EasyEDA Tutorial

2017.10.10

EasyEDA Editor: https://easyeda.com/editor

EasyEDA Editor Beta: https://beta.easyeda.com/editor

Instruction:

- This document will be updated according to the updated EasyEDA editor.
- The latest edition please refer to https://easyeda.com/Doc/Tutorial/ .
- The Editor beta version will release the new future and enhancement first, but maybe have some bugs, please using carefully.

Update Record:

Update Date	Editor Version	Description
2017.10.10	v4.9.3	Local auto router support Linux(64)
2017.10.09	v4.9.3	Add OpenFileFormat section
2017.10.08	v4.9.3	Change SpiceSimulation to Simulation, Add Simulation eBook section at Simulation section
2017.09.20	v4.9.3 Beta	Update export Altium Designer format description
2017.09.18	v4.9.3 Beta	Update local auto router description
2017.09.08	v4.8.5	First release, Add "Essential Check" section, add "Essential Check Before Placing a PCB Order" of PCBOrder section
NA		

Introduction to EasyEDA

What's EasyEDA

Welcome to EasyEDA, a great web based EDA tool for electronics engineers, educators, students, makers and enthusiasts.

There's no need to install any software. Just open EasyEDA in any HTML5 capable, standards compliant web browser.

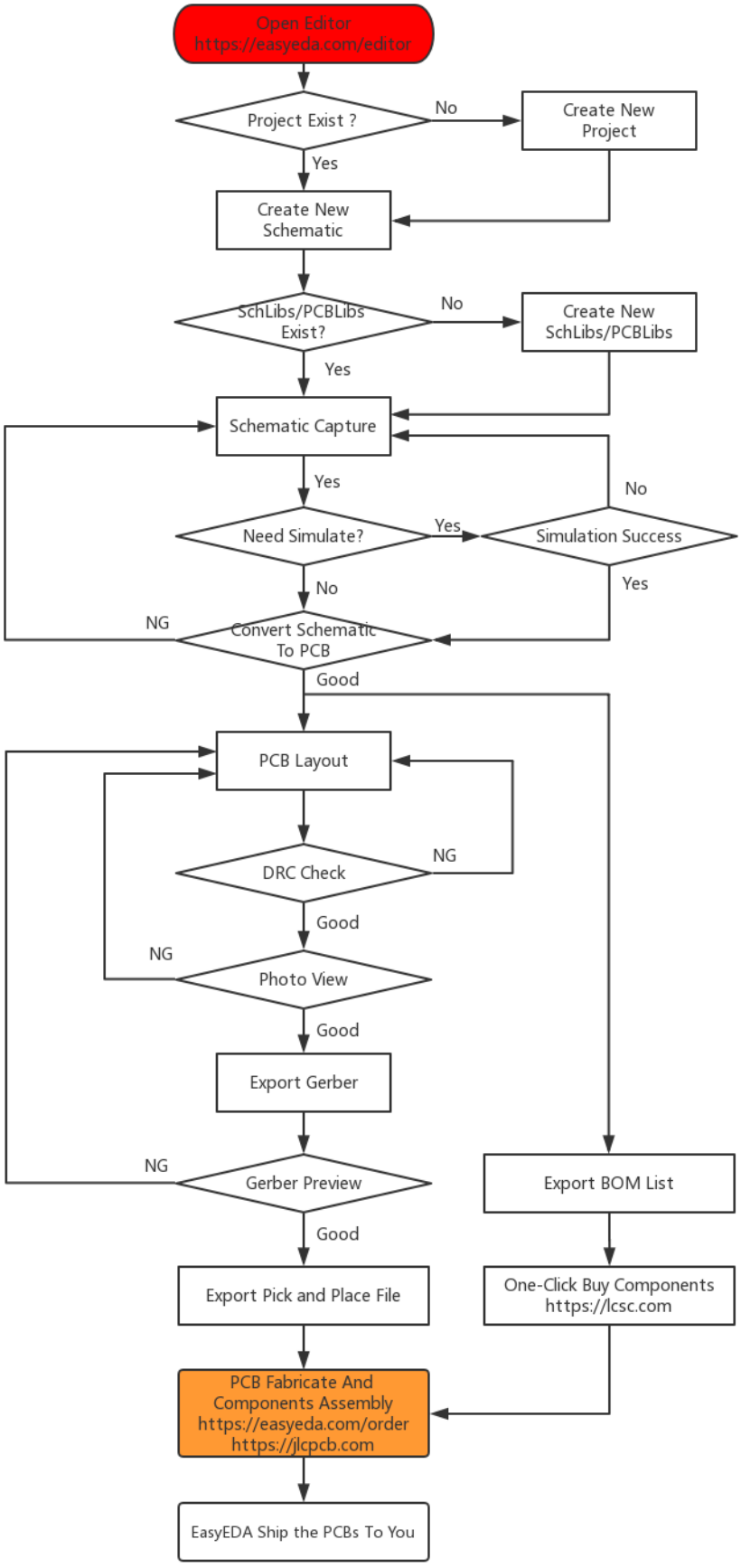
Whether you are using Linux, Mac or Windows; Chrome, Firefox, IE, Opera, or Safari. Highly recommend to use Chrome and Firefox. EasyEDA has all the features you expect and need to rapidly and easily take your design from conception through to production.

EasyEDA provides:

- Simple, Easier, Friendly, and Powerful general drawing capabilities
- Working Anywhere, Anytime, Any Device
- Real-time Team Cooperation
- Sharing Online
- Thousands of open source projects
- Integrated PCB fabrication and Components purchase chain
- API provide
- Script support
- Schematic Capture
  - NgSpice-based Simulation
  - Spice models and subcircuits create
  - WaveForm viewer and data export(CSV)
  - Netlist export(Spice, Protel/Altium Designer, Pads, FreePCB)
  - Documentation export(PDF, PNG, SVG)
  - EasyEDA source file export(json)
  - Altium Designer format export
  - BOM export
  - Mutil-sheet and hierarchical schematics
  - Schematic module
  - Theme setting
  - Document recovery
- PCB Layout
  - Design Rules Checking
  - Mutil-Layer
  - Documentation export(PDF, PNG, SVG)
  - EasyEDA source file export(json)
  - Altium Designer format export
  - BOM export
  - Photo view
  - Gerber output
  - Pick and Place File output
  - Auto Router
  - PCB module
  - Document recovery
- Import
  - Altium/ProtelDXP ASCII Schematic/PCB
  - Eagle Schematic/PCB/Libraries
  - LTspice Schematic/Schematic Libraries
  - DXF
- Libraries
  - More than 500,000 Libraries(Symbol and Footprint)
  - Libraries management
  - Symbol/Subpart create and edit
  - Spice symbol/model create and edit
  - Libraries management
  - Footprint create and edit

Design Flow By Using EasyEDA

You can create circuits design easily by using EasyEDA. The design flow as below:

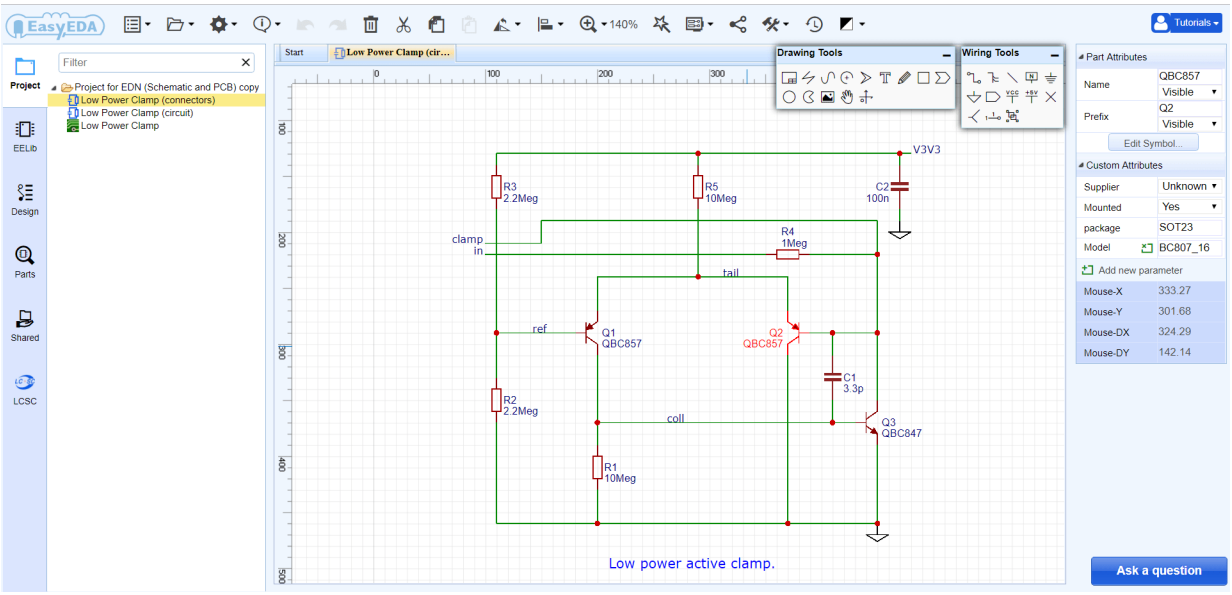


UI introduction

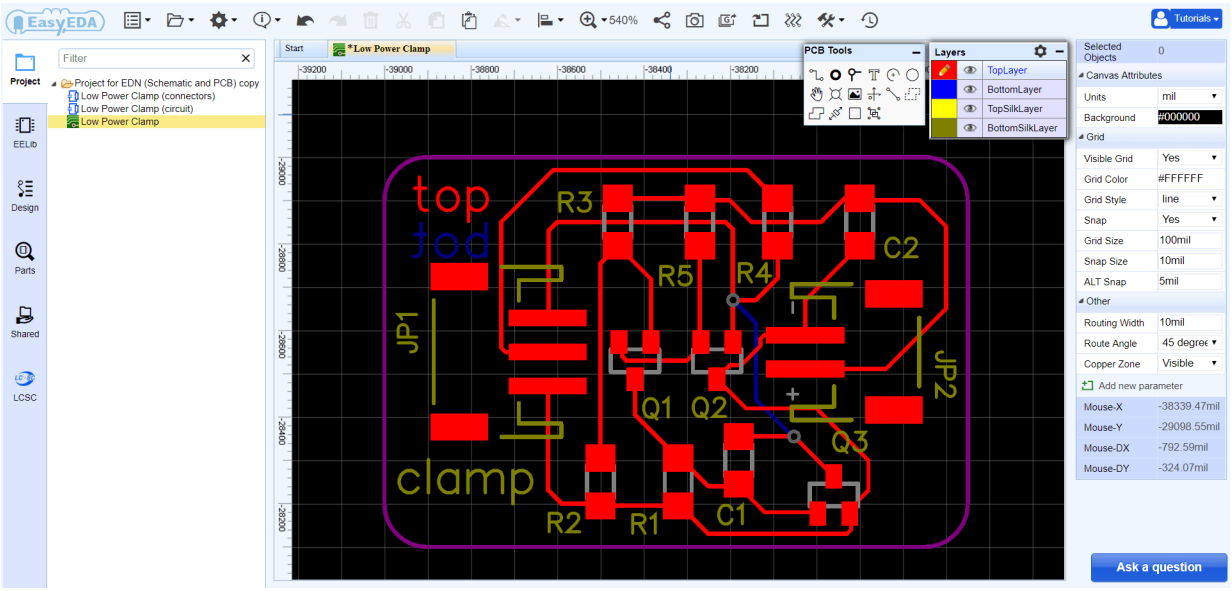
EasyEDA Editor has a clearly and friendly user interface. You can use its every function very easily when you become familiar with EasyEDA.

Global UI

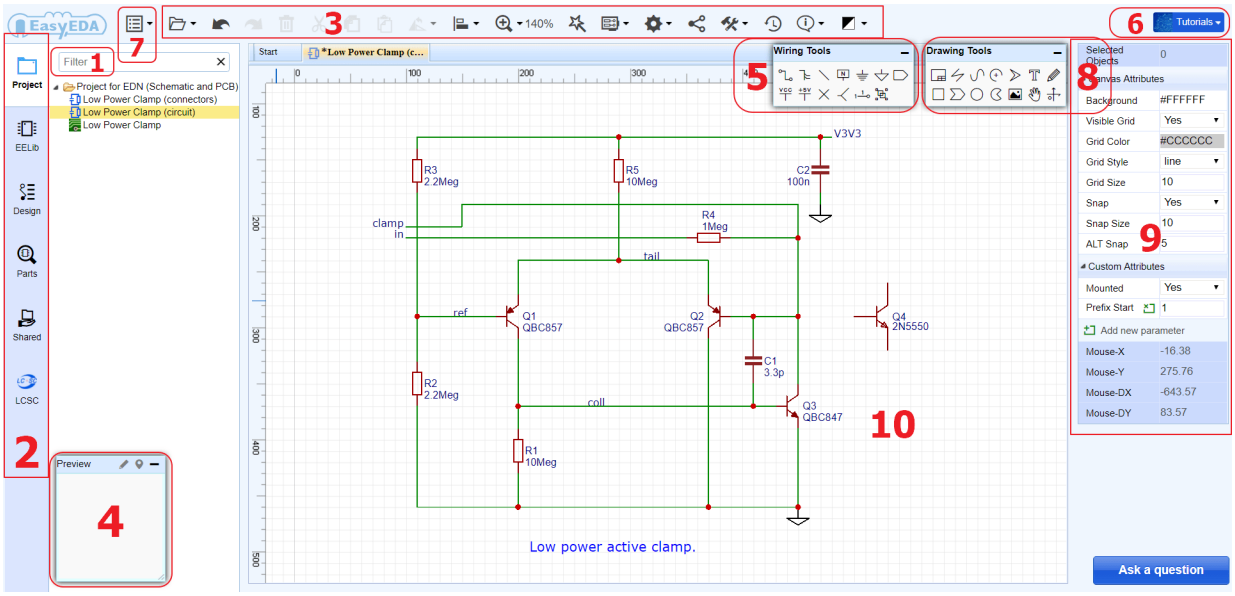
Schematic UI



PCB UI

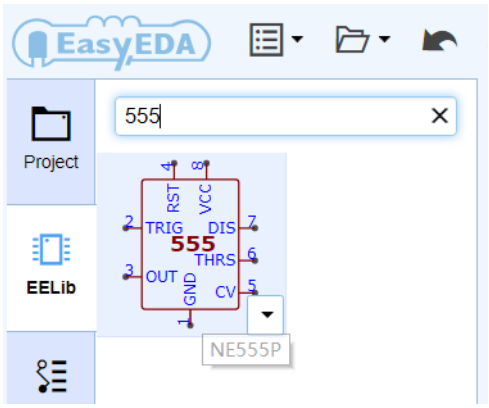


The Clear EasyEDA UI



1. Filter

Before using the Filter, you need to select what module you need in the left navigation panel, and then you can quickly and easily find projects, files, parts and footprints, just by typing a few letters of the title. For example, if you want to find all files containing "NE555" in the title, just type "555", it is non-case-sensitive.

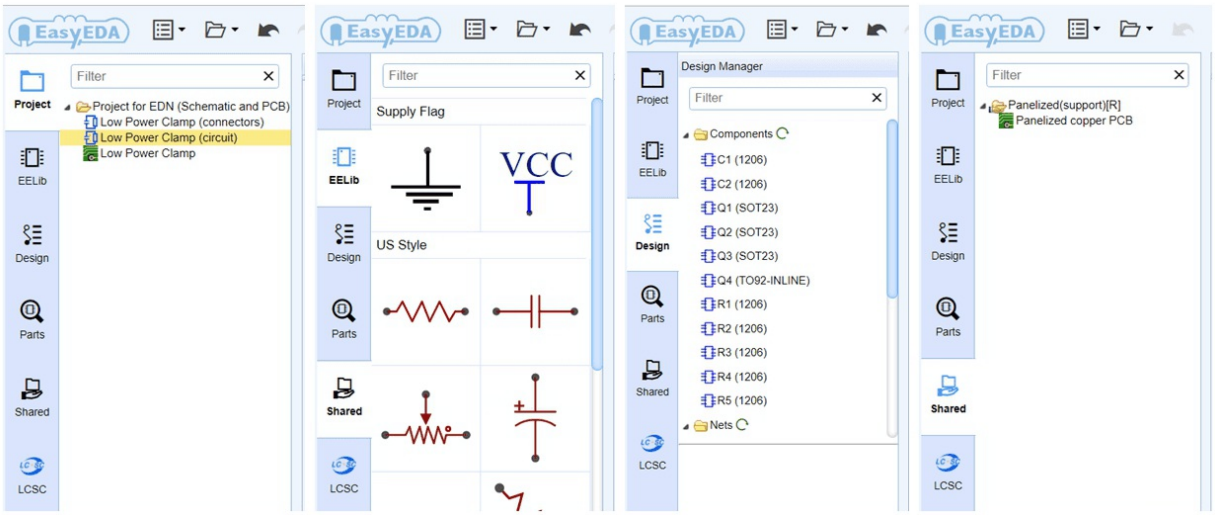


The Filter only searches project, file and part titles and names. It does not search the Description and Content fields.

Click the X to clear the filter.

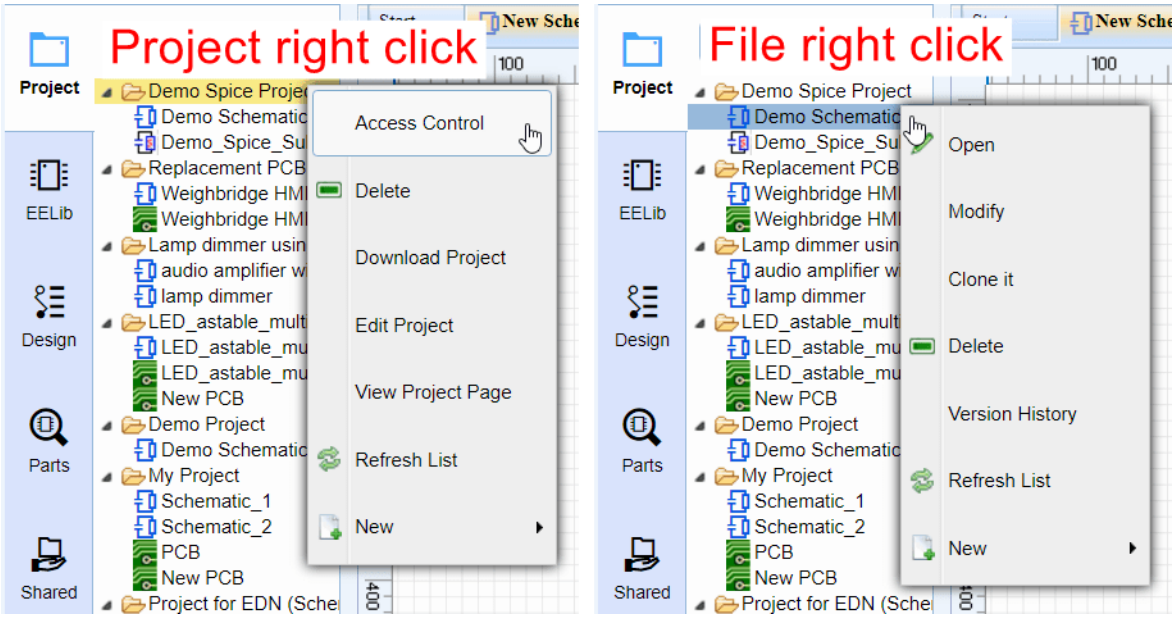
2. Navigation Panel

The Navigation panel is very important for EasyEDA: it is from here that you can find all your projects, files, parts and footprints.



Project Here, You can find all of your projects that are private or shared with the public, or fork from someone else's.

Except for System IC, these options have a content menu. For example, if you drop down to My Projects and right click an item, you will get a tree menu like :



EELib

EElib means EasyEDA Libraries, It provides lots of components complete with simulation models, many of which have been developed for EasyEDA to make your simulation experience easier.

Design

Design Manager, you can check each component and net easily, and it will provide DRC(Design rule check) to help your design better.

Parts

Contains schematic symbols and PCB footprints for many readily available components and projects. And your own libs and modules will show up here.

Shared

About this module, if your partner shared his/her private project with you by using the **Access Control** option, then the project will show here.

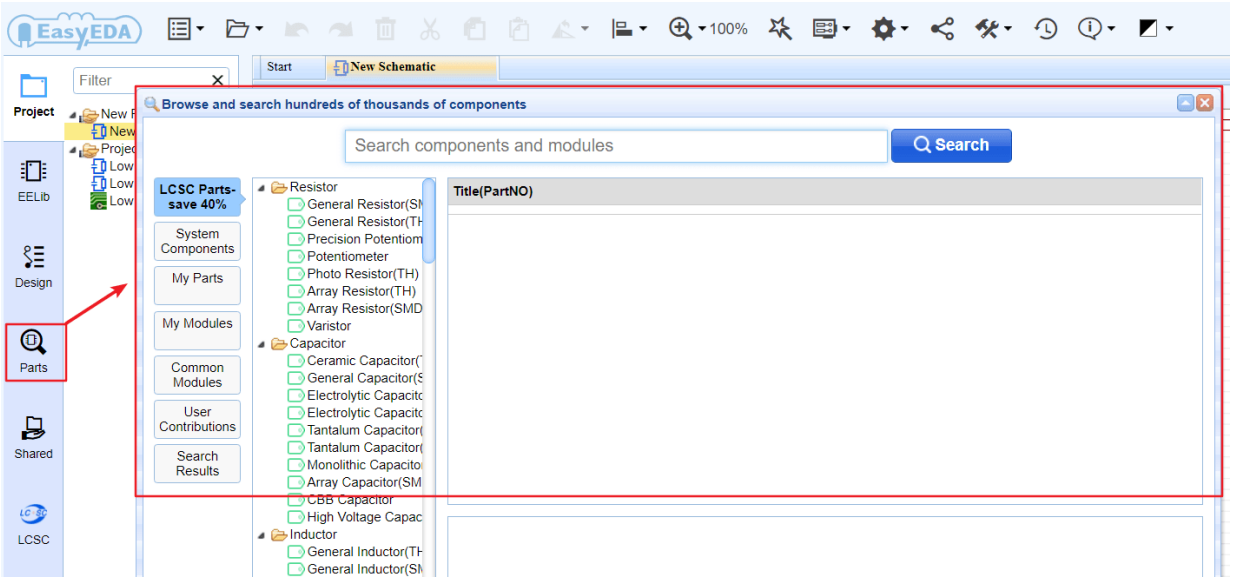
For more information you can refer to the Access Control section.

