CleWin 4.1

User Guide

CleWin 4.1 User Guide

Publisher

WieWeb software

Managing Editor

Cisca Weber

Technical Editors

Remco Wiegerink

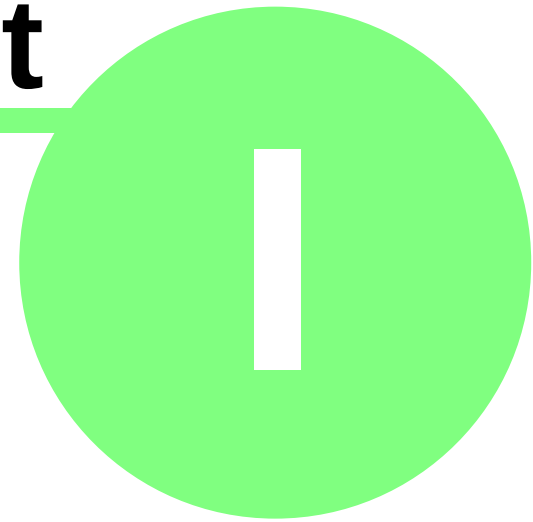
Cisca Weber

Table of Contents

CleWin 4.1

User Guide

Part



1 Introduction

CleWin 4.0 is a layout editor designed to run under the Windows 2000, XP and Vista operating systems. CleWin uses the well-known Caltech Intermediate Format (CIF) as its native file format. Furthermore, CleWin can read and write Calma GDS-II files. Both CIF and GDS-II are supported by all major layout editors, thus layouts designed in CleWin can easily be imported in other applications.

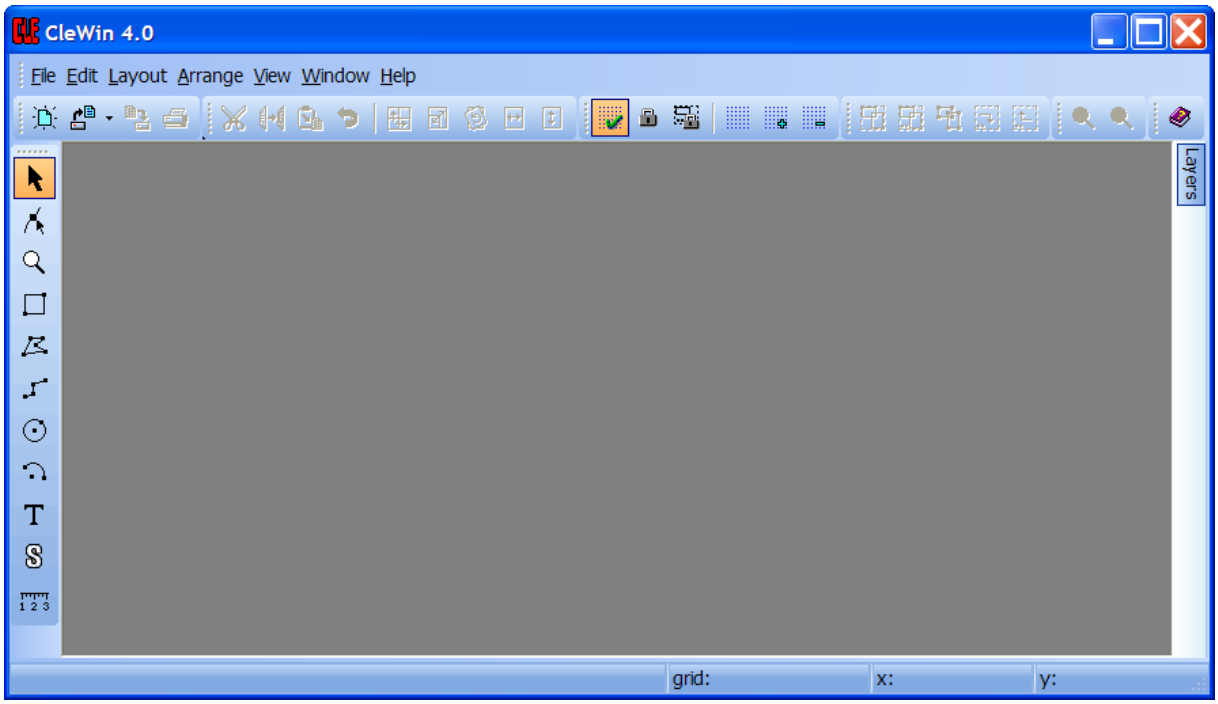
CleWin was developed by WieWeb software in close cooperation with mask manufacturer DeltaMask and the MESA+ institute. Thanks to the input from DeltaMask, mask manufacturers should have no problem producing masks from layout files created by CleWin.

1.1 Getting started

After installing CleWin on your computer using the InstallShield installation utility provided on the CD-ROM you can start CleWin by selecting the icon from the Start menu. Alternatively you can double-click on the program icon in the program folder (usually: `C:\Program Files\CleWin4`).

If you installed CleWin manually by copying the files into an empty directory you can start the program by double-clicking on the program icon. You can include CleWin in the Start menu by copying a shortcut to the program in the `C:\Windows\Start Menu directory`.

After starting the program the following window appears:



You can now open a new layout file by selecting the File|New command.

In CleWin, a layout file may contain up to 256 layers. Some of the layers correspond to the masks made by your mask manufacturer. Other layers may be used to store other shapes that may be useful

during the design but should not appear on the masks, for example the outline of a chip or silicon wafer. In principle you can choose yourself which layers you want to use for what purpose. Simply tell your mask manufacturer which layers should be used for mask generation.

You can use the Layout|Add layers... and Layout|Remove layers... commands to change the number of layers.

After setting up the layers you want to use, you may want to select a mask size using the Layout|Mask size... command. Using this command is optional. It allows CleWin to draw the mask outline in the background and warn you about structures that are outside the mask dimensions. It does not have any influence on the actual contents of your layout file.

Next you can start drawing the masks. First select the desired layer and draw objects using one of the insert objects modes. Objects can be grouped together using the Arrange|Group command. Alternatively, objects can be combined into a named symbol using the Arrange|Combine... command. A symbol can be placed in the layout several times but is only stored once in the layout file. This has the advantage that changing a symbol will automatically change all occurrences of the symbol.

When you have finished drawing your masks, save the layout in CIF format, which is CleWin's default file format. When you are ready to send your layout to a mask manufacturer, you can also use any of the other file formats provided by CleWin. Discuss this with your mask manufacturer. More information can be found in the section about Mask Fabrication.

1.2 Mask fabrication

Any mask manufacturer should be able to produce masks from the files created by CleWin. For mask manufacturing at the MESA+ Research Institute you can contact DeltaMask: http://www.deltamask.nl

Usually, for mask fabrication the CIF or GDS-II file format is used. DeltaMask will prefer CleWin's CIF format, but other mask manufacturers may prefer GDS-II. Besides the mask dimensions and mask type, e.g. chrome or emulsion, your mask manufacturer will need to know some more information depending on the file type.

When using CIF, the mask manufacturer only needs to know the layers from which you want to have masks. CleWin uses the so-called "one call convention". This means that the entire design is defined in a symbol definition and at the end of the file there is one call to this symbol. All other symbols you may have defined are called from within this top-level symbol. CIF layer names are limited in length. Therefore, in CleWin you can specify a "long" and a "short" name for a layer (using the Layout|Layers|Layer properties... command). The long name is used internally by CleWin, your mask manufacturer may only see the short name (unless they also use CleWin). Therefore, be sure to provide your mask manufacturer with the short name of the layers. By default the short names are simply the number of the layer preceded by a capital letter L.

When using GDS-II, your mask manufacturer will need to know the name of the "top-level" or "main" symbol. GDS-II files are simply a collection of symbol definitions and there is no direct way to see which symbol is the main symbol. By default, CleWin uses the name "MainSymbol" for the main symbol, but you can change this by editing the name in the treeview panel of the layout window. GDS-II does not allow layer names. Instead, layers are indicated by their index.

Summarizing:

File format	Provide mask manufacturer with:
CIF	"short" layer names, i.e. L1, L2, etc.
GDS-II	main symbol name, e.g. "MainSymbol", and layer numbers

For low-cost mask fabrication using a photoplotter the Postscript (*.ps), Encapsulated Postscript (*.eps) or Extended Gerber (*.gbr) file formats may be used. In that case you have to use the File|Export layer... command and create a separate file for each layer.

1.3 License terms

CleWin is protected by copyright law and international copyright treaty. Therefore, you must treat this software just like a book, except that you may copy it onto a computer to be used and you may make archive copies of the software for the sole purpose of backing up your software and protecting your investment from loss.
By saying 'just like a book', it is meant that the software may be used by any number of people, and may be freely moved from one computer or location to another, so long as there is no possibility of it being used by more than the number of licensed persons at a time. Just as a book can't be read by two different people in two different places at the same time, neither can the software be available for use by two different people in two different places at the same time.

You may add users by purchasing multi-user versions or additional licenses for the software, so long as the number of persons who are able to use the software at one time isn't more than the number of authorized users specified in the package or license.

For universities and non-commercial research institutes the number of users is not restricted as long as they belong to the research group that purchased the license. If in doubt, please contact WieWeb software for the exact license terms. Other users should refer to the detailed license agreement that has been sent to them by PhoeniX BV.

Limited warranty

WieWeb software warrants the physical media provided by WieWeb software to be free of defects in materials and workmanship for a period of ninety (90) days from the original purchase date. If WieWeb software receives notification within the warranty period of defects in materials or workmanship, and determines that such notification is correct, WieWeb software will replace the defective media.

The entire and exclusive liability and remedy for breach of this limited warranty shall be limited to replacement of defective media and shall not include or extend to any claim for or right to recover any other damages, including but not limited to, loss of profit, data, or use of the software or special consequential damages, or other similar claims, even if WieWeb software has been specifically advised of the possibility of such damages. In no event will WieWeb software's liability for any damages to you or any other person ever exceed the lower of the list price or the actual price paid for the package or the license to use the software, regardless of the form of the claim.

WieWeb software specifically disclaims all other warranties, representations, or conditions, express or implied, including but not limited to, any implied warranty or condition of merchantability or fitness for a particular purpose. All other implied terms are excluded.

Specifically, WieWeb software makes no representation or warranty that the software or documentation are 'error free,' or meet any user's particular standards, requirements, or needs. In all events, any implied warranty, representation, condition, or other term is limited to the physical media

and is limited to the 90-day duration of the limited warranty.

1.4 Support

If you have a support and update contract with PhoeniX BV, the easiest way to get support is by sending an e-mail to their support department: support@phoenixbv.com

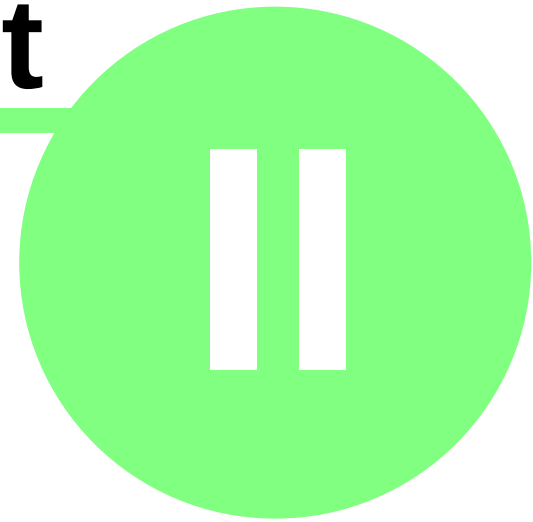
Universities and (non-commercial) research institutes usually do not have a support contract. They may contact WieWeb software directly by e-mail: support@wieweb.com. Suggestions for improvements can also be sent to this address.

In both cases you should receive an answer within 24 hours.

CleWin 4.1

User Guide

Part



2 The user interface

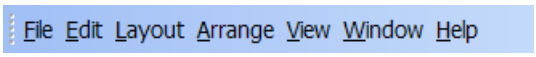
One of the most important things that distinguishes CleWin from other layout editors is the clear and intuitive user interface. Although the number of features has grown considerably during the last 10 years, working with CleWin has never been so easy.

The main window contains five major components:

Menu bar	The menu bar is located at the top of the main window and provides access to almost all commands.
Tool bar	The tool bar is a row of buttons that provides quick access to some frequently used commands.
Mode selector	The mode selector defines the current editing mode.
Layer panel	The layer selector defines the currently active layer. It also contains shortcuts to the Single layer and Lock symbols /groups commands.
Library panel	The library panel is only visible when one or more library files are opened.

2.1 The menu bar

Most commands can be selected from the menu bar at the top of the main window.



CleWin has the following menu's:

File menu	The File menu provides commands for creating new files, opening existing files, saving files, printing files, and exiting the application.
Edit menu	The Edit menu provides commands to undo edits, access the clipboard, and to edit selected objects.
Layout menu	The Layout menu provides commands to define the way in which a layout is edited. It contains snapping options, a grid can be specified and default settings can be changed.
Arrange menu	The Arrange menu provides commands to change the layer of objects and to arrange objects in groups and symbols.
View menu	The View menu contains commands which affect the appearance of a layout on the screen.
Window menu	The Window menu provides commands to control the position and size of the open layout windows.
Help menu	The Help menu provides access to the help system and the about dialog.

2.1.1 The File menu

The *File menu* provides commands for creating new files, opening existing files, saving files, printing files, and exiting the application.

New	Create a new, untitled, layout.
Open...	Open an existing layout.
Open library...	Open a file for use as library
Save	Save the current layout if its contents have changed.
Save as...	Save the current layout under a new name.
Export layer...	Save a layer into EMK, Mann, Postscript or Gerber format.
Read layer map...	Read layer settings.
Save layer map...	Save current layer settings to file.
Create Metafile	Save the contents of the active window in a metafile.
Print...	Print the current document.
Print Setup...	Set printer characteristics.
Exit	Exit CleWin.

File|New

The *File|New* command opens a new, untitled document, and makes it the active window. The application prompts you to name untitled documents when they are saved.

Toolbar button: 
Shortcut: Ctrl+N

File|Open...

The *File|Open…* command displays the *Open file dialog box* so you can select a file to load into a new window.
CleWin accepts files in the following formats: CIF, GDS-II, DXF, EMK, MANN, and Gerber RS-274X. Bitmap files can be converted using the separate utility.

Toolbar button: 
Shortcut: Ctrl+O

File|Open library...

The *File|Open library…* command opens a CIF or GDS-II file to be used as a symbol library. The contents of the file will appear in the library panel. Symbols from the library can then be dragged into the open layout windows.

File|Save

The *File|Save* command saves the document in the active window to disk. If the document is unnamed, the *Save as dialog box* is displayed so you can name the file, and choose where it is to be saved.

Toolbar button: 
Shortcut: Ctrl+S

File|Save as...

The *File|Save as…* command allows you to save a layout under an new name, or in a new location on disk. The command displays the *Save as dialog box*. You can enter the new file name, including the drive and directory. If you choose an existing file name, you are asked if you want to overwrite the existing file.

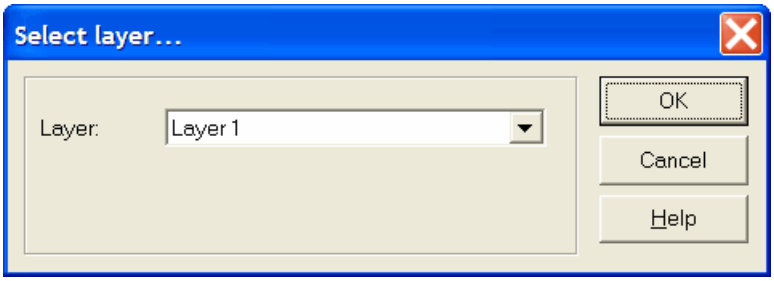
At the bottom of the dialog there are 4 options, which are specific for CleWin and only apply to CIF files:

1. Do not use CleWin extensions
   If this option is checked, all CleWin extensions are removed from the CIF file. This may be useful if your mask manufacturer does not have CleWin, although normally other layout tools will simply ignore all CleWin specific extensions.
2. Flatten file structure
   This option will flattens the layout, that is all groups are ungrouped and all symbols are uncombined. The same result is obtained by using the Layout|Special|Flatten command, but using this option will be much faster since flattening a layout in memory may be extremely time consuming.
3. Remove empty symbol definitions
   This option removes all symbol definitions that do not contain any objects from the file.
4. Remove unused symbol definitions
   This option removes all symbol definitions that are not used, i.e. not called by any other symbol.

File|Export layer...

The *File|Export layer...* command saves a single layer into CIF, EMK, Mann, Postscript or Gerber format.

The command first displays the following dialog, in which you select the layer that you want to export:



Next, a standard Save as... dialog appears that allows you to select the file name.

Shortcut: Ctrl-E

File|Read layer map...

The *File|Read layer map…* command reads previously stored layer settings from file. It is possible that the active window contains objects in a layer that is not in the layer map file. These objects will remain in the layout but will only be drawn as a black outline and the status bar will indicate "invalid layer" when one of these objects is selected. Use the Layout|Layers|Add layers... command to (re-)create the layer(s) containing the objects.

The format of a layer map file is as follows. It starts with a comment line containing the version number of CleWin that created it. Next, each line indicates a layer. A line starts with the layer index followed by

the name of the layer. The index and name are separated by one or more spaces. The end of the name is indicated by a slash followed by a hexadecimal number which contains the layer's style and color information. The slash and hexadecimal number are optional and you may omit them if you create or edit a layer map file with another application or text editor.

Example of a layer map file:

```
# CleWin 4.0 Layer Map
0 Layer 0/0f808000
1 Layer 1/0fe08080
2 Layer 2/0f60e060
3 Layer 3/0fa0a020
```

When saving a design in GDS-II format, a layer map file with the same name is automatically created since the GDS-II format does not allow the use of layer names.

File|Save layer map...

The *File|Save layer map…* command saves the current layer settings in a layer map file.

File|Create metafile...

The *File|Create metafile...* command allows you to save the contents of the active window as a drawing in a window metafile. Metafiles can be imported in many other applications.

CleWin supports two types of metafiles:

Windows metafiles (extension: .wmf)
The Windows-metafile format is supported to maintain backward compatibility with applications that were written to run with Windows version 3.x.

Enhanced metafiles (extension: .emf)
Enhanced metafiles can only be read by 32-bit Windows applications.

File|Print...

The *File|Print…* command prints the contents of the active window. Use File|Print setup... to select a printer, and to set printer options.

Toolbar button: 
Shortcut: Ctrl+P

File|Print setup...

