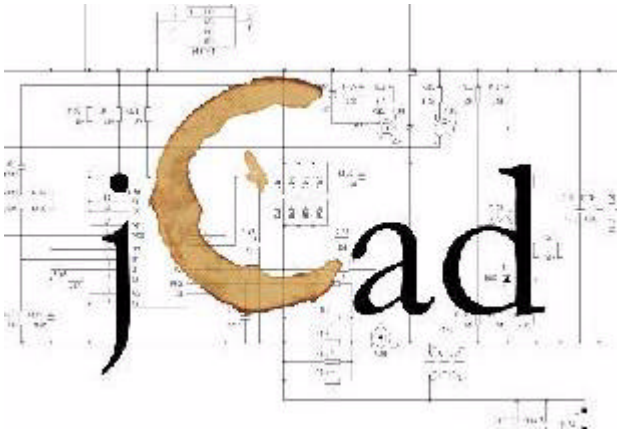


jCad Users Guide

updated for jCad 0.27

2002-01-06

Table of Contents

CHAPTER 1

Legal Stuff

1.1 jCad Background

jCad is modeled after the commercial “Vanguard” electrical CAD system. Vanguard was originally developed by CASE Technology Inc, then taken over by TERADYNE Inc. The product was withdrawn form the European and US markets by Teradyne at the beginning of the 1990’s, but was taken over by Sophia Systems, Japan, who still provide a Japanese only version of the system.

What I have done is a from scratch design of a very similar system - called jCad. jCad has a very similar user interface, and uses exactly the same file formats for all work files as Vanguard version 4.1 I have not used (or even seen) a single line of code from the original Vanguard product - evidence of this is that jCad is entirely written in Java, which did not exist at the time of this Vanguard version.

Some files in the distribution have some connection to Vanguard:
The main configuration file “config.g” is a copy from my licensed Vanguard copy, but it has been modified in several aspects.
The font files are copies from my licensed Vanguard copy, but they have been modified (scaled for more narrow appearance).
The component libraries shipped with the distribution were originally drawn using Vaguard - but are not the libraries shipped together with Vanguard.
All other parts of the jCad distribution are completely written/designed from scratch for jCad.

To my understanding this should be OK - no one can claim stolen property of any kind. If anyone does not agree in my view on this, please tell me so I can correct. (address, see below).

1.2 Disclaimer

jCad is a piece of software under construction. I provide no support, and take no responsibility for how it might affect your computer or files. Use at own risk.

1.3 License

I have not figured this out yet, but the intention is to keep everything open source software (GNU GPL?). Until further notice, use it for any purpose.

Mats Barkell

email: Mats.Barkell@era.ericsson.se

CHAPTER 2

Introduction

2.1 Overview and Status

jCad might eventually become a complete electrical CAD system with schematics editor, simulation and PCB layout editor. The following table gives an overview of what pieces of such a system are working within jCad.

TABLE 1. jCad Development Status

Function	Works	%ready	Comment
Schemantics Editor	Yes	90%	
Schematics Library Editor	Yes	90%	
BOM Creation	Yes	90%	
Schematics Printing	Yes	80%	
Spice Interface	Yes	80%	ngspice
Schematics => Layout Transfer	No	30%	"Vanguard" netlisting works, but not packager or "Layout Update"
Layout Editor	Yes	30%	Only "simple mode" (no connection checks), buggy.
Gerber, Drill files Generation	No	0%	
Layout => Schematic Back Annotation	No	0%	

2.2 Requirements

You need a JVM (Java Virtual Machine) to be able to run jCad. You can get it from many places, for example http://www.blackdown.org/ for Linux or http://www.javasoft.com/ for Windows and Solaris. The JVM is included in both JRE (Java Runtime Envorinment) or JDK (Java Development Kit) packages. If you do not plan to develop java programs youreself, JRE is recommended. Version 1.3.0 or higher is required.

You also need a three-button mouse. The user interface depends rather much on this - I did not even figure out whether it is possible/practical to run it with a two-button mouse.

2.3 Installation

The software is distributed as a gzipped tar file in two versions, for Linux/Unix and for Windows.

Note: If you have a previous installation of jCad, and have made any changes to the startup script (jcadx or jcadw.bat) or to the configuration file config.g, first copy these files to a safe place. This so you can do the same modifications to the new startup script and/or configuration files. The README file should contain information how these file have changed.

Expand the gzipped tar file into /usr/local/jcad for Linux/Unix or at C:\jcad for Windows systems. You must also adjust your PATH to include “/usr/local/jcad/bin” (Linux/Unix) or “C:\jcad\bin” (Windows).

2.4 Configuration

This User Guide assumes that you have installed jCad according to the Installation instructions above, with the distributed startup script and config.g file. It also assumes you organize your working files in a special way.

jCad can however be reconfigured in many ways, but that is described in a planned separate document - jCad Reference Guide.

2.5 Starting/Stopping jCad

Normally, you would first create a “working directory” for your project - how to do so is described in the next chapter “Schematics Editor”. In order to make a quick check that everything works, you could however download the “examples.tar.gz” file and unpack it somewhere in your file system.