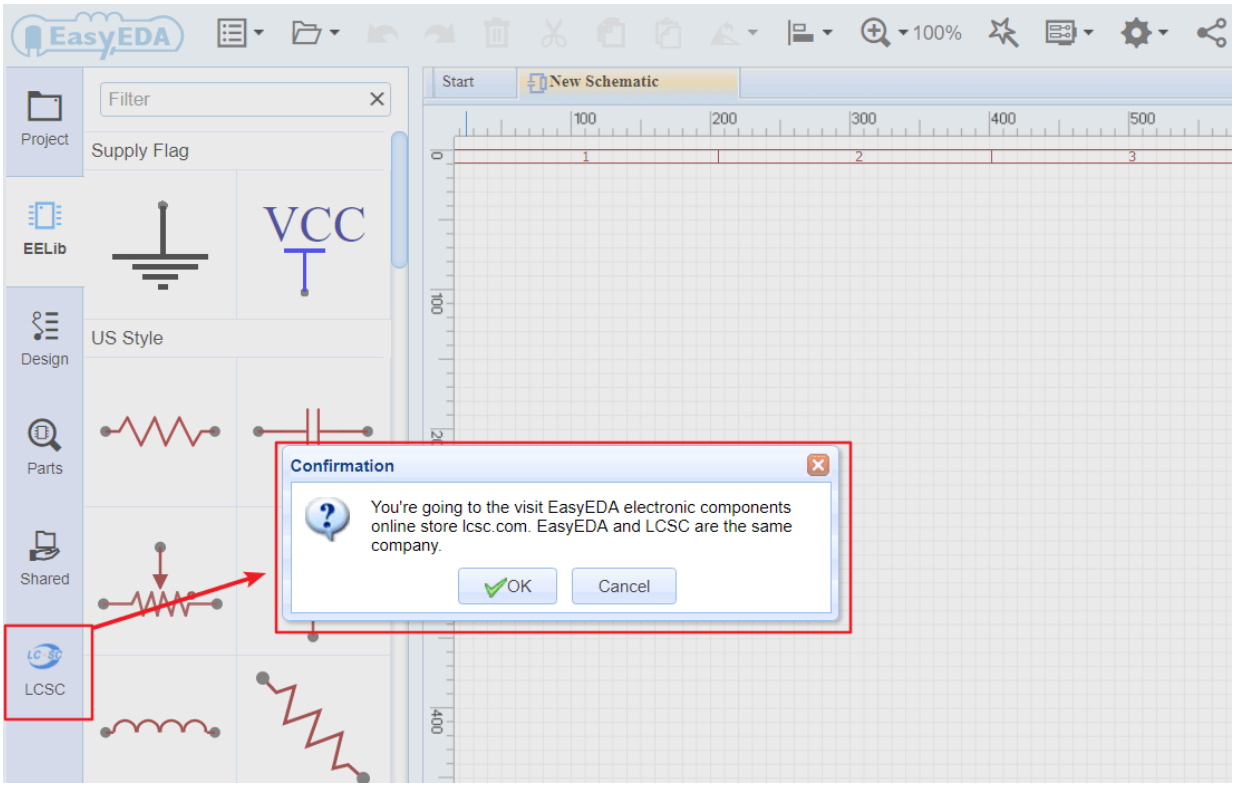
LCSC

If you want to buy components to finish your PCBA, you should try the **LCSC** module, LCSC.com and EasyEDA are the same company.

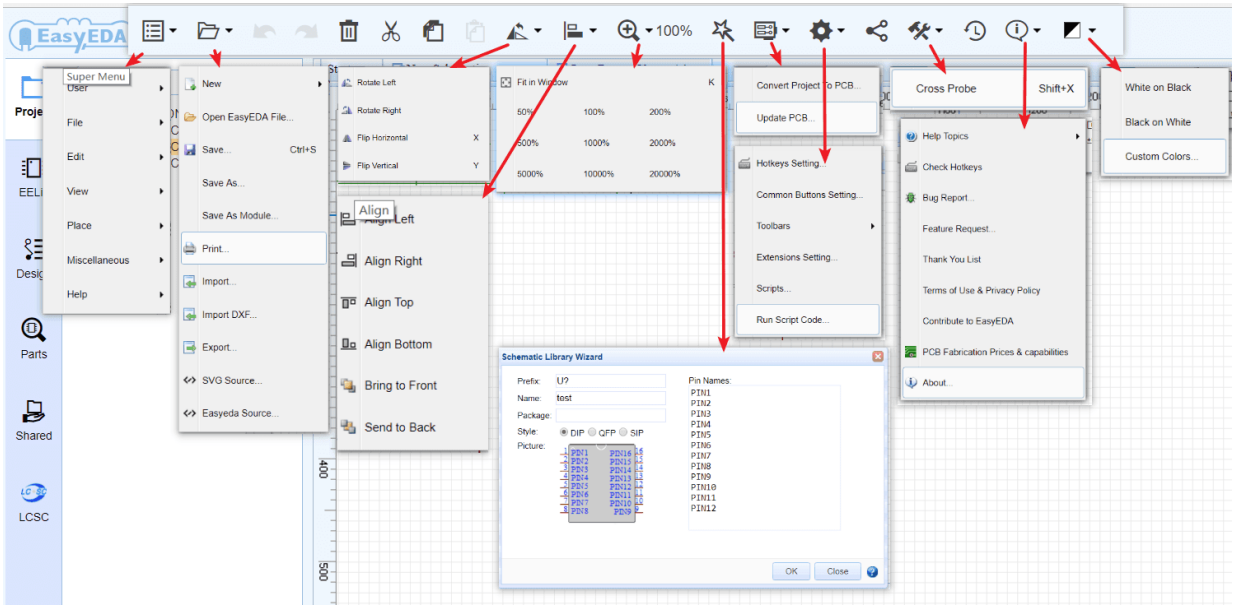
EasyEDA partners with China's largest electronic components online store by customers and ordering quantity launch https://lcsc.com. LCSC means **L**ove **C**omponents? **S**ave **C**ost! We suggest to our users to use LCSC parts to design. Why?

- 1. Small Quantity & Global Shipping.
- 2. More Than 25,000 Kinds of Components.
- 3. All components are genuine.
- 4. It is easy to order co after design.
- 5. You can save 40% cost at least.
- 6. You can use our symbols and package.

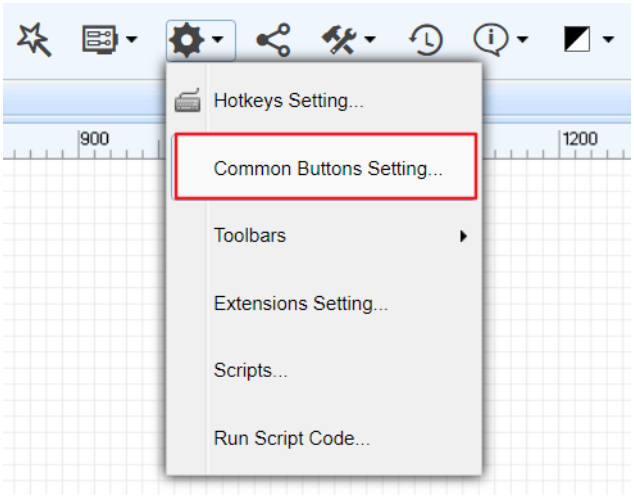




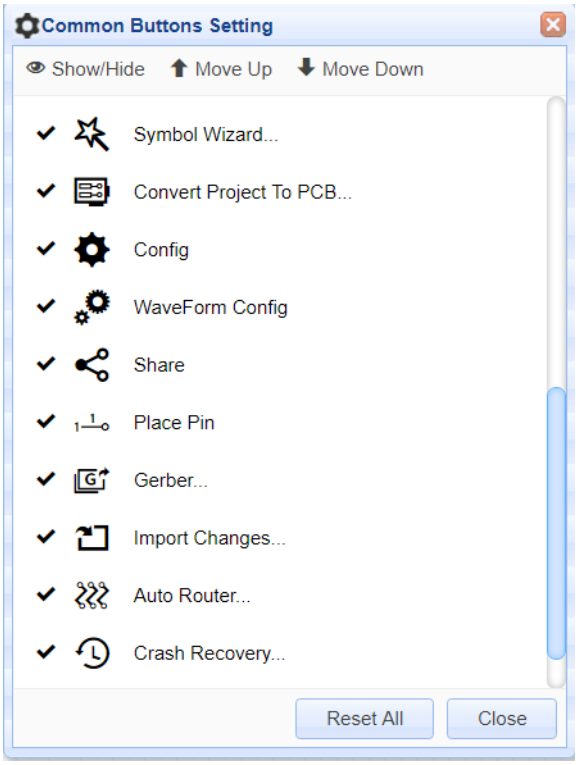
3. ToolBar



EasyEDA's toolbar can be reconfigured via Common Buttons Setting...



The configure dialog is also easy to use:



Click on a button to select it. Then you can toggle button visibility by clicking on Show/Hide or by clicking on the tick space to the left of the button icon. You can change the button position using Move up and Move Down.

Many of the buttons have been assigned hotkeys, so you can use those to replace the button actions.

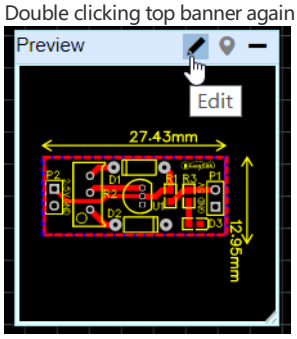
4. Preview Dialog

The Preview dialog will help you choose components and packages and can help you to identify schematics and PCB layouts.

You can close or open this dialog via:

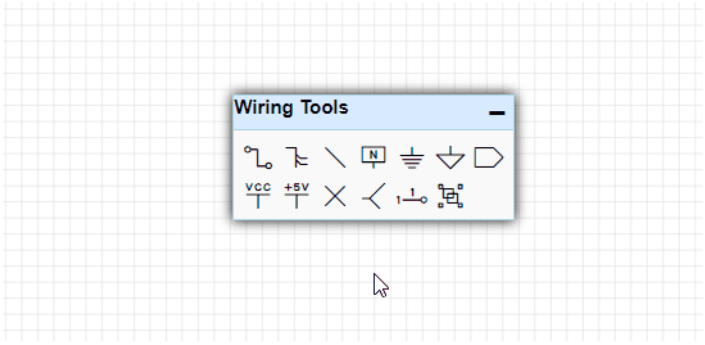
Super Menu > **View** > **Toolbars** > **Preview** or on the top toolbar **Config Icon** > **Toolbars** > **Preview**.

- The Preview Dialog has a resizing handle in the bottom right corner.
- The Preview Dialog can't be closed but double clicking on the top banner will roll up the panel or you can click the top right corner . Double clicking top banner again toggles it back to the selected size.



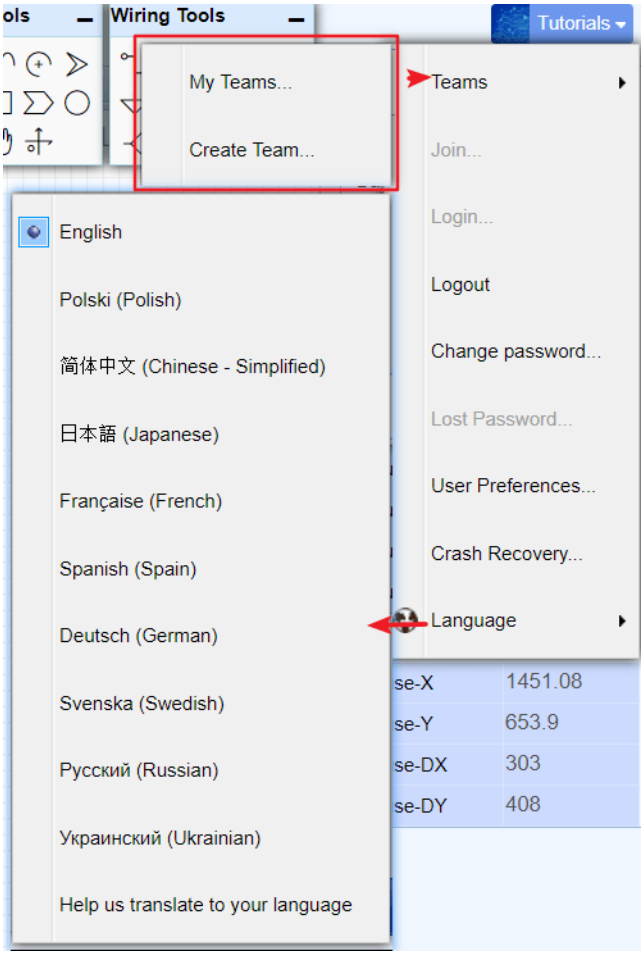
- Clicking on the little pencil edit tool opens the item in the preview for editing. Clicking on the location place tool in the top right corner of the preview dialog places the item onto the canvas. If you try to place PCB footprint into a schematic it will not provide any action and message.

5. Wiring Tools



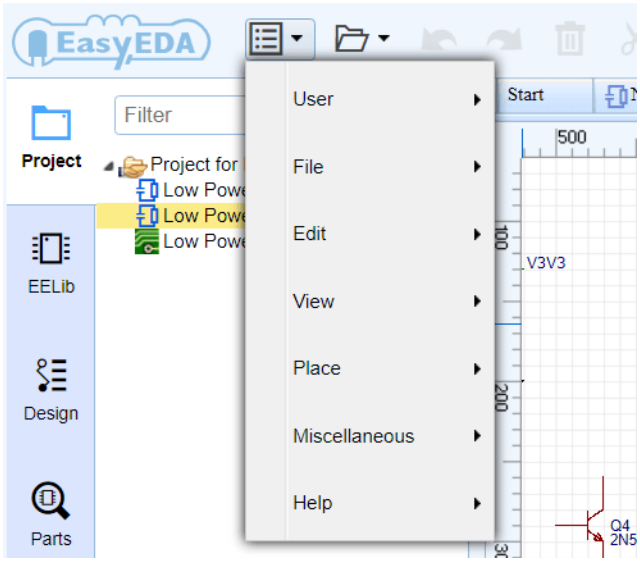
Wiring Tools are document type sensitive: different document types have different tools.

6. User management menu

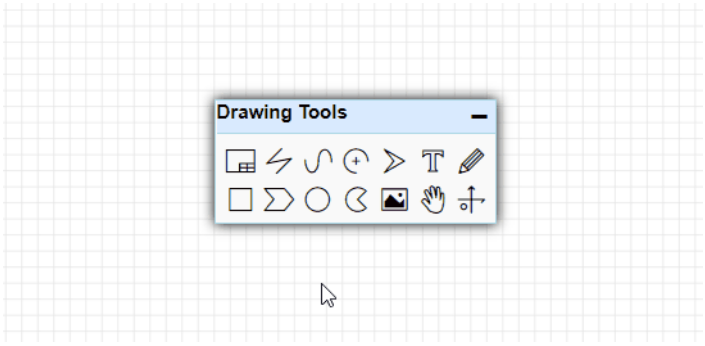


7. Super menu

All EasyEDA's menus can be found here. Most of the time, we hope you can access these options via the Hotkeys or from the top toolbar but if you find that you use some of the more specialized options from this menu frequently then may want to set them as you own hotkeys.



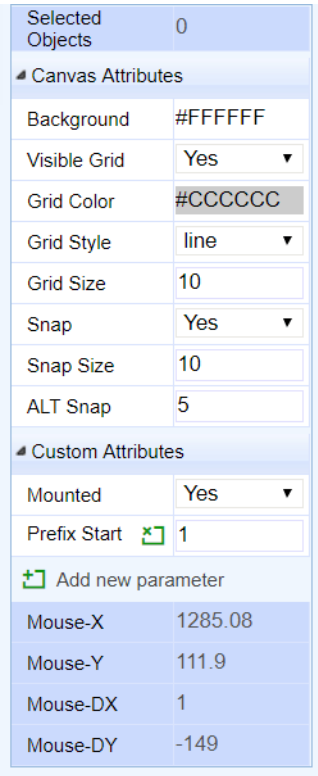
8. Drawing Tools



To keep EasyEDA's UI clean and sharp, the Wiring and Drawing tools palettes can be resized horizontally, rolled up or hidden so if you want to focus on drawing, you can roll up or hide the others to make more space and reduce the clutter.

9. Canvas Attributes

You can find the canvas Properties setting by clicking on any of the blank space in the canvas.



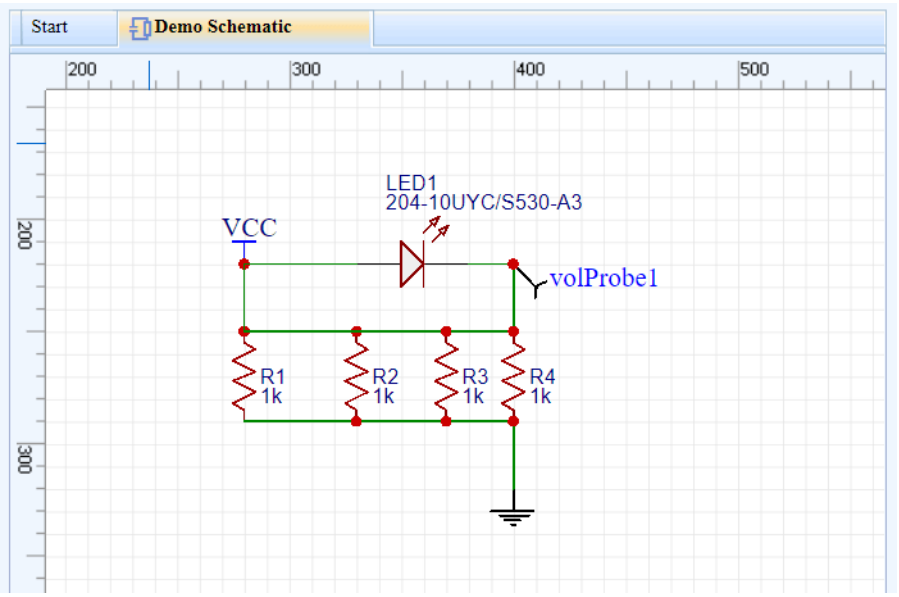
Background and grid colors and the style, size, visibility and snap attributes of the grid can all be configured.

The canvas area can be set directly by the Width and Height or from available preset frame sizes.

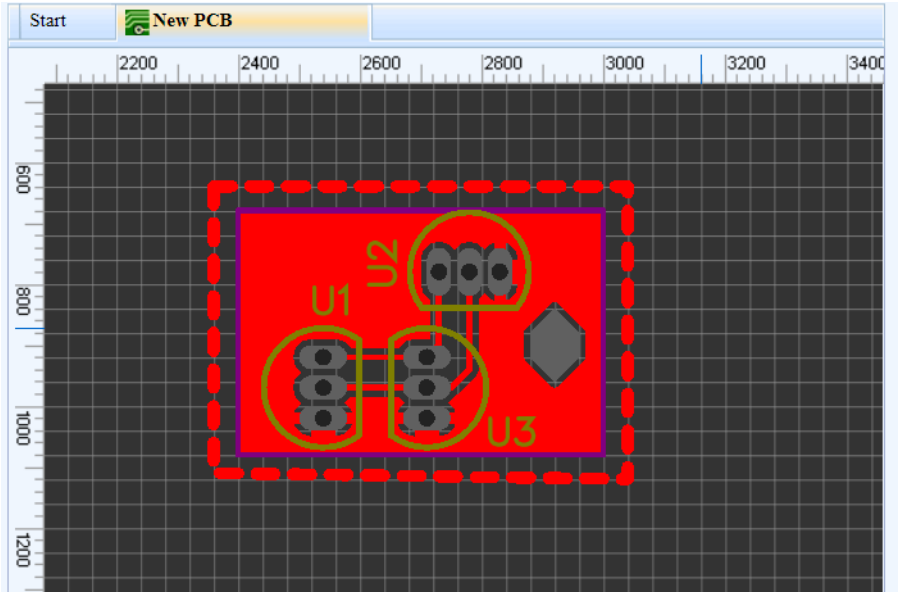
10. Canvas

This is where it all happens! This the area where you create and edit your schematics, PCB layouts, symbols, footprints and other drawings, run simulations and display WaveForm traces.