The *File|Printer setup…* command displays the Printer setup dialog box which allows you to select and configure the printer to be used.

File|Exit

The *File|Exit* command exits CleWin. If you've modified documents without saving, you'll be prompted to save before exiting.

2.1.2 The Edit menu

The *Edit menu* provides commands to undo edits, access the clipboard, and to edit selected objects.

Undo	Undo the previous operation.
Cut	Delete selected objects and move them to the clipboard.
Copy	Copy selected objects to the clipboard
Paste	Move objects from the clipboard to the current layout.
Delete	Delete the selected objects.
Duplicate...	Duplicate the selected objects.
Transform...	Shows the Transform... dialog.
Shaping (sub-menu)	Provides access to the polygon operations.
Properties...	Show and edit properties of the selected object(s).

Edit|Undo

The *Edit*|*Undo* command reverses the last edit operation in the currently active symbol definition. Undo inserts any objects you deleted, deletes any objects you inserted, etc.. CleWin keeps track of the changes in each symbol definition. Thus, you can edit a symbol, switch to and edit other symbols... and switch back to the first symbol and still undo the change.
CleWin 4 has a smart undo engine, i.e. changes are recorded using a minimum amount of memory. Still, when working for a long time with CleWin the undo list can become large. Therefore, you are advised to periodically save your work to file. The File|Save and File|Save as... commands clear the undo list after completion of the save operation.

Toolbar button: 
Shortcut: Ctrl+Z

Edit|Cut

The *Edit*|*Cut* command removes the selected objects from the currently active symbol and places the objects in the clipboard. Choose Edit|Paste to paste the cut objects into another symbol. The objects remain in the clipboard, and can be pasted multiple times.

The objects can also be pasted in other Windows applications, in which case they will be inserted as a picture.

Toolbar button: 
Shortcut: Ctrl+X

Edit|Copy

The *Edit*|*Copy* command leaves the selected objects intact and places an exact copy of them in the clipboard. To paste the copied objects into another symbol, choose Edit|Paste.

The objects can also be pasted in other Windows applications, in which case they will be inserted as a picture.

Toolbar button: 

Shortcut: Ctrl+C

Edit|Paste

The *Edit|Paste* command inserts the objects currently selected in the clipboard into the current symbol.

Toolbar button: 
Shortcut: Ctrl+V

Edit|Delete

The *Edit|Delete* command deletes the currently selected objects from the active symbol definition. The objects are not placed in the clipboard. Use the Edit|Undo command to restore the symbol.

Shortcut key: Del

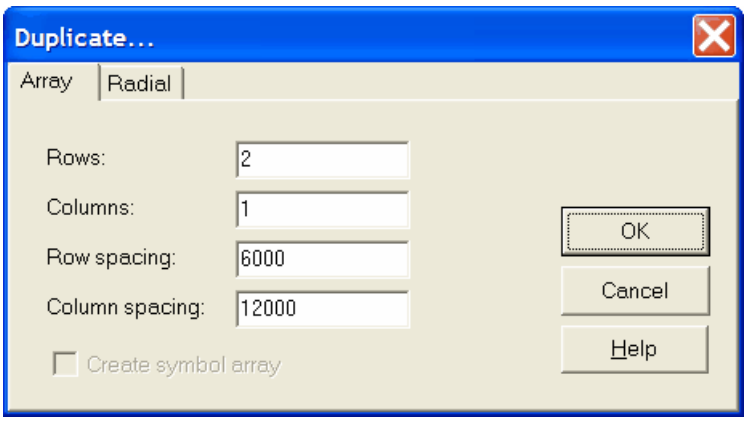
Edit|Duplicate...

The *Edit|Duplicate…* command is used to insert multiple copies of the currently selected objects at regular distances from each other.

A dialog appears with two Tab Pages.
The first page, "Array", allows you to specify 4 parameters:

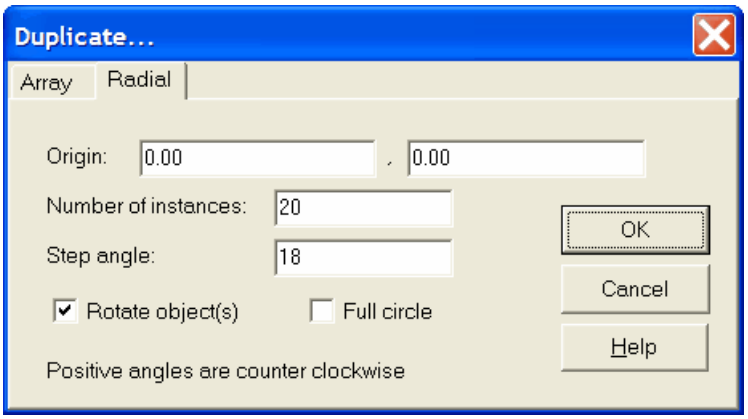
Rows	Number of rows
Columns	Number of columns
Row spacing	The spacing between the rows in micrometers
Column spacing	The spacing between the columns in micrometers

The result will be a rectangular array. If the selected object is a symbol instance, the option "Create symbol array" can be checked. The dimensions of a symbol array can be edited later, using the Edit|Properties... command. The duplicate command remembers the last used number of rows and columns. The row and column spacing are by default equal to the dimensions of the selected object(s).



The second tab page, "Radial", allows you to duplicate the objects around the circumference of a circle. You need to specify the center of the circle, the number of copies of the selected objects, and the angle between the objects in degrees. The option "Rotate object(s)" will rotate the copied objects so that all objects keep the same orientation with respect to the center. The option "Full circle" will spread the opjects evenly around the circumference and the angle between objects is calculated

automatically.



Toolbar button:
Shortcut: Ctrl+D

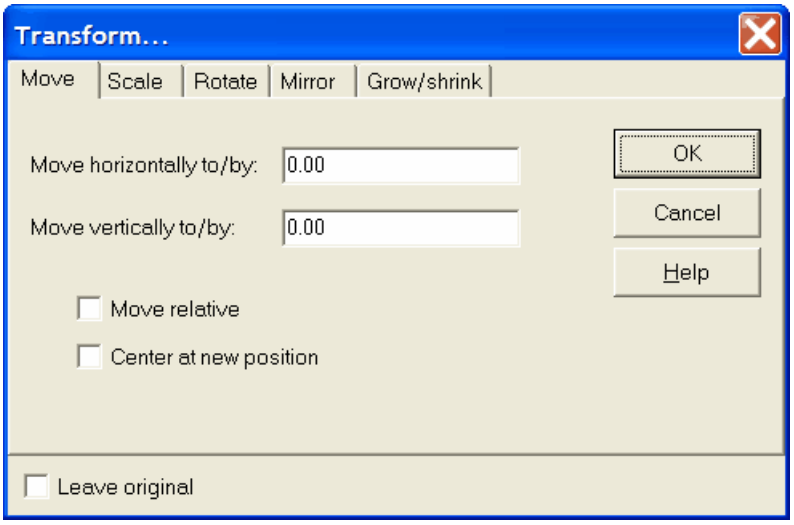
Edit|Transform...

The *Edit*|*Transform* command shows the *Transform... dialog*. The command can be used to move, scale, rotate, mirror and grow/shrink objects. Checking the *Leave original* box will leave a copy of the original object(s).

Moving objects

The *Move* page contains the following items:

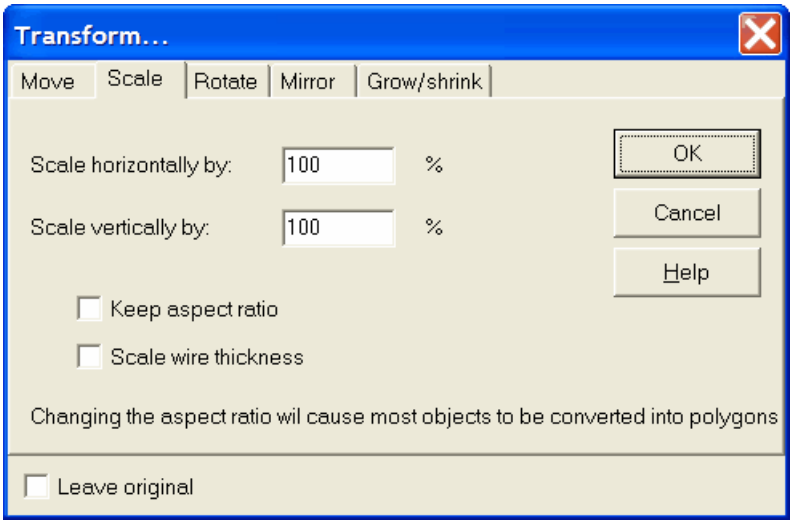
Move horizontally to/by	The new horizontal coordinate or the distance to move horizontally
Move vertically to/by	The new vertical coordinate or the distance to move vertically
Move relative	Select this option to move relative to the current position
Center at new position	Select this option to center the object(s) at the new position



Scaling objects

The *Scale* page contains the following items:

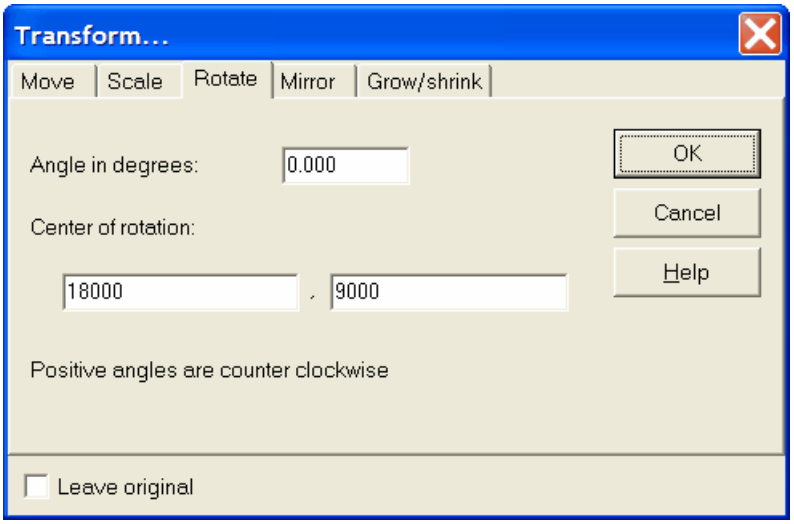
Scale horizontally by	The percentage by which to change the horizontal dimensions.
Scale vertically by	The percentage by which to change the vertical dimensions.
Keep aspect ratio	Select this option to preserve the current aspect ratio. The scale vertically percentage will be disabled.
Scale wire thickness	If the selection contains wires, this option defines whether the wire thickness will be scaled too.



Rotating objects

The *Rotate* page contains the following items:

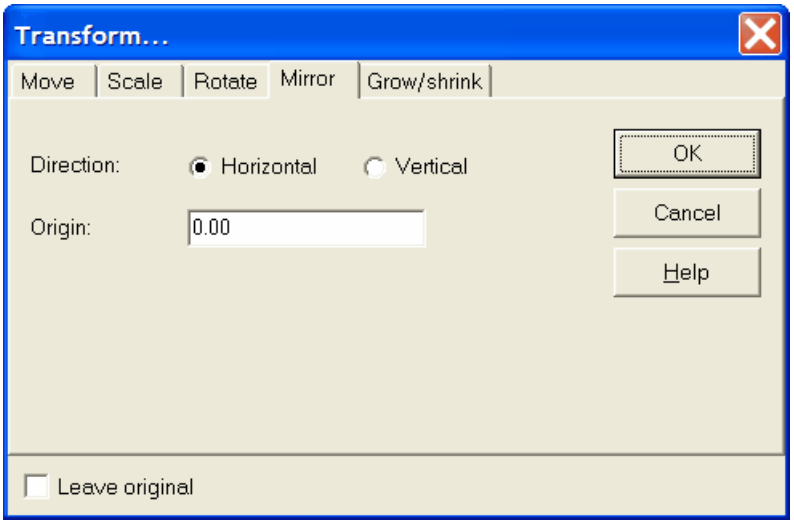
Angle in degrees	The angle in degrees by which to rotate. Positive angles are counter clockwise. The resolution is 0.001 degrees.
Center of rotation	The center of rotation in micrometers.



Mirroring objects

The *Mirror* page contains the following items:

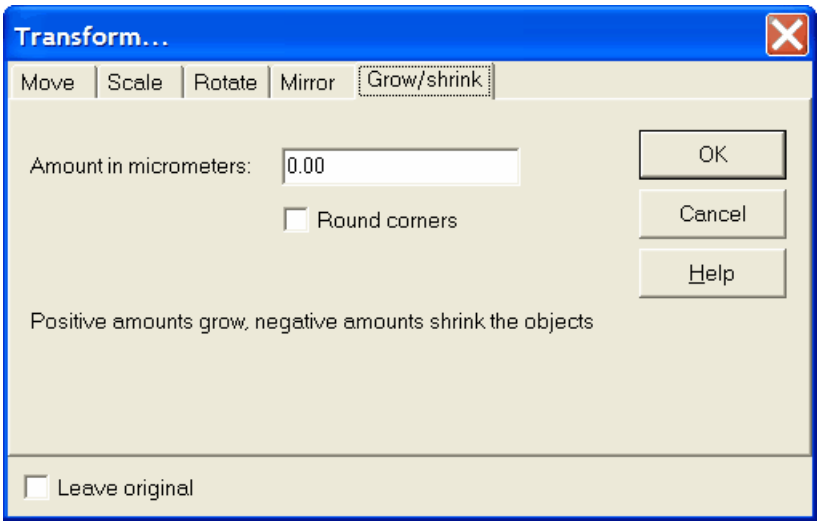
Direction	Horizontal: mirror in a vertical line at x=origin.
	Vertical: mirror in a horizontal line at y=origin.
Origin	The origin in micrometers.



Growing and shrinking objects

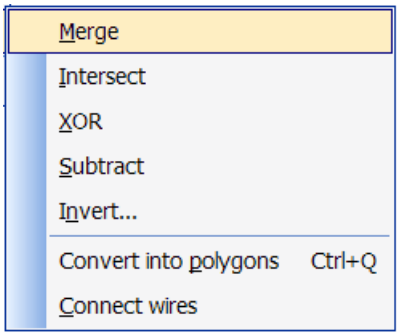
The *Grow/shrink* page contains the following items:

Amount in micrometers	The amount by which the object(s) should grow or shrink.
	Positive values cause objects to grow.
	Negative values cause objects to shrink.
Round corners	This option will round all convex corners.



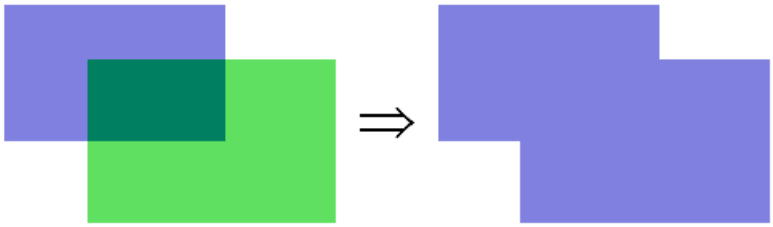
The Edit|Shaping menu

The *Shaping menu* contains the boolean operations Merge, Intersect, XOR, Subtract and Invert..., and the commands Convert into polygon(s) and Connect wires. The boolean operations as implemented in CleWin are based on the algorithm developed by K. Holwerda, see http://boolean.klaasholwerda.nl/bool.html.



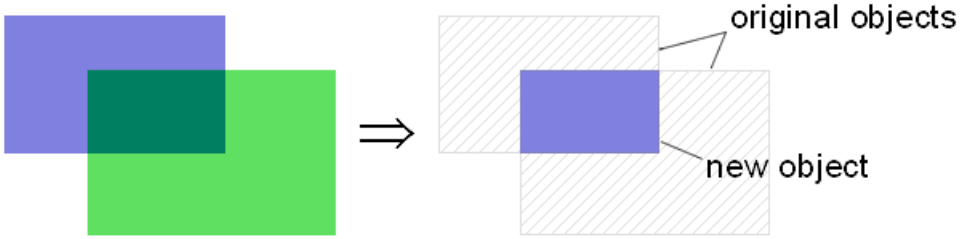
Edit|Shaping|Merge

The *Edit|Merge* command combines the currently selected objects in one or more polygons. The resulting polygon(s) appear(s) in the layer of the first selected object or in layer 0 if the first object is a group or symbol instance. The command works with all types of primitive objects (rectangles, boxes, polygons, wires, circles and rings) and with groups and symbol instances containing these objects. Symbol instances will be flattened but the symbol definition will remain untouched.



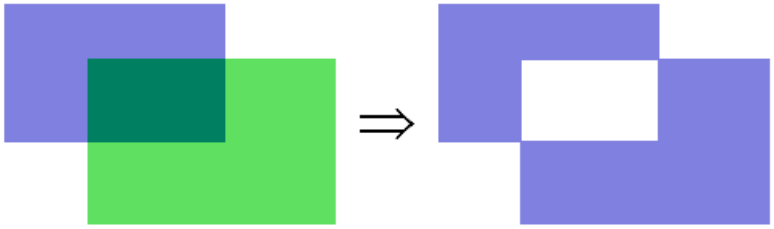
Edit|Shaping|Intersect

The *Edit|Intersect* command calculates the intersection of two selected objects. The resulting polygon(s) appear(s) in the layer of the first selected object or in layer 0 if the first object is a group or symbol instance. The command works with all types of primitive objects (rectangles, boxes, polygons, wires, circles and rings) and with groups and symbol instances containing these objects. Symbol instances will be flattened but the symbol definition will remain untouched.



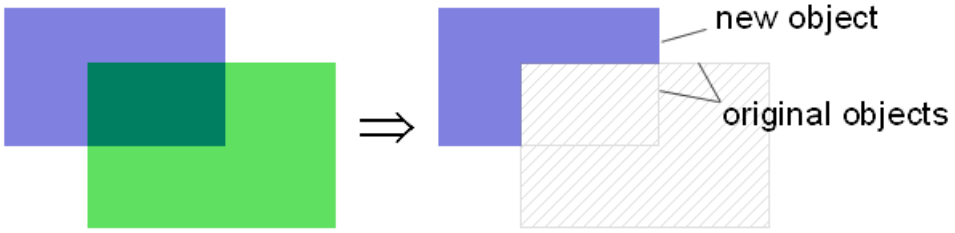
Edit|Shaping|XOR

The *Edit|XOR* command calculates the non-overlapping area of two selected objects. The resulting polygon(s) appear(s) in the layer of the first selected object or in layer 0 if the first object is a group or symbol instance. The command works with all types of primitive objects (rectangles, boxes, polygons, wires, circles and rings) and with groups and symbol instances containing these objects. Symbol instances will be flattened but the symbol definition will remain untouched.