Open a terminal window (xterm on Linux/Unix, or DOS window on Windows) and navigate to your wherever you put the example files. For Linux/Unix, typeType:

jcadx<return>

For Windows, type:

jcadw<return>

Once the program has started, you should see the a splash window on top of a black schematic editor window having a grid. If you click anywhere on the schematic editor window, the splash window will disappear. If you click your right mouse button or press the keyboard “H” button, a pop-up menu will appear (possibly a bit slow). If you choose FILE>>READ>>Drawing on the pop-up menu, you will be prompted for a drawing file name to open. Type:

natv<return>

and the file natv.drw will be displayed. While you have jCad running, please read the next subchapter in order to acquaint yourself a bit with the jCad user interface.

It is recommended starting jCad from a terminal window, though of course creating an “icon” or “button” for it should be very simple. This is because jCad is still a work in progress, and all diagnostics are not shown in its text fields. In case something goes wrong, there is usually more information what happened written to standard output (i.e. the terminal window).

Another good advice is to save your drawing often. There is no “undo” function, and there are certainly also bugs that may cause unexpected results.

Leaving jCad is done by selecting TOOLBOX>>Quit (Break) from the pop-up menu. You will always get a last “Are you sure (y/n): “ prompt before leaving. And if you have unsaved changes, you will also get a comment about that.

2.6 jCad User Interface

The user interface of jCad might seem a little bit special if you are used to modern GUI software, but the reason I kept it unchanged from Vangurad version 4.10 is:

- coding jCad was much quicker, it spared me lots of decisions
- the user interface is very efficient, actually more efficient than most modern GUI software

As you can see, there is no menu bar or tools bar. Instead, by pressing the keyboard “H” or right mouse button, you always get a pop-up menu at the place of the cursor. This means you do not have to move the cursor all over your screen, just a click instead.

Also when you have to type some text response, you do not need to move the cursor to the text field. jCad instead shifts to “line editor mode” completely and takes input from anywhere on its screen area - one more way to reduce long cursor moves.

Almost all the commands of the pop-up menus have keyboard shortcuts, and what buttons to use are written in the pop-up menu. You will likely learn some of these shortcuts very soon. A “C-” or “A-” in front of the shortcut key means that the Ctrl or Alt keys should be pressed together with the shortcut key.

The mouse cursor is moved automatically sometimes. This is when a new window is opened, it is then moved to the last known position (or top left) of that window. Most windows are also automatically scrolling, and the mouse is then “kicked back” to maintain its position relative to the window text. This effective, but also annoying when the mouse gets outside the mouse mat.

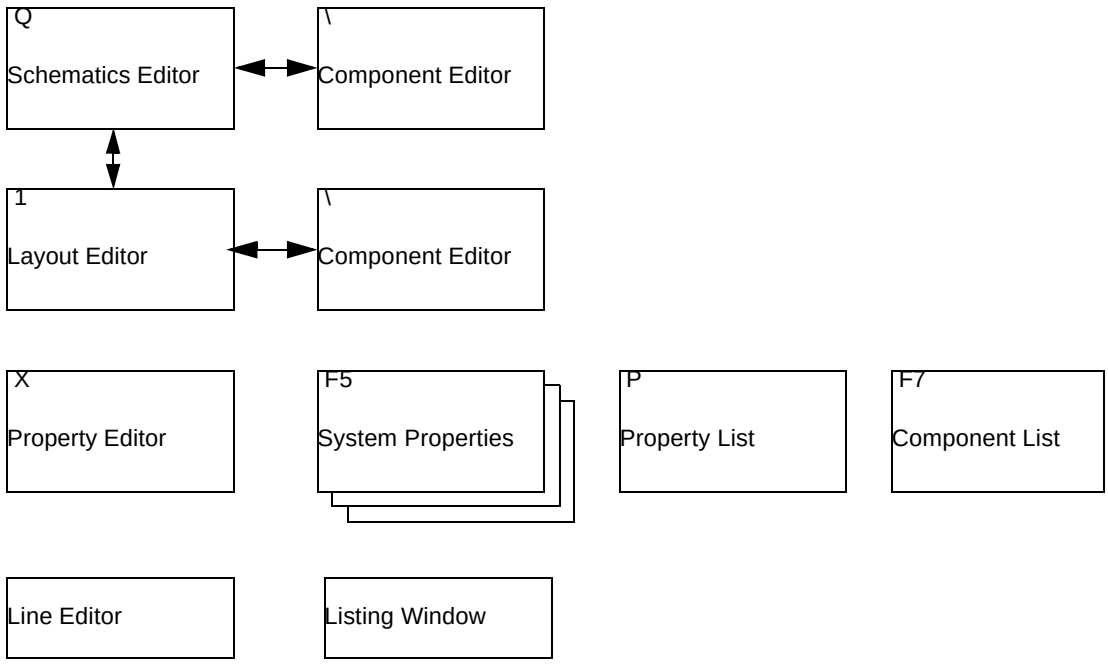
You also may have noted there are several cursors:

- a “cross” cursor, shoving the closest snap-grid location
- a “triangle” cursor, shoving the closest signal point or unconnected component pin
- a “square” cursor, showing the closest component
- a “dot” cursor, showing the closest component property value or signal point name

These cursors show which elements will be affected by any commands issued from the pop-up menus or shortcut keystrokes. Any command concerning a component (such as V=attach, C-V=delete, X=show properties,...) will always affect the component having the square cursor, any command concerning signal points/segments (such as R=attach, C-R=delete, X=show properties,....) will always affect the signal point/segment having the triangle cursor, and so on.