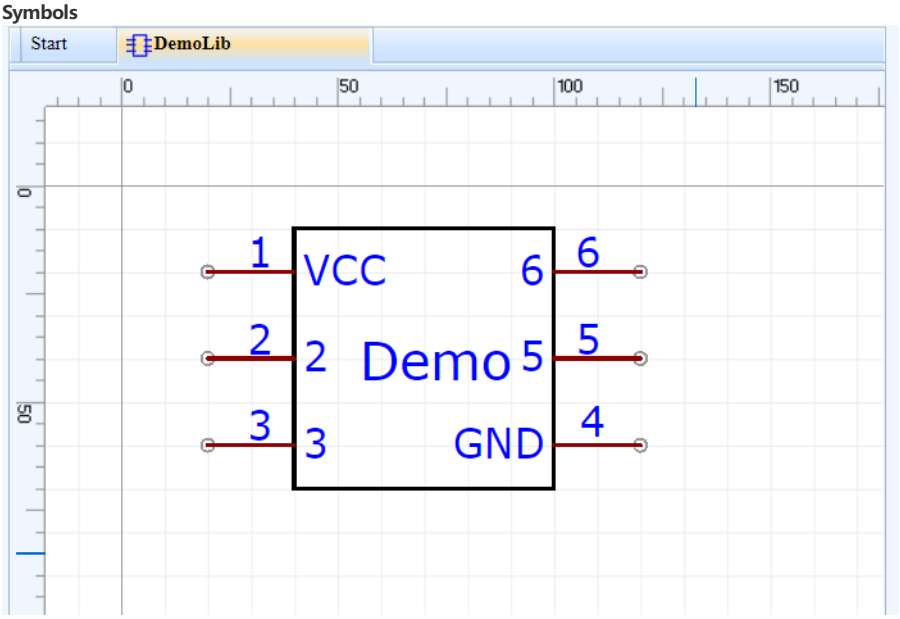
Schematic



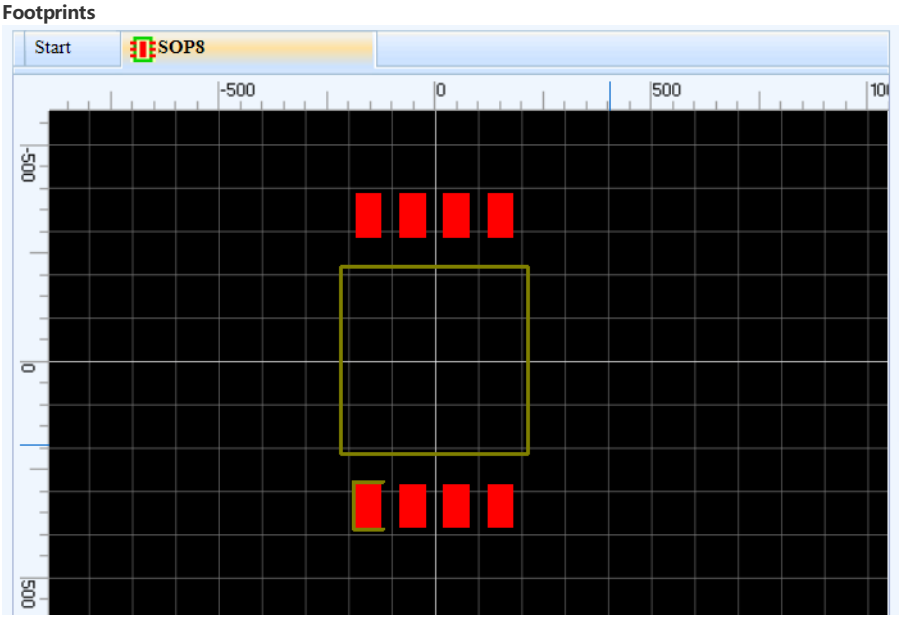
PCB



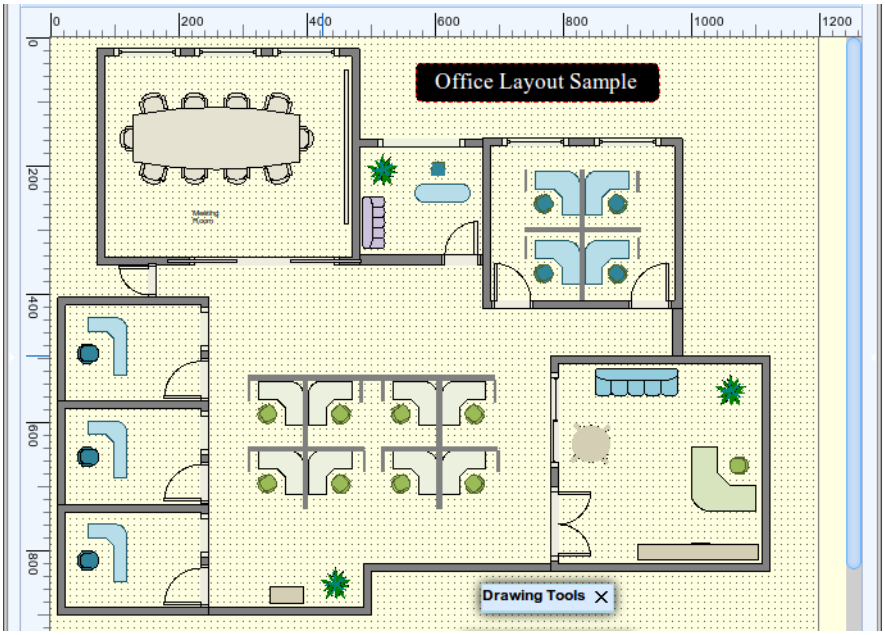
Symbols



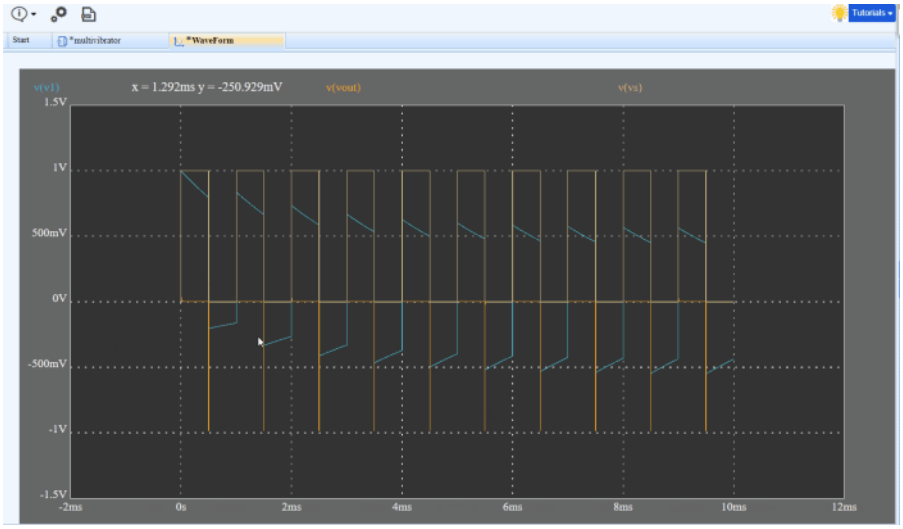
Footprints



Other Drawing

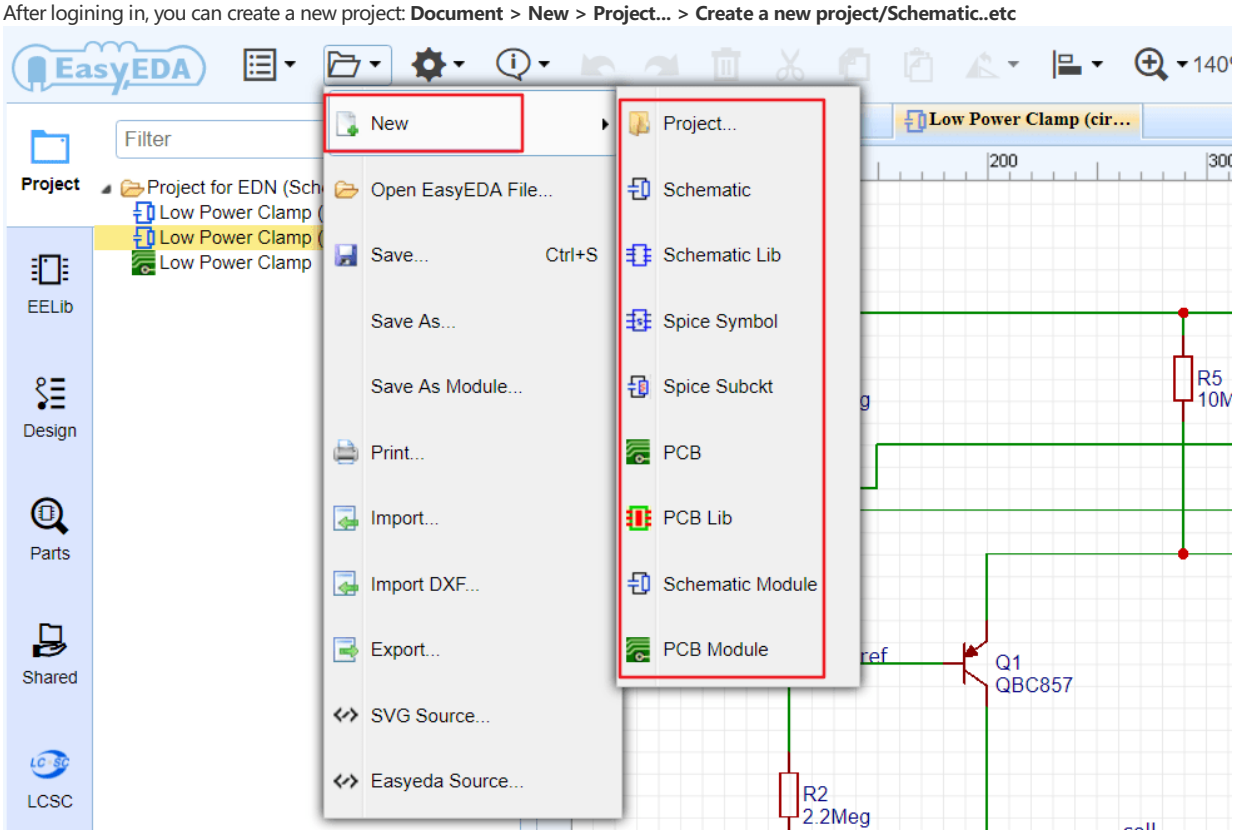


Simulation WaveForm

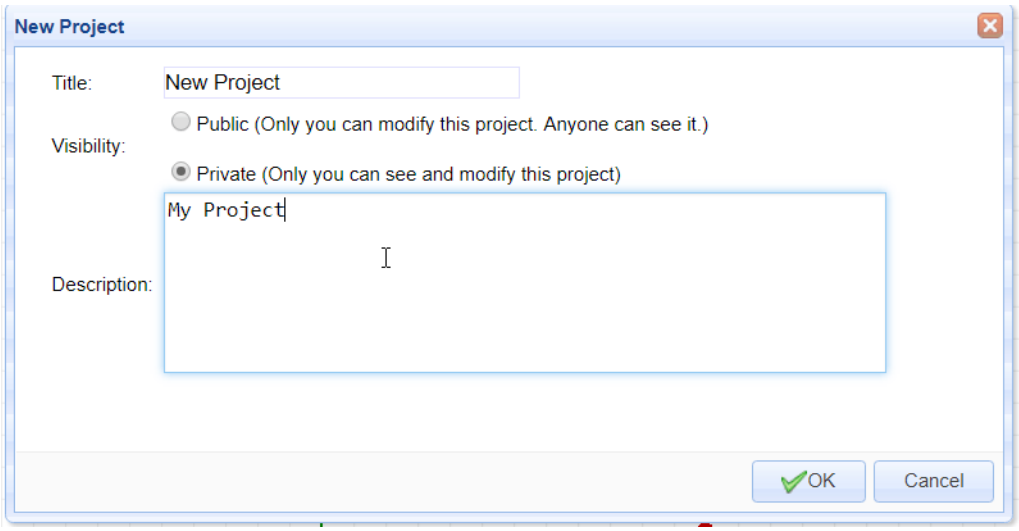


How to Create A New Project or File

After logining in, you can create a new project: **Document > New > Project... > Create a new project/Schematic..etc**



The Project concept is important in EasyEDA because it is the foundation of how to organise your designs.



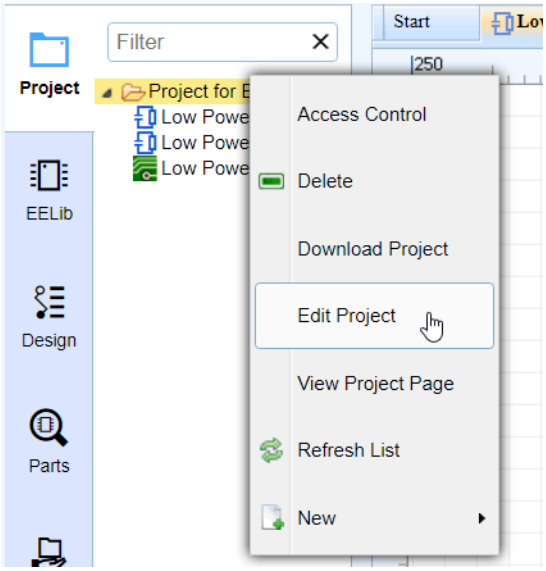
Give it a title: this will show in the project tree in the left hand panel.

You can make your project public or private by setting its Visibility.

If you choose to make you project Public, Categories allows you to select which category you want your project to be listed under on our website. If you keep your project private then the category is still applied but has no direct use in sorting your projects because this field is not searched in the Filter box in the left hand panel.

Adding a short description helps you and anyone you are sharing this project with understand what the project is about.

Once created, to modify your project, right click on it in the project tree in the left hand panel,



then will open a web page in which you can edit your project:

just test

Tags

Add Tag

Attachments

Upload Files Or drag and drop files here

Accept jpg,gif,png,jpeg,zip,rar,tar,gz,7z,doc,docx,txt,xml files. Maximum size is 8MB per file.

Public

Allow other people to see this project

Allow Comment

Allow other people to comment

License

Unkown License

Status

Not Set

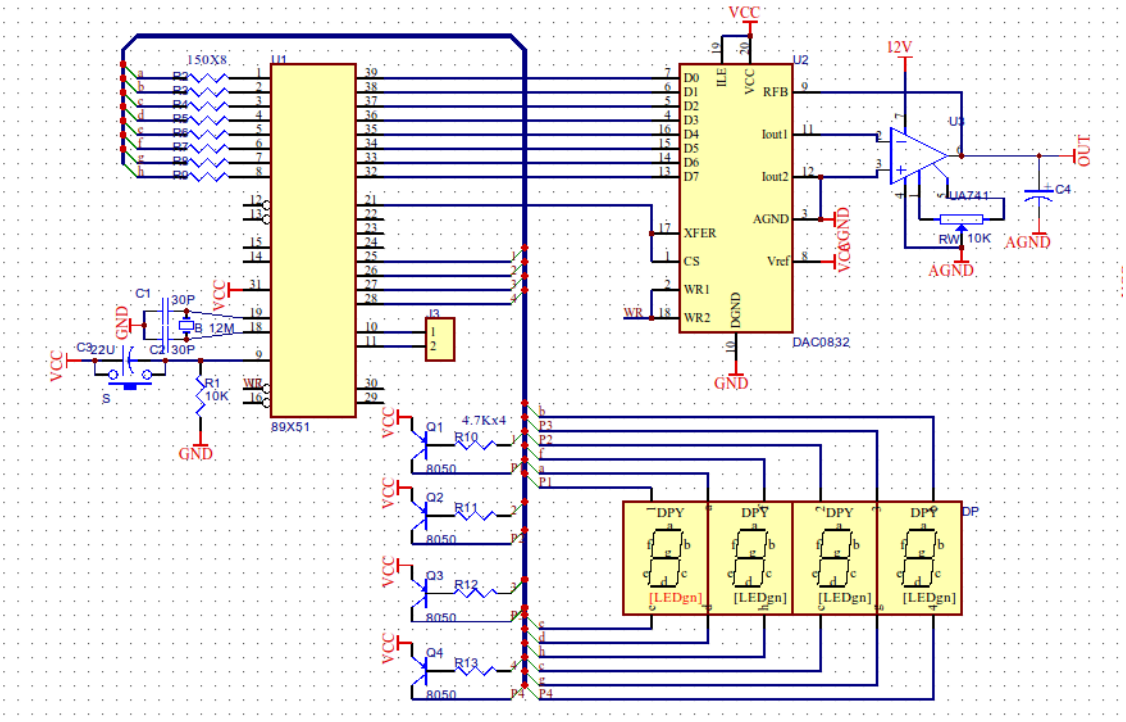
Save

From here, you can change the Visibility, allow other people to comment on your project and type a more detailed description of the project content. To help you make your project stand out or to maybe simply make a detailed description of your project easier to read, you can use Markdown syntax. If you need more information on Markdown syntax, click on Markdown Syntax? just above the Content box.

Function introduction

Schematics

EasyEDA can create highly professional looking schematics.

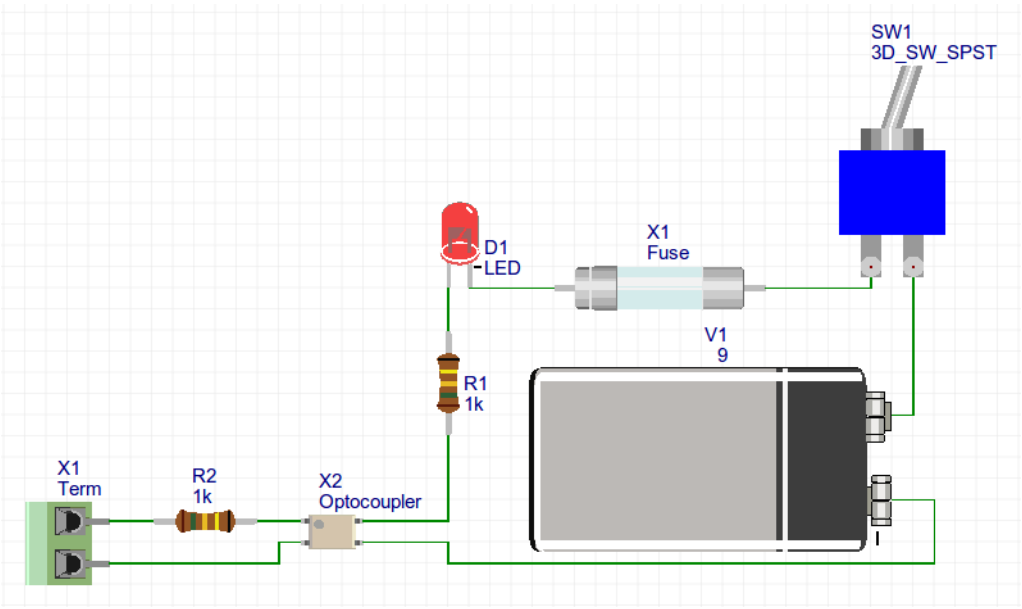


Because EasyEDA has some simple but powerful drawing capabilities, you can create your own symbols either by copying existing symbols into your own library and then editing and saving them, or by drawing them from scratch.

There is also a **Symbol Wizard** to quickly draw new symbols for `DIP`, `QFP` and `SIP`

A feature of EasyEDA is that as well as extensive libraries of the usual simple "2D" graphical schematic symbols, it has a library of drawn `3D` component symbols, i.e. symbols that look like the physical components that they represent.

If you have enough time and patience using the drawing features to full effect in symbol creation, your schematic can be built like this:



Another powerful feature is that it is also possible to import symbols from `Kicad`, `Eagle` and `Altium` libraries.

Libraries management

Thanks to the Free and Open Source Kicad Libs and some Open Source Eagle libs, EasyEDA now has 100,000+ components, which should be enough for most projects.

Now you can enjoy using EasyEDA without having to spend so much time hunting for or building schematic symbols and PCB footprints.

Search symbols

On the left hand Navigation panel you will find "**EElib**" and "**Parts**", just type what components you want and search.

Create symbols

EasyEDA supports creating symbols by yourself, after created you can find out your components at **Parts** > **My parts**, and it is easy to manage your parts.

To prepare for the final assembly stage you can create a Bill of Materials (**BOM**) via:

Super Menu > Miscellaneous > BOM Report...

and you can produce professional quality `SVG`, `.PNG` or `.PDF` output files for your documents.

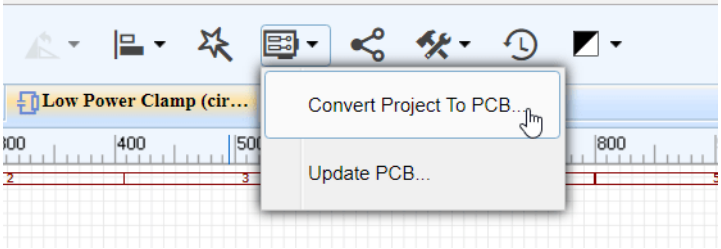
All EasyEDA Schematic Symbol and PCB Footprint libs are public, so after you have created and saved a new symbol or footprint, others will be able to find your part and you will be credited as a contributor. https://easyeda.com/page/contribute

PCB Design

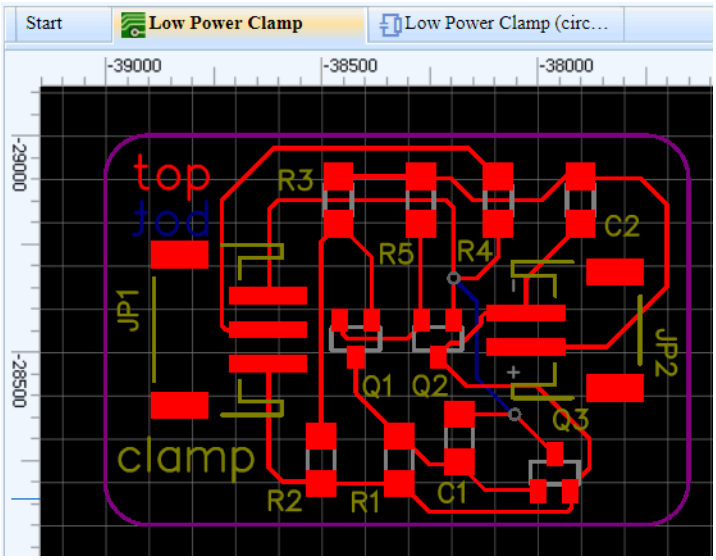
When you are satisfied with your schematic design and simulation results, you can then quickly proceed to produce your finished and populated PCB without leaving EasyEDA.

EasyEDA's PCB Design canvas helps you to quickly and easily lay out even complex multilayer designs from schematics you have already created in the Schematic canvas or directly as a layout with no schematic.

Passing an EasyEDA Schematic into the PCB Design editor is as easy as clicking a button: Just click the **Convert Project to PCB** PCB icon on the top toolbar!



EasyEDA has extensive libraries of footprints. You can also build up your own library of unusual and specialized parts by copying and modifying existing parts or from scratch using EasyEDA's powerful footprint creation and editing tools.



In a similar way as in the Schematic design canvas, to help you locate items and navigate your way around when working in the PCB Design canvas there is a PCB Design Manager.

Left Navigation Panel > Design

The PCB Design Manager is a very powerful tool for finding components, tracks (nets) and pads (Net Pads).

Clicking on any item highlights the component and pans it to the center of the window.

You can set up layers used in the PCB and their display colours and visibility using

Super Menu > Miscellaneous > Layer Options...

The active layer and layer visibility can be selected using the Layers Toolbar.

Default track widths, clearances and via hole dimensions can all be configured in the Design Rule Check dialog which is opened via:

Super Menu > Miscellaneous > Design Rule Setting...

From first setting up the Design Rule Check (**DRC**) at the start of your board layout, running a DRC is almost the last step in checking your PCB design before you generate **Gerber** and **Drill** files for board manufacture ready to place your order for a finished PCB.

The last step is to check the Gerber and Drill files using an easy to install and use Free and Open Source Software Gerber Viewer: Gerbv: http://gerbv.geda-project.org/

While you are waiting for your PCB to be delivered, you can create a Bill of Materials (BOM) via:

Super Menu > Miscellaneous > BOM Report...

and you can produce professional quality `SVG`, `.png` or `.pdf` output files for your documentation.

PCB Designs can be shared with colleagues and made public in the same way as Schematics.

The size of PCB that you can produce using EasyEDA is almost unlimited: designs of over 100cm * 100cm are possible ... but you might need a powerful computer for that.

EasyEDA supports up to 6 layer PCBs by default but it is capable of handling more, so if you need more layers then please contact us as shown in the section on How to get Help?.

Search footprints

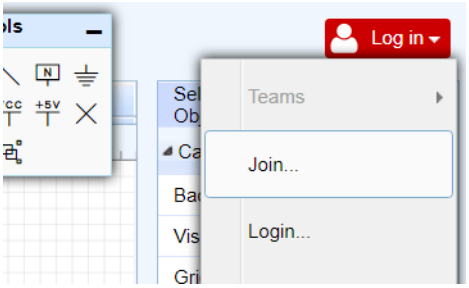
Searching footprints is the same as searching symbols by using **Parts** in the Schematic. You can place the selected footprints in the canvas after the search.

Account Management

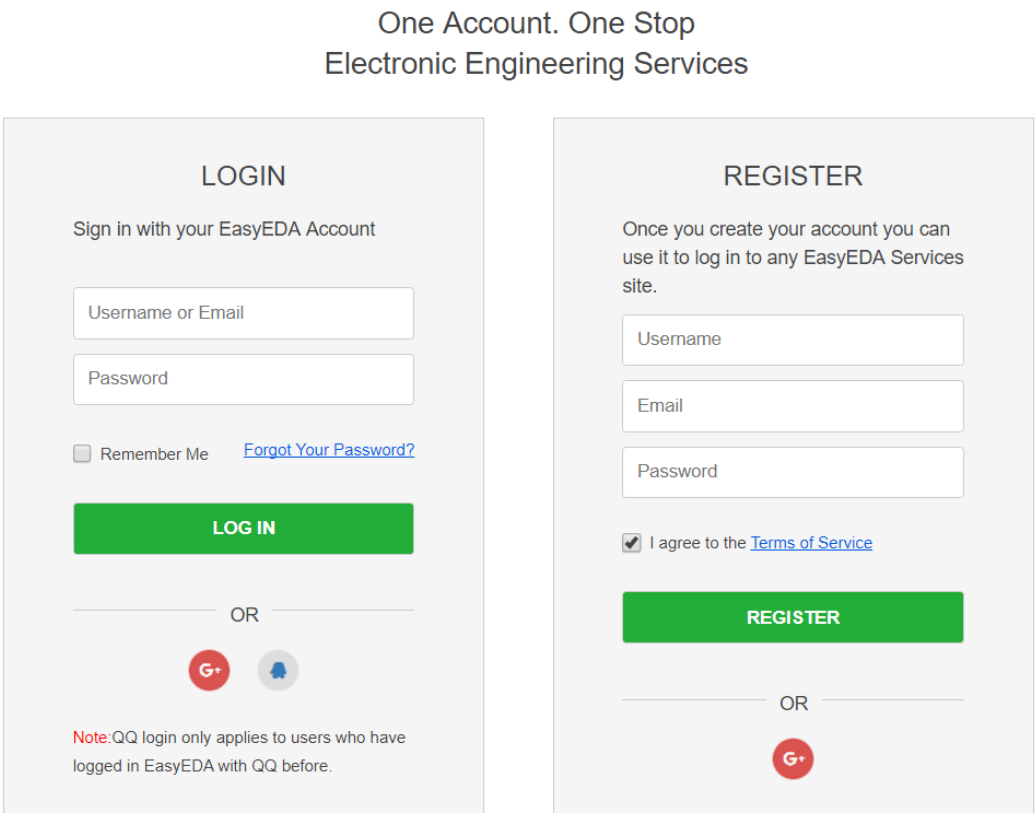
EasyEDA is a web-based service and although you are free to use it in Anonymous mode which you can do without creating an account, you are much better off creating an account to manage your own designs and parts libraries. Creating an account is easy and gives you free access to the full power of EasyEDA for as long as you wish.