Edit|Shaping|Subtract

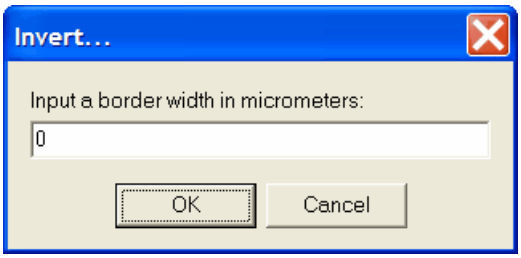
The *Edit|Subtract* command subtracts the second selected object from the first selected object. The resulting polygon(s) appear(s) in the layer of the first selected object or in layer 0 if the first object is a group or symbol instance. The command works with all types of primitive objects (rectangles, boxes, polygons, wires, circles and rings) and with groups and symbol instances containing these objects. Symbol instances will be flattened but the symbol definition will remain untouched.



Edit|Shaping|Invert...

The *Edit|Invert...* command inverts the area covered the selected objects. The resulting polygon(s) appear(s) in the layer of the first selected object or in layer 0 if the first object is a group or symbol instance. The command works with all types of primitive objects (rectangles, boxes, polygons, wires, circles and rings) and with groups and symbol instances containing these objects. Symbol instances will be flattened but the symbol definition will remain untouched.

The following dialog appears which allows you to enter a border for the area to be inverted:



Edit|Shaping|Convert into polygon(s)

The *Edit|Convert to polygons* command converts the selected object(s) into one or more polygons. The resulting polygon(s) appear(s) in the layer of the first selected object or in layer 0 if this object is a group or symbol instance.

Shortcut: Ctrl-Q

Edit|Shaping|Connect wires

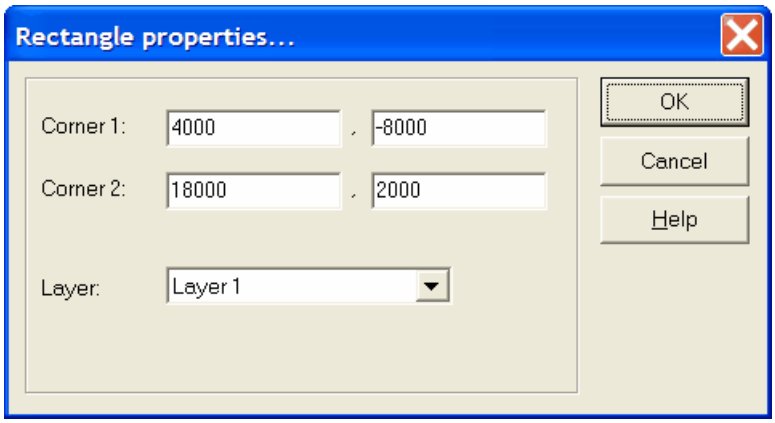
The *Edit|Connect wires* command connects wires in the current selection that share begin/end nodes. This command is especially useful for editing DXF files generated by other applications than CleWin. These files often contain a large number of separate line pieces that should form a long wire or polygon. In the latter case the wires usually have a zero width. Therefore, when connecting zero-width wires results in a closed shape CleWin will convert them into a polygon.

Edit|Properties...

The *Edit|Properties…* command allows you to edit the settings of the currently selected object. A special dialog appears for each type of object. If more than one object is selected, the Change layer... dialog will appear.

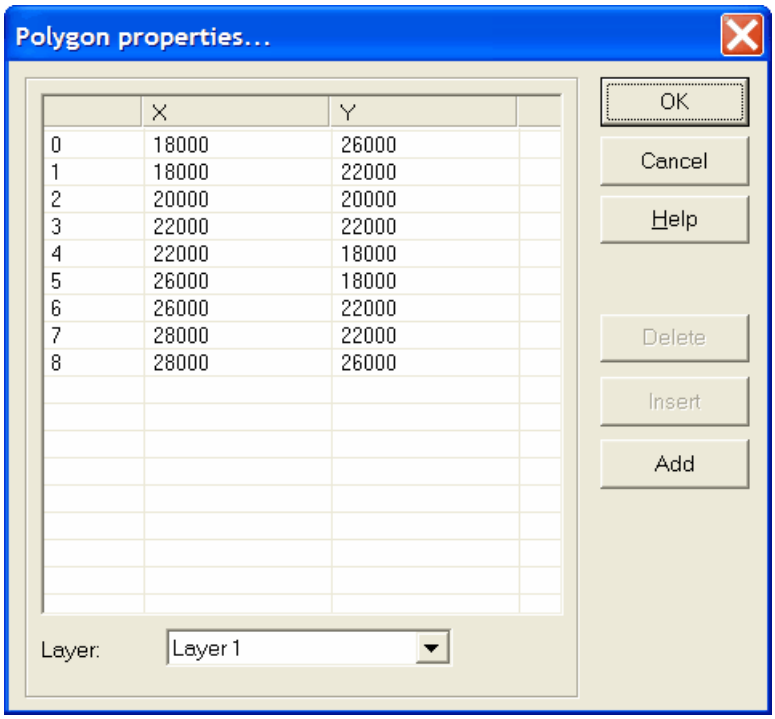
Rectangle properties

In the *rectangle properties dialog* you can edit the coordinates of two opposite corners of the rectangle. You can also change the layer.



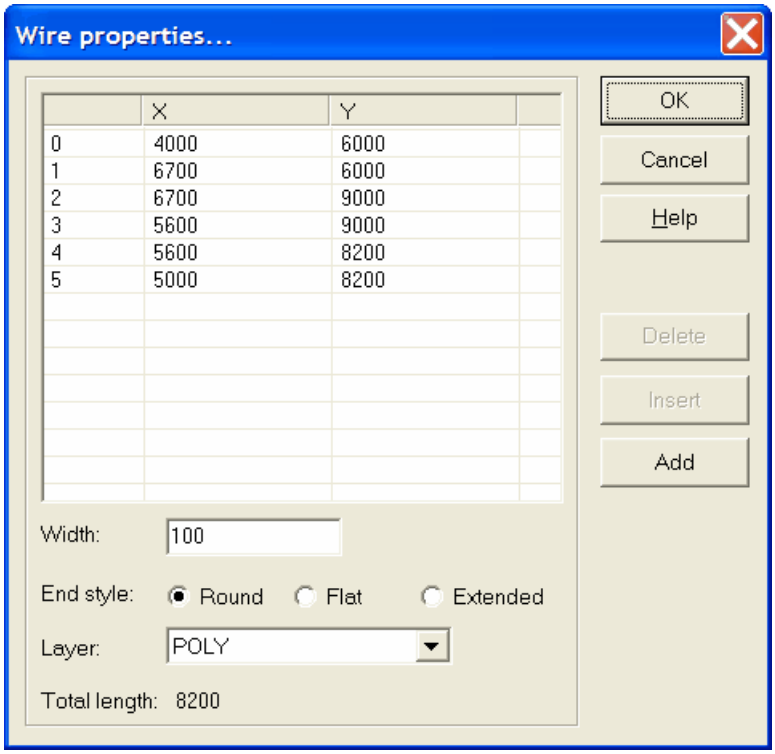
Polygon properties

The *polygon properties dialog* allows to edit the polygon nodes and set the layer.



Wire properties

The *wire properties dialog* allows to edit the nodes of a wire and to set the layer, width, and end style of a wire. It also calculates the total length of a wire.

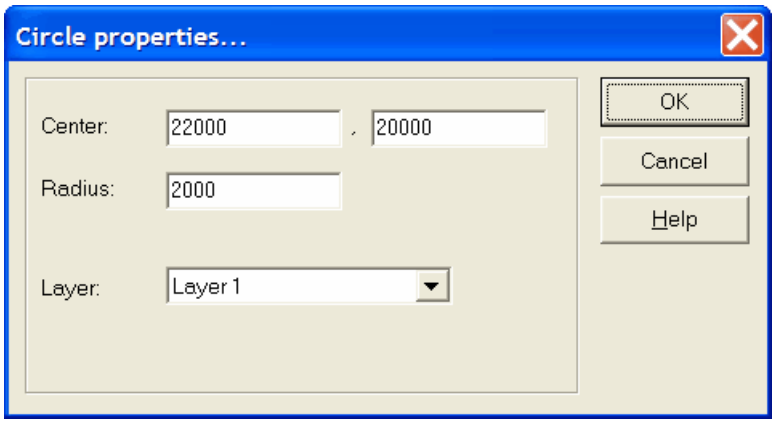


Note that the end style is usually not supported by mask manufacturers! You need to contact your

mask manufacturer about this. When using CIF files, wires will probably have "Round" ends on the mask, independent of the setting in CleWin. The GDS-II format supports all wire end styles, however they are not always supported by mask making equipment.

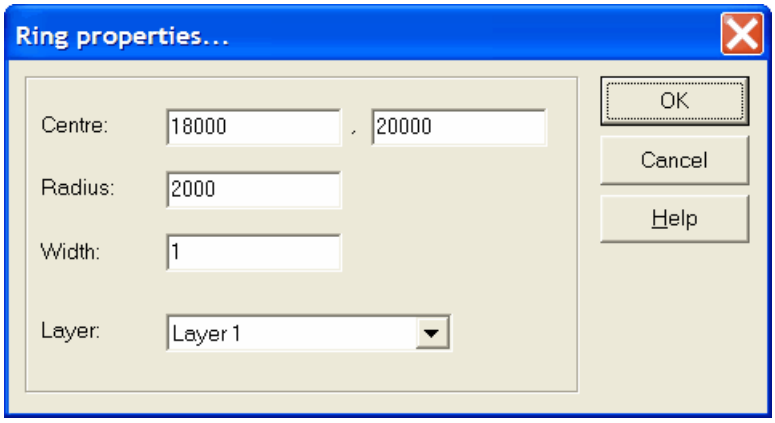
Circle properties

The *circle properties dialog* allows editing of the center position and radius of a circle. You can also select a different layer.



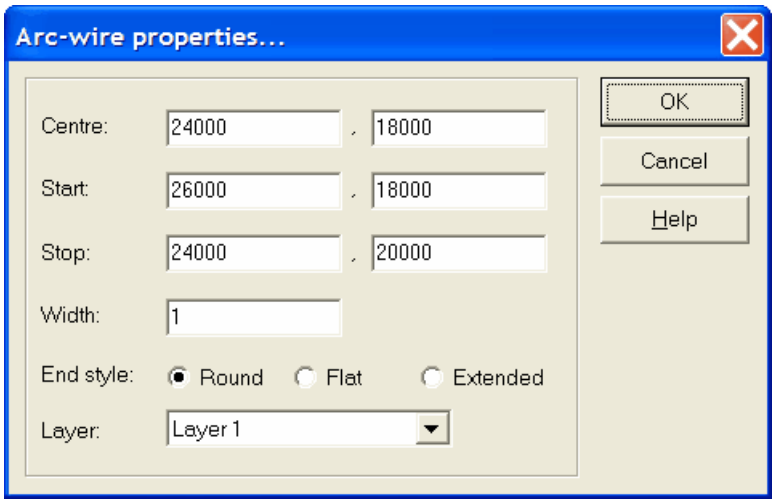
Ring properties

The *ring properties dialog* allows editing of the center position, radius and width of a ring. You can also select a different layer.



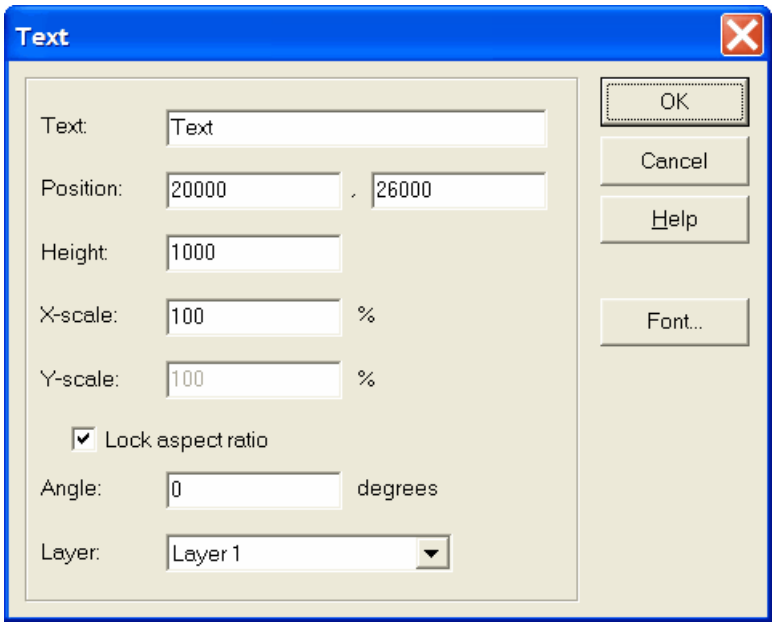
Arc-wire properties

The *arc-wire properties dialog* allows editing of the center position, the start node, and the stop node of the arc-wire. Furthermore, you can select a new end-style or layer.



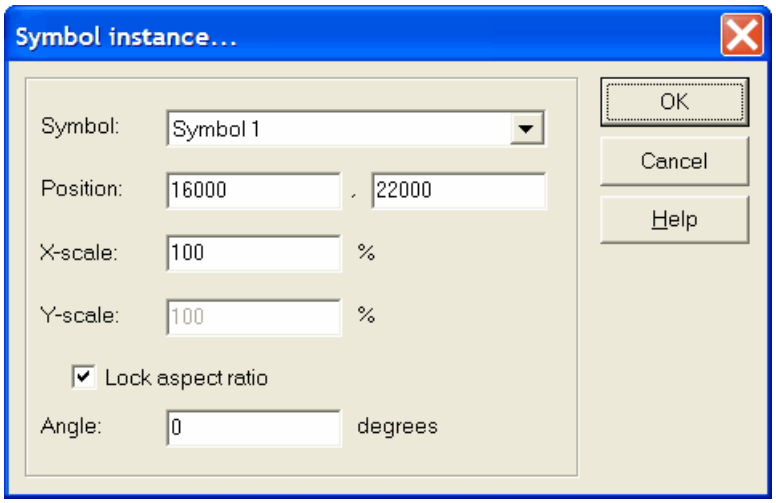
Text properties

The *text properties dialog* allows editing of a text object. Use the Font... button to select a different font.



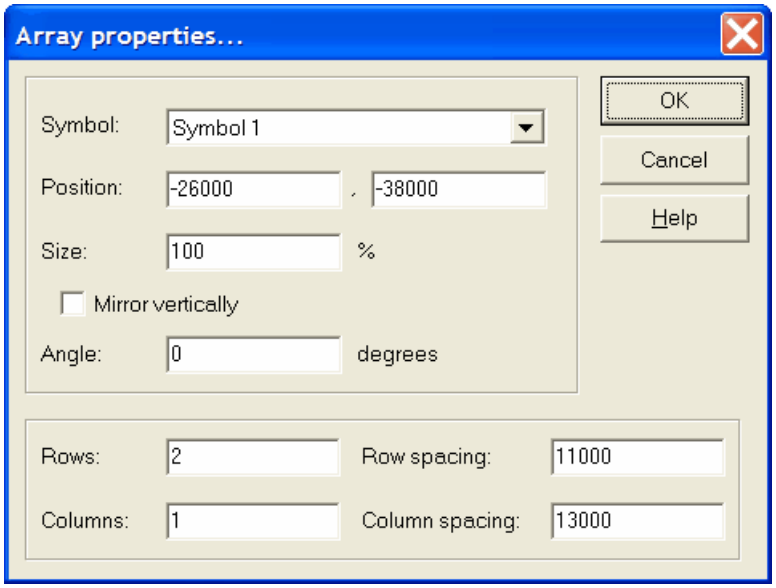
Symbol instance properties

The *symbol instance properties dialog* allows editing of a symbol instance. Pleas note that a scale other than 100% is not supported by all file formats. In CIF files scaling of symbol instances is not allowed and CleWin will use special extensions to the file format. As a result, a CIF file with scaled symbol instances may not be accepted by your mask manufacturer. In such a case a solution might be to flatten the design or at least uncombine the scaled instances. In GDS-II scaling of symbol instances is allowed, however the values for *X-scale* and *Y-scale* need to be identical.



Symbol array properties

The *array properties dialog* allows editing an array of symbols that was created by the Edit|Duplicate... command.



The Change layer... dialog

The *change layer dialog* appears when more than one object is selected.

2.1.3 The Layout menu

The *Layout menu* provides commands to define the way in which a layout is edited. It contains snapping options, a grid can be specified and default settings can be changed.

Snap to (sub-menu)	Select the snap settings.
Single layer	Restrict editing to the active layer.
Lock symbols / groups	Disable editing symbol instances and groups.
Grid setup...	Select a new grid spacing.
Grid +	Increase the grid spacing.
Grid -	Decrease the grid spacing.
Layers (sub-menu)	Contains the add/remove layers and layer properties commands.
Mask size...	Define the mask and wafer dimensions.
Preferences...	Set some preferences.
Special (sub-menu)	Contains the Center at (0,0), Flatten and Insert text commands.

The Layout|Snap to menu

The *Layout|Snap to* menu contains the Snap to grid, Snap to guidelines and Snap orthogonal commands.

Shortcut: Ctrl+Y

Layout|Snap to|Snap to grid

The *Layout|Snap to grid* command is used to turn on or off snapping of coordinates to multiples of the grid value.

Toolbar button:
 (snap on)
 (snap off)

Shortcut: Ctrl-Y

Layout|Snap to|Snap to guidelines

The *Layout|Snap to guidelines* command is used to turn on or off the snapping of coordinate values to guidelines. More information about using guidelines can be found in the section about Rulers and guidelines.

Layout|Snap to|Orthogonal

The *Layout|Snap orthogonal* command affects the way polygons, (arc-) wires and circles are entered. When snap orthogonal is active, angles are rounded to the nearest multiple of 45 degrees.

Layout|Single layer

The *Layout|Single layer* command allows you to restrict editing to objects in the selected layer.