The following figure shows an overview of jCad’s different windows or “operating modes”.

FIGURE 1. jCad “Windows” or “Modes”



For each of the Windows/Modes, the shortcut key to enter that Window/Mode is shown at the top left of its box in the figure. You can use these shortcuts or selecting the corresponding pop-up menu items to navigate between these modes. Once you have left the startup Scematics Editor mode, you can also use the backspace key to get back to previous Windows/Modes. The title bar of the main window always shows what Window/Mode is active. A short overview description of the Windows/Modes:

The four top Windows/Modes all use the main jCad window - so “modes” are a relevant term:

- Shematics Editor. This mode is always entered when jCad is started. Use the FILE>>READ>>Drawing (,) command to open a the drawing to be edited. How to use it is further described in chapter XX.
- Component Editor. This mode is used to create or edit new library components. It works slightly different when invoked from the Schematics- or Layout Editors (among other things, it uses different search paths for library files). How to use it is described in chapter XX (as schematic component editor) and chapter XX (as layout component editor).
- Layout Editor. How to use it is further described in chapter XX.

The other windows are:

- Property Window. This is a separate pop-up window, used for editing of: signal names, component properties and component pin properties. When it is invoked from a Component Editor, it edits global component or component pin properties, and when it is invoked from a Schematic or Layout Editor, it edits local signals, components or component pin properties. “global” and “local” is described further below.
- System Properties. This is a separate pop-up window, having three modes: “flags and values”, “layer information” and “color palette”. It controls the behaviour of jCad in different ways. Most of the changes made here are however not saved to disk, they are valid for the current session only.
- Property List. This is a separate window, in which you can view and edit selected property values of a filtered (based on their LOC property value) list of components in the drawing.
- Component List. This window is implemented as a pop-up menu. It contains the components available for this drawing. It is normally used to pick new components for use in the drawing. In order to fill the Component List window with more components for use in the drawing, use the FILE>>READ>>System Lib (.) command.

- Line Editor. This mode uses the main window bottom text field. You can not select line editor mode manually, it is only selected automatically when user input is required. You can not select text with the mouse - or do not have to select the text field in order to enter text - your keybord typing goes there anyhow. Entering nothing, just a <return>, aborts most commands. Entering F3 in most cases enters the old value of whatever parameter is being edited.
- Listing Window. This window is a separate scrollable text window. It used for commands that generate much user feedback, like netlist or parts lists generation or execution of other programs, like spice simulations. Any of these commands normally end by the text “hit any key to continue” being displayed in this window, doing so closes the window.

2.7 Overview of File Types

The following files edited/created by jCad are the following types:

- Drawing Files (.drw). ASCII format. These files are edited by the Schematics and Layout Editors. They basically contains (somewhat simplified) a list of components and a list of signalpoints. The list of components refers to library files used for graphics information and global properties, the drawing file only contains: information on position and rotation within the drawing, local properties and connection information. Together with the referred Library Files, all other information can be generated from the Drawing File. When opening a drawing file, it is searched for in the directories listed under “Drawing search path” in the system property window.
- Library Files (.lib). ASCII format. These files contain one or more components each. For each component, these files contain the graphic information, information about location of its “pins” (connection points) if needed, and global properties (global component properties as well as global pin properties). When opening a library file, it is being searched for in the directories listed under “Library search path” in the system property window.
- WDP Files (.wdp). ASCII format. These files are a remnant from earlier versions of the CAD system, and are a half-way product in creation of netlists or parts lists. They contain the connection information and all property information, but not the graphics information. This file is stored in the same directory as the .drw file it was generated from.
- Netlist Files (.net). ASCII format. These files contain all the component, connectivity and property information needed for making a PCB layout matching the schematics. They may be derived from a single schematic drawing or from a set of drawings. The are the result of a successful EXECUTE>>COMPILE>>Simple Netlist command. This file is stored in the same directory as the .drw file it was generated from.
- Parts List Files (.prt, .csv). ASCII format. These files contain the bill-of-material listing of the schematics or a combined set of schematics. They are generated by a successful EXECUTE>>COMPILE>>Simple Partslist command. If the “Normal” format was chosen, a .prt file intended for printing is generated. If the “CSV” format was chosen, a .csv intended for use by a spreadsheet program is generated. Both formats contain a listsing of the LOC, MNAME and VALUE properties plus a quantity column. This file is stored in the same directory as the .drw file it was generated from.
- Circuit Files (.cir). ASCII format. These files contain the circuit information for the Spice (ngspice, Winspice) simulator. They are created by a successful EXECUTE>>ANALYSIS>>Write .cir or EXCUTE>>ANALYSIS>>Run Spice commands. This file type is stored in the same directroy as the .drw file it was generated from.
- Spice Library files (.lib) ASCII format. These files contain subcricuit definitions and or model statements needed by the Spice simulator. These files are being searched for in the spicelib subdirectory of the current directory. They are aupdate/saved bu a successful EXECUTE>>ANALYSES>> Write .lib command.
- Spice Rawdata files (.dat) ASCII, binary or mixed format. These files contain results from successful simulations. They are saved in the same directory as the .drw file that was simulated. jCad does not contain any tool for these files, but there is a separate program “jProbe” for viewing and printing of them at the same site as jCad was downloaded from.

