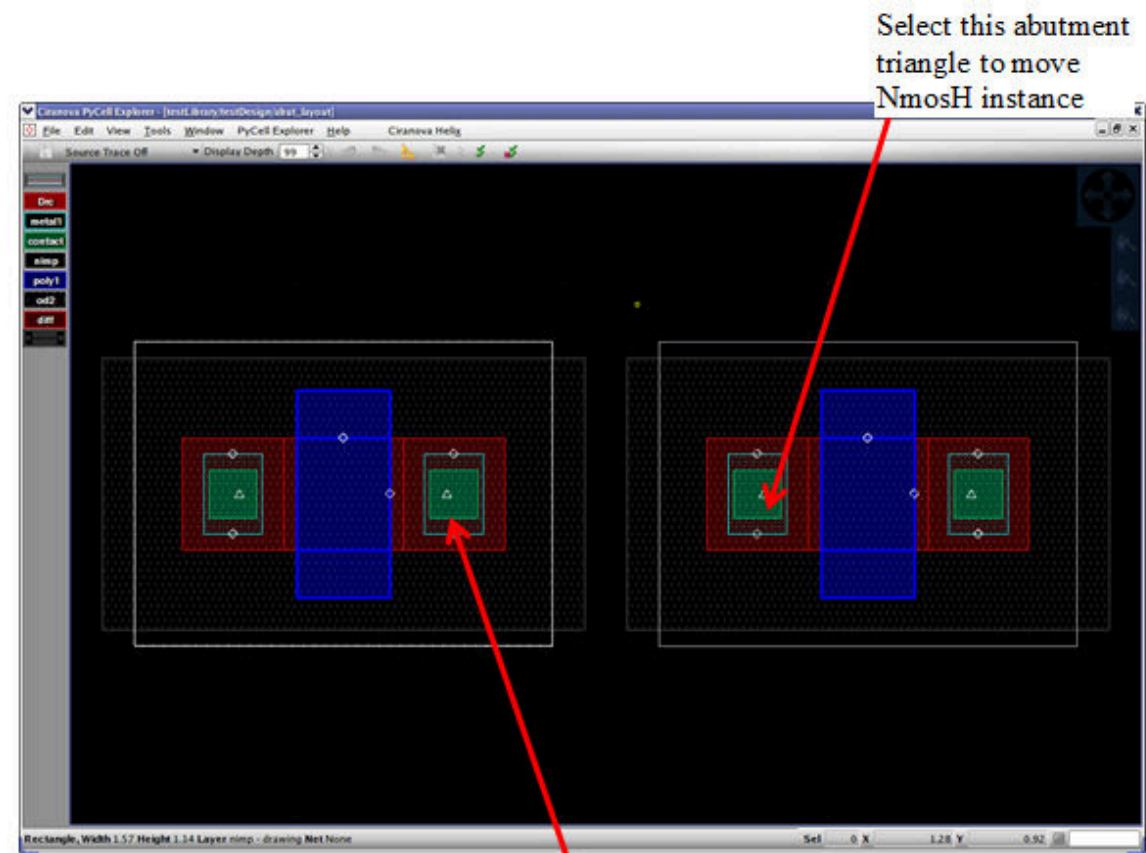
The Pyros viewer supports graphical auto-abutment handles, which are attached to shapes which can be abutted in the generated layout for a PyCell. These graphical auto-abutment handles are represented by small triangle shapes which are attached to the center location of the shape, which can be selected by the mouse and then moved to place the shape to overlap another shape in the layout which can be abutted.

This should be done by placing this abutment triangle directly on top of the abutment triangle for the other shape in the layout. Note that this is different from some layout editing tools, which may only require that the two shapes to be abutted are merely overlapped, and do not make use of such graphical abutment triangles.

For example, the following example screen shots show how the source and drain diffusion for two NmosH transistor instances can be abutted (so that source and drain diffusion can be shared to minimize layout area) using these graphical abutment triangles.

Figure 24 shows two NmosH transistor instances with same dimensions. Use abutment triangles to place diffusion rectangle of second instance to overlap first instance diffusion.