Toolbar button: 

Layout|Lock symbols / groups

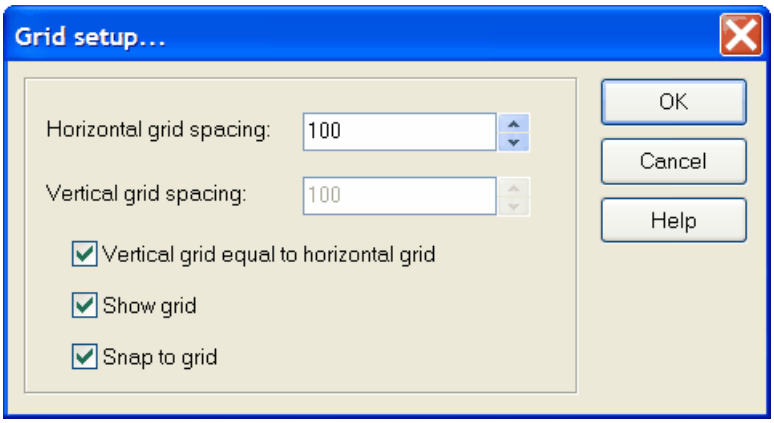
The *Layout|Lock symbols / groups* command allows you to disable editing of groups and symbol instances.

Toolbar button: 

Layout|Grid setup...

The *Layout|Grid setup…* command allows you to change the spacing of the grid.

The following dialog appears:



The dialog allows you to specify separate values for the horizontal and vertical grid spacing. Selecting the option 'Vertical grid equal to horizontal grid' will result in a square grid and the edit box for the vertical grid spacing will be disabled. The options 'Show grid' and 'Snap to grid' can also be changed using the View|Show grid and Layout|Snap to|Snap to grid commands, respectively.

Toolbar button: 

Layout|Grid +

The *Layout|Grid* + command increases the grid spacing to the nearest logical value in a 1-2-5 sequence.

Toolbar button: 

Layout|Grid -

The *Layout|Grid* - command decreases the grid spacing to the nearest logical value in a 1-2-5 sequence.

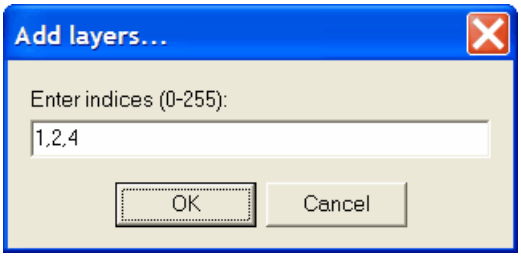
Toolbar button: 

The Layout|Layers menu

The *Layout|Layers* menu contains the Add layers..., Remove layers... and Layer properties... commands.

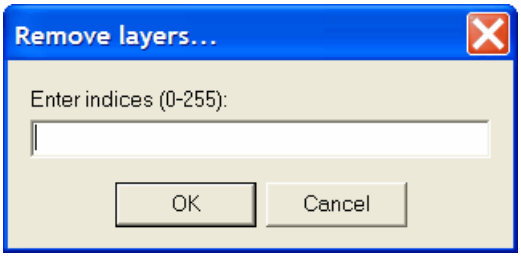
Layout|Layers|Add layers...

The *Layout|Layers|Add layers…* command allows you to add one or more new layers. You have to enter layer indices separated by spaces or semicolons (the example below would create three new layers with indices 1, 2 and 4).



Layout|Layers|Remove layers...

The *Layout|Layers|Remove layers…* command allows you to remove one or more layers. You have to enter layer indices separated by spaces or semicolons.



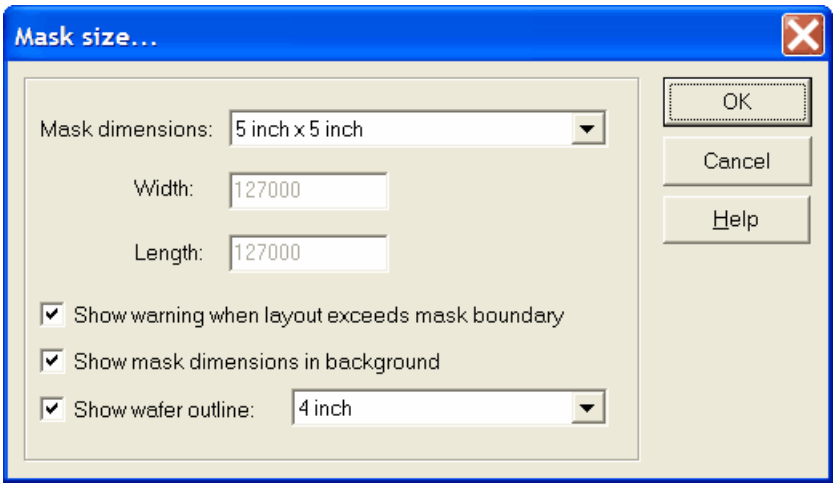
Layout|Layers|Layer properties...

The *Layout|Layers|Layer properties...* command shows the Layer properties... dialog, which allows you to change the names, colors and line styles of the layers.
For the CIF file format you can also specify a short name for the layer.
The GDS-II file format does not support layer names. When reading or writing in GDS-II format only the layer indices are used, but CleWin will automatically read or create a layer map file with the same name.

You can also select this command from the pop-up menu that appears when clicking with the right mouse button on a layer button in the layer bar.

Layout|Mask size...

The *Layout|Mask size…* command allows you to select an optional mask size. Two check boxes allow you to display the dimensions of the mask at the background and to specify that CleWin should warn you when you save a layout that is larger than the mask.



Layout|Preferences...

The *Layout|Preferences…* command allows you to select some preferences.

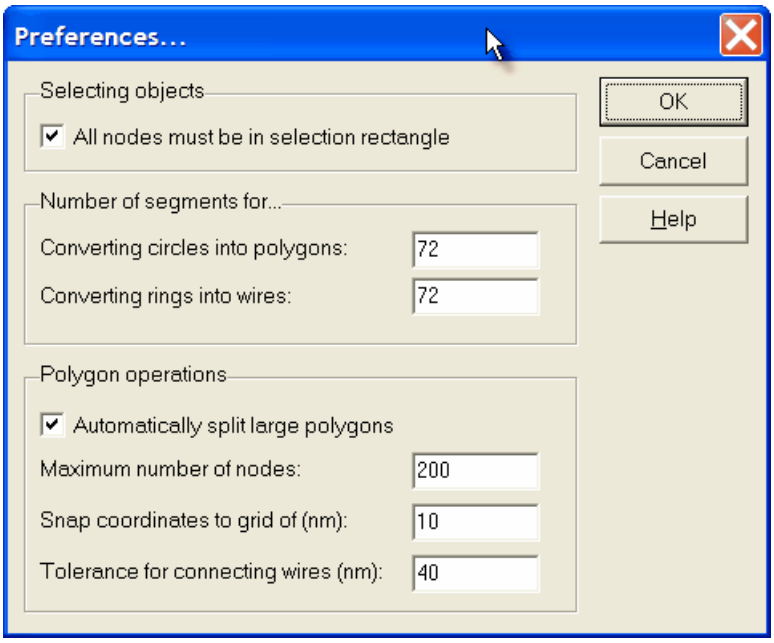
The *All nodes must be in selection rectangle* option specifies whether an object is selected if one of its nodes is inside the selection rectangle, or only if all nodes are inside the rectangle.

The number of segments for converting circles and rings into polygons is used whenever such a conversion is needed, e.g. during the polygon operations (Merge, XOR, Subtract, etc.) and when saving into GDS-II format.

When the option *Automatically split large polygons* is active, CleWin checks whether the result of a polygon operation gives a polygon with more than the specified maximum number of nodes. If so, the polygon is split into a collection of smaller polygons.

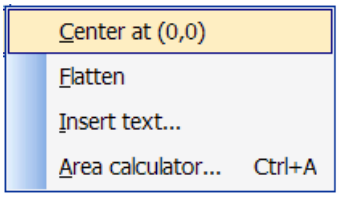
Snap coordinates to gride of (nm) specifies the resolution of the polygon operations. All coordinate values of polygons resulting from these operation will be a multiple of this value. A larger value will result in faster calculations, but may also result in rounding errors. A value of zero disables the rounding of coordinates.

Tolerance for connecting wires (nm) is used by the Edit|Shaping|Connect wires command. It specifies the distance between nodes that is allowed so that the nodes are still considered to be the same.



The Layout|Special menu

The *Layout*|*Special* menu contains the Center at (0,0), Flatten, Insert text... and Area calculator... commands.



Layout|Special|Center at (0,0)

The *Layout*|*Special*|*Center at (0,0)* command centers the entire layout around the origin (0,0). If one or more objects are selected these objects will be centered (all other objects will move by the same distance).

This command only works for the entire layout. Therefore, the command is only active when the top symbol in the hierarchy (usually "MainSymbol") is the active symbol.

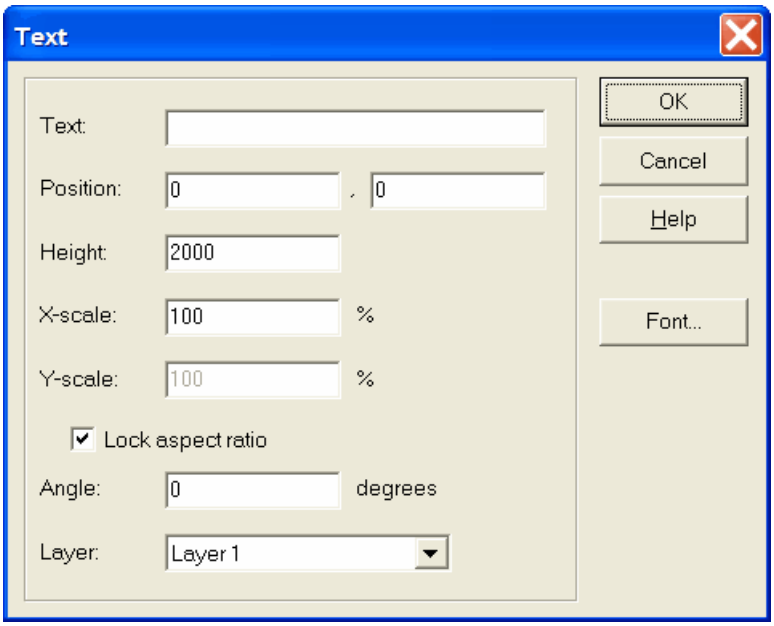
Layout|Special|Flatten

The *Layout*|*Special*|*Flatten* command flattens the layout, that is all groups are ungrouped and all symbols are uncombined.

This command only works for the entire layout. Therefore, the command is only active when the top symbol in the hierarchy (usually "MainSymbol") is the active symbol.

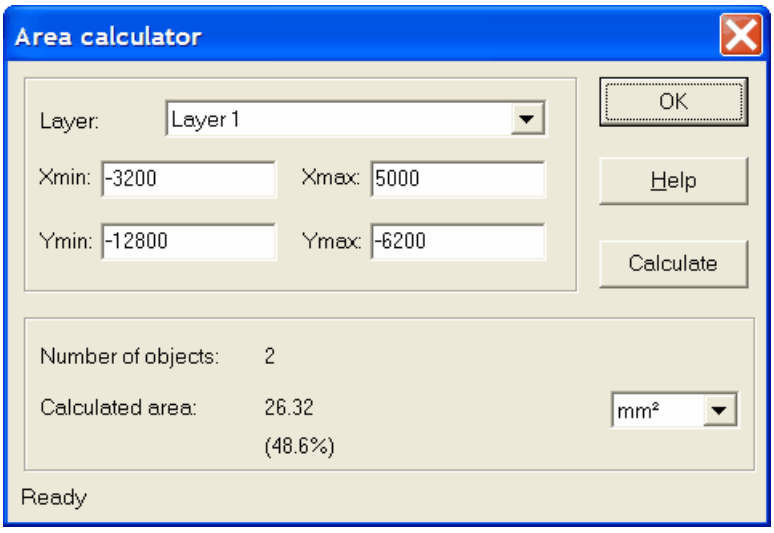
Layout|Special|Insert text...

The *Layout|Special|Insert text...* command allows to insert text at an arbitrary position in the currently active symbol definition. The following dialog appears (see also Edit|Properties, Text):



Layout|Special|Area calculator...

The *Layout|Special|Area calculator...* displays the Area Calculator dialog which allows you to calculate the surface area covered by a certain layer in a certain area.



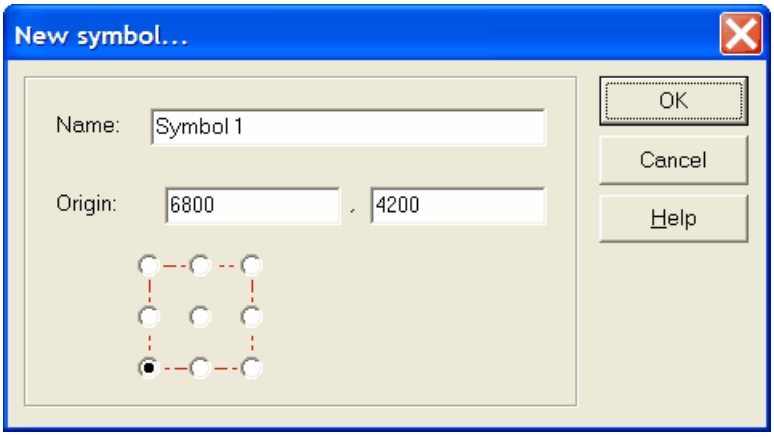
2.1.4 The Arrange menu

The *Arrange menu* provides commands to change the layer of objects and to arrange objects in groups and symbols.

Combine...	Combines the selected object into a named symbol.
Group	Groups the selected objects.
Uncombine / ungroup	Breaks apart the selected group or symbol instance.
Flatten selection	Recursively breaks apart all groups and symbol instances in the current selection.
Open symbol	Opens the definition of the selected symbol instance for editing.
Close symbol	Closes the symbol definition and returns to the calling symbol.
Clone definition...	Copies the symbol definition under a new name

Arrange|Combine...

The *Arrange|Combine…* command combines the selected objects into a new symbol definition and replaces the objects by a reference to the new symbol.
Symbol definitions may be nested, i.e. a symbol may contain references to other symbols. A symbol may not contain a reference to itself.
The following dialog appears:



Name	The name of the new symbol definition.
Origin	The origin of of the new symbol in micrometers with respect to the currently selected objects. By default this is the lower-left corner of the selection area.

The radio-buttons can be used to quickly select another origin.

Toolbar button:
Shortcut: Ctrl-L

Arrange|Group

The *Arrange|Group* command arranges the currently selected objects in a group. From then on the objects can be edited as if they were a single object.
Use the Arrange|Ungroup command to obtain the individual objects again.
Groups may be nested, i.e. a group of objects may contain other groups.

Toolbar button:

Arrange|Break apart

The *Arrange|Break apart* command breaks apart a selected group, symbol, symbol array or script object:

- A selected group is replaced by the original objects.
- A reference to a symbol definition is replaced by a copy of the objects contained by the symbol. The command does not affect the definition of the symbol. Thus, other references to the symbol remain valid and new references to the symbol can be created.
- A symbol array is replaced by separate symbol instances.
- A script object is replaced by the generated objects. The script itself is deleted.

Toolbar button: 
Shortcut: Ctrl-K

Arrange|Flatten selection

The *Arrange|Flatten selection* applies the Arrange|Break apart command to all groups and symbol instances in the current selection. That is, all selected groups and symbol instances are exploded into their individual objects, including those nested inside other groups or symbol instances .

Arrange|Open symbol

The *Arrange|Open symbol* command opens the currently selected symbol instance for editing. Instead of selecting this command from the menu, you can also double-click on the selected symbol using the left mouse button.
Editing the symbol changes the definition of the symbol. Thus, other instances of the symbol will change simultaneously.

Toolbar button: 
Shortcut: PgDn

Arrange|Close symbol

The *Arrange|Close symbol* command closes the symbol definition that is currently being edited and returns to the calling symbol, i.e. one level higher in the symbol hierarchy.

Toolbar button: 
Shortcut: PgUp

Arrange|Clone definition...

The *Arrange|Clone definition…* command creates a new symbol definition with the same contents as the selected symbol instance. It shows the "New symbol..." dialog so that you can enter a new name and, optionally, change the symbol origin.

2.1.5 The View menu

The *View menu* contains commands which affect the appearance of a layout on the screen.

Zoom in...	Select the snap settings.
Zoom out	Restrict editing to the active layer.
Zoom all	Disable editing symbol instances and groups.
Show grid	Shows or hides the grid.
Fill	Defines whether the interior of objects is filled.
Show coordinates	Shows or hides the current coordinates at the mouse pointer.
Show rulers	Shows or hides the rulers.
Lock aspect ratio	Locks the aspect ratio in the layout window
Symbols (sub-menu)	Contains commands related to how symbol instances are shown.
Groups (sub-menu)	Contains commands related to how groups are shown.
Background (sub-menu)	Contains options on how to draw the background.
Redraw window	Immediately redraws the contents of the active window.

View|Zoom in...

The *View|Zoom in…* command allows you to zoom in on a detail of the active symbol. After selecting the command, you can draw a rectangle in the layout window while pressing the left mouse button.

Toolbar button: 
Shortcut keys: F2, I and +

View|Zoom out

The *View|Zoom out* command zooms out to the previous zoom level, i.e. the zoom level before the last *Zoom in* command was executed.

Toolbar button: 
Shortcut keys: F3, O and -

View|Zoom all

The *View|Zoom all* command zooms out in order that the entire active symbol definition becomes visible.

Shortcut key: F4

View|Show grid

The *View|Show grid* command allows you to show or hide the grid.

Shortcut: Ctrl-G

View|Fill

The *View|Fill* command defines whether the interior of objects is filled or not.

Shortcut: Ctrl-F

View|Show coordinates

The *View*|*Show coordinates* command defines whether the current coordinates are shown next to the mouse pointer.

View|Show rulers

The *View*|*Show rulers* command defines whether the rulers are visible at the top and left of the layout area.

View|Lock aspect ratio

The *View*|*Lock aspect ratio* command locks the aspect ration in the layout window. That is, if this option is active the scale at which the design is drawn is the same in horizontal and vertical direction.

The View|Symbols menu

The *View*|*Symbols* menu contains the Show interiors, Show borders, and Number of levels... commands.

View|Symbols|Show interiors

The *View*|*Symbols*|*Show interiors* command defines whether the interior of groups and symbols is drawn or not.

View|Symbols|Show outlines

The *View*|*Symbols*|*Show outlines* command defines whether the boundaries and names of symbol instances are drawn or not.

View|Symbols|Number of levels...

The *View*|*Symbols*|*Number of levels…* command defines the number of symbol nesting levels that are drawn on the screen.

The View|Groups menu

The *View*|*Groups* menu contains the command.

View|Groups|Show outlines

The *View*|*Groups*|*Show outlines* command defines whether the boundaries of groups are drawn or not.

The View|Background menu

The *View*|*Background* menu contains the Show wafer outline, Show mask dimensions, Background white and Background black commands.