CHAPTER 3

Schematics Editing

(It is assumed you have read the introduction chapter, and thus know how to start jCad and how to select commands form the menus, befor reaching here.)

3.1 Starting a New Project

Start a new project by creating a “work directory” for it - jCadTest might be a good choice. In this directory, create the following subdirectories:

“schdrw“ for your schematics files
“schlib” for components that are unique to this project
“laydrw” for layout drawing files
“laylib” for layout components that are unique to this project
“spicelib” if you are going to do simulations and need to define models or subcircuits unique to this project

Even if creating the above subdirectories is not necessary or you do not intend to do all these things, it is a good habit to always create these subdirectories. The risk is that otherwise you will end up in a single directory cluttered with files, and possibly name conflicts (because you do not also want to use long file names).

After creating this “work directory” (jCadTest), change directory to it and start jCad from there (it is assumed you know how to do this from the previous chapter).

The next thing to decide is the drawing sheet size of your drawing. Choices are A1L, A2L, A3L, A4S (L for lying=landscape, S for standing=portrait). Assuming A3 is a good format, we select FILE>>READ>>Drawing (,) from the pop-up menu. At the prompt “Drawing file to read: “, we type:

a3l<return>

As you can see at the jCad window title bar, we have now opened a library drawing for editing. We therefore save it in our local schdrw directory before starting to edit it. We do so by selecting FILE>>WRITE>>Drawing (C-W) from the pop-up menu. At the prompt “Drawing file to write: “, we type:

schdrw/test1<return>

From now on, we are editing the file schdrw/test1.drw. We can save changes by selecting FILE>>Save from the pop-up menu. And if we are starting jCad again later from the same directory, we can open it by doing FILE>>READ>>Darwing (,) and just typing “test1” at the “Drawing file to read: “ prompt, since schdrw is in the drawing search path.

Now is a good opportunity to edit the title block fields of the test1 drawing. And since these are Properties....

3.2 Editing Properties

Properties are just text name/value pairs, but with the extra functionality that the value text can also be positioned, fonted (size, visibility), justified (left, center, right), and assigned to a drawing layer. They are typically used to hold the location value (the LOC property, ex. “R10”, “C100”,..), component values (the VALUE property, ex. “100k”, “10p”,...). Some of these properties have a special meaning to jCad, other are used by convention, and still others can be used freely just to display text or ather purpose. See “Scehmatics Special Properties” on page 23 for a listing of special properites.

Using the newly created test1 drawing, move the cursor to the top left corner of the drawing, until the square cursor is postioned exactly at the upper left corner of the green drawing frame. Select COMPONENT>>Describe (X) from the pop-up menu, and the property window appears. Check that the top line of the property windows says “Component name: TITLE BLOCK 1” - if it shows something else, close the property window by selecting Previous/Rtn (BackSpace) from the pop-up menu, and reposition the cursor as described above.

As you move the mouse (or use the keyboard arrow keys to move the cursor - not working) over the property window, you note that some fields change colour as you pass them. These fields are all editable.

You also see that the properties are divided into “global” and “local”. These two concepts are somwhat like “Class” and “Instance” of object oriented programming languages. Global properties are defined in the library file of a component (in their .lib file) - common for every application/instance of it, while local properties are defined for every application/instance of it in every drawing (.drw file). A global property can be overriden by a new value, position, font, justification or layer locally. The “effective” (i.e. visible in the drawing, netslists and so on) parameters of a property is the “last defined” (i.e. local if such exist, otherwise global, or none if neither exist). New local properties, that have no global definition can also be created.

The local properties that already exist for the TITLE BLOCK 1 component are all created automatically (if you followed the instructions so far) - do not change any of these.

Overriding a global property is done by clicking the left mouse button (or by pressing keyboard Y - not working) on its value. Do so for the TITLE property. At the “Property value: “ prompt, enter for example:

Test1<return>

Note that there is now a local property TITLE having value Test1. Having created this local property, we can also change layer, font and justification. This is done by left mouse button clicking (or by pressing keyboard Y - not working) at the appropriate field. If we just wanted to change any of the font, layer or justification parameters, we could have typed “TITLE<return>” or just pressed F3<return> (preserve old value) at the “Property value: “ prompt. To remove a local property, change its value to an empty string (i.e just a single <return>).