Join

Click **Join...** on the **User Management** menu:



After clicking on **Join**, a new webpage about **Create an account** opens.



Just enter a username, invent a password, confirm it and type in an email address. A valid email address is needed so that we can send you a confirmation email before we create your account. This is also the address we will use to contact you with information or any questions about your PCB orders.

Login

The Login dialog image can be seen in the Join section above.

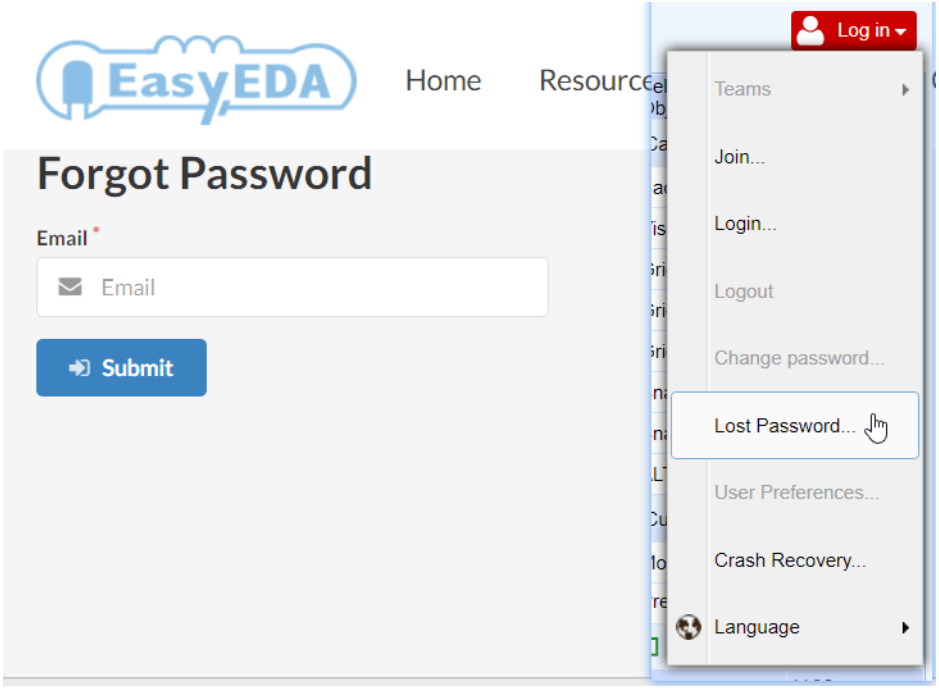
After clicking on Login, you can enter the username or email and password to login to EasyEDA. If you use a private device, you can check **Remember Me**, so you don't need to login again each time you open EasyEDA.

Alternatively, if you have a Google or Tencent QQ account, you can login in using http://en.wikipedia.org/wiki/OpenID; it is safe and easy.

Note: *QQ login only applies to users who have logged in EasyEDA with QQ before.*

Reset Your Password

Your password is encrypted, so EasyEDA team doesn't know it, but you can reset it via your email. Click the **Lost Password** menu and type your email. If you can't get the email after 10 minutes, please send an email to us.

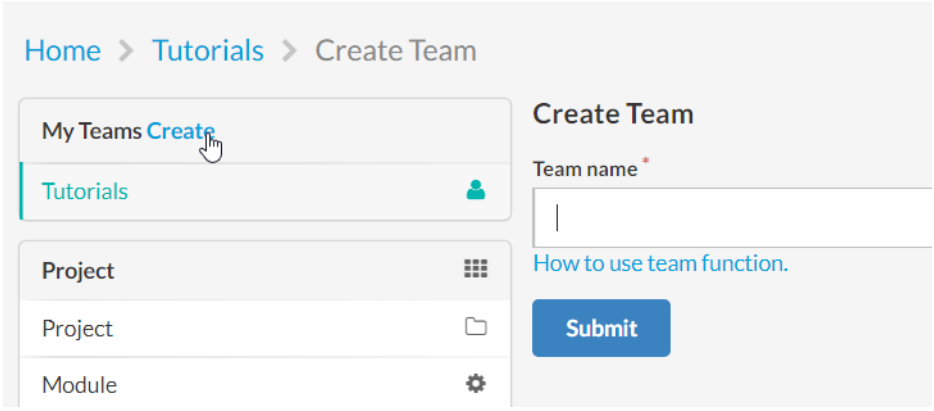


Teams

EasyEDA provides a team feature with which you can work seamlessly with your partners. You can work as if everyone is logged in under the same account, with full access to all components, Schematics, PCBs and Projects.

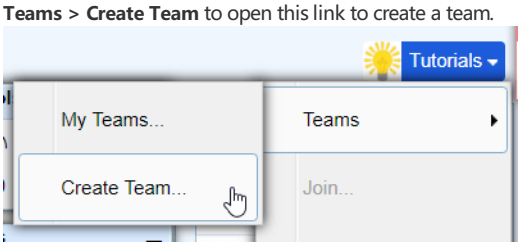
How to find the team function

Under the dashboard , there is a team section.

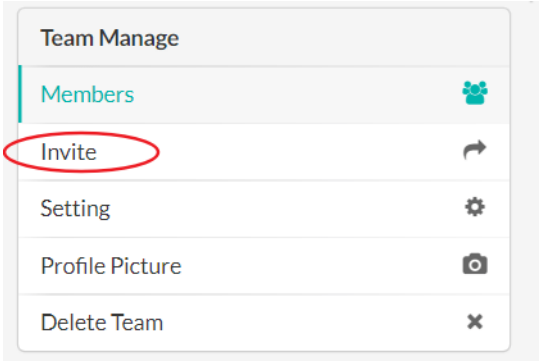


How to create a team

There is a link as shown in the image above, or click https://easyeda.com/teams/create after you login, Or you can click **User Management** > **Teams** > **Create Team** to open this link to create a team.

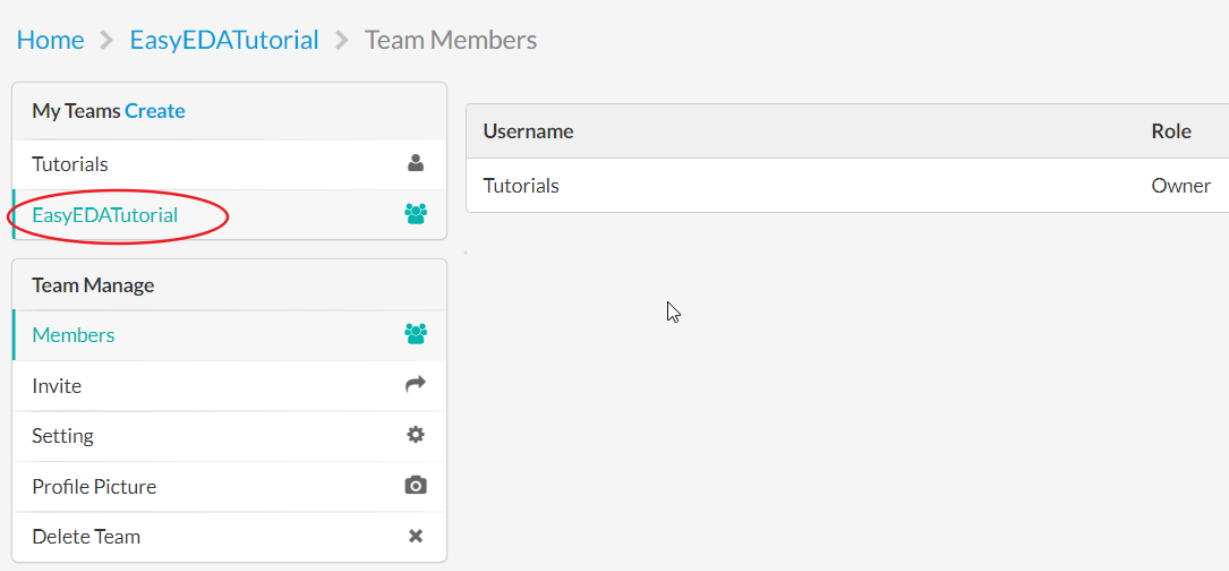


And then you need to invite your partner(s) to join this team at Team Manage of the dashboard:



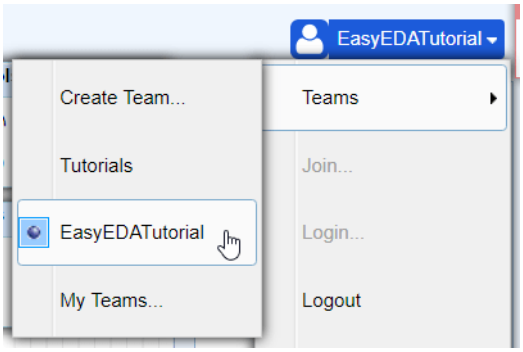
How to switch to team model

1) switch to the dashboard. After you have created a team, click the team name and the dashboard will switch to showing only the team projects, files and components.



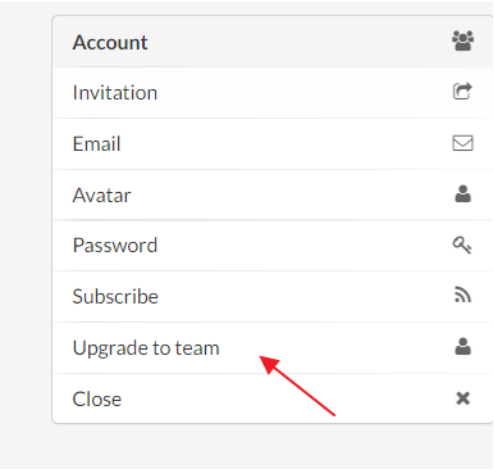
After switching to a team, there is a team management section where you can manage your team members, invite new team members and even delete the team.

2) switch to the editor. Under your personal menu, there is a sub menu allowing you to switch to a team or to your personal account.



How to Upgrade to a team If you want to contribute all of your designs to a team, you can use this function. First you need to create a team, then click the link, shown below, under the dashboard.

Be careful ! , because after you do that, **all** of your components, projects will be moved to your team.



Tips about the team function.
1.If you switch to a team, you can't automatically use any Packages/Footprints which you have created under your personal account. You need to **Favorite** your personal package/components first.

2.You need to be aware that your team and your personal accounts are the different, separate accounts and that you can't use them both at the same time.

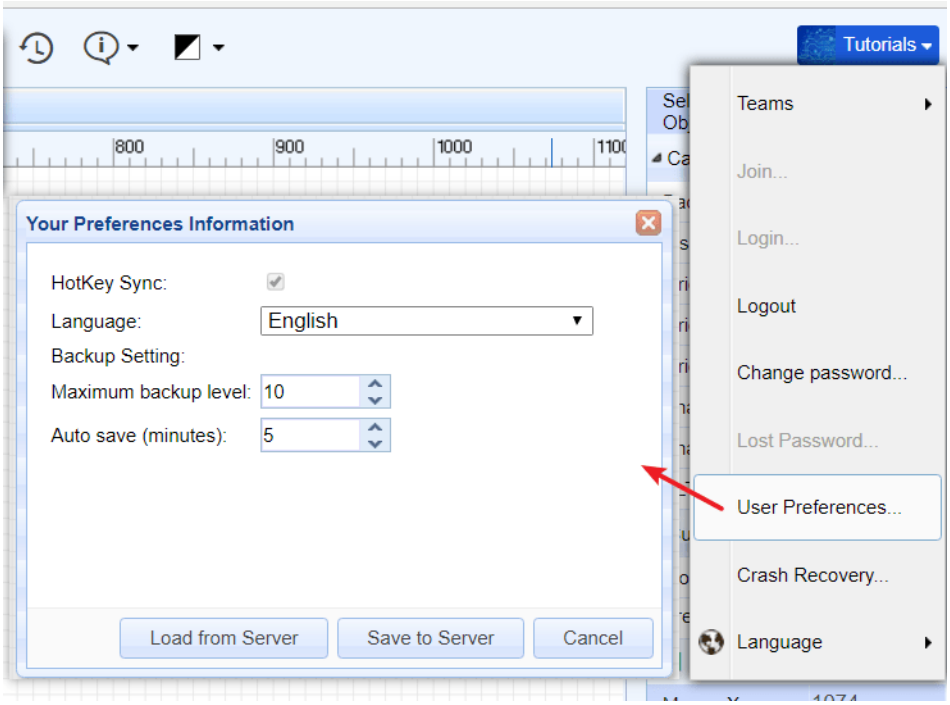
3.After making yourself the owner of a team, it is best to create any Components and Packages needed by the team under that team.

4.If you add a member, nominated to be your accountant, to your team then they can deal the team billing and invoices.

User Preferences

When EasyEDA shows up the login success popup in the bottom right of the window, your user management menu will be look like this:

Click on **User Preferences**,



Maximum backup level: every open document can be saved at up to this number of different revisions.

Auto save (minutes): this is the time interval between auto saves of all your open documents.

Save to Server: Save your preferences (Toolbar configurations, EasyEDA libs, Hotkey settings, language and so on) to the EasyEDA Server.

Load from Server: EasyEDA can't load your Preferences automatically but once you have saved them, you can load them manually. Then, when you change to a different computer or browser, you can load your preferences from the EasyEDA Server.

If you have not saved any preferences then **Load from Server** will have no effect.

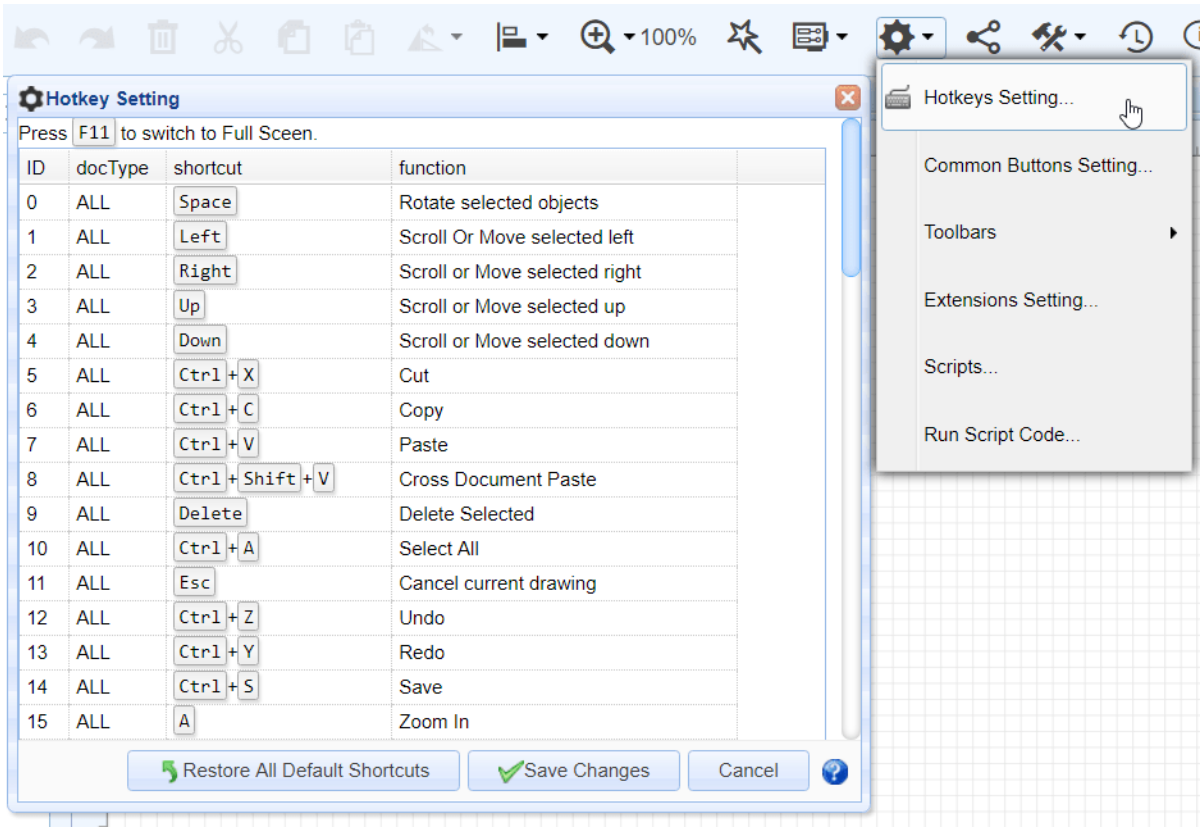
Close Account

If you want to close your account, you can go to https://easyeda.com/account/close
Before you close your account, please let us know why, and that will make us to be better.

Hotkeys

After a while of using an EDA tool suite, clicking all over the place with a mouse gets very tedious and seriously reduces your productivity. Keyboard shortcuts or Hotkeys avoid much of that. EasyEDA not only provides lots of hotkeys, but also every hotkey can be reconfigured.

Under the Config toolbar, click the Hotkeys Setting... Menu which will open the Hotkey Setting dialog.



To change a Hotkey, click anywhere in the row for the hotkey you want to change and then press your new key.

For example, if you want to use R instead of space to rotate selected objects, click on the first row, then press `R`.

After you change the hotkey, don't forget to click Save Changes button.

The **docType** column describes which type of EasyEDA document each hotkey applies to. **docType** has three types:

- **ALL**: any document type in EasyEDA.
- **SCH**: schematic and schematic libs
- **PCB**: PCB and PCB libs.

The functions of some hotkeys may change between docTypes. For example, the hotkey `c` draws an Arc in SCH, but draws a circle in PCB.

A list of all the available default hotkeys is given below.

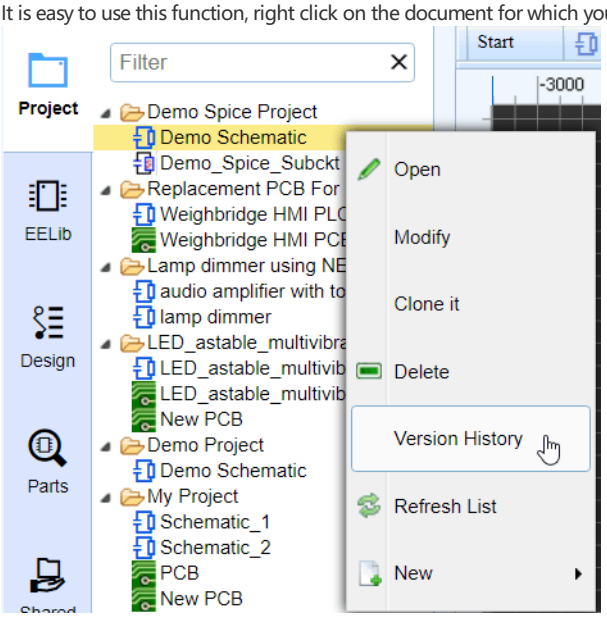
ID	docType	shortcut	function
0	ALL	Space	Rotate selected objects
1	ALL	Left	Scroll Or Move selected left
2	ALL	Right	Scroll or Move selected right
3	ALL	Up	Scroll or Move selected up
4	ALL	Down	Scroll or Move selected down
5	ALL	Ctrl+X	Cut
6	ALL	Ctrl+C	Copy
7	ALL	Ctrl+V	Paste
9	ALL	Delete	Delete Selected
10	ALL	Ctrl+A	Select All
11	ALL	Esc	Cancel current drawing
12	ALL	Ctrl+Z	Undo
13	ALL	Ctrl+Y	Redo
14	ALL	Ctrl+S	Save
15	ALL	A	Zoom In
16	ALL	Z	Zoom Out
17	ALL	X	Flip Horizontal
18	ALL	Y	Flip Vertical
19	ALL	G	Snap
20	ALL	Ctrl+F	Find Component
21	ALL	Ctrl+D	Design Manager
22	ALL	D	Drag Tool
23	SCH	W	Draw Wire
24	SCH	B	Draw Bus
25	SCH	U	Bus Entry
26	SCH	N	NetLabel
27	SCH	Ctrl+Q	NetFlag VCC
28	SCH	Ctrl+G	NetFlag GND
29	SCH	P	Place Pin
30	SCH	L	Draw Polyline
31	SCH	O	Draw Polygon
32	SCH	Q	Draw Bezier
33	SCH	C	Draw Arc
34	SCH	S	Draw Rect
35	SCH	E	Draw Ellipse
36	SCH	F	Freehand Draw

37	SCH	T	Draw Text
38	SCH	I	Edit Selected Symbol
39	SCH	Ctrl+R	Run the Document
40	PCB	W	Draw Track
41	PCB	U	Draw Arc
42	PCB	C	Draw Circle
43	PCB	N	Draw Dimension
44	PCB	S	Draw Text
45	PCB	O	Draw Connect
46	PCB	E	Draw copperArea
47	PCB	T	Change To TopLayer
48	PCB	B	Change To BottomLayer
49	PCB	1	Change To Inner1
50	PCB	2	Change To Inner2
51	PCB	3	Change To Inner3
52	PCB	4	Change To Inner4
53	PCB	P	Place Pad
54	PCB	V	Place Via
55	PCB	M	Measure
56	PCB	L	Change Route Angle
57	PCB	-	Decrease Routing Width
58	PCB	+	Increase Routing Width
59	PCB	Alt+-	Decrease Snap Size
60	PCB	Alt++	Increase Snap Size
61	PCB	H	Highlight Net
62	PCB	Shift+M	Remove All Copper Area
63	PCB	Shift+B	Rebuild All Copper Area

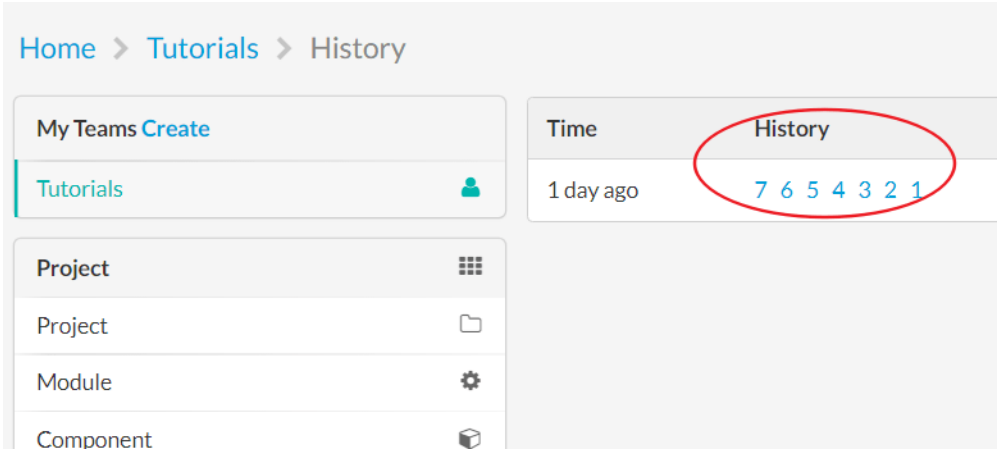
Basic Driving Skills.