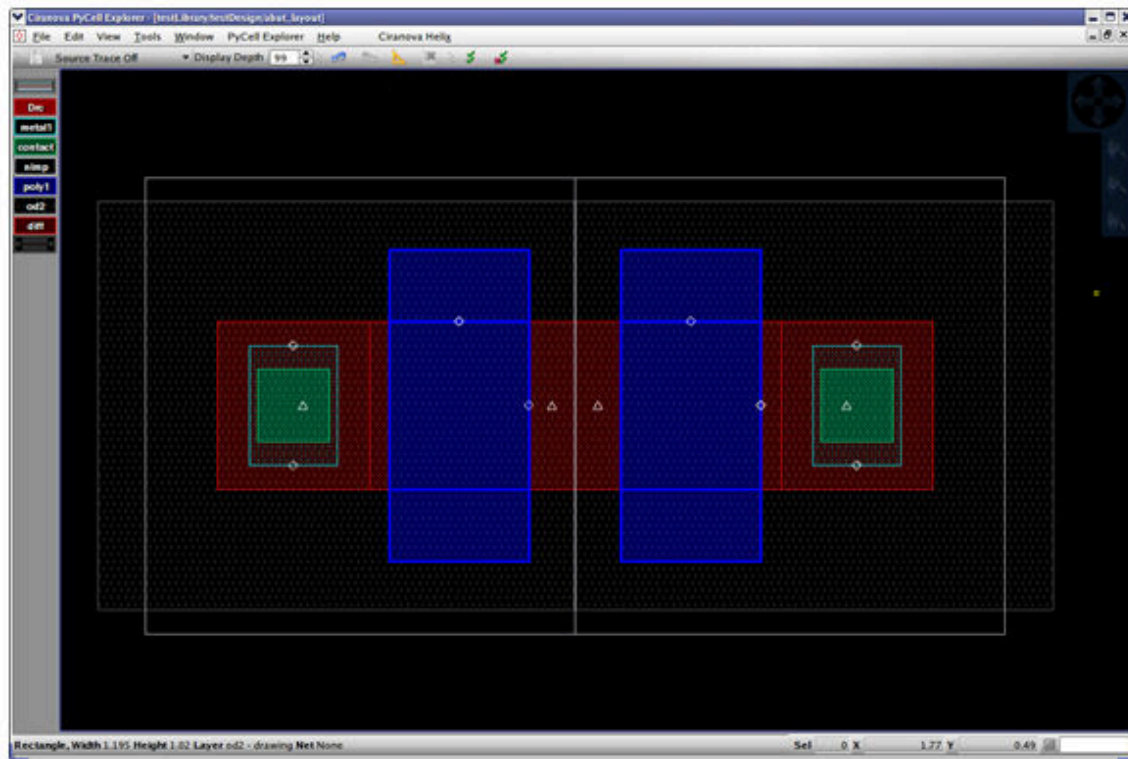
Figure 24 Auto-Abutment (Before)



Drag the abutment triangle over to the other abutment triangle. Select OK to the prompt to abut the two instances.

Figure 25 shows two NmosH transistor instances with abutted source and drain diffusion, which is now shared.

Figure 25 Auto-Abutment (After)



A

Santana Technology Display File

This appendix discusses the Santana technology display file and its Layer Purpose Pair display structure.

The Santana technology display file is discussed in the following sections:

- [Creating the Santana Technology Display File](#)
- [Layer Purpose Pair Display Structure](#)

Creating the Santana Technology Display File

The Santana technology display file is an ASCII text file that contains display information for the process technology information stored in the Santana technology file. Instead of combining technology and display information into a single large file, two separate files are used for this purpose. The Santana technology file stores all of the process related technology information, while the Santa technology display file only stores display information which is used by the graphical tools, such as the Pyros graphical layout viewer.

The Santana technology file can be created in a number of different ways:

- Use the information contained in this section to create this technology display file using a standard text editor program.
- Convert an existing Cadence DF II technology file into the Santana technology display file format, using a conversion utility which is supplied by Synopsys. The technology display information contained in the Cadence DF II technology file is converted and stored in this separate technology display file.
- Use a combination of these approaches, where some sections of the Santana technology display file are created through the conversion utility, and other sections are created using a text editor.