Now that you know how to change property values, you may change any of the NR (drawing number), PREP(department), RESP(responsible designer), REV(revision designation), SWTITLE (title in swedish) to the values you prefer. If you intend to do a multi sheet design, where several drawing are merged to one parts- or netlist, you must assign SHEET to unique values for each sheet, OF to the total number of sheets and PROJECT to the same value on all sheets.

If you want to create a new property, which does not already have a global definition, you should press the left mouse button (or keyboard Y - not working) on the active field below the bottom local property. Doing so will first produce a prompt “Property name: “ - enter for example “REF<return>”, and then a new prompt “Property value: “ - enter for example “See doc XXXX<return>”. After that (unless you gave an empty response to any of the prompts) , you will be returned to the schematics editor window with the new created REF property attached to the cursor. Move the cursor, with the property attached, to the “Ref.” field of the title block, and release it there by selecting TEXT>>COMP PROP>>Drop Att (Space-Bar). As you may note, the newly dropped text “See doc...” does not look like the other texts - this is because you were never asked about font, layer or justification - the new property was created using “current values” for these. We could have set these to correct values before creating the property (by TEXT>>COMP PROP>>CHANGE>>LAYER>>Set Curr (F4), TEXT>>COMP PROP>>CHANGE>>Font (C-T), TEXT>>OPTIONS>>Set Justify (A-T)), but it is easier to do that in the proerty window whenjust a single property is created. To do so, again move the cursor to the TITLE BLOCK 1 component and select COMPONENT>>Describe (X). In the property window, you then can change font, layer and justification for this new REF property by clicking the left mouse button on each of those fields. Once you are done, leave the property window by selecting Previous/Rtn (BackSpace) on the pop-up menu.

What fonts are available in jCad is described in chapter “Fonts” on page 23.

Conventions for use of layers are described in chpter “Scematics Layer Conventions” on page 23.

Positioning of properties is done in the schematics editor window. Once the cursor gets close enough to a property, the small dot cursor is visible at its position point. You can then select Select/Drop (A) from the pop-up menu to attach the property, move it by moving the cursor, and drop it by selecting Select/Drop (A) from the pop-up menu again.

Once you have done the changes to the title block, it’s time to start to enter youe components to the schematic.... But it is a good habit to write your drawing to disk often, do so by selecting FILE>>Save on the pop-up menu.

3.3 Entering Components

Entering components into your drawing is a two step process. You first read the libraries you want to use - doing so makes the components appear in the Component List window. You can then pick the components from there into you drawing.

In order to read the libraries, we select FILE>>READ>>System Lib (.) from the pop-up menu. Assume we want to use resistors, so we type “res<return>” at the “Library file to read: “ prompt. You can repeat this procedure to read capacitors - by responding “cap<return>” at the prompt - and inductors and inductors - by responding “ind<return>” at the prompt.

But how did we know that resistors are in res.lib, capacitors in cap.lib, and so on? Well, that is a slight shortcoming, you simply have to know. Perhaps there will be a library browser window in jCad in the future, but until then you have to use your file manager or whatever to get familiar with what ius in the libraries shipped (in the lib subdirectory of where you installed jCad).

Once libraries are read, you can pick components by selecting COMPONENTS>>List Comps (F7) from the pop-up menu. The Component List window (actually another pop-up menu) then appears. From that menu, select for example res.lib>>RESD. The Component List window then disappears and you are back in the schematics windows with a RESD component attached to the cursor. You can drop the component wherever you want in the schematic drawing by selecting COMPONENT>>Drop Att (SpaceBar) from the pop-up menu. You can attach it again by moving the cursor so the square cursor is on the component and select COMPONENT>>ATTACH>>With Signals (V). While it is attached, you can also make copies of it by selecting COMPONENT>>Create/Copy (B) or rotate it by selecting COMPONENT>>Rotate (G) from the pop-up menu.

After you have you have entered some components, you probably wish to draw some connections....

3.4 Editing Signals

Editing signals is done mainly by the LINE>>CREATE>>Segment (U) menu command, which is repeated for each segment. After the last segment is done, it is dropped by selecting LINE>>Drop Att (SpaceBar) from the pop-up menu. These commands also have mouse shortcuts, the center mosue button does the same as U, while the left mouse button does the same as the SpaceBar when a signal is attached. The pop-up menu commands do currently not work well here, so use keyboard or mouse shortcuts instead.

As you note, two signal segments and one bend are always attached while you draw the signal. By creating a new segment (LINE>>CREATE>>Segment (U)), the first segement and bend is freezed, and a new bend and segment is created closer to the cursor. If the bend points the wrong way or has the wrong angle (45 vs 90 degrees), you can select LINE>>Flip X/Y (G) until it goes the way you want (four alternaitves are cycled) while the signal is still attached. If you want to remove a segment, move the cursor until the triangle cursor is on the segment you want to remove and select LINE>>DELETE>>Segment (C-R) from the pop-up menu.