View|Background|Show wafer outline

The *View*|*Background*|*Show wafer outline* command defines wether the outline of a wafer is shown in

the background. The size of the wafer is selected by the Mask size... command

View|Background|Show mask dimensions

The *View|Background|Show mask dimensions* command defines wether the outline of the masks is shown in the background. The mask size is selected by the Mask size... command

View|Background|White

The *View|Background|White* command selects a white background.

View|Background|Black

The *View|Background|Black* command selects a black background.

View|Redraw window

The *View|Redraw window* command redraws the contents of the active window.

Shortcut key: F12

2.1.6 The Window menu

The *Window menu* provides commands to control the position and size of the open layout windows.

Toolbars (sub-menu)	Hides or shows the tool bars.
Mode selector	Hides or shows the mode selector.
Layers	Hides or shows the layer panel.
Libraries	Hides or shows the library panel.
Cascade	Resize and position all windows in an overlapping pattern.
Tile	Resize and position all windows in a non-overlapping pattern.
Minimize all	Minimize all windows in icons.
Arrange all	Align all iconized windows along a grid.

Window|Toolbar

The *Window|Toolbar* command hides or shows the toolbar.

Window|Mode selector

The *Window|Mode selector* command hides or shows the mode selection panel.

Window|Layers

The *Window|Layers* command hides or shows the layer panel.

Window|Libraries

The *Window|Libraries* command hides or shows the library panel.

Window|Cascade

The *Window|Cascade* command resizes and positions all windows in an overlapping pattern.

Window|Tile

The *Window|Tile* command resizes and positions all windows in a non-overlapping pattern.

Window|Minimize all

The *Window|Minimize all* command minimizes all windows into icons.

Window|Arrange all

The *Window|Arrange all* command aligns all iconized windows along a grid.

2.1.7 The Help menu

The Help menu provides access to the help system and the about dialog.

CleWin Help	Help topic contents.
About	Shows the About dialog.

Help|CleWin Help

The *Help|CleWin Help* command shows the help topic contents.

Help|About

The *Help|About* command shows the About dialog.

2.2 The tool bars

The *tool bars* are the rows of buttons at the top of the main window which represent commands that are also available through the menu bar. Clicking one of the buttons is a quick alternative to choosing a command from the menu. Buttons on the toolbars activate and deactivate according to the state of the application.

The toolbar can be undocked and used as floating window, or it can be docked to any edge of the main CleWin window.
The Window|Toolbars menu can be used to hide or show the toolbars.

2.2.1 Toolbar commands

The toolbars contain the following commands:

New
Open
Save
Print

Cut
Copy
Paste
Undo

Duplicate
Scale
Rotate
Horizontal mirror
Vertical mirror

Snap to grid
Single layer
Lock symbols / groups
Grid settings
Increase grid
Decrease grid

Group
Combine into symbol
Ungroup/uncombine
Open symbol
Close symbol

Zoom in
Zoom out

Help contents

2.3 The mode selector

The *mode selector* is a row of buttons which allows you to select the edit mode.
Like the toolbar, the mode selector may be moved with the mouse and left as a floating palette, or docked to any edge of the main window.

Button	Mode
	Select and edit objects
	Edit object nodes
	Zoom in/out
	Insert rectangles
	Insert polygons
	Insert wires
	Insert circles
	Insert arc-wires and rings
	Insert text
	Insert script objects
	Measure distances

2.3.1 Selecting and editing objects

The *edit object mode* allows you to select and edit objects. Objects can be selected by either

clicking on an object node or by *drawing a selection rectangle*. To be selected by a selection rectangle, objects have to be either completely inside the rectangle or have one node inside the rectangle, depending on the settings in the Preferences dialog.
Objects can be added to or removed from an existing selection by pressing the Shift-key, while selecting the objects.

Once one or more objects have been selected, the objects can be edited. When the mouse cursor moves over the selected objects, the cursor shape changes to one of the following:

 When the cursor has this shape the objects can be dragged to another position using the left mouse button. Clicking the right mouse button once during dragging or pressing the '+' key will leave a copy of the original objects. When Snap to grid is active the objects will move by a multiple of the grid size except when the Shift-key is pressed. Pressing the Shift-key while dragging causes the lower-left corner to be snapped to the grid. Pressing the Ctrl-key restricts the movement to either horizontal or vertical displacements.

 The mouse cursor has this shape at the lower left and upper right corners of the selection area. These corners can be dragged to a new position, thus resizing the objects.

 The mouse cursor has this shape at the upper left and lower right corners of the selection area. These corners can be dragged to a new position, thus resizing the objects.

 The mouse cursor has this shape at the left and right sides of the selection area, however only if the Ctrl button is pressed. These sides can be dragged to a new position, thus stretching the objects horizontally.

 The mouse cursor has this shape at the top and bottom sides of the selection area, however only if the Ctrl button is pressed. These sides can be dragged to a new position, thus stretching the objects vertically.

 The mouse cursor has this shape at the corners of the selection area, however only if the Ctrl button is pressed. The objects can now be rotated by dragging the corner to a new position.

Pressing the Shift-key while sizing an object causes the sizing to be symmetrical with respect to the center of the object.

2.3.2 Edit object nodes

 The *edit object nodes mode* allows you to edit the nodes of a single object. A small rectangle is drawn at each node of the selected object. These nodes can now be dragged to a new position using the left mouse button. Clicking the right mouse button once during dragging will leave a copy of the original object.

In the edit object nodes mode only one object can be selected. Selecting a new object will automatically deselect a previously selected object.

2.3.3 Zoom in/out

 The *zoom mode* allows you to zoom in at details of your layout. Just draw a rectangle in the window while pressing the left mouse button and CleWin will automatically zoom in to this rectangle.

Panning the layout window
The zoom mode also allows you to pan the window contents, i.e. shifting it on the display. This can be done in two ways:
1. Using the left mouse button: hold down the Ctrl-key, press the left mouse button, and drag the window contents to a new position.
2. Using the right mouse button: just press the right mouse button and drag the window contents to a new position.

Zoom out
Clicking the right mouse button once causes CleWin to zoom out to the previously visible area.

2.3.4 Insert rectangles

 The *insert rectangles mode* allows you to insert new rectangular objects in the layout.

Inserting rectangles can be done in two ways:
- Click once at a desired corner of the new rectangle using the left mouse button, move the mouse to the desired position of the opposite corner and click the left button again, or
- Press the left mouse button at the first corner, move the mouse, and release the mouse button at the opposite corner.

Boxes (rotated rectangles):
Boxes (rotated rectangles) cannot be entered directly. First you have to enter a normal rectangle. Next, you can use the Edit|Transform... command to rotate the rectangle and transform it into a box. Alternatively, you can select the Edit objects mode and rotate the rectangle using the mouse while pressing the Ctrl-key.

2.3.5 Insert polygons

 The *insert polygons mode* allows you to insert new polygons in the layout.

To enter a new polygon: click the left mouse button at every node of the polygon. After entering the last node click the right mouse button to stop.

2.3.6 Insert wires

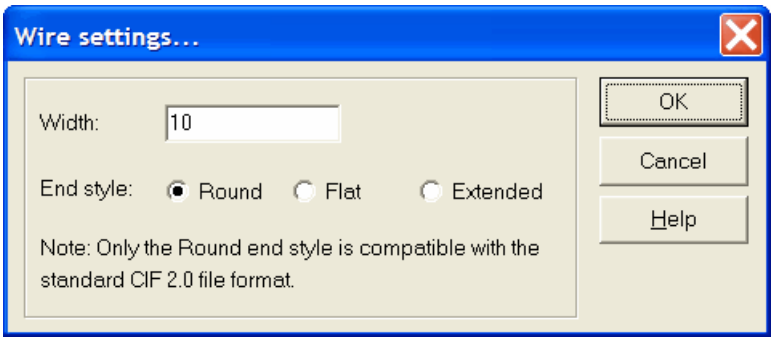
 The *insert wires mode* allows you to insert new wires in the layout.

To enter a new wire: click the left mouse button at every node of the wire. After entering the last node click the right mouse button to stop.

To change the wire settings: click on the insert wires button using the right mouse button or select the Wire settings... command from the popup menu that appears when the right mouse button is clicked in the edit window.

The Wire settings... dialog

The wire settings dialog changes the default wire thickness and end style:



Note that the wire end style is often not supported by mask manufacturers! You need to contact your mask manufacturer about this. The standard CIF file format only supports "Round" end styles. Therefore, when using CIF, wires will probably have "Round" ends on the mask, independent of the setting in CleWin. CleWin stores the end style information in CIF files by inserting additional comment lines. The GDS-II format supports all wire end styles, however they are not always supported by mask making equipment.

2.3.7 Insert circles

The *insert circles mode* allows you to insert new circles in the layout.

Inserting circles can be done in two ways:
- Click once at a desired center of the new circle using the left mouse button, move the mouse to the desired position of the edge of the circle and click the left button again, or
- Press the left mouse button at the center, move the mouse, and release the mouse button at the edge of the new circle.

2.3.8 Insert arc-wires and rings

The *arc-wires mode* allows you to insert wires with a shape corresponding to a section of a circle. Also rings can be inserted in this mode.

First you have to click with the left mouse button at the center of the circle. Now, when the mouse is moved a dashed ring appears. Next, click the left mouse button to define the starting point of the arc. Finally, click the left mouse button again to define the end point.
When the starting point and end point are equal, a ring instead of an arc-wire will be inserted.

2.3.9 Insert text

The *insert text mode* allows you to insert text in the layout.

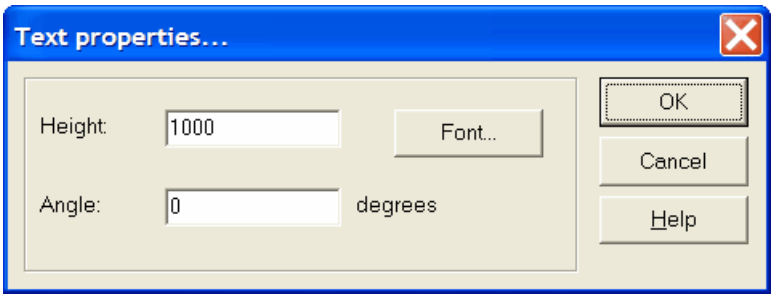
Click the left mouse button at the point where you want to insert text. A dialog will appear prompting you to enter the desired text.

The text will appear in the currently active font. You can change the text and/or the font by selecting the Edit|Properties... command.

To change the current text settings: click on the *insert text mode* button using the right mouse button or select the Text properties... command from the popup menu that appears when the right mouse button is clicked in the edit window.

The Text properties... dialog

The *text properties... dialog* changes the default text height, text angle and font:

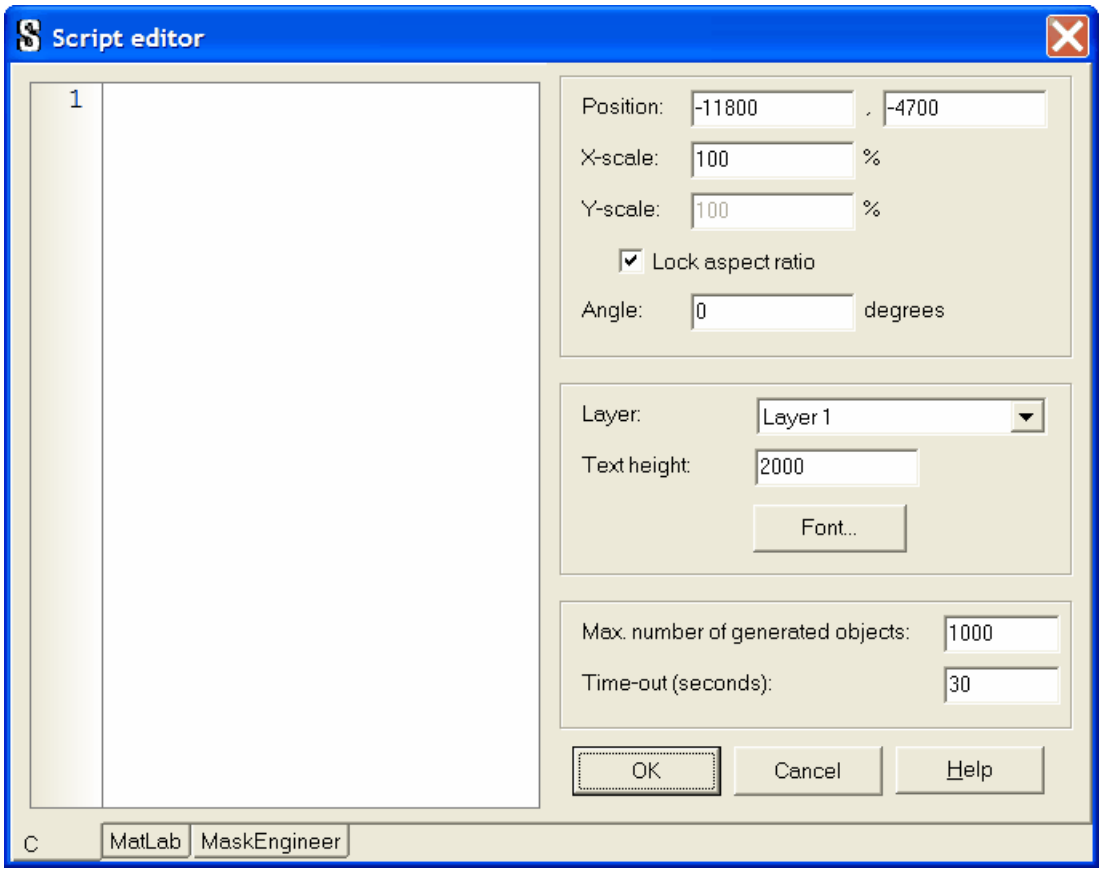


2.3.10 Insert script objects

The *insert script objects mode* allows you to generate objects using a scripting language. This mode is only available in the "pro" version of CleWin. Two scripting languages are included with CleWin: "C" and "MaskEngineer". Furthermore, if you have MatLab installed on your computer, you will also be able to execute scripts in the MatLab engine.

Click the left mouse button at the point where you want to insert the script object. A dialog will appear in which you can enter the script and specify a transformation (scaling factors, rotation angle) to apply to the objects generated by the script. A script object belongs to a layer and by default the objects are generated in this layer. It is however possible to overrule the default layer inside the script. The Tab Pages at the bottom of the dialog can be used to switch between the available scripting engines.

More information on using scripts can be found in the chapter Using scripts.



2.3.11 Measure distances

The *measure distances mode* allows you to easily inspect the distance between objects and view cross sections of the design.

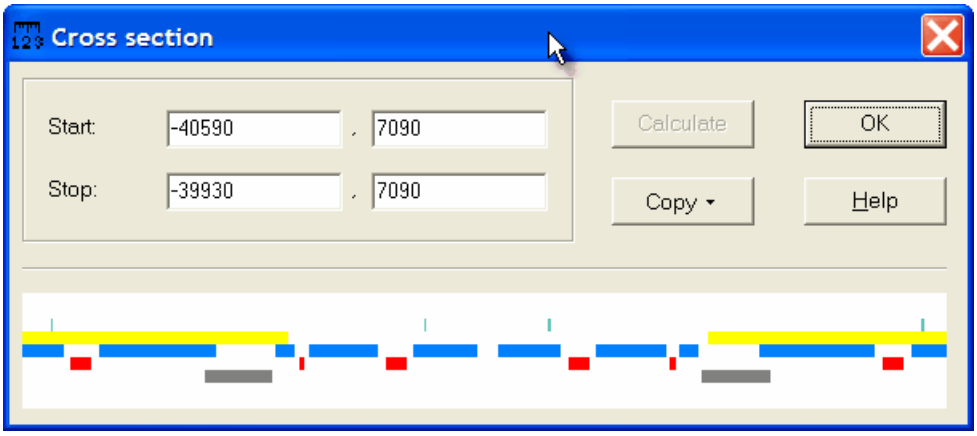
Click the left mouse button at the point from where you want to measure the distance or view a cross section. Now CleWin will display the distance from this point to the mouse. Clicking the left mouse button again on a second point completes the measurement. The distance between the two points remains on the screen until a new measurement is started (by clicking the left mouse button), or until it is removed by clicking the right mouse button and selecting *Hide measurement* from the popup menu. The popup menu also contains the command *View cross section*, which activates the Cross section viewer.

A completed measurement, i.e. when two points have been selected, will remain visible in the other edit modes. A completed measurement can be adjusted by clicking on one of the nodes. This node can then be placed at a new position. In this way you can make accurate measurements over large distances: first roughly position the measurement, then switch to the zoom mode and zoom in on one of the nodes, finally switch back to the measure distances mode and fine-position the measurement node.

2.3.12 Cross section viewer

The cross section viewer can be accessed from the Measure distances mode. Simply draw a

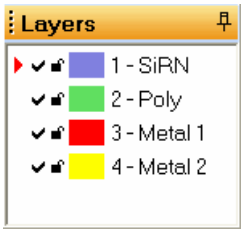
measurement ruler, click the right mouse button, and select *View cross section* from the popup menu. The following dialog will appear:



The lower part of the dialog shows a cross section of the layers along the measurement ruler. The coordinates of the ruler are shown in the top part of the dialog and can be edited. Press the *Calculate* button to calculate a new cross section image.
The *Copy* button activates a drop down menu that allows you to copy the cross section to the windows clipboard (as text or as figure) or to the process simulator FlowDesigner from PhoeniX BV.
When you are ready, select the *OK* button to close the dialog.

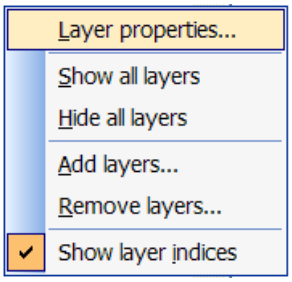
2.4 The layer panel

The layer panel is located at the right side of the main CleWin window and allows you to change the active layer or change layer properties. It can automatically slide in and out when needed, but it can also be locked in a fixed position.



Each line in the layers panel represents a layer. Clicking on the ✔ symbol will hide or show the layer. Clicking on the 🔓 symbol will lock/unlock the layer. A locked layer cannot be edited.
The currently selected layer is indicated by the red arrow (▶). Only one layer can be selected at a time. A layer is selected by clicking on the colored square or on the name of the layer.

Clicking with the right mouse button on one of the layer buttons shows the following popup menu:



The Layer properties..., Add layers..., and Remove layers... commands are a shortcuts to the same commands in the Layout|Layers menu.