Note that in some cases, it may be necessary to modify the data which is generated by the conversion utility. Thus, a typical approach to creating the Santana technology file would involve using a combination of the conversion utility and text editing.

The Santana technology display text file is defined using the standard syntax of "Symbolic expressions" ("S-expressions"). Parentheses are used to group data into different sections, so that each section can then be viewed as a standalone element.

Note that the Santana technology display file contains a single predefined section:

```
LAYER PURPOSE PAIR DEFINITION section:  
dispLayerPurposePairs  
Defines how purposes should be displayed
```

After this Santana technology display file has been completed, it should be stored in a directory, along with the corresponding Display Resource file (.drf file). This Display Resource file contains display information which is also used by the Santana technology display file; that is why these two files should be located in the same directory. The environment variable CNI_DISPLAY_DIR should then be set to the location of this directory, before the graphical tools are invoked.

Layer Purpose Pair Display Structure

The layer purpose pair (LPP) display group for the Santana technology display file contains all of the display information about the layer purpose pairs which are displayed in the graphical tools. This display group is used to define Layer Purpose Pairs, along with the "packet" display information, which specifies how each Layer Purpose Pair should be displayed in the graphical tools.

This purpose display group makes use of a single section, which is defined by the `dispLayerPurposePairs` section keyword, described as follows:

Description: The `dispLayerPurposePairs` section is used to provide all of the information which is needed to graphically display a Layer Purpose Pair in the graphical environment. This information includes the name of the layer, the name of the purpose and the name of the "packet" which specifies how the Layer Purpose Pair should be displayed. In addition, there are two Boolean fields, the Visible and Selectable fields, which can be used to control the visibility and selectability of the Layer Purpose Pair.

Note that for the current release of the graphical tools, these Visible and Selectable field values are not used to control visibility and selectability; however, they can still be specified in the Santana technology display file. Also note that the "blink" attribute which is defined in the Display Resource file for each packet is not currently used by the graphical tools.

The name of the layer should be the layer name which was used when this layer was defined in the Santana technology file, or should be the predefined name used by any predefined system layer. The name of the purpose should be the purpose name which was used when this purpose was defined in the Santana technology file, or should be the predefined name used by any predefined system purpose. The name of the "packet" should be the packet name which is defined in the Display Resource file (.drf file) which

is associated with this Santana technology display file. This packet name field should be unique, so that each different Layer Purpose Pair defined in this section has a unique packet name. The Visible and Selectable fields are Boolean fields; the values for these two fields are specified using "t" and "f" characters

Example

```
dispLayerPurposePairs (
; ( layerName      Purpose      Packet      Vis      Sel      )
; ( -----      -
```

layerName	Purpose	Packet	Vis	Sel
User-Defined Layer Purpose Pairs				
pwell	drawing	creamshortDash	t	t
nwell	drawing	creamthickLine2	t	t
diff	drawing	readbackSlash_S	t	t
poly	drawing	bluecrossthick	t	t
contact	drawing	greencross_S	t	t
metall	drawing	cyandot4_S	t	t
metall	pin	cyanXthickLine2	t	t
metall	dummy	cyandot4thickLine2	t	t
vial	drawing	creamshortDash	t	t
metal2	drawing	creamshortDash	t	t
via2	drawing	creamshortDash	t	t
metal3	drawing	forestdot4_S	t	t
flightLine	drawing	white	t	t
drc	drawing	reddot3_S	t	t
System-Reserved Layer Purpose Pairs				
background	drawing	background	t	f
text	drawing	text	t	t
border	drawing	border	t	t
y0	drawing	y0	t	t
y1	drawing	y1	t	t
y2	drawing	y2	t	t
y3	drawing	y3	t	t
y4	drawing	y4	t	t
y5	drawing	y5	t	t
y6	drawing	y6	t	t
y7	drawing	y7	t	t
y8	drawing	y8	t	t
y9	drawing	y9	t	t
pin	drawing	pin	t	t
hilite	drawing	hilite	t	t
select	drawing	select	t	t
drive	drawing	drive	t	t
hiz	drawing	hiz	t	t
resist	drawing	resist	t	t
supply	drawing	supply	t	t
unknown	drawing	unknown	t	t
y0	flight	y0Flt	t	t
y1	flight	y1Flt	t	t