You can also attach and move a signal after it has been dropped. To do so, move the cursor until the triangle cursor is at the point you want to move and select LINE>>Attach Pnt (R) from the pop-up menu. When you have moved it, drop it again by selecting LINE>>Drop Att (SpaceBar).

Signals normally get default names in the style %n%, where n is a number automatically incremented by jCad. But you can also invent your own. In order to do so move the cursor so the triangle cursor is on the signal you want to name, and the cross cursor is where you want the name to appear. Then select TEXT>>SIGNAL>>Create/Edit (T). At the prompt “New Text: “, type the name you want the signal to have.

Signal names are very important from one aspect - signals having the same name are considered connected, regardless if they are connected by signal segment in the drawing or not. This means you can for example end a signal and name it “+5V”, and it will be connected to all other signals with the sam náme in the netlist. This is also the technique used for drawing buses - there is for example a library bus.lib containing the symbols for connecting signals to buses, but these symbols have no connectivity information - this works as long as the signals are named correctly.

Connection between signals and from signal to pin should occur automatically. The connection ring symbols for signal crossings and the comments issued in the text field below the drawing should indicate that connections were noted properly by jCad. There is however also a check for connectivity available - it is available in the System Property window (select TOOLBOX>>Options (F5)) and is controlled by the “Highlight No-Connects” Yes/No flag. Changing this flag to “Yes” and returning to the schematic editor window, you will note that all signal segments that do not have a name and are not connected are highlighted by a cross, and also that all unconnected component pins are highlighted by a small box.

3.5 Numbering LOC Properties

As you entered new components into your drawing, they were automatically numbered. This numbering was according to the “xppn” format (see %LOC=1 in “Scehmatics Special Properties” on page 23). This numbering was because all the shipped libraries were created with %LOC=1.

This might not have been what you wanted. If so, there is a simper way than redesigning all library components to %LOC=2. That simpler way is to eneter a new %GLOC to the TITLE BLOCK component, and assigning its value =2. Such a %GLOC property overrides all component %LOC values. Doing so changes the LOC number format for all new components entered. (Should have told you from the beginning?...well, read on).

Note that there is also a TITLE BLOCK property %LOCMIN to which you assign a start value for the LOC numbering, so you do not have to start from 1.

Since you have also probably entered you components a bit unstructured, the LOC numbering may not arranged in the nice left to right increasing order you want. The solution for this problem, and also for the above problem of changing %LOC numbering format, is to use the EXECUTE>>Renumber LOC command. It renumbers all LOC values from left to right, using the present %LOC (or %GLOC override) and %LOCMIN. Components having the same LOC before the renumbering will have the same LOC also after the renumbering - useful for connectors or other multiple section components.

Do not run the EXECUTE>>Renumber LOC after you have started making a layout of your design! Any LOC number changes at that state will cause big trouble!

Several components using the same LOC value is only allowed for components having either property CONN=1 and different pin names (used with connectors), or for components having different SECT property values and non-overlapping pin names (used with multiple section components such as OP-amps, buffers, transformer windings,...).

3.6 Tips

Change of many property values on many components is easiest performed in the Property List window. It is opened by shortcut keyboard P (no menu entry yet). This window displays a filtered list of all components in the drawing. The filtering is based on the LOC properties of the components and the match string at the top of the window (initially “*”). The match string is changed by clicking on it and honours the normal use of the “*”(any number of chars) and “!” (one char) wildchards. Initially, only the LOC properties of the components are shown, but new columns can be added and deleted. Values can be edited by clicking on them, or copied and pasted between components. Clicking the “HL” column marks the component in the Schematic Editor windows (if both windows are made visible at the same time). Property List window keeps its settings (match string, columns shown) between invocations within the same jCad session.

Do not use the “local library” directly for placement for picking components into your drawing! Instead always use the FILE>>READ>>System Lib (.) command. Using local library components in your drawing blocks (these components can not be removed from the local library) this library from its intended use - which is to edit library files.

3.7 Creating Parts Lists and Netlists

Creating a parts list (BOM) is performed by simply selecting EXECUTE>>COMPILE>>Simple Parts List from the pop-up menu. Two formats are supported: “Normal” and “CSV”. The Normal format is intended for printing and has columns for LOC, Quantity, MNAME and VALUE. The CSV (Comma Separated Variable) format is intended for use with spreadsheet programs and has the same columns.

Creating a netlist is performed in a similar way by selecting EXECUTE>>COMPILE>>Simple Netlist from the pop-up menu. The only format supported presently is the Vanguard flat netlist format (see however Spice interface in a later chapter).

For both these commands, a Listing window is shown during the execution. Check that the final lines say Warnings: 0, Errors: 0 or check higher in the text what might have went wrong befor accepting the result.

Both these commands accept mutiple sheet drawings. To make that work, you must create a small text project file containing either the .drw files or the .wdp files you wish merge into one parts or netlist. By convention, such a file is given the .prj extension. An example such file “test.prj” for our project “test” might contain:

```
schdrw/test1.drw+schdrw/test2.drw+test3.drw
```

When performing the EXECUTE>>COMPILE>>Simple Parts List or EXECUTE>>COMPILE>>Simple Netlist commands, you then enter “@test<return>” as a response to the “Design file(s) (@project, [current]): “ prompt.