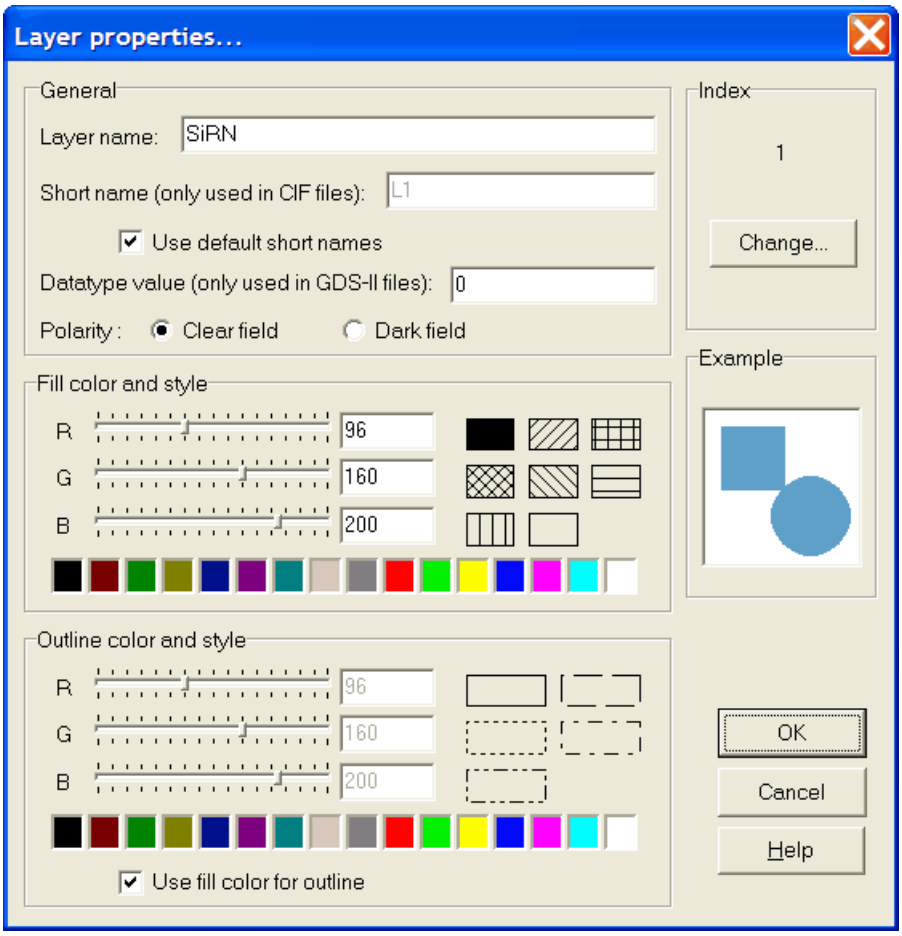
The *Show all layers* command will make all layers visible.
The *Hide all layers* command will hide all layers.

When the option *Show layer indices* is selected, the layer panel will show the layer indices in front of the layer names.

2.4.1 Layer properties

The *Layer properties...* dialog allows you to change the names, colors and line styles of the layers.

You can activate the *Layer properties...* dialog by double-clicking on a layer button in the layer panel or by selecting the Layout|Layers|Layer properties... command.



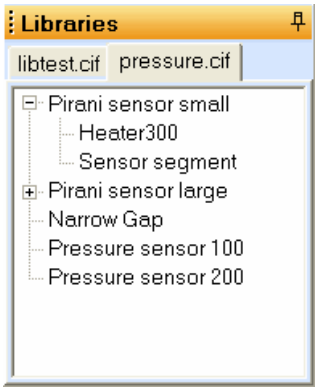
In the "general" part of this dialog you can specify a name for the layer. When using the CIF file format, other layout tools than CleWin often assume that layer names are short mnemonics of only 4 or 6 characters. Therefore, you can also specify a short name for the layer to be used in CIF files.
The GDS-II file format does not support layer names at all; only the layer indices can be stored in the file. Therefore, when reading or writing in GDS-II format, CleWin will automatically read or create a layer map file with the same name.
The value for *Datatype* is only used in GDS-II files. Although GDS-II allows to specify a *Datatype* value for each individual object, the current version of CleWin will assign the same value to all objects in a layer.
The polarity of a mask can be either *Clear field* or *Dark field*. *Clear field* means that the mask will be transparent everywhere except for the structures drawn in CleWin. *Dark field* means that the structures will become transparent. For the moment this option is only used for the cross section calculation in the Cross section viewer.

2.5 The library panel

CIF and GDS-II files can be opened as symbol libraries with the File|Open library... command. When one or more libraries are opened a list of their symbols appears in the library panel. These symbols can now be dragged by the mouse and dropped on the edit area of layout windows. The result is the same as a copy and paste operation between two layout windows. The symbol definition is copied and a new instance of the symbol appears in the layout.

The library panel looks similar to the tree view in layout windows. At the top it contains tab pages to

switch between the open library files. The panel can be moved with the mouse and used as a floating panel, or it can be docked to the right edge of the main window.



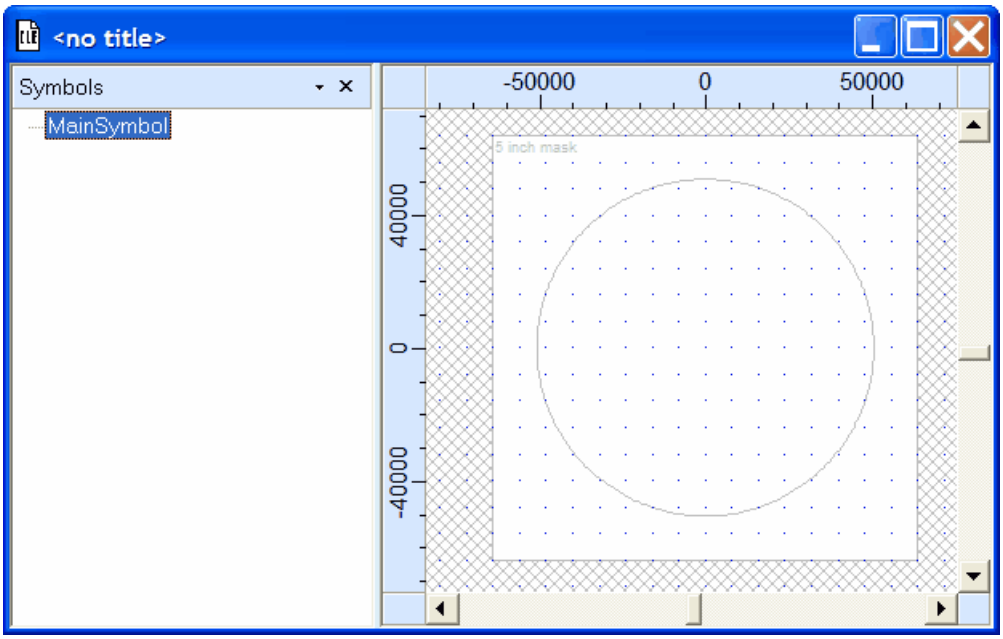
Opening more library files will simply add additional tabs in the library panel. A library file can be closed by clicking on the library panel with the right mouse button and selecting *Close library* from the popup menu.

2.5.1 Using library symbols

Symbols can be dragged from the library panel and dropped on the edit area of a layout window. To start dragging a symbol, press the left mouse button on the symbol name. Next, move the mouse while keeping the button pressed. Release the mouse button when the mouse is over the edit area of a layout window to drop the symbol.

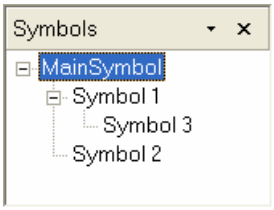
2.6 Layout windows

After opening a design with the File|Open... command or creating a new design with the File|New command a layout window will be opened.
At the left side of this window you see a tree-view panel with a hierarchical overview of the defined symbols. In the case of a new layout, only one symbol will be defined: "MainSymbol".
At the right side of the window you see the layout area, which shows the contents of the selected symbol definition.



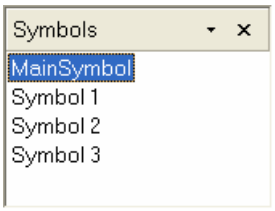
2.6.1 The treeview

The *treeview* at the left of the layout window shows a hierarchical view of the symbol definitions. At the top of the list is the main symbol, which usually is the symbol that is used for producing masks. Selecting a symbol in the treeview will show its contents in the layout area so that it can be edited. Clicking on a symbol name with the right mouse button will show a popup menu with only one command: *Properties*. Selecting this command will allow you to change the name of a symbol. Furthermore, it will show the number of objects in the symbol and the amount of memory occupied by the symbol definition.



Clicking on the downward pointing arrow will show the following menu:

By selecting the item *List*, the treeview will show an alphabetically ordered list of all symbol definitions in the layout. This list also contains symbol definitions that are not used and, therefore, do not appear in the hierarchical view.



Selecting a symbol will again show the contents in the layout area. Clicking on a name with the right mouse button will again show a popup menu, however in this case it also contains the command *Delete*. Symbol definitions that are not used, i.e. that are not called in any other symbol definition, can be deleted. However, be careful because this action cannot be undone.

2.6.2 The layout area

The layout area is the area that shows a drawing of the selected symbol definition. In this area you can edit existing objects or insert new objects, depending on the current edit mode.

2.6.3 Rulers and guidelines

The rulers at the top and left of the layout area show the coordinate range that is currently visible. Clicking on the rulers will insert a guideline. Coordinates will snap to guidelines if the option Layout|Snap|Snap to guidelines is active. A guideline is deleted by dragging it beyond the end of the ruler.

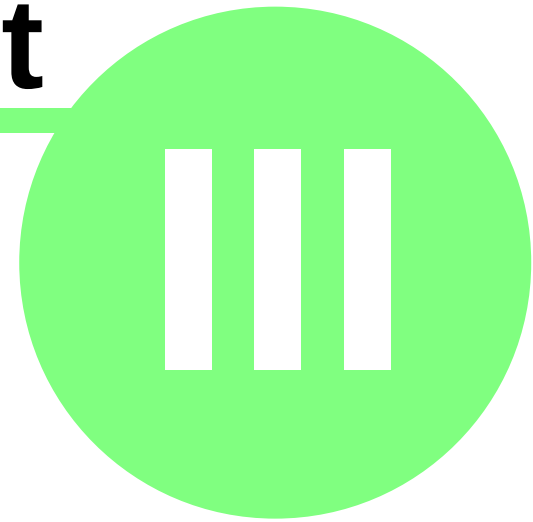
2.7 Useful hints and tricks

- Pressing the space bar will switch from any of the edit modes into the basic  mode and back.
- Pressing the N(odes), Z(oom), R(ectangle), P(olygon), W(ire), C(ircle), A(rc wire), T(ext), S(cript) or M(easure) key will switch to the corresponding edit mode.
- The I(n) and O(ut) keys can be used to zoom in and zoom out.
- Pressing the Ctrl-key will allow stretching and rotation of selected objects. Avoid scaling and stretching of symbol instances (see Editing symbol instances)
- Pressing the Ctrl-key while moving objects will restrict movement to either the horizontal or vertical direction.
- The cursor keys can be used to shift the visible area in the layout window (when nothing is selected), but also to move selected objects by a distance equal to the grid spacing.
- Clicking the right mouse button once while dragging a selection will leave a copy of the original objects in place.
- Quite often objects share a node and clicking on the node may select the wrong object. In that case, click again at the same position and the next object will be selected. This also works with more than 2 objects.
- Pressing the Esc-key will deselect all currently selected objects.

CleWin 4.1

User Guide

Part



3 Creating a layout

Creating a design consists largely of drawing objects. In most cases you will use the mouse pointer to do this. Object combinations that need to be repeated can be combined into a symbol, as is explained in the section Working with symbols.

3.1 Drawing objects

CleWin supports the following basic objects: (rotated) rectangles, polygons, wires, circles, and arc-wires or rings. Before drawing an object you need to select the layer in which you want to draw it. Each object type has a corresponding edit mode in which new objects can be inserted.

	Mode	Objects
	Insert rectangles	Rectangles. Boxes (rotated rectangles) are created by rotating an existing rectangle
	Insert polygons	Polygons
	Insert wires	Wires
	Insert circles	Circles
	Insert arc-wires and rings	Arc-wires (ring segments) and rings
	Insert text	Text
	Insert script objects	Scripts for automatic generation of the other objects

There is no special mode for inserting symbol instances. Symbol instances can be created by simply dragging a symbol from the list in the treeview into the layout area of the layout window.

3.2 Editing objects

To edit objects by using the mouse pointer you need to select one of the following modes:

 Select and edit objects
In this mode you can move, stretch and rotate selected objects. An unlimited number of objects can be selected and editing will affect all selected objects.

 Edit object nodes
In this mode you can edit the individual nodes of objects. Only one object can be selected.

For precise editing of object coordinates you may prefer to use the Edit|Properties... command so that you can use the keyboard for entering coordinate values.

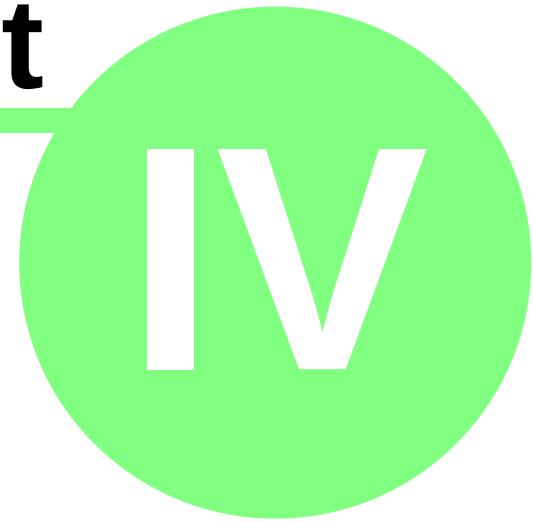
Objects can be grouped together by using the Arrange|Group command. Grouped objects can be edited as if they were a single object.

Symbol instances can be moved and rotated like any other object. Scaling and/or stretching of symbol instances should be avoided as explained in the section "Editing symbol instances".

CleWin 4.1

User Guide

Part



4 Working with symbols

CleWin is a hierarchical layout editor. That means that you can define symbols that contain not only basic drawing primitives like rectangles, polygons, etc., but also calls to other symbol definitions. Working with symbols is a powerful way to limit the file size when a design contains large numbers of identical structures. The symbol definition is only stored once and for all occurences of the symbol only the position and sometimes rotation and scaling information needs to be stored.
At the left side of each layout window you see a tree-view panel with a hierarchical overview of the defined symbols.

4.1 Creating a new symbol

To create a new symbol, you can simply select the objects that should be inside the new symbol and apply the Arrange|Combine... command. A dialog will appear that allows you to specify a name for the new symbol. Furthermore, you can specify the coordinates of the "origin", i.e. the coordinates that will correspond to (0,0) inside the symbol definition.

4.2 Editing a symbol definition

An existing symbol definition can be opened for editing in two ways:

- You can select the name of the symbol in the tree-view panel. The symbol contents will now be drawn in the layout window and can be edited.
- You can select an instance of the symbol and apply the Arrange|Open symbol command or double-click on the symbol instance. This will select the symbol in the tree-view panel and the contents can be edited.

After editing a symbol definition, all instances of the symbol will change automatically.

4.3 Deleting a symbol definition

A symbol definition can only be deleted if it is not used anymore, i.e. if it is not called by any other symbol. In that case, the symbol name may not be visible in the treeview. You need to switch to the "list" representation to be able to select the symbol. To delete the symbol definition: click on the symbol name using the right mouse button and select "Delete" from the popup menu that apears.

4.4 Creating symbol instances

An instance of a symbol can simply be created by dragging the symbol from the treeview into the layout area of the layout window. If there are not yet any other instances of the symbol it does not appear in the hierarchical view and you may need to switch the treeview into "list" mode.

4.5 Editing symbol instances

Symbol instances can be edited using the mouse pointer like any other object in the Select and edit objects mode. Alternatively, the Edit|Properties... command allows editing of a symbol instance.

Please note that scaling of symbol instances is not supported by all file formats. In CIF files scaling of symbol instances is not allowed and CleWin will use special extensions to the file format. As a result, a CIF file with scaled symbol instances may not be accepted by your mask manufacturer. In such a case a solution might be to flatten the design or at least uncombine the scaled instances. In GDS-II scaling of symbol instances is allowed, however the values for *X-scale* and *Y-scale* need to be identical. Only the DXF format supports arbitrary scaling of symbol instances.

CleWin 4.1

User Guide

Part V

5 Using scripts

The "pro" version of CleWin 4 currently supports four scripting languages: C, MatLab, MaskEngineer and Lua. Each of these languages has its own merits and which language you should choose depends very much on the specific application. The programming language C is well known and rather easy to learn. The same is true for MatLab, but to use MatLab scripts you will need a valid MatLab license. The MaskEngineer language is less known but extremely powerful for generating mask layouts.

In C, MatLab and Lua scripts, all basic CleWin objects, rectangles, boxes (rotated rectangles), polygons, wires, circles, rings, texts, and symbol references, can be generated. MaskEngineer scripts can generate a variety of predefined shapes. In CleWin these shapes will appear as polygons or wires. In all script languages, coordinate values and dimensions are expected to be in micrometers. Coordinates are relative to the point where the script is inserted in the design.

Use the CIF format to store designs containing scripts
Note that scripts cannot be stored in standard file formats like CIF, GDS-II and DXF. CleWin uses comment lines in CIF to embed script objects in the file. However, this is not possible in the other file formats, therefore, if you want to edit a script later you should always use the CIF format to save your work. In all other file formats only the objects generated by a script will be stored and not the text of the script itself.

5.1 Using the programming language C

The C programming language as it is implemented in CleWin is based on the freely available C interpreter EiC ("Extensible Interactive C"), which is an open source portable C interpreter. Unfortunately development of EiC seems to have stopped and the web site for it has disappeared. Still, the source code of EiC can be found at several sites on the internet. The source code of the `clescrpt.dll` used by CleWin is available from WieWeb software.