Version History

It is easy to use this function, right click on the document for which you need the version history in like in the image below:



After clicking on the version history link, you will get a list of all of the versions like in the image below.



Click the version number, you can open the saved file in the editor, if this is what you need, you can save it to your project and delete your bad file.

Note:
1. For now all of the versions are marked as number, we will allow you to add a tag soon.
2. Don't save your files too frequently, or you will get lots of versions and it will be hard to find the exact one you want.

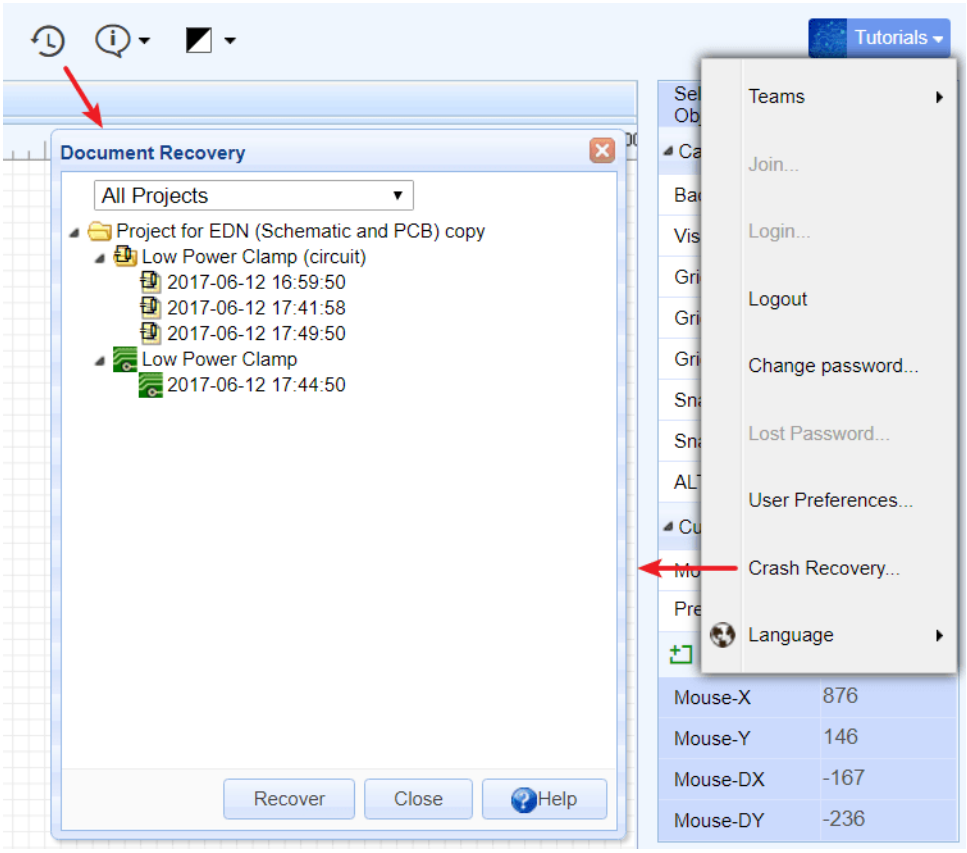
Crash Recovery

No operating system, software or network is perfect, so sometimes things can go wrong. Having your Desktop or web browser freeze or your broadband connection drop, two hours into laying out a PCB, could spoil your day.

However, with EasyEDA, your day will be just fine.

This is because EasyEDA auto saves and makes backups of all your open files to your computer so crash recovery is built into EasyEDA.

In **user management menu**, click on **Crash Recovery**. Or you can click **Crash Recovery** button on the top Toolbar as below:



Select the file which you would like to **recover**, then click the Recover button; your file will be opened in a new tab.

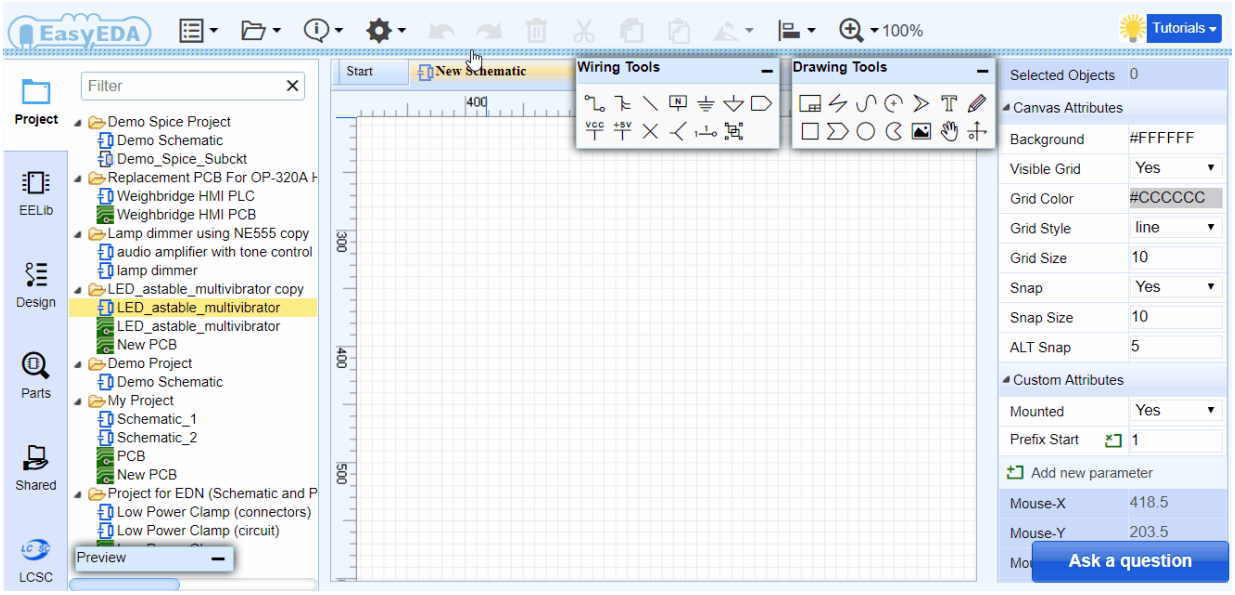
Please note:
- EasyEDA saves these crash recovery files on your computer and not on the EasyEDA server. Therefore you cannot recover files from a crash on one computer or browser by changing to a different computer or browser.
- And if you cleaned your browser's cache, the recovery files will disappear. - If you make a mistake to delete a file and remove the cache already, maybe you can find your document back via : https://easyeda.com/document/recycle.

To use EasyEDA, you need to be familiar with a few basic terms and concepts. The best way to learn them is to open up EasyEDA, open a new schematic:

Document > **New** > **Schematic** , and play!

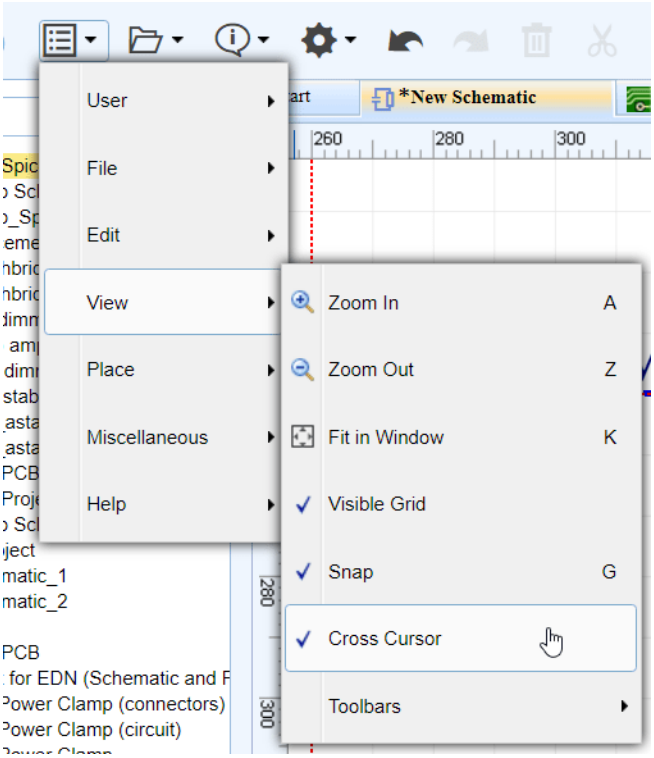
Resizing the canvas area

Hovering the mouse cursor over the areas indicated by the three green ellipses will bring up blue toolbar toggle lines. Clicking on them will toggle the visibility of their associated top, right and left toolbar areas to expand the canvas area. The vertical lines can also be dragged horizontally to resize the panels.

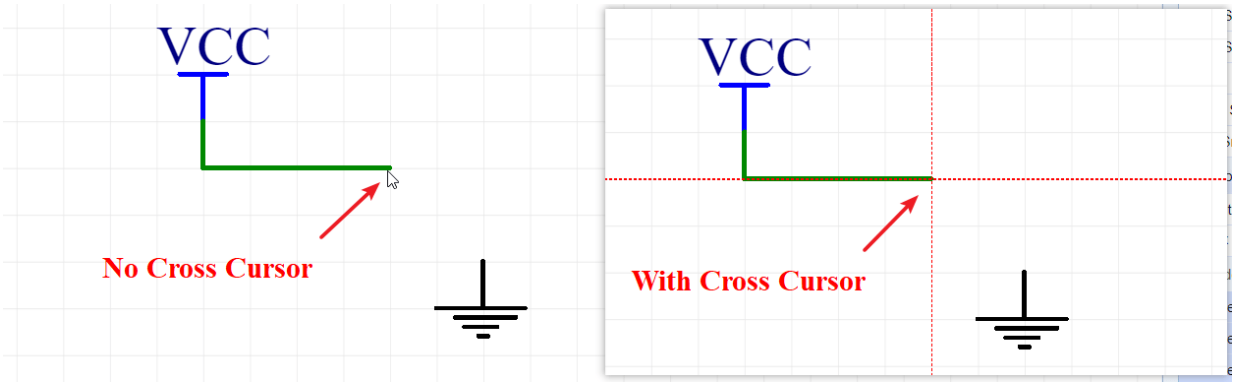


Cursor Style

Some users don't like the cross cursor, so you can change it to arrow cursor like in the image below.



These difference between these options is as below:

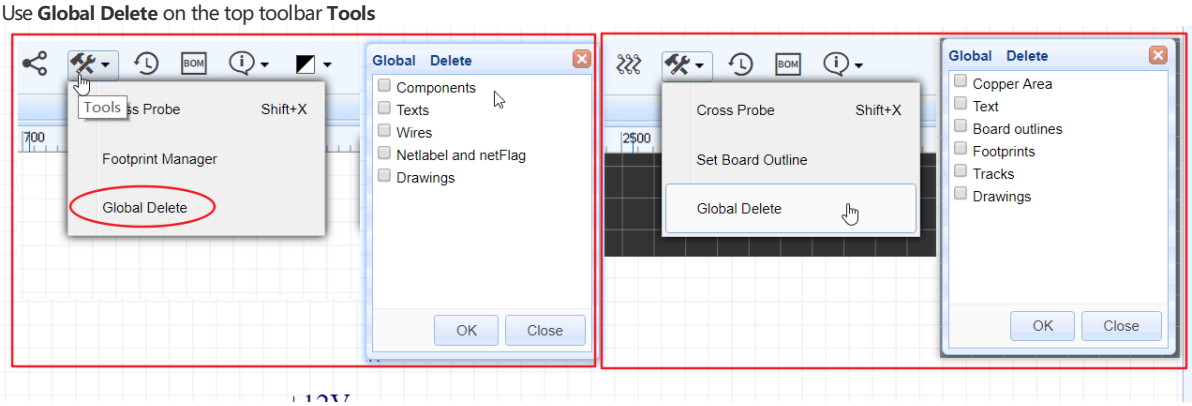


Clear

If you think your schematic or PCB looks terrible, and you want to redraw all units, you can:

- **Super Menu > Edit > Clear**.

- Delete this schematic and create a new one.
- Use **Global Delete** on the top toolbar **Tools**



Left clicking

Similar to other EDA software:

- Click on an item to select it;
- If over a selected item, click and hold to drag a selected item;
- If not over a selected item, clicking and holding while dragging creates a selection box;
- the selection box, using click and drag to the right, selects everything inside the box;
- the selection box, using click and drag to the left, selects everything inside and intersected by the box;
- Double click on a text area to edit it;
- The exact left click functionality depends on what item is being selected and in what Canvas the item exists (Schematic or PCB).

Right clicking

EasyEDA does not support right click context menus in the Schematic or PCB Canvas. Instead, right clicking executes a context sensitive command:

- When you are placing a symbol, after a right click, the active symbol will be removed;
- When you are drawing a shape such as a polyline, after a right click, the polyline will be stopped at the place where you right click but the mouse will remain as a **cross**, so you can draw another shape;
- To get out of the current active context sensitive command such as placement or drawing mode and go back to **select mode**, just double right click.

Ctrl+Right clicking anywhere in the Schematic, waveForm or PCB Canvas drags the canvas around within the EasyEDA window.

ESC key

Pressing the `ESC` key ends the current drawing action but does not exit the current active context sensitive command mode (i.e. it does not return the cursor to select mode).

Select more shapes

- Ctrl+left clicking on items adds those items to your selection;
- Clicking and holding creates a selection box;
- Creating a selection box, using click and drag to the right, selects everything inside the box;
- Creating a selection box, using click and drag to the left, selects everything inside and intersected by the box;

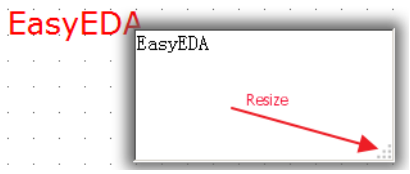
Zoom in and Zoom out

- Using the middle mouse button:
- Roll forward to zoom in;
- Roll back to zoom out;
- Using hotkeys, the default hotkey `A` for zoom in, `Z` for zoom out.

Please note:
Do not roll your mouse at the same time as pressing the CTRL key. Some browsers will zoom the whole site, not just the canvas in the EasyEDA window. If this happens, just press `Ctrl+0` to reset the browser zoom.

Double clicks

Double clicking any text area opens a resizable text box to allow you edit the text inline.



Press enter to create new line. Click outside the text box to close it.

Pan

- Right click anywhere in the Schematic, WaveForm or PCB Canvas and Hold down right button to drags the canvas around within the EasyEDA window.

- If your canvas is bigger than the EasyEDA window and is showing scroll bars, you can use either the scroll bars or the Arrow keys to scroll the canvas to pan.
- When drawing a wire, a graphic line or shape that you wish to extend beyond the edge of the EasyEDA window holding down the left mouse button after starting the line will pan the canvas to keep the drawn item inside the window.

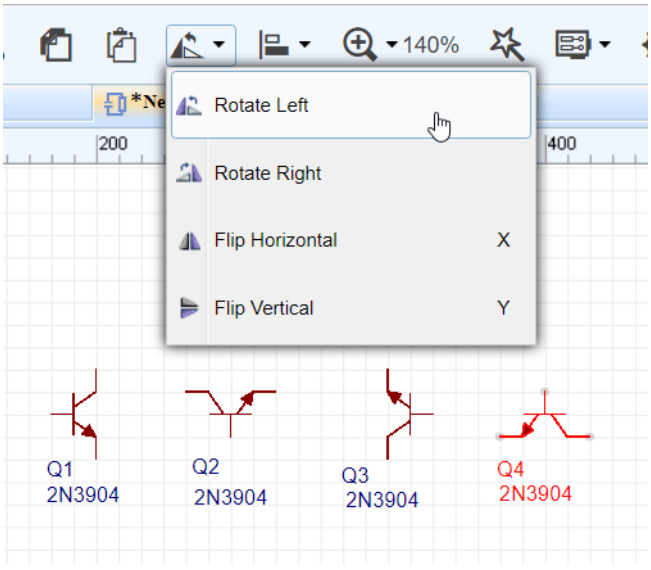
Tip: *If you use Chrome, and cursor is in the canvas while pressing CTRL or ALT key and rolling your mouse, the canvas will move vertically, and when pressing SHIFT and rolling your mouse, the canvas will move horizontally.*

Rotate

After selecting one or more items, you can rotate the selected items using:

Super Menu > **Edit** > **Rotate** or click top ToolBar **Rotate and Flip** > **Rotate Left or Rotate Right**

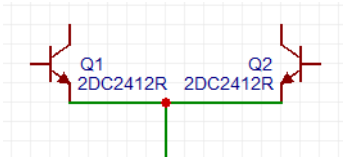
or by pressing the default rotate hotkey: `Space`.



Please note:
Rotating a multiple selection rotates each item about its own symbol origin. It does not rotate the items about the centroid of the group of items.

Flip

To place a Q2 as shown in the schematic below you need to Flip the item.



You can Flip one or more selected items using:

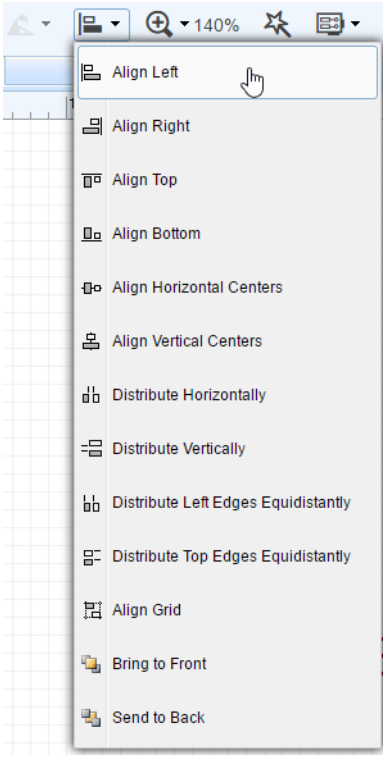
Rotate and Flip > **Flip Horizontal or Flip Vertical** from the toolbar,

or by pressing the default flip hotkeys: `X` to Flip Horizontal, `Y` to Flip Vertical.

Align

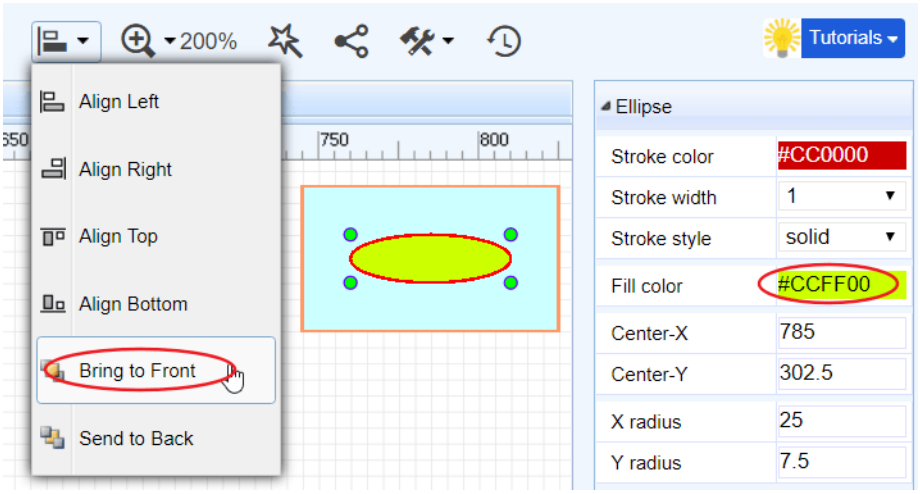
EasyEDA provides many align option features, you can align your components or footprints very easily, it include:

- Align Left
- Align Right
- Align Top
- Align Bottom
- Align Horizontal Center
- Align Vertical Center
- Distribute Horizontally
- Distribute Verticall
- Distribute Left Edges Equidistantly
- Distribute Top Edges Equidistantly
- Align Grid



Bring to Front and Send to Back

In the image below, both the rectangle and the ellipse are filled.



If you draw the ellipse before drawing the rectangle, the rectangle will overlap and therefore hide the ellipse. To reveal the ellipse, select the rectangle and then use:

Align > Send to Back from the toolbar.

To bring the rectangle to the front again, you could select it and use:

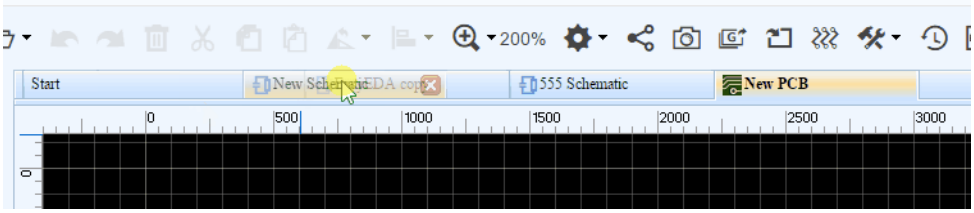
Align > Bring to Front

or select the ellipse and then use:

Align > Send to Back

Documents Tab Switch

It's easy to fit your documents tab location.



Saving Your Work Locally

Although EasyEDA saves all your files on our Server, sometimes you may want to save your work locally and EasyEDA provides a hack way to do this.

More detail you can view at Export EasyEDA Source section.

About upgrade

Version Rule

EasyEDA version number is ReleaseYearsCount.ReleaseMonth.ReleaseCountOfThisMonth. For example, v4.9.3 is the fourth year release of EasyEDA, and at the ninth month of this year, EasyEDA had released 3 times.

Version Upgrade

If you use EasyEDA online, it can seamlessly upgrade by itself. However,EasyEDA uses an App Cache technique to allow you to use EasyEDA offline (W3C HTML5 Offline Web Applications) which may delay the automatic upgrading process. Therefore, if you want to upgrade to the latest version immediately, you can follow the two simple steps below.

1. Check the About... dialog;
2. If the Built Date is older than 2017/06/01:

Close your browser open EasyEDA again.

If the Built Date is still showing older than 2017/06/01:

Close your browser and open EasyEDA again.

If the Built Date is at or newer than 2017/06/01, you don't need to do anything.