Hierarchical Parts Lists or Netlists (i.e. components referring to other schematic drawings) are not currently supported by jCad.

CHAPTER 4

Schem. Comp. Editor

4.1 Overview

Component libraries can be both read (by using FILE>>READ>>Local Lib (C-.)) into and written from (by using FILE>>WRITE>>Library (A-.)) the “local library” only. Editing of library components and creation of new components can also only be done in the local library. Thus, the mechanism for editing of or creation of new library files is based on this libraray. It is always shown at the top of the Component List window (shown by the COMPONENT>>List Comps (F7) command) regardess if it is empty or not.

The local library is stored together with the schematics drawing, withing its .drw drawing file.

A schematic component library file normally contains more than one library component. The most normal use is to have several versions of the same component. These versions might be with different rotation - since rotating a component in the schematics normally means you have to adjust text positions, and you wish to avoid that work. The versions could also for different use - for example a 8-bit buffer might be used as 8 separate buffers with their different pin numbers ready or as a combined single 8-bit component depending on application. Other examples are the existing “trabuild” or “relbuild” libraries, by which you can build transformers or relays from pieces - by then assigning the same LOC number to all pieces, but different pin numbers and SECT values, they will appear as one component in the net- or partslists.

You start by entering the Component Editor window, this by selecting COMPONENT>>Editor (\) from the pop-up menu.

4.2 Composing a Library File

Composing a library file means to first check that the local library is empty, or that it does not contain anything that should not be in the new library file. If there are components that are not not intended for the new library file, they must be deleted one by one by using the COMPONENT>>Del from lib (C-B) command.

If you want to include a component that already exists in another library file, you must read that library file into the local library by the FILE>>READ>>Local Lib (C-.) command. Such reads are accumulative, the components of the library file you read are added to those already present in the local library. This means you afterwards have to remove each of the components you did not want to include in the new library file by the COMPONENT>>Del from lib (C-B) command.

Creating new components in the library file is done by the COMPONENT>>Create (A-\) command. It creates an empty component by the name you specify, adds it to the local library and selects it for editing in the Component Editor. How to edit the graphics, global properties and pins of this new component is described in chapter “The Component Editor” on page 20.

You can also select any of the other local library components for editing by use of the COMPONENT>>Edit (C-\) command.

Once your done composing your library and editing its components, you save it using the FILE>>WRITE>>Library (A-.) command. You typically save it to the schdrw subdirectory of your current project, and if it is a library of common interest for other projects, you separately copy the library file to where the system libraries are stored.

4.3 The Component Editor

You enter the component editor window by the COMPONENT>>Editor (\) command.

All components of the local library are accessible for editing. You select which component to edit by the COMPONENT>>Edit (C-\) command, or you create new components by the COMPONENT>>Create (A-\) command, whichs also selects the component for editing.

The component editor is similar to the schemtics editor in many ways. Some differences are:

- It has an alignment “diamond” symbol at its center. This becomes the point where you attach the component when you later use it in a schematics (or re-use it for building a new component). The convention is to make this the centerpoint of small (up to three pins) components, while it is at the upper left corner of bigger box-style components.
- You can only edit global properties. (Local properties only exist at their instances in a drawing).
- You can add/delete pins.
- You can edit the graphics of the component.

When you start from scratch editing a new component (newly created by the COMPONENT>>Create (A-\) command), it is wise to first check if there is a very similar component that you can re-use directly, by rotation or other small changes. If it is so, you can “pick” that component from the Component List window (available by COMPONENT>>List Comps (F7) command), align it (by mouse moves) or rotate it (by COMPONENT>>Rotate (G) commands) as wanted before dropping it into the new component by the (COMPONENT>>Drop Att (SpaceBar) command). Note that this is the only situation where you can move or rotate an entire component in the components editor, once you have dropped it you need to handle its graphics lines, texts, pins and properties individually.

You probably start by drawing/editing the component graphics. This is done very similar to drawing signal lines in the schematics editor. Differences here are that you edit only one line segment at a time, and you can draw the any angle. You can also draw arcs - this is done by first drawing a straight line, and then converting it to an arc. You can also draw circles. You can also always toggle the cursor grid by the DISPLAY>>CURSIR/GRID>>Reduced Grid (C-A), or you can move to any coordinate (relative, absolute or back to grid) by the ctrl-G commands. When you enter the pin stubs, make sure you are on grid so you can end signal lines in the schematics editor at the same location. Also honour the layer conventions when drawing the component graphics, see “Scematics Layer Conventions” on page 23.

You create pins by the PIN>>Create (A-Z) command. When doing so, make sure you are at the same position as the pin stub line end (by the LINE>>Move To (A-R) command), or the symbol will appear confusing when used in a shematics. There is no jCad internal interpretation of the pin stub lines, the pin is just the point where you do the PIN>>Create (A-Z) command. You can not move pins (just their text), if you fail in placing it, you have to delete it by the PIN>>Delete (C-Z) command and again create it at the correct position. Also honour the the layer convention for pin text layers, see “Scematics Layer Conventions” on page 23. (The pin itself does not have a layer, just the pin text).