Besides the basic C language structure the following CleWin-specific functions are implemented:

```
int layer(char *name);
int rectangle(double x1, double y1, double x2, double y2);
int box(double xcentre, double ycentre, double width, double height, double
angle);
int polygon(int numnodes, double *nodes);
int wire(int numnodes, int style, double width, double *nodes);
int circle(double xcentre, double ycentre, double radius);
int ring(double xcentre, double ycentre, double radius, double width);
int text(char *text, TM *matrix);
int symbol(char *symbolname, TM *matrix);
```

The first function can be used to change the current layer. The function expects a pointer to a null-terminated string containing either the name of the layer or the character '#' followed by the layer index. The other functions are used to generate the basic CleWin objects: rectangles, boxes (rotated rectangles),polygons, wires, circles, rings, texts, and symbol references. All coordinate values and dimensions are expected to be in micrometers. Coordinates are relative to the point where the script is inserted in the design. The last two functions require a reference to a transformation matrix structure to specify the position and orientation of the text or symbol reference. This structure is defined as follows:

```
typedef struct TransMatrix {
  double m11;
  double m21;
  double m12;
  double m22;
```

```
  double m13;
  double m23;
} TM;
```

Working with transformation matrices

A transformation matrix like this is commonly used for 2D transformations using homogeneous coordinates. The new, transformed coordinates are calculated as follows:

```
x' = m11*x + m12*y + m13
y' = m21*x + m22*y + m23
```

Any 2D transformation (rotation, scaling, stretching, translation) or combination of transformations can be realized with a single TransMatrix structure. For example, for translation one would use m13 and m23, for scaling m11 and m22, and for rotation through an angle θ one would use m11 = cos θ, m12 = -sin θ, m21 = sin θ and m22 = cos θ. m13 and m23 have the unit micrometer. m11, m12, m21, and m22 are dimensionless multiplication factors. More information on using matrix calculations for 2D transformations with homogeneous coordinates can be found in many textbooks about computer graphics and on the internet.

To simplify working with transformation matrices, the following functions are implemented:

```
void unityTM(TM *matrix);
void translateTM(TM *matrix, double dx, double dy);
void rotateTM(TM *matrix, double xcentre, double ycentre, double angle);
void scaleTM(TM *matrix, double xcentre, double ycentre, double xfactor, double
yfactor);
```

For example, to generate the text "hello" at position (1000,0) and at an angle of 15 degrees, one would type the following lines:

```
TM m;
unityTM(&m);
rotateTM(&m, 0, 0, 15);
translateTM(&m, 1000, 0);
text("hello", &m);
```

Example: Using a C script to generate device numbers

On a mask it is often very useful to mark each individual device by a unique number, so that after production of the devices it is alway possible to locate the original position on the wafer. Usually, devices are located at regular positions and then it is rather easy to write a script that generates the numbers. For example, the following script generates numbers in a matrix with 3 rows and 7 columns, having a column and row spacing of 8000 micrometers:

```
#define W 8000
#define H 8000
#define NROWS 3
#define NCOLS 7

int irow,icol;
char str[3];

// Define transformation matrix:
TM m;
unityTM(&m); // initialize with unity matrix

// Start numbering at 1:
int n=1;
```

```
// Use a double for-loop to generate the text objects:
for(irow=0; irow<NROWS; irow++) {
  for(icol=0; icol<NCOLS; icol++) {
    // Generate text object:
    sprintf(str,"%d",n++);
    text(str,&m);
    // Next column:
    translateTM(&m,W,0);
  }
  // Next row:
  translateTM(&m,-NCOLS*W,-H);
}
```

The Math library

In addition to the CleWin-specific functions, the following standard C math functions and constants are implemented (these are usually defined in math.h):

```
double acos(double);
double asin(double);
double atan(double);
double atan2(double, double);
double cos(double);
double sin(double);
double tan(double);
double cosh(double);
double sinh(double);
double tanh(double);
double exp(double);
double frexp(double, int *);
double ldexp(double, int);
double log(double);
double log10(double);
double modf(double, double *);
double pow(double, double);
double sqrt(double);
double ceil(double);
double fabs(double);
double floor(double);
double fmod(double, double);

#define M_E         2.71828182845904523536
#define M_LOG2E     1.44269504088896340736
#define M_LOG10E    0.43429448190325182765l
#define M_LN2       0.69314718055994530942
#define M_LN10      2.30258509299404568402
#define M_PI        3.14159265358979323846
#define M_PI_2      1.57079632679489661923
#define M_PI_4      0.78539816339744830962
#define M_1_PI      0.31830988618379067154
#define M_2_PI      0.63661977236758134308
#define M_1_SQRTPI  0.56418958354775628695
#define M_2_SQRTPI  1.12837916709551257390
#define M_SQRT2     1.41421356237309504880
#define M_SQRT_2    0.70710678118654752440
```

5.2 Using MatLab

If you have MatLab installed, CleWin adds a Tab Page in the "Insert script..." dialog that allows you to enter MatLab scripts. Besides the full MatLab functionality, you can use the following functions to

generate objects in CleWin:

```
rectangle(x1, y1, x2, y2)
box(x, y, width, height, angle)
polygon(nodes)
wire(style, width, nodes)
circle(x, y, radius)
ring(x, y, radius, width)
text(text, matrix)
symbol(symbolname, matrix)
```

For example, to draw a rectangle of 1000 by 1000 micrometers you could use the following line in a script:

```
% Draw a rectangle of 1000x1000um:
rectangle(0, 0, 1000, 1000)
```

Note that in MatLab the percentage sign % indicates a comment line.

CleWin automatically starts the MatLab engine the first time a MatLab script needs to be executed. The MatLab command window will be visible on the screen, which can be useful for debugging a script. For example, by typing a variable name in the MatLab command window, you may check whether the variable has the expected value.

Changing the layer

By default, objects are generated in the layer that is selected in the "Insert script..." dialog. You can however change the current layer by calling the function:

```
setlayer(layername)
```

The parameter layername must be the name of the new layer or it should start with a '#' character followed by the layer index. So,

```
setlayer('#2')
```

will select the layer with index 2.

Working with transformation matrices

The functions to generate text or insert a symbol instance expect (besides the text or symbol name) a transformation matrix. This is a 3x3 matrix that defines the position and size of the text or symbol instance. For example, to insert a text rotated by 15 degrees you could use the following script:

```
% Define a transformation matrix:
m = [1 0 0; 0 1 0; 0 0 1]
% Rotate by 15 degrees around the origin (0,0):
theta = 15.0*pi/180
rotmatrix = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1]
m = rotmatrix*m
% Show rotated text:
textstring = 'text at an angle of 15 degrees'
text(textstring,m)
```

If, in addition the text should be scaled to 50% using the point (20000,1000) as reference, you could add:

```
% A matrix to translate by (-20000,-1000), so that the point (20000,1000) is at
the origin:
m1 = [1 0 -20000; 0 1 -1000; 0 0 1]
```

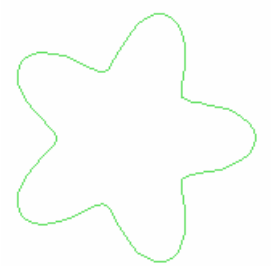
```
% A matrix to scale by a factor 0.5:
m2 = [0.5 0 0; 0 0.5 0; 0 0 1]
% A matrix to translate back by (20000,1000):
m3 = [1 0 20000; 0 1 1000; 0 0 1]
% Show rotated and scaled text:
text('text at an angle of 15 degrees and scaled',m3*m2*m1*m)
```

Example: Creating a polygon

In MatLab, all variables are represented as matrices and a single expression can operate on all elements of a matrix. This allows you to generate complex polygons with just a few lines of code. For example, the following 4 lines:

```
% Create a vector consisting of 100 elements with values ranging from 0 to 2pi
angle = linspace(0,2*pi,100);
% Create a radius vector giving a radius for each angle:
r = 6000 * (1+0.3*cos(5*angle))
% Calculate the nodes of a polygon, i.e. a matrix with 2 columns and 100 rows:
nodes = [r.*cos(angle); r.*sin(angle)]';
% Generate the polygon in CleWin:
polygon(nodes);
```

generate the shape:



If you would like to rotate the shape over a certain angle you may simply change the last line into:

```
% Create a transformation matrix:
rotangle=10*pi/180
T=[cos(rotangle) -sin(rotangle); sin(rotangle) cos(rotangle)]
% Transform the nodes and generate the polygon:
polygon((T*nodes')');
```

5.3 Using MaskEngineer scripts

MaskEngineer, which uses the PhoeniX Script Language, is a powerful software package to generate mask layouts. Starting with version 4 this script language is available in CleWin.

Basics of the PhoeniX Script Language

Variables

Variables are used to store values so they can be used later on. Variables can have "any" name, but when <empty space> or <special characters> are used, then the single quote ' is needed. The following names are therefore valid: a, MyVariable, 'My variable with some spaces and a ;'. We recommend to use descriptive names, which are not too short.
Variables can be defined using the statement var ..., where the ... refers to the variable name. For example:

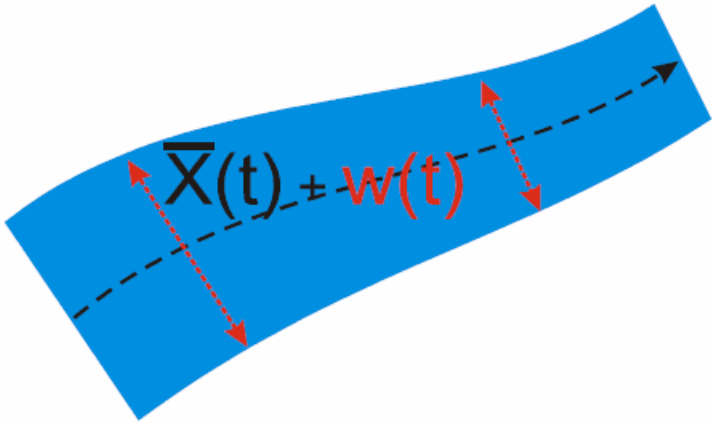
```
var a=10,b,MyVar=a+b;
```

Equations
To define a value, you can use an extensive library of functions. Something like

```
var a=cos(tan(b)+10.0);
```

is no problem. All normal calculation rules apply to the PhoeniX script language.

Elements
Elements are basic parts of a layout. They can either be a control element or an actual graphic / mask element.
Most elements are defined using a center path and a width profile, as shown below.



Defining elements as function of width and length reduces the amount of element definitions considerably. A unique name can be assigned to each element, which can be used later to refer to the element, and, for example connect another element to it. Using relevant names for elements simplify the readability of the design considerably.
The following pre-defined mask elements are available in CleWin:

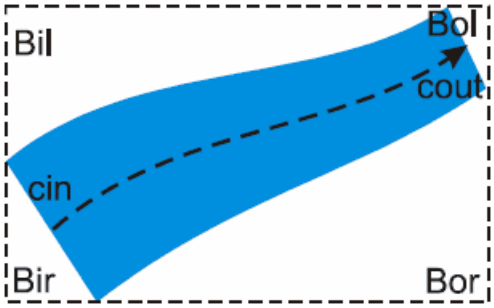
```
Bend, BendCosine, BendPolar, BendS, BendSine, CenterPath, Circle,
CurveBoundary, CurveSymmetric, CurveUpDown, Donut, Ellipse,
InBetween, Offset, Polygon, Polynom, RectangleRounded,
SBBendPolar, SB_Bend, SBend, Straight, StraightOffset,
Triangle, Triangle2LA, Triangle3L, TriangleL2A, Wire, Yblunt
```

More elements can be licensed from PhoeniX BV.

Ports
Elements can be positioned using two methods:
- absolute positioning (absolute coordinate)
- positioning relative to another element (relative coordinate)

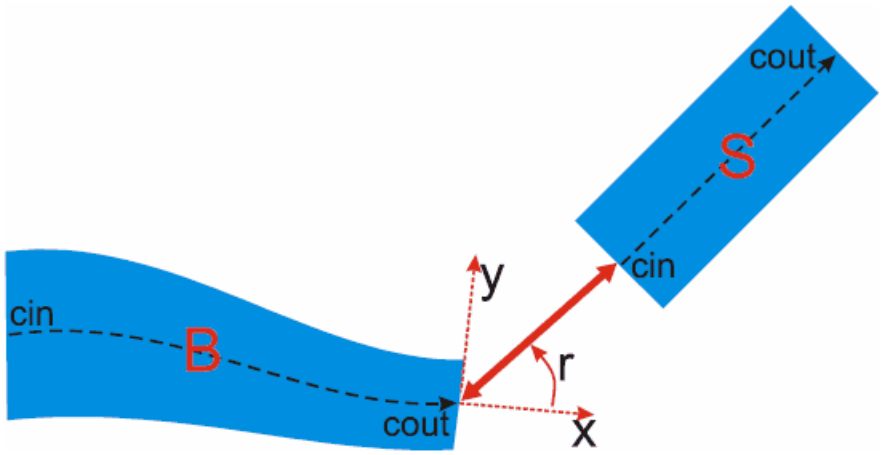
To position elements, each element has "ports" associated with it. Any port can be used to define the element's absolute or relative position. The main ports for most elements are:

cin	'center input' port
cout	'center output' port
Bir	bounding box port, 'input right'
Bil	bounding box port, 'input left'
Bor	bounding box port, 'output right'
Bol	bounding box port, 'output left'



Port names can be at most 4 characters long; longer names can be used, but the remaining characters are ignored. The reason for this limitation is that it allows the MaskEngineer engine to operate more efficiently (faster and consuming less memory) and, since the number of ports of an element is limited, there is no need for using long names. To refer to a specific port of a specific element, simply use the element's name and append the @ sign and the port name to it. So, port 'cout' of element 'B' would read 'B@cout'.

In the illustration below, the element B is positioned using cin -> [0,0] while the element S is defined to be cin -> B @ cout + [x,y,r]. If the properties of element B or its position change (e.g. cin -> [0,5]) then element S will adapt its location too.



Element width
The width of an element can be constant, a pre-defined function (linear, parabolic or exponential) or any other function. For example, consider the following lines, which all generate a straight element with a length of 10 micrometers:

```
ml::Straight(cin->[0, 0] : wfix(1.1),10) m0;
ml::Straight(cin->[0,10] : wpar(1,5),10) m1;
ml::Straight(cin->[0,20] : wlin(1,5),10) m2;
ml::Straight(cin->[0,30] : wexp(1,5,1.91),10) m3;
ml::Straight(cin->[0,40] : wfun{t -> 1.5+(1+t*0.8)+sin(t*physics::PI*6)},10) m4;
```

The first line generates a rectangle with a fixed width of 1.1 micrometer. The other lines generate objects with a width that is changing along the length of the element. In the last line a function is specified for the width, where the parameter t changes from 0 to 1 along the length of the element.

Defining functions
Some elements allow a function to be used to define the coordinates, e.g. *CenterPath*, *CurveBoundary*, *CurveSymmetric* and *CurveUpDown*.

```
ml::CenterPath( last : {t -> 1000*t, 200*sin(6*t)+200*t}, wfix(100), 100);
```

Connecting elements to each other

To connect an element to another element, you can simply replace the `cin->[x,y]` by something like `cin->name@cout+[x,y]`. For example, the following lines generate a straight segment with a bend element connected to its end:

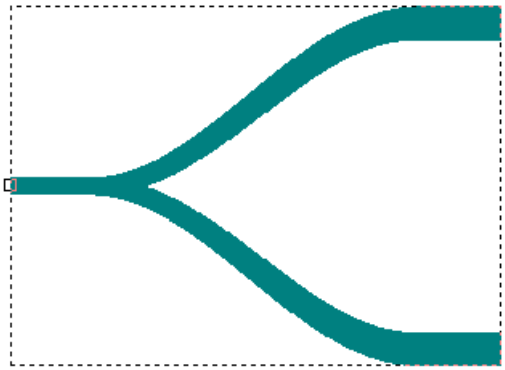
```
ml::Straight(cin->[0, 0] : wfix(1),5) inp;
ml::BendCosine(cin->inp@cout : wlin(1,2),20,10);
```

For convenience, you may use the keyword last to indicate that the standard input of the element should be connected to the standard output of the previous element. So we could also write:

```
ml::Straight(last : wfix(1),5) inp;
ml::BendCosine(last : wlin(1,2),20,10);
```

Example: Y junction

As an example, we will now generate a simple Y-junction as shown below.



We start with defining a few variables for the widths of the input and output segments, for the separation between the outputs, and for the length of the curved elements. This will allow us to change these dimensions later. The complete script looks as follows:

```
var  wIn=1,  // The width of the element inp
     wOut=2, // The width of the element outTop
     Sep=20, // The seperation between the upper and lower output
     Len=20; // The horizontal length of the cosine bend
ml::Straight(cin->[0,0] : wfix(wIn),5) inp;
// Top curve:
ml::BendCosine( last : wlin(wIn,wOut),Len, Sep/2.0);
ml::Straight( last : wfix(wOut),5) outTop;
// Bottom curve:
ml::BendCosine( cin->inp@cout : wlin(wIn,wOut),Len, -Sep/2.0);
ml::Straight( last : wfix(wOut),5) outBot;
```

The first *Straight* element forms the input of the device. A combination of a *BendCosine* and a *Straight* element form the top and bottom outputs.