Note: 2017/06/01 *is just an example.*

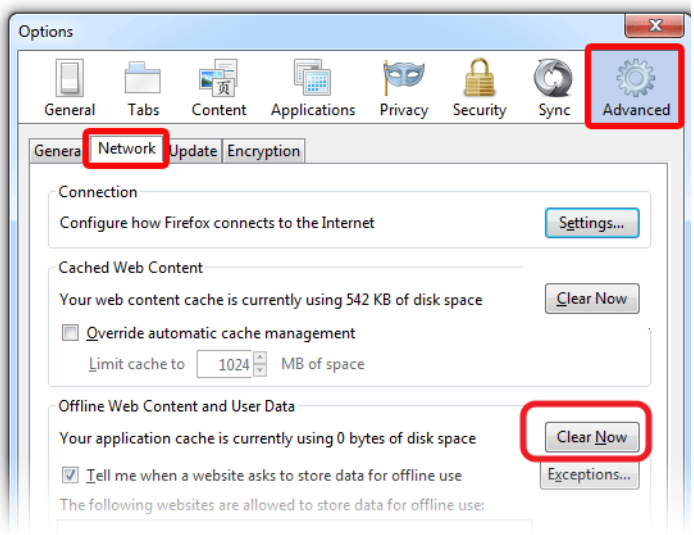
If those two steps don't work, you may need to clear your browser's cache:

1.Mozilla Firefox

- Go to "Preferences... > Advanced > Network > Offline Storage",

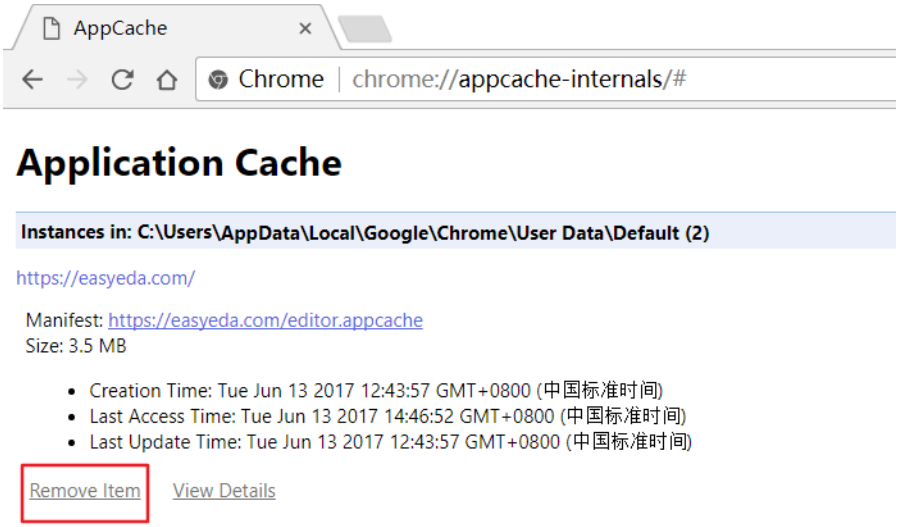
-Click on "Clear now",

-Reload easyeda again.

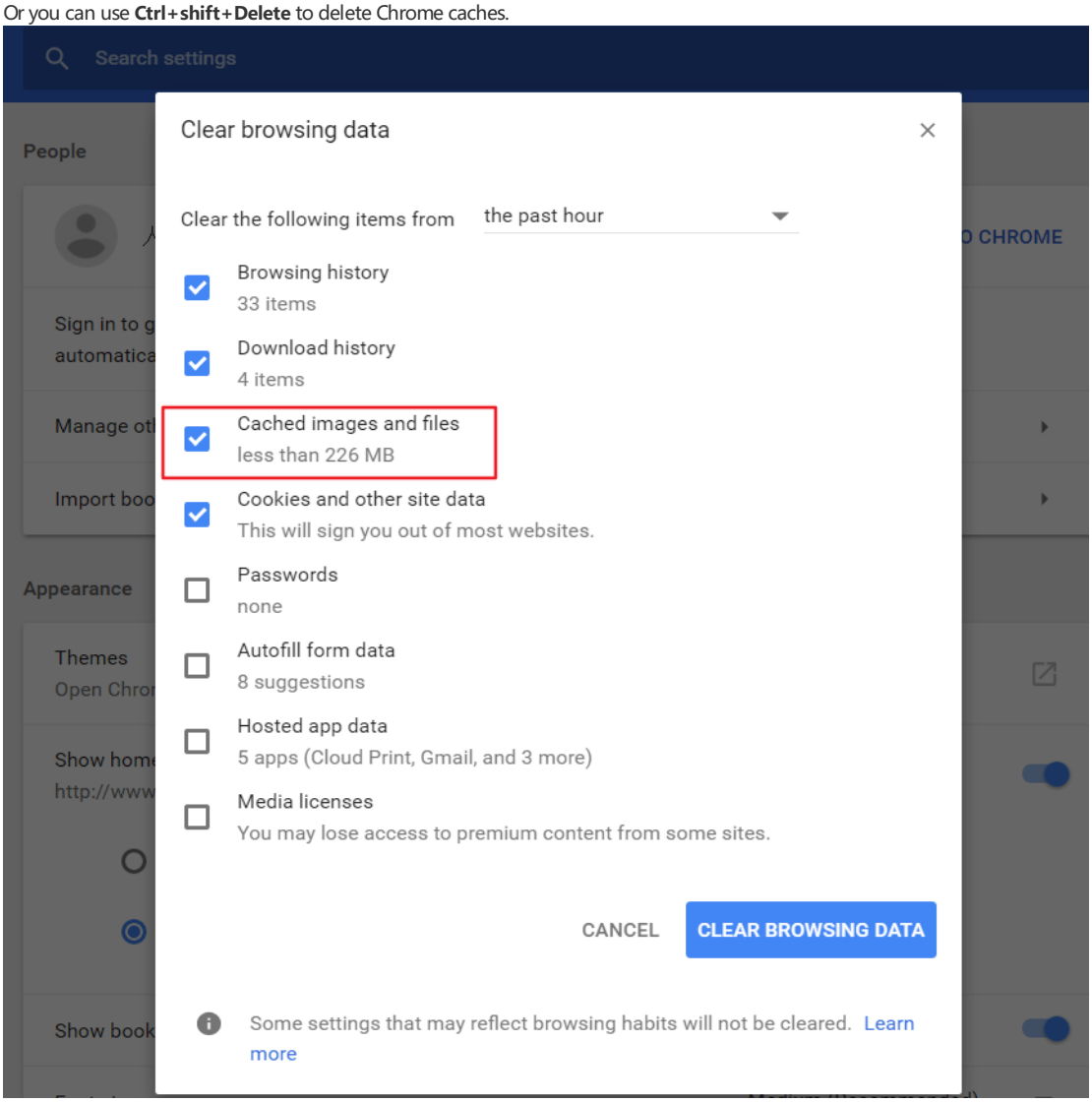


2.Chrome

- Open the following URL: **chrome://appcache-internals/**
- Look for easyeda.com and click "Remove"
- reload easyeda again.

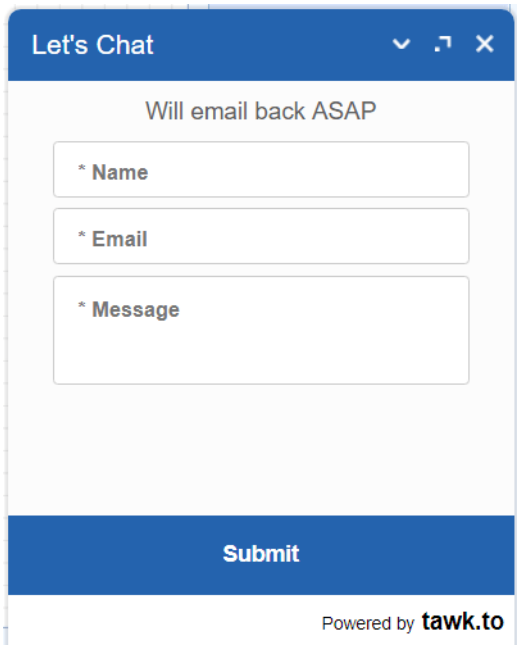


- Or you can use **Ctrl+shift+Delete** to delete Chrome caches.



How to get help

It is easy to ask for help for any problem with EasyEDA, just click on **Let's Chat**, and then complete and Submit the Support request:



Please ask your questions in English or Chinese and don't worry if your English is not good! (Or your Chinese!)

1. You can also ask your questions directly in the EasyEDA forum. We will try to respond to every post but please be patient. Maybe EasyEDA team is in a different timezone and we are a bit busy, so you may need to wait for a while.
2. If you don't want your help requests to be public then you can drop us an email to support@easyeda.com
3. If maybe you have a design that you know worked in some other EDA package and you are having problems importing it to EasyEDA, let us know and we will take a look and try to help you to fix them.

Please note that:
EasyEDA team may not have the time or resources to help you fix all your problems; we may just be able to help you to fix problems commonly encountered by newbies, such as using a drawing polyline in place of a wire, finding a spice model for a simulation or selecting the right PCB footprint.

[1] *Please note that although some browsers or plug-ins allow you to use gestures, EasyEDA does not work with gestures, so you should disable this function.*

[2] *Simultaneous editing is not yet fully supported: care must be taken because the last save by any collaborator overwrites all previous saves.*

[3] *It can also find the value text but it cannot step through multiple components with the same value.*

[4] *Take a few moments to think about your username because this is the name that other users will see on your designs and posts if you choose to share them or make them public. Once you have created an account, you cannot change your username.*

[5] *You can use upper and lower case letters, numbers and symbols to make a strong password but don't forget that the password entry is case sensitive.*

[6] *Except ordering of PCBs directly from EasyEDA.*

[7] *If you always open EasyEDA in the same browser on the same machine, your Anonymous files will appear under the Anonymous Files folder in the left hand panel but you should not rely on this as a way of keeping track of Anonymous files.*

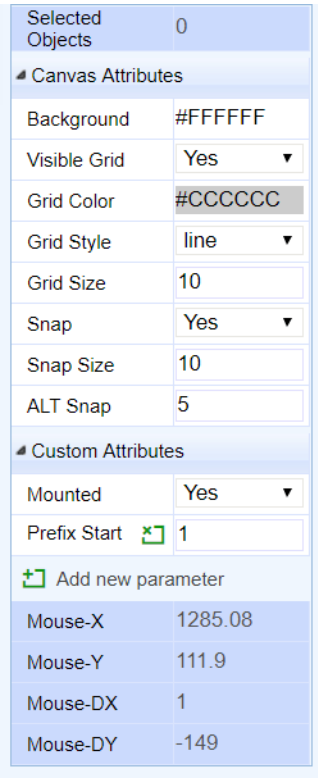
Please email support@easyeda.com when you need any help.

Creating The Schematic

During this tutorial we will to guide you in using EasyEDA Schematic capture.

Canvas Settings

You can find the canvas Properties setting by clicking on any the blank space in the canvas.



As described earlier, background and grid colours and the style, size, visibility and snap **attributes** of the grid can all be configured.

The canvas area can be set directly by the Width and Height or by using the available preset frame sizes.

Grid

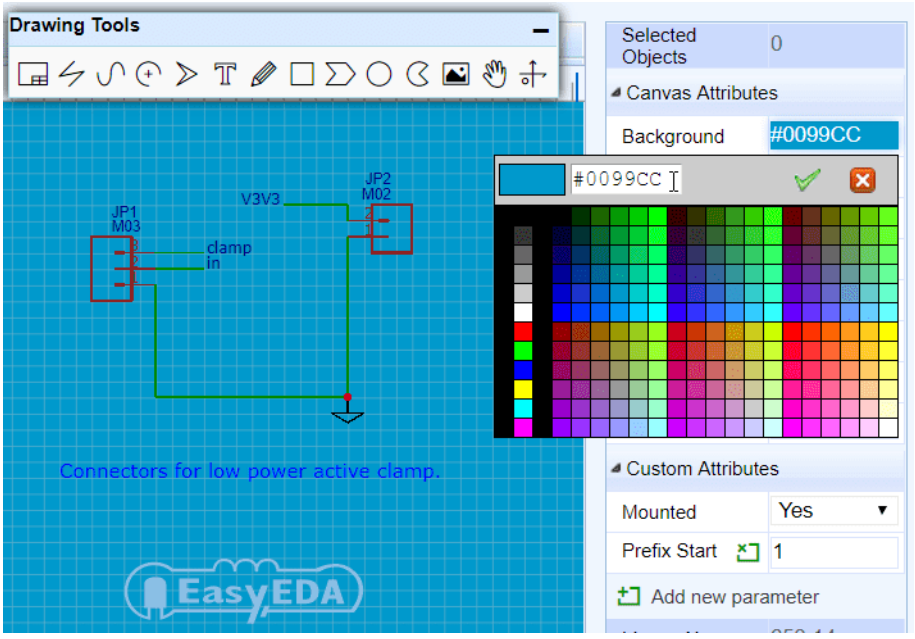
Visible Grid : Yes or No

Grid Color: Any valid colour

Grid Style: Line or Dot

Grid Size: To ensure proper alignment of all EasyEDA parts, it is advisable to set in 10, 20, 100.

Grid (and background) colour can be set directly by entering the hexadecimal value of the colour you want or by clicking on a colour in the palette that opens when you click on the colour value box:



Snap

Snap: Yes or No. The default hotkey is G. Pressing this key toggles switching snap to grid on and off.

Snap Size: To ensure proper alignment of all EasyEDA parts, it is advisable to set in 10, 20, 100 but any valid number can work, such as 0.1, 1, 5.

It is strongly recommended that you keep **Snap = Yes** all the time. Once items are placed off-grid it can be very difficult to reset them back onto the grid. Off-grid placement can result in wires looking as though they are joined when in fact they are not and so causing netlisting errors that can be hard to track down.

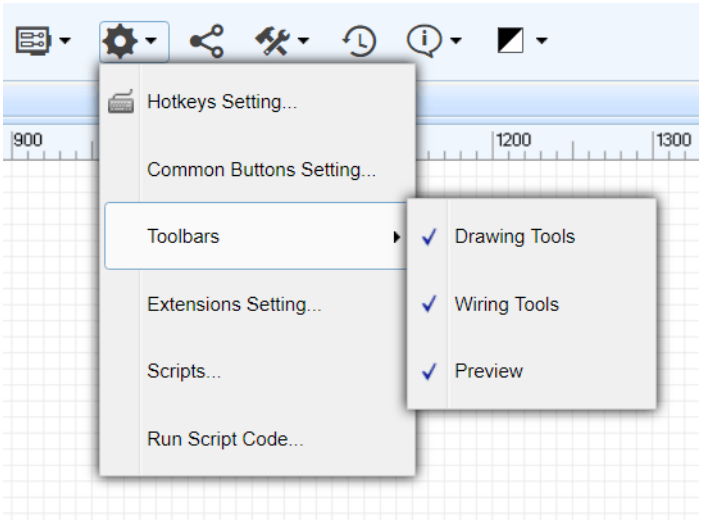
If you need to draw detailed parts of new symbols or footprints that need to go between grid points, try to reduce the grid spacing to draw these elements and then reset the grid back to your chosen default value as soon as you have completed that part of the drawing. Setting

Snap=No should only really be used as a last resort.

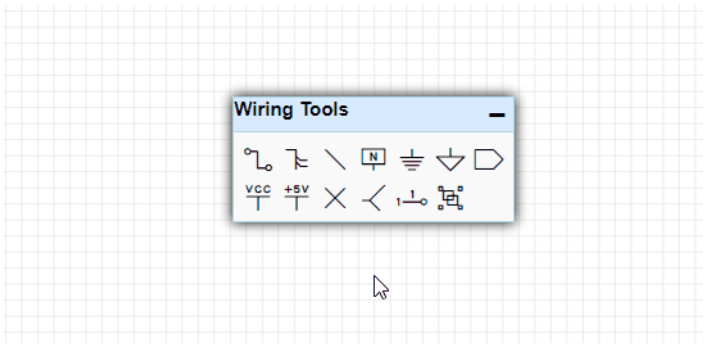
ALT Sanp: Snap size when pressing the `ALT` key.

Wiring Tools

If you have hidden your tools , you can open them from here: Top toolbar **Config Gear Icon** > **Toolbars** > **...**



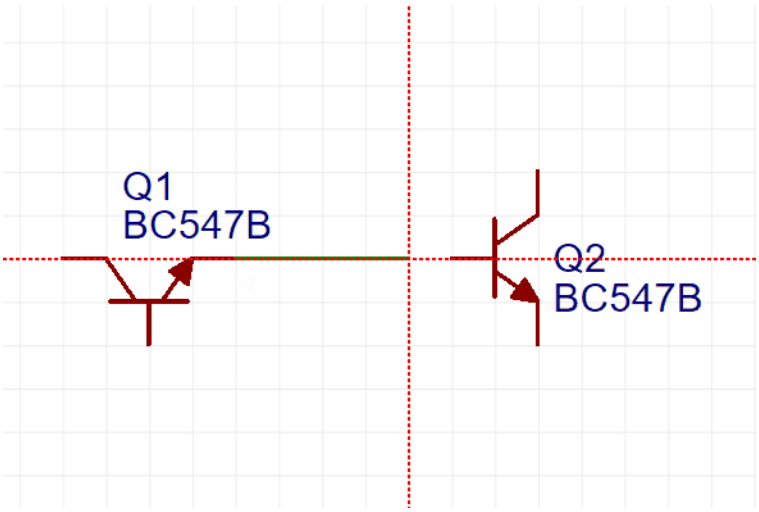
Note: *All of the commands in Wiring Tools are electronics related. Don't use a wire when you just need to draw a line, shape or an arrow: use Drawing Tools instead.*



Wire

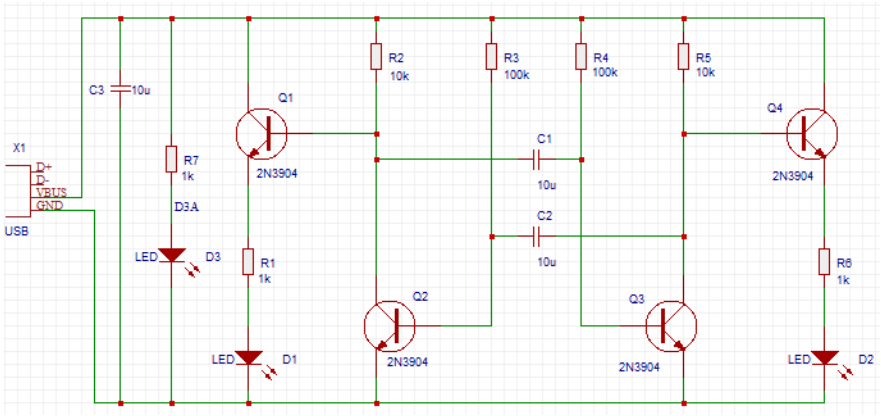
There are three ways to enter the wire mode in EasyEDA.

1. Click the **Wire** button from the **Wiring Tools** palette.
2. Press the `W` hotkey.
3. Click on the end of a component pin (where the grey pin dot appears if you select the component):



EasyEDA automatically enters **Wire** mode.

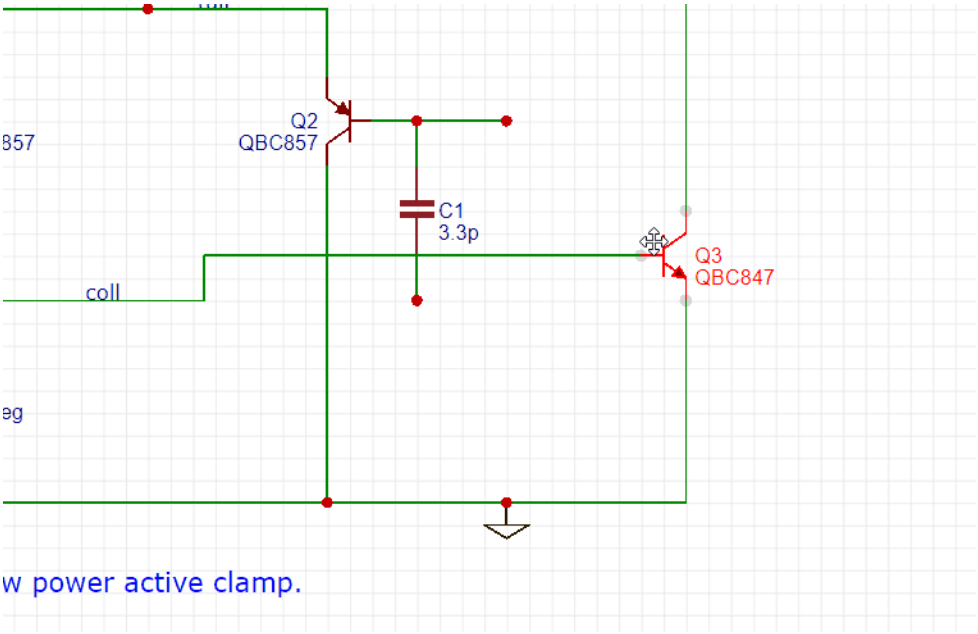
Here is a screenshot of the **Astable Multivibrator LED project schematic** after wiring:



Moving Components And Wires:

If you place a component, such as a resistor, on top of a wire then the wire breaks and reconnects to the ends of the component.

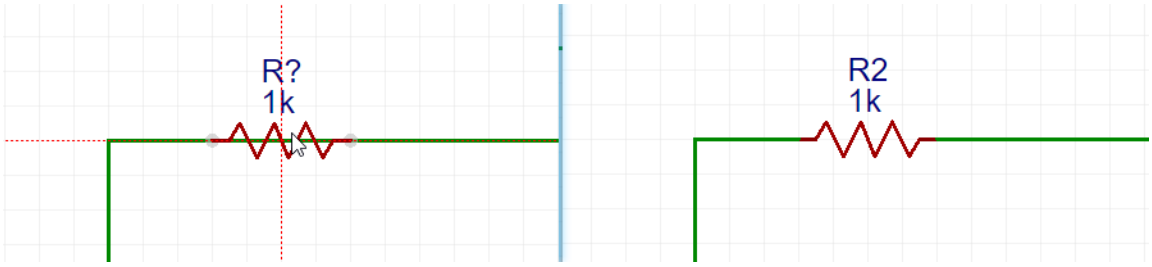
When moving selected components using the mouse, they will drag attached wires with them ("rubber band") to some extent but please be aware that the rubber banding feature has some limitations. When moving selected components most wire will move vertically and horizontally. Using the arrow keys will not rubber band. Selected wires do not rubber band.