You create/edit properties exactly as in the schematics editor, the only difference is that here you only see the global properties. Again, honour layer conventions, see “Scematics Layer Conventions” on page 23.

4.4 Tips

It is not so easy to clear the local library - you have to delete each component individually by name. Thus, it is a good idea to save your schematics drawing before you start using the local library. If yo do so, you only have to re-read your schematics drawing to empty the local library.

CHAPTER 5

Simulation

5.1 Spice Interface

In order to be able to do spice simulations, you need to have the appropriate simulator installed on your computer. The shipped configuration is set up for use with ngspice (Unix version) or winspice (Windows version). The simulator executable must be located within your PATH.

The spice interface of jCad has the following main commands:

- run spice. Runs the spice simulator from within jCad. This includes first saving the circuit definition in spice format to a .cir file.
- write .cir. Saves the current drawing as a spice format circuit definition .cir file, without running the simulator.
- write .sbc. Save the current drawing as a subcircuit definition .sbc file.
- write .lib. Saves the current drawing as a subcircuit definition into an existing or newly created .lib file - for storage together with other subcircuit definitions. If a definition of the same subcircuit already exists in the .lib file, it is replaced.

All spice commands show a “listing window” when executed. This window shows the progress of the conversion as well as the stdout and stderr output from the simulator.

For a deeper description of the spice interface of jCad, see the “jCad Reference” document.

5.1.1 Performing a Simple Simulation

In the currently shipped libraries, only a few of the components of the RES, CAP and IND libraries are set up with the appropriate properties for use with spice, so in order to make an example we need to restrict ourselves to a simple passive circuit.

Draw a circuit similar to the following:

5.1.2 Using Subcricuits

CHAPTER 6

Appendix

6.1 Fonts

The following table gives on overview of the fonts used within jCad. The meaning of size is maximum up*right*down*left for the biggest character of the font in drawing coordinates.

TABLE 2. jCad Fonts

Font Name	Description
A	size 60*120*60*30 fixed width hole symbols mapped to ASCII codes for: a, b, c, d, e, f, g, h ,i ,j (all lowercase)
I	size 0 invisible text
0	size 7*6*2*0 fixed width line style ASCII characters
1	size 80*80*40*0 fixed width line style ASCII characters
2	size 120*120*60*0 fixed width line style ASCII characters
3	size 53*53*26*0 fixed width line style ASCII characters
4	size 34*34*17*0 fixed width line style ASCII characters
5	size 63*63*32*0 fixed width line style ASCII characters

6.2 Scematics Layer Conventions

In the schamatics editor, the purpose of layers is mainly for simple determination of the purpose of graphics. Since layers can be assigned colours, their purpose of grahic objects are easily visible in the drawing. jCad uses 256 layers. The table below shows default colors and names for the layer conventiuons used in the shipped libraries.

TABLE 3. Schematic Layer Conventions

Layer	Colour	Name	Description
0	GRAY2		Component Placement
1	GRAY2	SIGLINES	Signal lines

6.3 Scehmatics Special Properties

All properties having a name started by a “%” sign have a special meaning either to the schematics editor or whenever any form of netlist or parts list is generated. They will be removed before the output is enerated.

All properties having a value identical to the property name are ignored, they just serve as placeholders. They are shown in the graphics editor, but have no meaning otherwise.

The “TITLE BLOCK*” component must always be proesent in a schematic drawing. It holds special properties that have a meaning for the entire drawing.

TABLE 4. TITLE BLOCKn special properties

Property Name	Description
%COLORS	Automatically updated
DATE	Automatically updated
FILE	Automatically updated
%GLOC	Overrides %LOC of all components in the drawing
%LAYERSHOW	Automatically updated.
%LAYERACTIVE	Automatically updated
%LNAMESn	
%LOCMIN	Sets the minimum value for assigning new LOC values.
OF	Drawing total sheets. Must be set to the total number of sheets in case several drawings are merged into one parts- or netlis
PROJECT	Project name. Can be freely chosen, but must be the same on all drawing that are intended to be merged into one parts- or netlist.
SHEET	Drawing sheet number. Must be set to a unique value for each drawing in case several drawings are merged into one parts- or netlist.
TIME	Automatically updated

Special Properties used in other components are:

TABLE 5. Standard Component Special Properties

Property Name	Description
CONN	CONN=1 tells this is a connector, several components using the same LOC value are allowed provided they instead have different pin names.
LOC	Is automatically assigned value according to %LOC property.
%LOC	Determines the method used for assigning LOC value: %LOC=1 => LOC=xppn (n not reused) %LOC=2 => LOC=xn (n reused) where x is the global LOC property, pp is a two digit page number and n is a running number.
%PIN	

The following properties have no special meaning to jCad, but are used by conventionse

TABLE 6. Standard Component Convention Properties

Property Name	Description
VALUE	Holds the “major parameter” (resistance for a resistor, capacitance of a capacitor,...).