Example: defining new elements

In the above example, we could also have defined a new element *Yjunction* as follows.

```
layout Yjunction( wIn=1,  // The width of the element inp
                  wOut=2, // The width of the element outTop
                  Sep=20, // The seperation between the upper and lower output
                  Len=20  // The horizontal length of the cosine bend
                )
{
     ml::Straight(cin->[0,0] : wfix(wIn),5) inp;
     // Top curve:
     ml::BendCosine( last : wlin(wIn,wOut),Len, Sep/2.0);
     ml::Straight( last : wfix(wOut),5) outTop;
     // Bottom curve:
     ml::BendCosine( cin->inp@cout : wlin(wIn,wOut),Len, -Sep/2.0);
     ml::Straight( last : wfix(wOut),5) outBot;
}
```

By doing this, we can use *Yjunction* in the remainder of the script just like other elements. We can place it by typing:

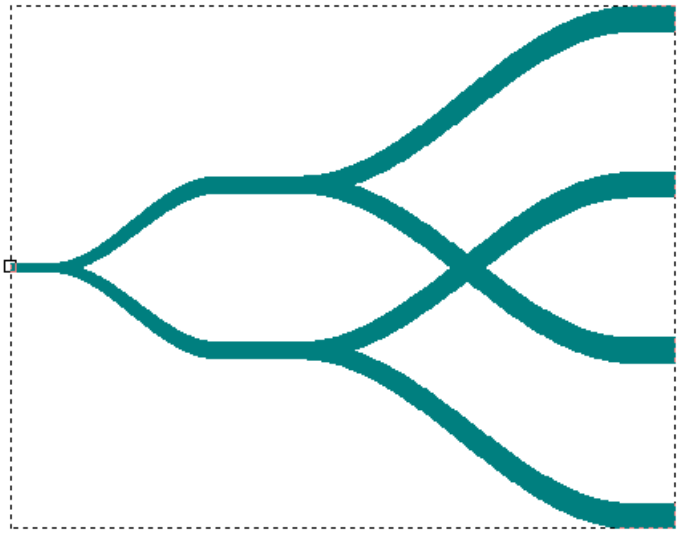
```
ml::Yjunction ( : 1,2,20,20) left;
```

Note that here we do not specify anything before the ':' character, i.e. we do not specify a position for the *Yjunction* element. In fact, the first *Straight* element is still connected to [0,0]. To avoid this limitation we need to give the *Yjunction* an input port and connect the first *Straight* to this input. This is done rather easily by changing the line for the *Straight* element as follows:

```
ml::Straight(cin->this@Inp : wfix(wIn),5) inp;
```

Note that the only difference is that [0,0] is replaced by `this@Inp`. This means that the *Straight* element now starts at the port *Inp* of the current definition, which is the *Yjunction* element. The port *Inp* is created automatically. We can do a similar thing with the outputs by connecting the output ports of the two other *Straight* elements to `this@oTop` and `this@oBot`., which gives the *Yjunction* element two output ports, *oTop* and *oBot*. We can now connect *Yjunction* elements to each other as shown in the script below:

```
layout Yjunction( wIn=1,  // The width of the element inp
                  wOut=2, // The width of the element outTop
                  Sep=20, // The seperation between the upper and lower output
                  Len=20  // The horizontal length of the cosine bend
                )
{
    ml::Straight(cin->this@Inp : wfix(wIn),5) inp;
    // Top curve:
    ml::BendCosine( last : wlin(wIn,wOut),Len, Sep/2.0);
    ml::Straight( last, cout->this@oTop : wfix(wOut),5) outTop;
    // Bottom curve:
    ml::BendCosine( cin->inp@cout : wlin(wIn,wOut),Len, -Sep/2.0);
    ml::Straight( last, cout->this@oBot : wfix(wOut),5) outBot;
}
ml::Yjunction( Inp->[0,0] : 1,2,20,20) left;
ml::Yjunction( Inp->left@oTop : 2,3,40,40) midTop;
ml::Yjunction( Inp->left@oBot : 2,3,40,40) midLow;
```



5.4 Using Lua scripts

Lua is a powerful, fast, light-weight, embeddable scripting language. It is designed, implemented, and maintained by a team at PUC-Rio, the Pontifical Catholic University of Rio de Janeiro in Brazil. There is a special Lua website: http://www.lua.org. The current implementation in CleWin is based on Lua version 5.1.3.

Besides the basic Lua language the following CleWin-specific functions are implemented:

```
setlayer(layer)
rectangle(x1, y1, x2, y2)
box(xcentre, ycentre, width, height, angle)
polygon(nodes)
wire(style, width, nodes)
circle(xcentre, ycentre, radius)
ring(xcentre, ycentre, radius, width)
text(text, transformation)
symbol(symbolname, transformation)
```

The function `setlayer(layer)` sets the layer in which objects are inserted. The parameter `layer` is either a number indicating the index of the layer, or it is a string containing the name of the layer or the character '#' followed by the layer index.
The other functions are used to insert new objects. All coordinate values are in micrometers. Angles are in degrees.

`rectangle(x1, y1, x2, y2)`
Generates a rectangle having opposite corners (x1,y1) and (x2,y2).

`box(xcentre, ycentre, width, height, angle)`
Generates a box, i.e. a rotated rectangle.

`polygon(nodes)`
Generates a polygon. The parameter `nodes` contains a list of coordinate values x1, y1, x2, y2, x3, y3, ... For example, to generate a triangle one could enter the following statement:
`polygon({0,0,1000,0,0,1000})`

`wire(style, width, nodes)`
Generates a wire. The parameter `style` can have 3 values, 0, 1 and 2, corresponding to the end

styles 'round', 'flat' and 'extended'. The parameter `width` is the width of the wire in micrometers and the parameter `nodes` again contains a list of coordinate values.

`circle(xcentre, ycentre, radius)`
Generates a circle at position(`xcentre, ycentre`) with radius `radius`.

`ring(xcentre, ycentre, radius, width)`
Generates a ring at position(`xcentre, ycentre`) with radius `radius` and width `width`.

`text(text, transformation)`
Generates a text string where `text` is the string and `transformation` contains information on the position of the text. The parameter `transformation` can either contain the coordinates of a single point (x, y) at which the text should appear, or it can contain the 6 values needed for a complete 2D transformation (see below).

`symbol(symbolname, transformation)`
Generates a symbol instance. The parameter `symbolname` is a string containing the name of a symbol. The parameter `transformation` defines the position of the new symbol instance (see below).

Working with transformations

A transformation matrix is commonly used for 2D transformations using homogeneous coordinates. The new, transformed coordinates are calculated as follows:

```
x' = m11*x + m12*y + m13
y' = m21*x + m22*y + m23
```

Any 2D transformation (rotation, scaling, stretching, translation) or combination of transformations can be specified by the 6 coefficients m11, m21, m12, m22, m13, m23 of a transformation matrix. For example, for translation one would use m13 and m23, for scaling m11 and m22, and for rotation through an angle θ one would use m11 = cos θ, m12 = -sin θ, m21 = sin θ and m22 = cos θ. m13 and m23 have the unit micrometer. m11, m12, m21, and m22 are dimensionless multiplication factors. More information on using matrix calculations for 2D transformations with homogeneous coordinates can be found in many textbooks about computer graphics and on the internet.

For example, to insert the text "hello" at position (1000, 2000) the following line could be used in the script:

```
text("hello", {1,0,0,1,1000,2000})
```

The matrix elements are expected in the order m11, m21, m12, m22, m13, m23 so that the position is specified by the last two values. Since there is no further transformation needed, in this case the same result is obtained using:

```
text("hello", {1000,2000})
```

To simplify working with transformation matrices, four additional functions were implemented in the Lua interpreter:

```
transformation = identity();
transformation = translate(transformation, x, y);
transformation = rotate(transformation, xcentre, ycentre, angle);
transformation = scale(transformation, xcentre, ycentre, xfactor, yfactor);
```

The function `identity()` simply generates an identity matrix, having m11=m22=1 and all other elements equal to 0. The other functions adapt the transformation passed as the first parameter according to the specified values and return a new transformation.

For example, to generate the text "hello" at position (1000,0) and at an angle of 15 degrees, one would type the following lines:

```
matrix = identity()
matrix = rotate(matrix, 0, 0, 15);
matrix = translate(matrix, 1000, 0);
text("hello", matrix);
```

Example: Using a Lua script to generate device numbers

On a mask it is often very useful to mark each individual device by a unique number, so that after production of the devices it is alway possible to locate the original position on the wafer. Usually, devices are located at regular positions and then it is rather easy to write a script that generates the numbers. For example, the following script generates numbers in a matrix with 3 rows and 7 columns, having a column and row spacing of 8000 micrometers:

```
W = 8000
H = 8000
NROWS = 3
NCOLS = 7

-- Create transformation matrix:
transform = identity();

-- Start numbering at 1:
n=1;

-- Use a double for-loop to generate the text objects:
for irow=0,NROWS do
  for icol=0,NCOLS do
    -- Generate text string containing the number:
    str = string.format("%d",n)
    -- Generate the text object:
    text(str,transform)
    -- Increment the number:
    n = n+1
    -- Next column:
    transform = translate(transform,W,0)
  end
  -- Next row:
  transform = translate(transform,-(NCOLS+1)*W,-H);
end
```

CleWin 4.1

User Guide

Part

6 File formats

CleWin supports the following file formats:

6.1 CIF files: CleWin's default file format

Caltech Intermediate Format (CIF) is CleWin's default file format. CIF files created by CleWin are fully compatible with the standard CIF 2.0 format, except in a few special situations when certain objects cannot be stored in the standard format. In that case CleWin shows a warning message with more information.

CIF files are text files and can be edited with any text editing program, e.g. NotePad or WordPad in the Windows environment. Each statement in CIF consists of a keyword, which is usually abbreviated to a single letter, followed by parameters and terminated with a semicolon. CIF files created by CleWin contain only one statement per line, although this is not prescribed by the CIF format.

This section provides information on the CIF format as it is generated by CleWin. For a more general discussion of the format, please refer to the references given below.

Unit
All coordinate values in a CIF file are integer values. The default unit in CIF is the centimicron. The unit can be changed inside symbol definitions by applying a scaling factor (see below). CleWin versions 3 and later use this feature to set the unit size to 1 nanometer. Thus, in CIF files generated by CleWin 4.x all coordinate values are in nanometers.

CIF statements
There are only a few CIF statements:

L	LAYER	Selects a new layer
B	BOX	Draws a (rotated) rectangle
P	POLYGON	Draws a polygon
W	WIRE	Draws a wire
R	ROUNDFLASH	Draws a circle
DS	DEFINITION START	Indicates the starts of a symbol definition
DF	DEFINITION FINISH	Indicates the end of a symbol definition
C	CALL	Draws a symbol instance
DD	DELETE DEFINITION	Deletes a range of symbol definitions (not used by CleWin)
E	END	Indicates the end of the file

As you can see from this list, there are only 4 basic shapes that can be drawn: rectangles, polygons,

wires and circles. Still, CleWin manages to store other shapes like rings, arc-wires, and text, by inserting them as polygons and wires and adding comment lines so that CleWin can correctly retrieve the original object data. If another application reads a CIF file created by CleWin, it will only see the polygon/wire representation of rings, arc-wires and text.

Symbol names
Besides the CIF statements listed above, the digits 0 through 9 can be used to store additional information. There are a number of commonly accepted extensions (for a list see reference 2), but only one of them is used by CleWin (and by virtually all other software accepting the CIF format):

```
9 symbolname;
```

This statement is used directly after the DS statement to assign a name to a symbol definition. The basic CIF format only supports symbol numbers.

The LAYER statement
The LAYER statement (or the letter L) sets the current mask layer. All subsequent objects are drawn in this layer until another LAYER statements sets a new layer. For example, the following statement selects layer L1:

```
L L1;
```

In CIF, layer names are usually restricted in length to a few characters. Therefore, CleWin distinguishes between long and short layer names. The short layer name (by default the letter L followed by the layer index) is used throughout the CIF file and seen by other applications reading the file. The long file name is stored in a comment line close to the beginning of the file. For example, a CIF file created by CleWin containing two layers with index 8 and index 12 will contain a section like:

```
(Layer names:);
L L8; (CleWin: 8 Layer 8/0fe08080 0fe08080);
L L12; (CleWin: 12 This is layer 12/0f60e060 0f60e060);
```

Here, the first line "(Layer names:);" is just a comment indicating that the list of layer names follows. In fact, any statement between brackets is a comment line.
The next line contains a statement and a comment. The statement "L L8;" indicates that layer L8 should be selected, where L8 is the short name of the layer. The comment contains additional information for CleWin. Directly after the string "CleWin:" it contains the layer index followed by a space and the long name of the layer. The long name ends at the "/" character, which is followed by the color and line style information for the layer. Thus, the layer has index 8 and the long name is "Layer 8". Similarly, the last line contains the information for the layer with short name L12, index 12, and long name "This is layer 12".

The BOX statement
The BOX statement (or the letter B) describes a rectangle by giving its length, width, center position, and an optional rotation. The format is as follows:

```
B length width xpos ypos [rotation];
```

Without the rotation field, the four numbers specify a box the center of which is at (xpos, ypos). The length is in the x direction, and the width is in the y direction. The optional rotation field contains two numbers that define a vector endpoint starting at the origin. The default value of this field is (1, 0), which is a right-pointing vector. The length of the box is aligned with this vector. The magnitude of the vector has no meaning. As an example, the following line would describe a box at position (10, 30), with length 50 and width 40, and rotated counterclockwise by 30 degrees:

```
B 50 40 10 30 10 5;
```

The POLYGON statement

The POLYGON statement (or the letter P) takes a series of coordinate pairs and draws a filled polygon from them. Since filled polygons must be closed, the first and last coordinate points are implicitly connected and need not be the same. The CIF format does not impose any restrictions on the complexity and number of points of polygons. However, self-intersecting polygons should be avoided since the result may differ depending on the filling algorithm used by the mask manufacturer. Furthermore, some software packages use a limit of 200 points. CleWin can handle polygons with up to 32768 nodes, but can automatically split polygons that are larger than a specified number of points if necessary (see Layout|Preferences...).

The WIRE statement

The WIRE statement (or the letter W) is used to construct a path that runs between a set of points. The path can have a nonzero width and has rounded corners. After the WIRE keyword comes the width value and then an arbitrary number of coordinate pairs that describe the endpoints. In the CIF format, wires always have round ends. CleWin also supports flat and extended ends, and stores the end style information in comment lines. Most other software will not recognize this information and, as a result will probably give round end styles independent of the setting in CleWin. If you need to rely on a specific end-style, send your design to your mask manufacturer in GDS-II format or convert your wires into polygons just before submitting the file (see Edit|Convert into polygons). GDS-II supports the same end-styles as CleWin, however they are not always accepted by mask manufacturers.

The ROUNDFLASH statement

The ROUNDFLASH statement (or the letter R) draws a filled circle, given the diameter and the center coordinate. For example, the statement:

```
R 20 30 40;
```

will draw a circle that has a radius of 10 (diameter of 20), centered at (30, 40).

The CALL statement

Objects can be combined into symbols using the DS and DF statements. The CALL statement (or the letter C) can then be used to draw an instance of the defined symbol. For example, the statement:

```
C 4;
```

will draw an instance of the symbol definition with index 4.
In addition to simply drawing the symbol, a CALL statement can include transformations. Three transformations can be applied to a symbol instance: translation, rotation, and mirroring.

Transformation	Parameters	Result
T	x, y	Translation by x, y
R	x, y	Rotation by vector (x, y)
MX		Mirror x coordinate
MY		Mirror y coordinate

Transformations are applied in the same order as they are listed in the CALL statement. For example, the statement:

```
C 4 T 10 20 MX R 10 10;
```

draws an instance of symbol 4, translated by 10, 20, then the x coordinate is inverted (mirror in the y-axis), and finally the picture is rotated counterclockwise by 45 degrees.

In addition to the standard transformations, CleWin uses two addition transformations: scaling in the x and y direction.

Transformation	Parameters	Result
SX	factor	Multiply x coordinate by factor
SY	factor	Multiply y coordinate by factor

This extension to standard CIF is supported by a few other tools, e.g. LinkCAD (http://www.linkcad.com/). CleWin will avoid using this extension, but there is no other way to support scaling of symbol instances in CIF. If you use scaled symbol instances, you may consider sending the design to your mask manufacturer in GDS-II format. GDS-II allows symbol scaling, however only one scaling factor is allowed for both the x and y direction. GDS-II does not allow stretching of symbols. Alternatively, you may flatten scaled and/or stretched symbol definitions prior to sending the file to a mask manufacturer.

Defining symbols using the DS and DF statements
Defining symbols for use in a CALL statement is quite simple. The statements needed to draw the symbol are simply placed between DS (Definition Start) and DF (Definition Finish) statements. The DS statement needs 3 arguments:

```
DS 4 1 10;
```

The first argument is the symbol index which is needed by the CALL statement. The second and third argument are the numerator and denominator of a scaling factor that is applied to all coordinate values inside the definition. Thus, in the above example, all coordinates are multiplied by 1 and divided by 10 to give a coordinate value in centimicrons. CleWin always uses the values 1 and 10 to get a resolution of 1 nanometer inside the symbol definitions.

As an example, the following sequence of statements defines a complete symbol consisting of a circle in layer L1 and a box in layer L2:

```
DS 4 1 10;
9 BoxAndCircle;
L L1;
R 20 30 40;
L L2;
B 50 40 10 30;
DF;
```

Note that the statement following the DS statement (starting with the digit "9") assigns the name "BoxAndCircle" to the definition as described above.

References
1. Carver A. Mead and Lynn A. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.
2. Steven M. Rubin, *Computer Aids for VLSI Design*, 1994, Appendix B, http://www.rulabinsky.com/cavd/text/chapb.html.

6.2 GDS-II files

CleWin can read and write the Calma GDS-II stream format. However, when reading a GDS-II file text and node objects will be ignored. Furthermore, GDS-II only supports two primitive object shapes: wires and polygons. Therefore, when writing GDS-II files, CleWin automatically converts all other objects into wires and polygons. Please note that GDS-II does not allow layer names. Instead, layer indices are used. For a more detailed description of the format, please refer to one of the references below.

References
1. Steven M. Rubin, *Computer Aids for VLSI Design*, 1994, Appendix C, http://www.rulabinsky.com/cavd/text/chapc.html.

2. Calma Corporation, *GDS II Stream Format*, July 1984.
3. Klaas Holwerda, *GDSII format, description and examples*, http://boolean.klaasholwerda.nl/interface/bnf/gdsformat.html.

6.3 DXF files

Especially for MEMS devices CleWin is often used in combination with AutoCad. Therefore, CleWin can read and write the AutoCad DXF format. However, CleWin does not support all objects that may be present in a DXF file. This is because a lot of these objects do not have any meaning in mask design, like 3-dimensional objects or zero-width lines. The command Edit|Connect wires may be used to manually connect zero-width lines and convert them into polygons.

6.4 EMK and MANN files

CleWin has a fracturing engine to convert all structures - including complex polygons - into a set of (overlapping) rectangles, which is necessary for some pattern generators. The standard version of CleWin can read and write the EMK and Mann file formats. Other, similar file formats are available on request.

6.5 Gerber RS-274X files

The Gerber RS-274X file format was implemented (together with high-resolution postscript) to allow for low cost mask manufacturing using standard photo plotters. Furthermore, the Gerber format can be used to design simple printed circuit boards in CleWin.

6.6 NC drill files

NC (Numerical Controlled) Drill files are normally used in combination with Gerber files for printed circuit board (PCB) design. A drill file simply contains the coordinates and diameters of the holes needed in a PCB. When writing a drill file, all objects other than circles and rings are ignored.

6.7 Postscript output

CleWin can create high-resolution postscript (*.ps) and encapsulated postscript (*.eps) output files for low-cost mask fabrication. However, CleWin cannot read postscript files.

6.8 Importing bitmap files

Bitmap files can be imported in CleWin with the help of a separate utility that converts BMP, PCX and GIF bitmap files into CIF files. This feature is used mainly to include photographs and logo's in a layout.

Index

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- L -

- M -

- N -

- O -

- P -

- R -

- S -

- T -

- U -

- V -

- W -

- X -

- Z -