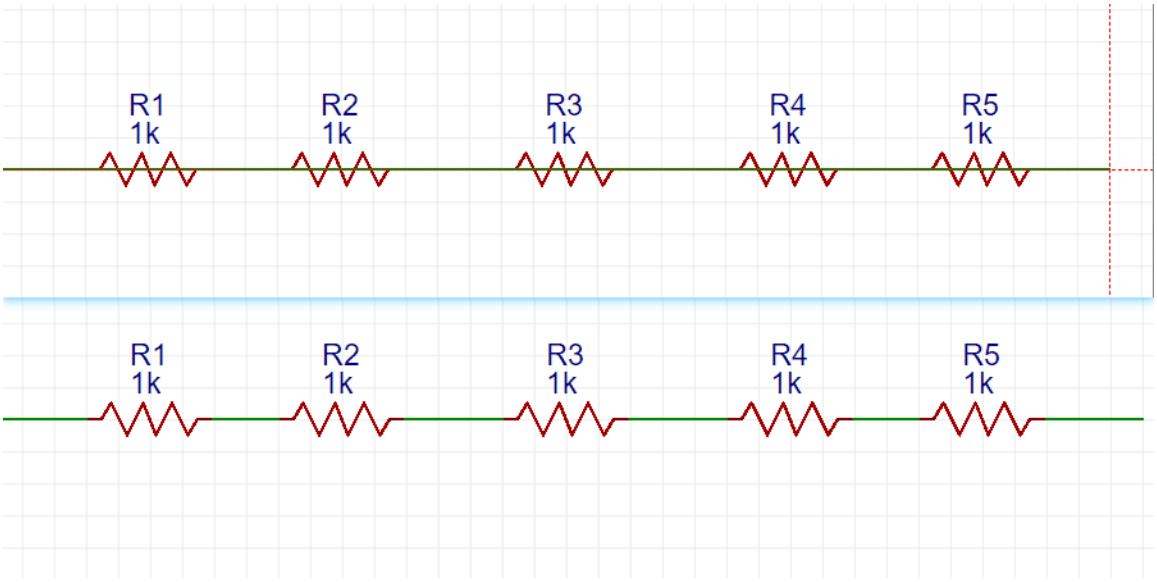
A selected wire can be moved directly by clicking on it using the mouse or by the arrow keys. If a wire is selected by clicking on it using the mouse then green grab handles will appear at the ends and vertices.

Auto adjust connection

If you put a resistor or capacitor on a wire, the wire will auto connect the pins as below:

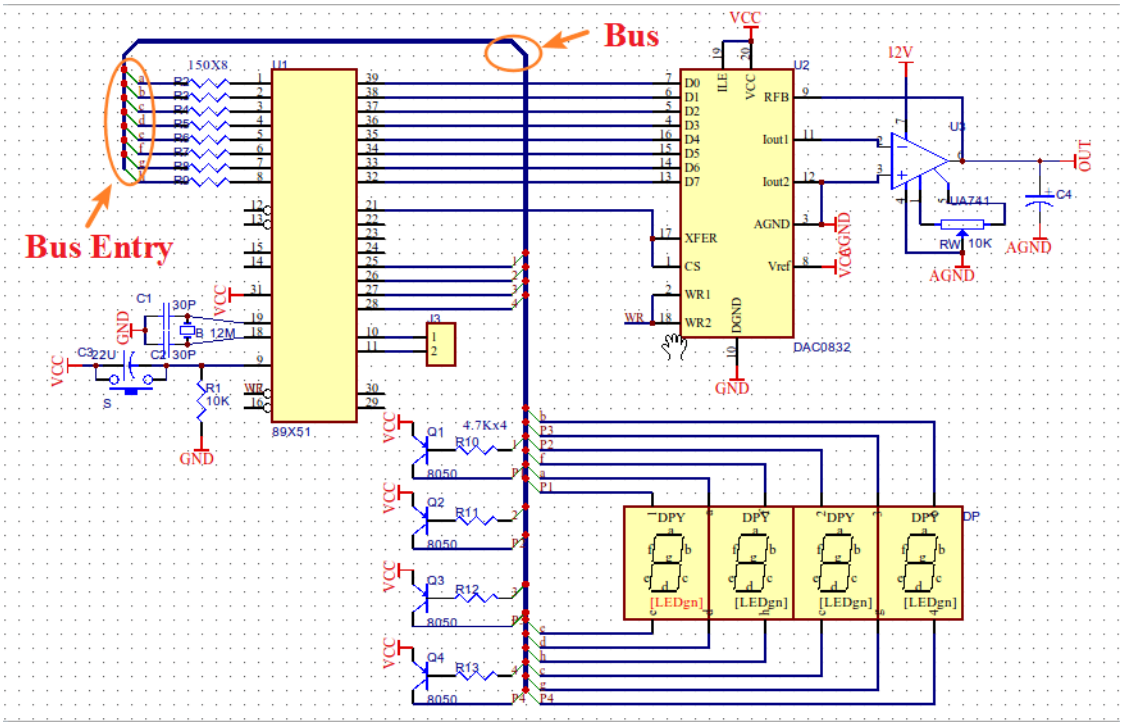


When you want to wiring a series of resistors which are in a row, you can just wire through them, and then you will find they all be connected.



Bus

When you design a professional schematic, perhaps it will use a lot of wires. If you're wiring one by one, much time would be wasted, and then you need to use `Bus`.



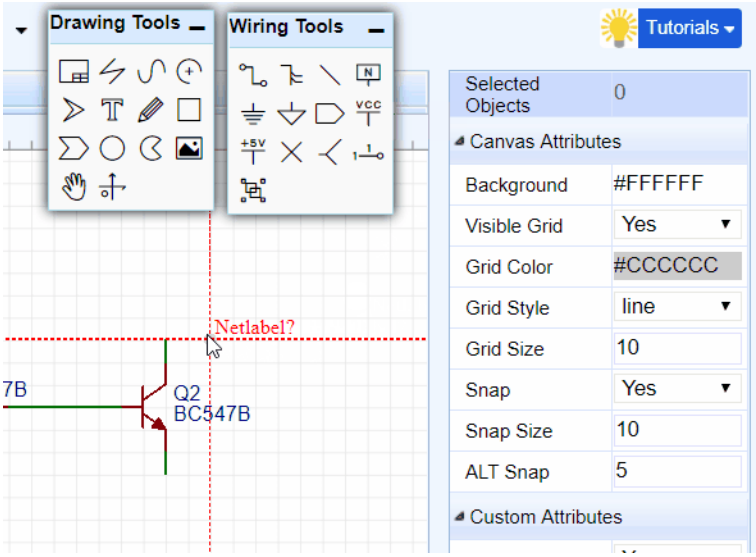
Bus Entry

If you decide to wire with `Bus` , the `Bus Entry` must connect to Bus and other nets with wires. such as in the above image.

Net Label

NetLabel and NetFlag

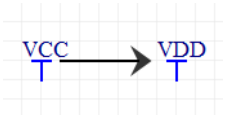
NetLabel can be used to give your wires names to help you find them and identify any misconnections. You can find the **NetLabel** from the Wiring Tools palette or by using the `N` hotkey. When selecting the netlabel, you will find its attributes in the right hand Properties panel:



You can change its name and colour. If you only want to change its name, it may be easier to just double click the netlabel.

Net Flag

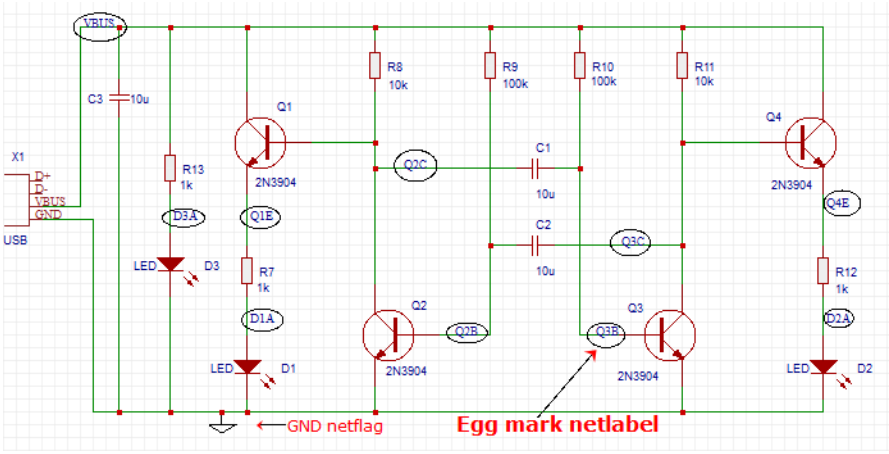
NetFlag is the same as NetLabel, you can find the NetFlag from the Wiring Tools palette or using the `Ctrl+G` hotkeys for **GND** or `Ctrl+Q` for **VCC**. You can also change its name, for example from **VCC** to **VDD**:



The screenshot below is after adding NetLabels

- indicated by the little **egg marks**
  -and a **GND** NetFlag

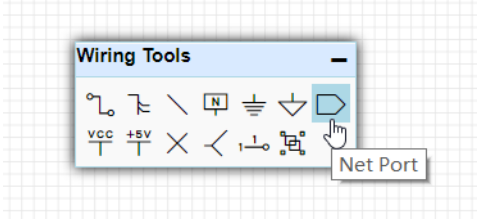
This schematic is almost finished.



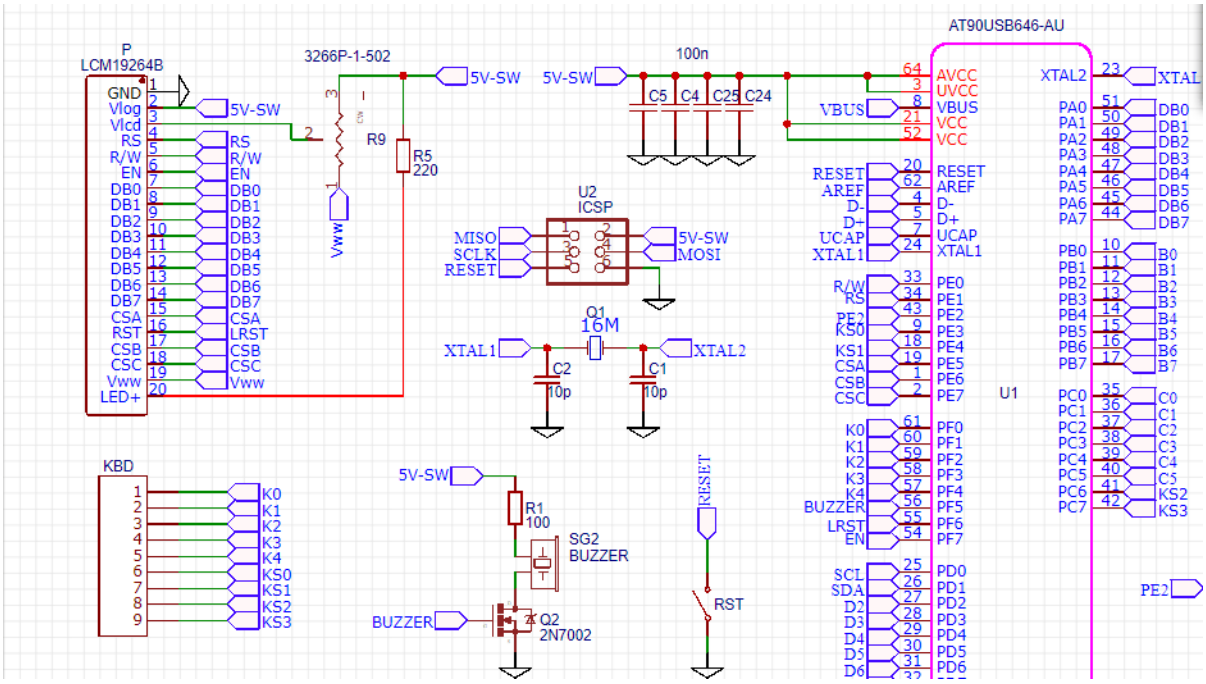
Wiring Tools palette provides NetFlag: Digital GND, Analog GND, VCC and +5V for your convenience.

Net Port

When you don't want to route too many wires, how about trying `Net Port` :

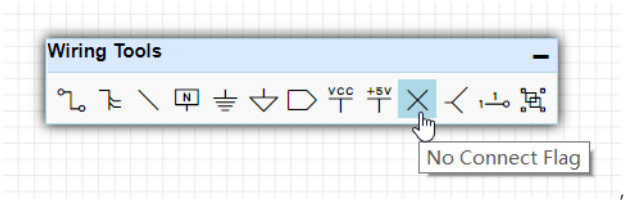


It will make your schematic look more clean, and you just need to set each Net Port a net name.

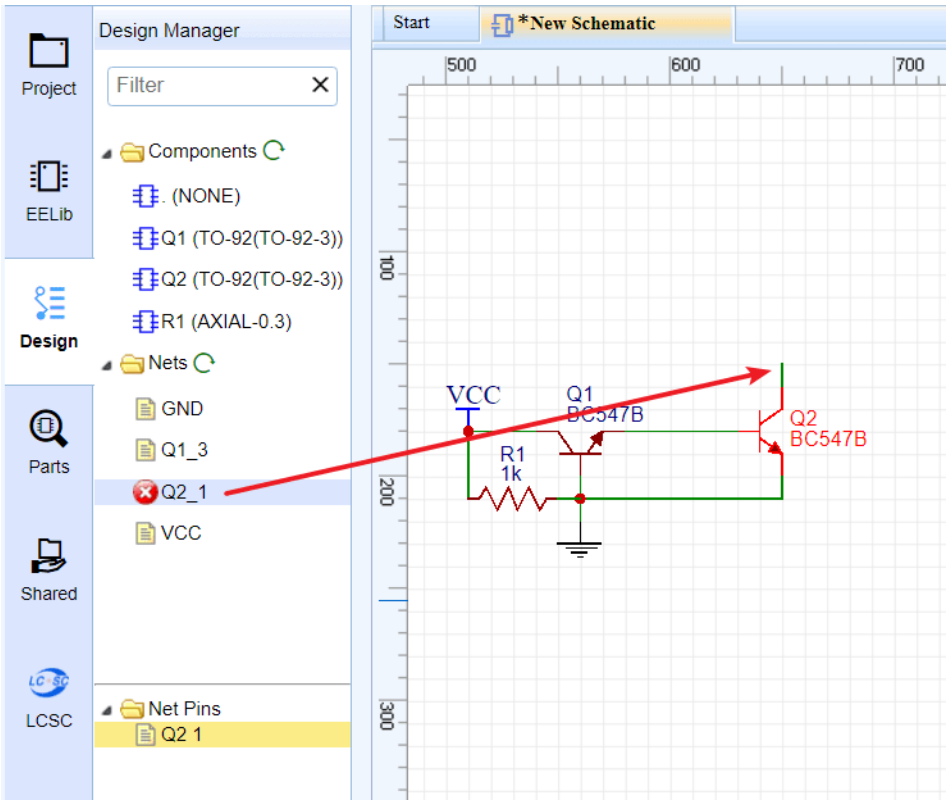


No Connect Flag

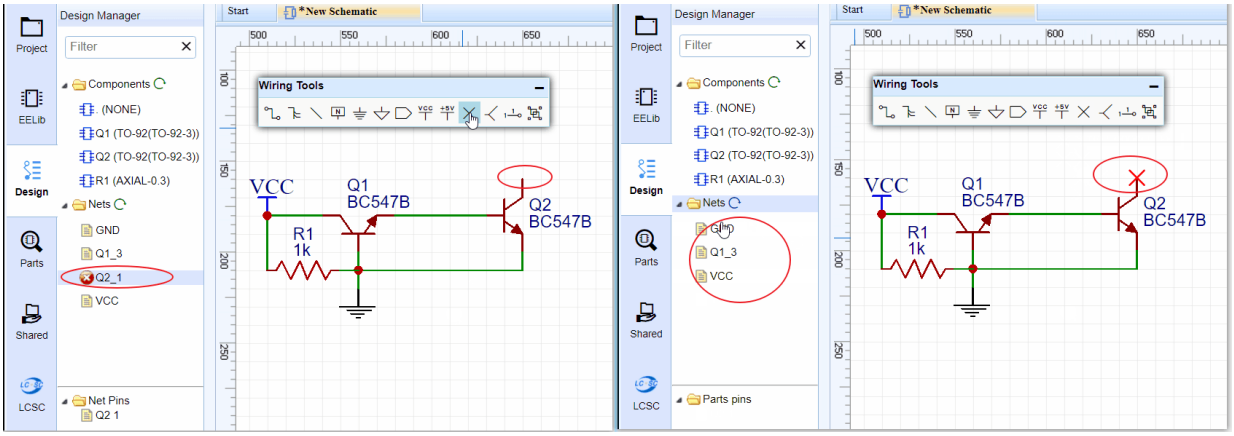
You can find the `NO Connect Flag` via wiring tool,



In the below schematic, if you don't add a `NO Connect Flag`, there is an error flag in the nets collection of the design manager.

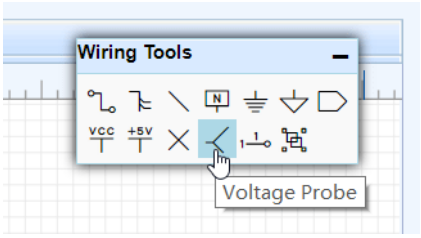


After adding a `NO Connect Flag`, the error disappears.

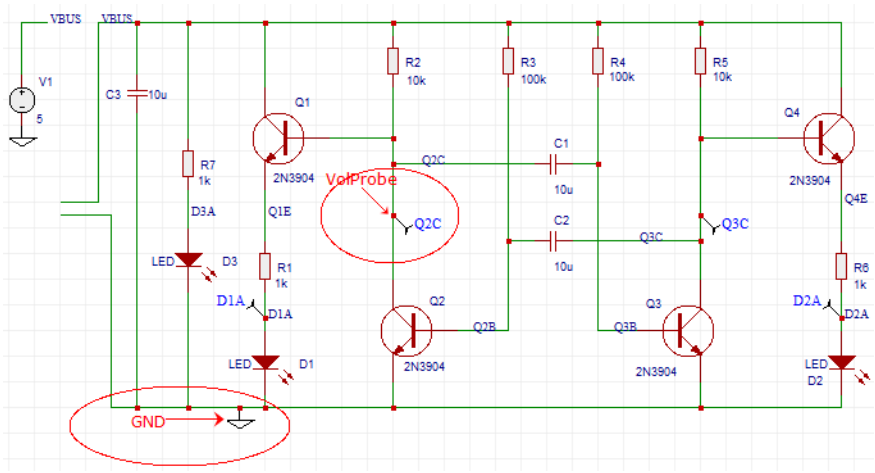


Note: `NO Connect Flag` *only works on the symbol's pin directly.*

Voltage Probe



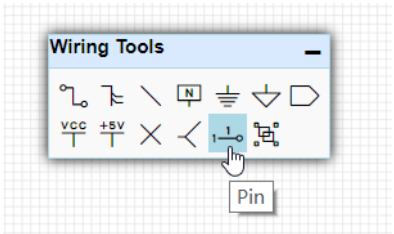
EasyEDA provides a simulation feature for the schematic. After the simulation is running, you will see the waveform where you placed the voltage probes in the circuit.



For more detail about the simulation, please check the Simulation section.

Pin

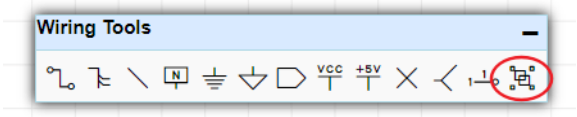
When you create a new symbol in schematic and schematic lib, you must use `Pin` to create pins for the new symbol, otherwise your symbol can't be wired with wires.



For more information please refer to the Schematic Lib: Pin section.

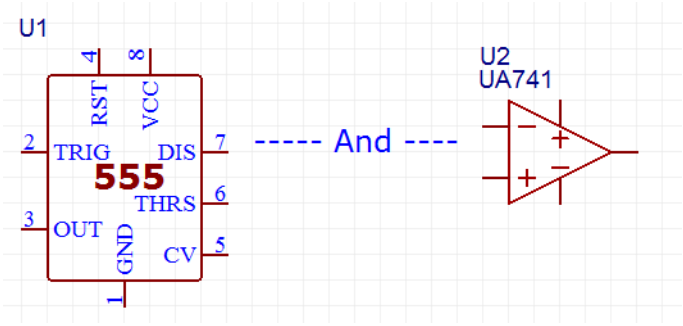
Group/Ungroup

On the **Wiring Tools** palette there is the **Group/Ungroup Symbol...** button.



Just like the **Symbol Wizard**, this tool is also for you to quickly create schematic library symbols.

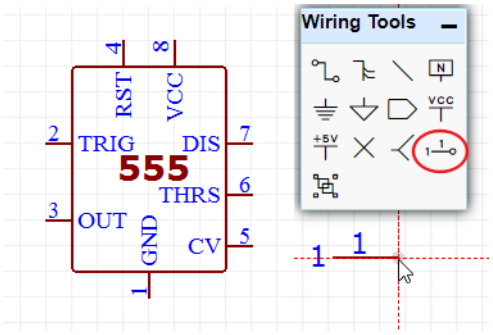
Using the **Symbol Wizard** you can only create generic symbols but how can you quickly and easily create symbols like these?



Here's how.

EasyEDA allows you to do something that very few other EAD tools support.

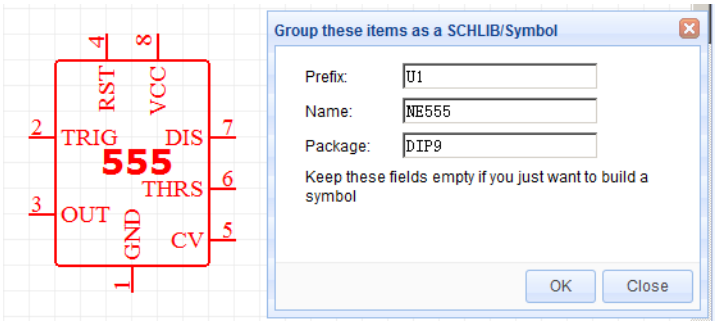
Using the P Hotkey, you are allowed to add a PIN directly onto the Schematic canvas. So you can add 8 Pins, draw a rectangle from the Drawing Tools palette and add 555 as text to form a symbol for the NE555 like the one shown below:



Now comes the clever bit.

Up to this point you have a collection of separate pins, a drawn rectangle and some text that are all separate items with no particular association with each other.

So now select all of the items and click the Group/Ungroup Symbol... button. A dialog will be opened:



After you click OK, all those separate elements will be grouped together to form your new symbol directly in the schematic.

Using the group function, you can create any symbol in the schematic, easily and quickly.

How cool is that?

So what does Ungroup do? Try selecting a symbol and then click the Group/ungroup command to see what happens!