Appendix A: Santana Technology Display File
Layer Purpose Pair Display Structure

```
( y2      flight  y2Flt      t      t      )
( y3      flight  y3Flt      t      t      )
( y4      flight  y4Flt      t      t      )
( y5      flight  y5Flt      t      t      )
( y6      flight  y6Flt      t      t      )
( y7      flight  y7Flt      t      t      )
( y8      flight  y8Flt      t      t      )
( y9      flight  y9Flt      t      t      )
); dispLayerPurposePairs
```

B

References

This appendix provides a reference for environment variables and the .cnitools directory.

This appendix discusses the following:

- [Environment Variables](#)
- [.cnitools Directory](#)

Environment Variables

The following variables are used:

Name	Purpose	Default
CNI_ROOT	Path where main installation hierarchy resides	none
PYTHONPATH	Path for Python modules	none
CNI_DISPLAY_DIR	Path where Santana display files reside	none
CNI_MAX_EVAL_TIMEOUT_S	Specifies the maximum PCell evaluation time limit (in seconds). This value is applied to each PCell evaluation.	no time limit
CNI_MAX_EVAL_USE_ALARM	Specifies how to time PCell evaluation: <ul style="list-style-type: none">• 0: CPU time (default)• 1: Elapsed time	0

The variable CNI_ROOT must be set properly to the location of the main installation directory or else the application does not function at all.

CNI_DISPLAY_DIR is the location of the following GUI display files:

- `santanaDisplay.tech` contains the technology related display information such as drawing order and layer abbreviations.
- `santanaDisplay.drf` contains colors and packet information. If the `santanaDisplay.drf` file does not exist, then the GUI reads the `display.drf` file, if it exists.

If the CNI_DISPLAY_DIR environment variable is not defined, then the GUI reads the OpenAccess technology database to see if the path to the Santana technology and display files are stored there. Lastly, the GUI checks for the existence of embedded technology display information and for a local `display.drf` file.

The product ships with a default technology node called `cni130` which includes display files. So CNI_DISPLAY_DIR can be set to:

```
${CNI_ROOT}/tech/cni130/santanaDisplay'
```

when using the `cni130` technology.

.cnitools Directory

The `.cnitools` directory is automatically created by the Pyros tools, and is used to store information about persistent configuration files, log files, and replay files. These configuration files contain start-up settings and information about the activation state of windows, as well as the precise placement for these active windows in the overall Pyros display. The log files contain logging information concerning the various Pyros-based tools. By default, the `.cnitools` directory is automatically saved in your local working directory. In addition, the `.cnitools` directory can be copied to your HOME directory or a corporate directory location.

The order in which the tool searches for a `.cnitools` folder is:

1. The path location specified by the `--systemScopePath path` command line argument.
2. The current working directory.
3. Your HOME directory.

Changing Start-Up Defaults

Start-up default settings such as grid settings, font, and so on can be changed by following this procedure:

1. Create an empty directory.
2. Change to that directory and start up the tool.

3. In the tool, change the values to their new default values, open, close or resize windows as necessary, and then exit the tool.
4. Copy the local `.cnitools` folder to your HOME directory or the corporate defaults path.
5. If a corporate path is used, start the tool with the `--systemScopePath path` command line argument.

Using Replay Files

Replay files contain the precise commands that were executed during the previous tool session. These replay files can later be used to replay the previous session. This is done by means of the `--rp` replay and `--rps` replay step (use Spacebar to step) command line options.

Note that the `--commands` command line option can be used to list all of the commands currently supported. In the case in which an unexpected bug is discovered, these replay files can sometimes be used by the development team to reproduce the problem.