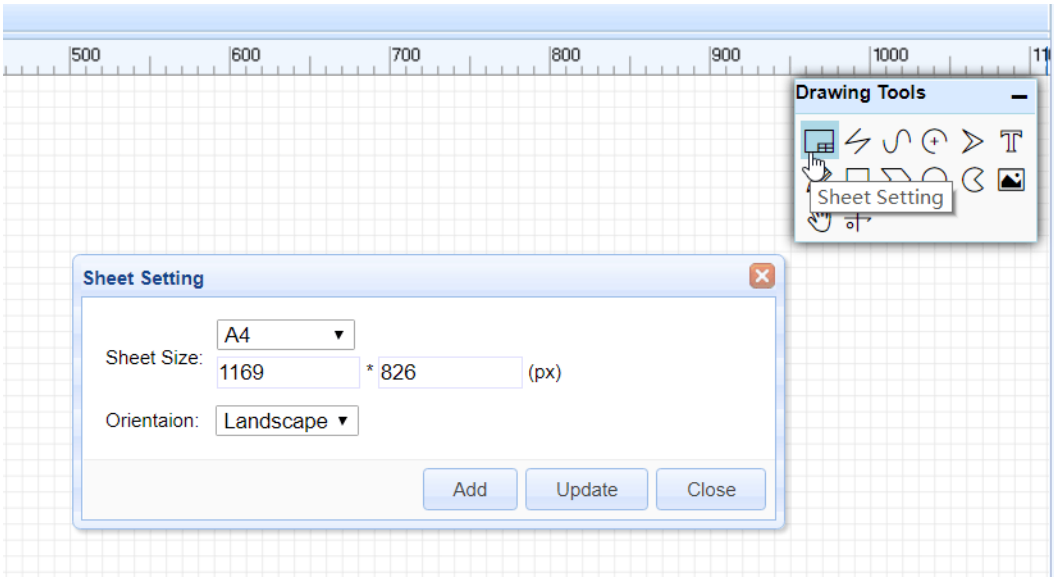
Note: *The pin numbers and names cannot be moved independently of the pin.*

Drawing Tools

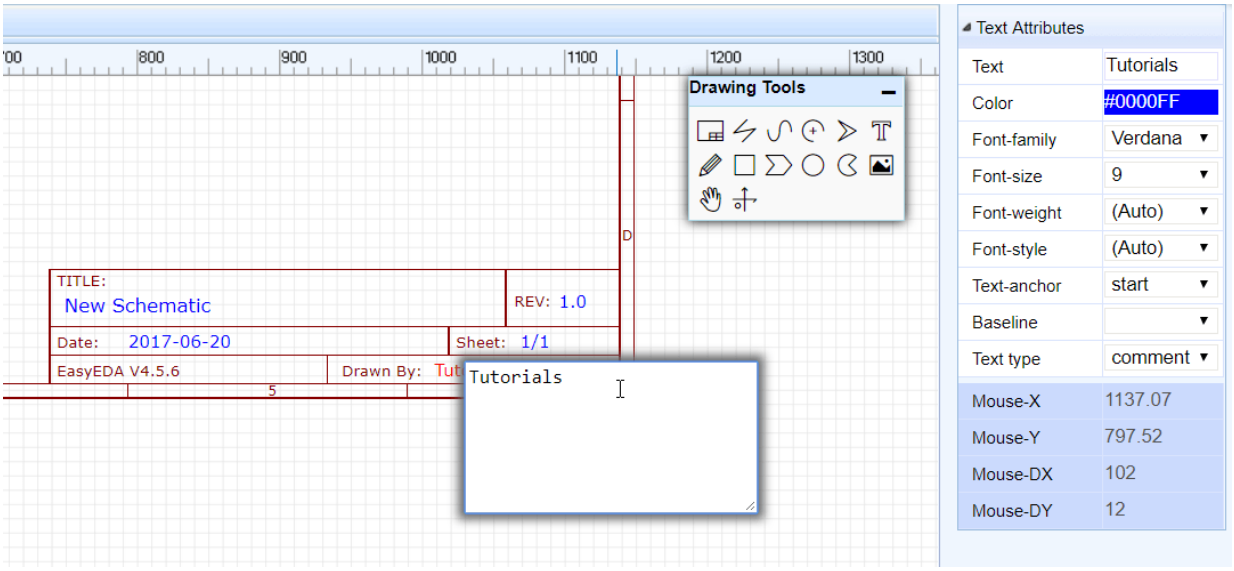
Sheet Setting

It is now possible to add design notes to the frame and the frame selection, for example A4, which can assist in aligning and improve the look of printed schematics and PCB designs.

Click the frame button like in the image below, Or via: **Super Menu > Miscellaneous > Sheet Setting**



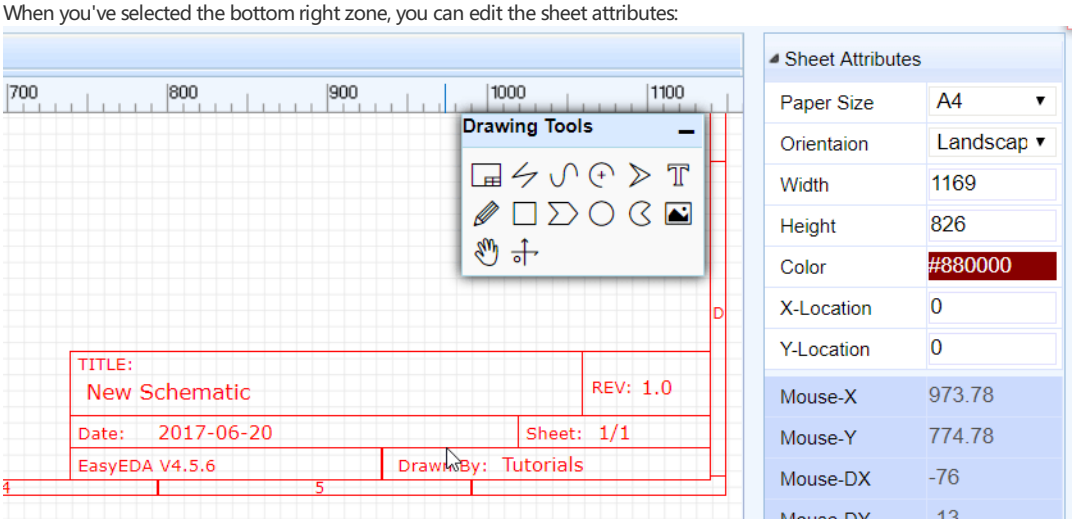
As illustrated in the image below:



And you can edit the blue text when you've selected the text attributes or double clicked it.

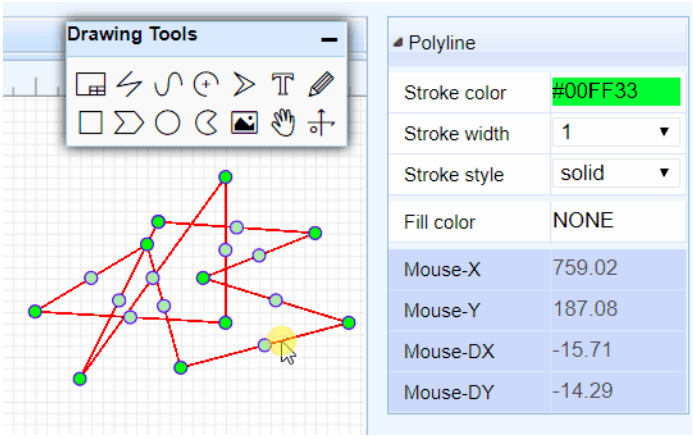
The bottom right zone can be selected and dragged or the frame can be dragged and deleted.

When you've selected the bottom right zone, you can edit the sheet attributes:



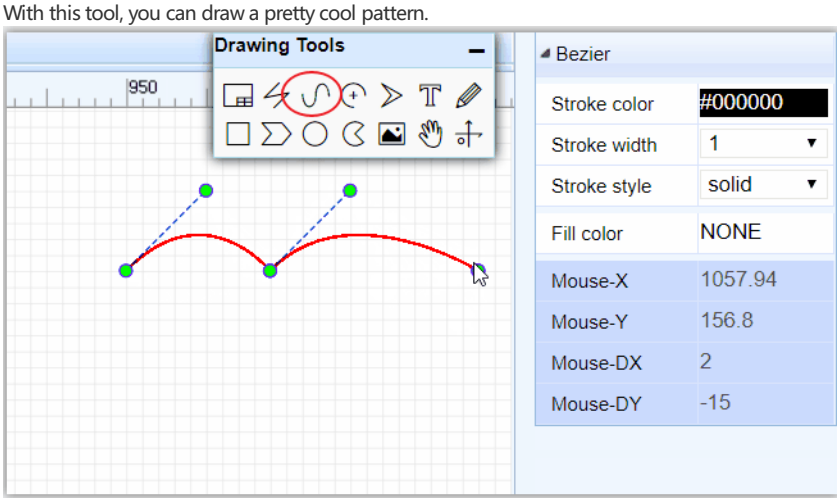
Line

In the Schematic editor, you can draw a line with any direction. You can change its attribute as in the image below:



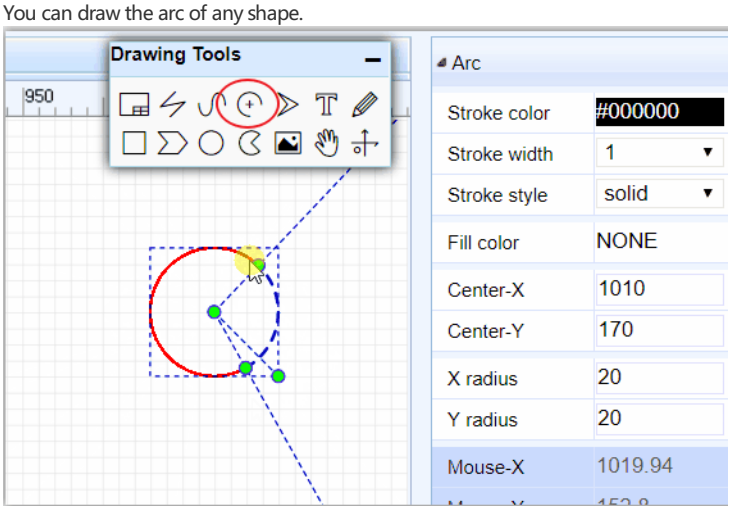
Bezier

With this tool, you can draw a pretty cool pattern.



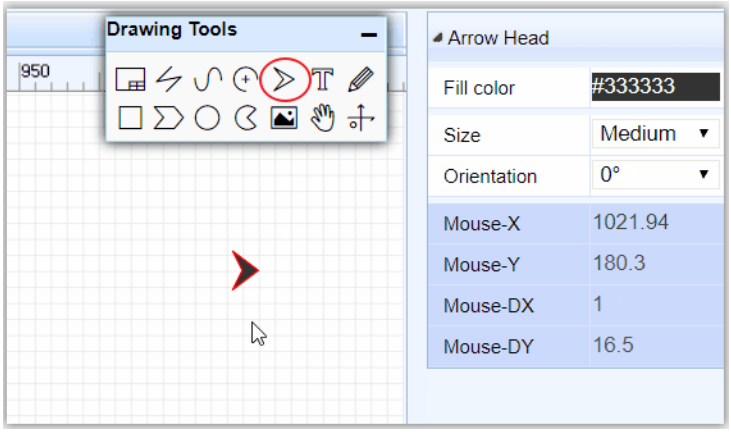
Arc

You can draw the arc of any shape.



Arrow Head

You can add arrow head to marking text or important part.

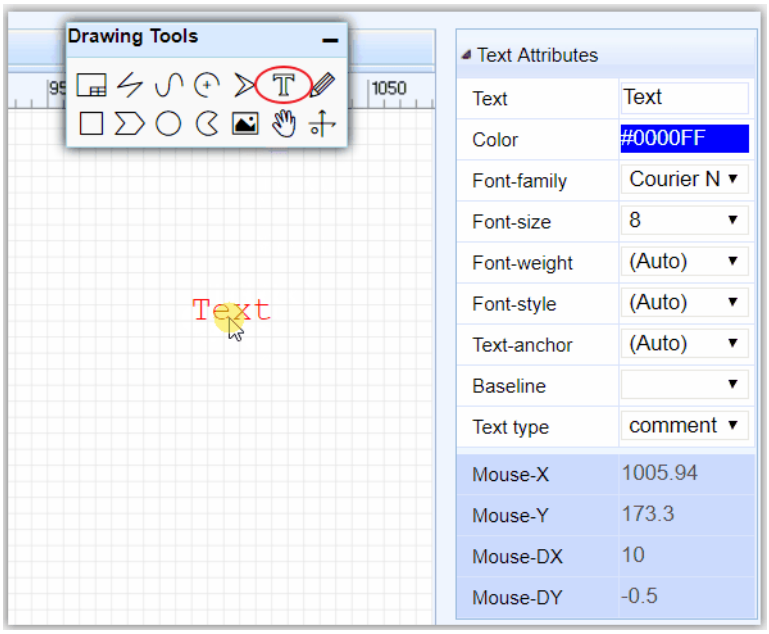


Text

Text attributes provide many parameters for setting:

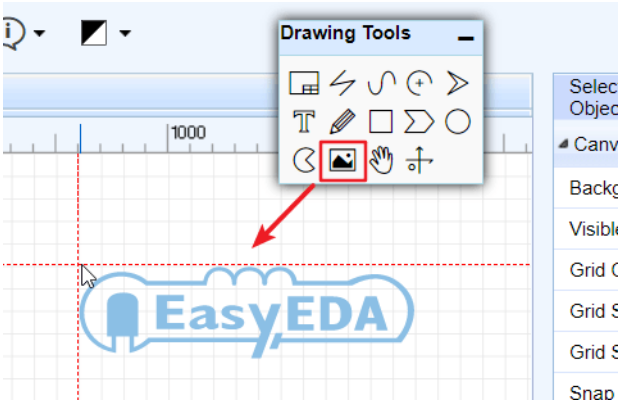
- **Text**: You can change text in inner box or double click the text. For every new text, the default text is `Text`. **-Color: Defines text color. -Font-family**: It provides 12 fonts for choosing. **-Font-Size**: Defines Text size. **-Font-weight: Defines Text weight. -Font-Style: It contains (auto), normal, italic. -Text-anchor: It contains (auto), start, middle, end, inherit. -Baseline: It contains (auto), use-script, no-change, reset-size ... and so on. -Text type**: types include comment and spice.

The editor will remember your last text parameters.

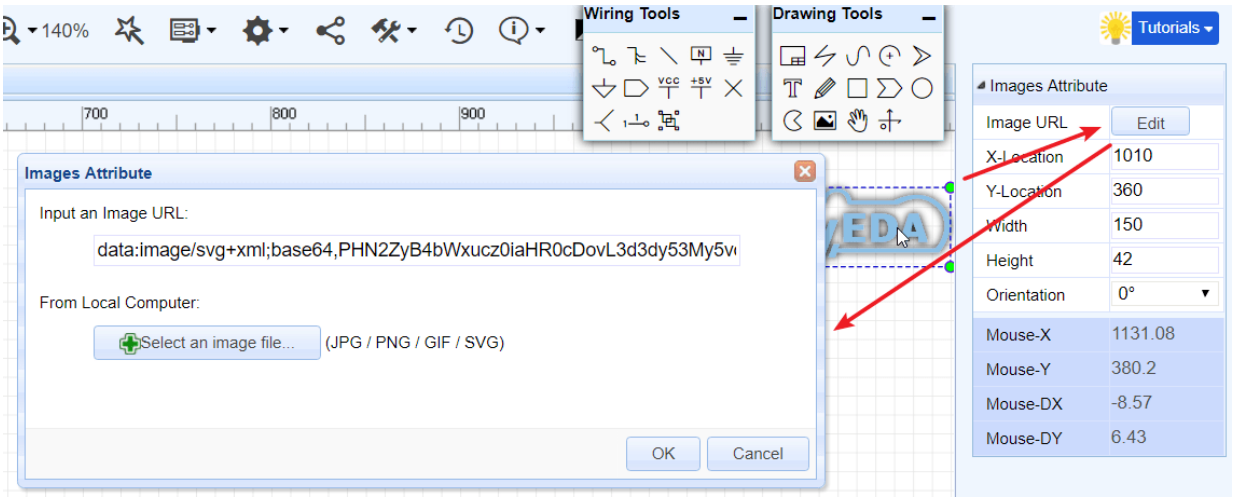


Image

When you select Image from the Drawing Tools palette, an image place holder will be inserted into the canvas:



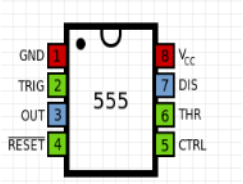
Select the place holder, so you can see the image's attributes in the right hand Properties panel:



Set the URL of your image. For example, setting the URL to:

http://upload.wikimedia.org/wikipedia/commons/thumb/c/c7/555_Pinout.svg/220px-555_Pinout.svg.png

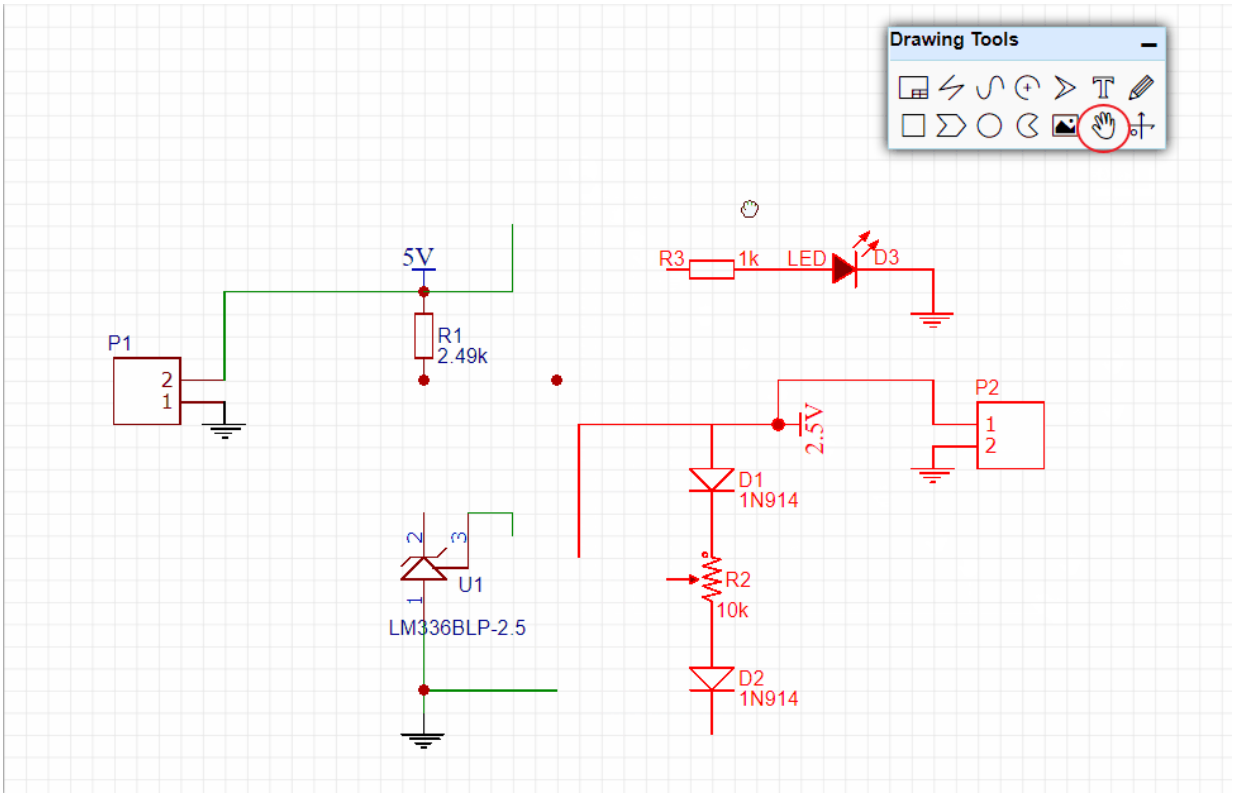
will make your image look like this:



Please note: at present, EasyEDA cannot host images, so you need to upload your images to an image sharing site such as http://www.imgur.com.

Drag

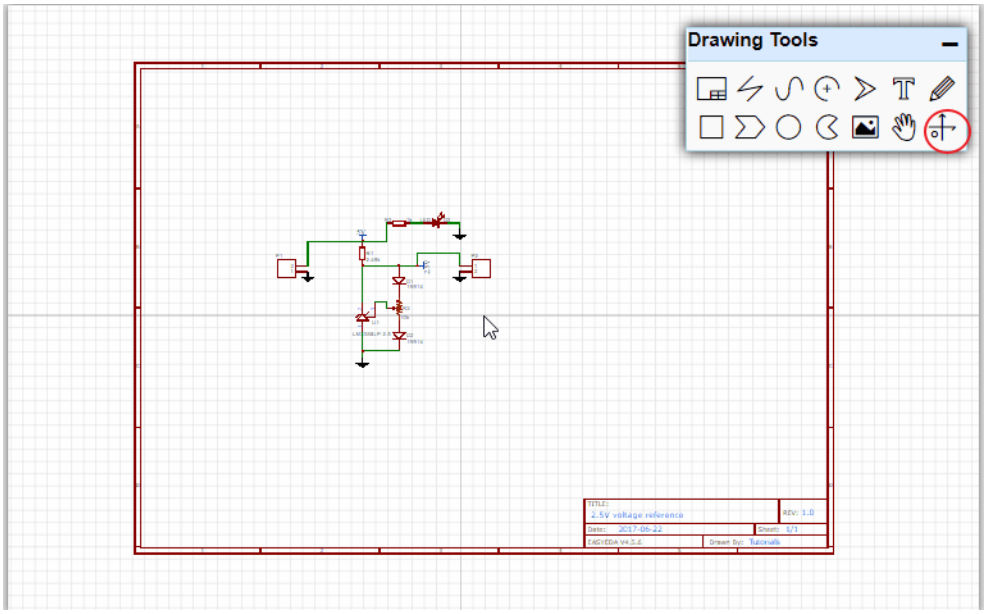
If you want to move some kind of parts and wires, you can use drag.
Or you can select the parts and wires area first and move them.



Canvas Origin

Canvas origin default is set at left top corner of the schematic sheet, but you can set it where you want via Canvas Origin.

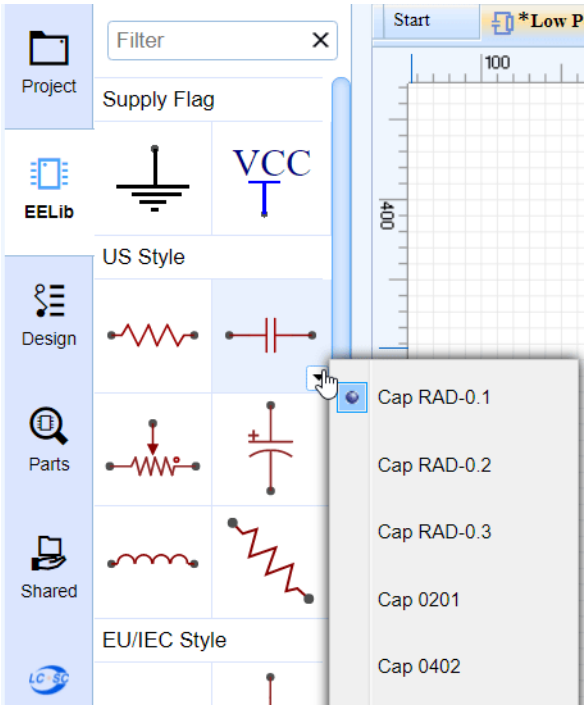
For another way to set canvas origin, you can try **Super Menu** > **Miscellaneous** > **Canvas Origin**.



Search symbols

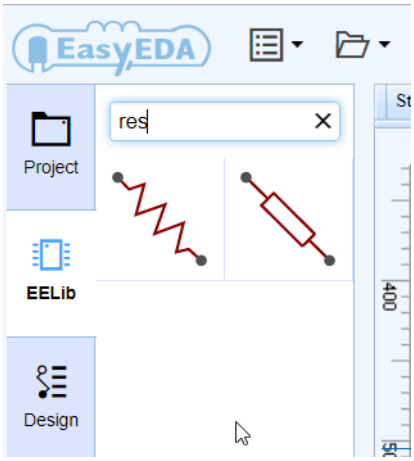
On the left hand Navigation panel you will find "**EElib**" and "**Parts**" ,

1) **EELibs** contains ready made symbols for a wide range of components and which can be simulated.

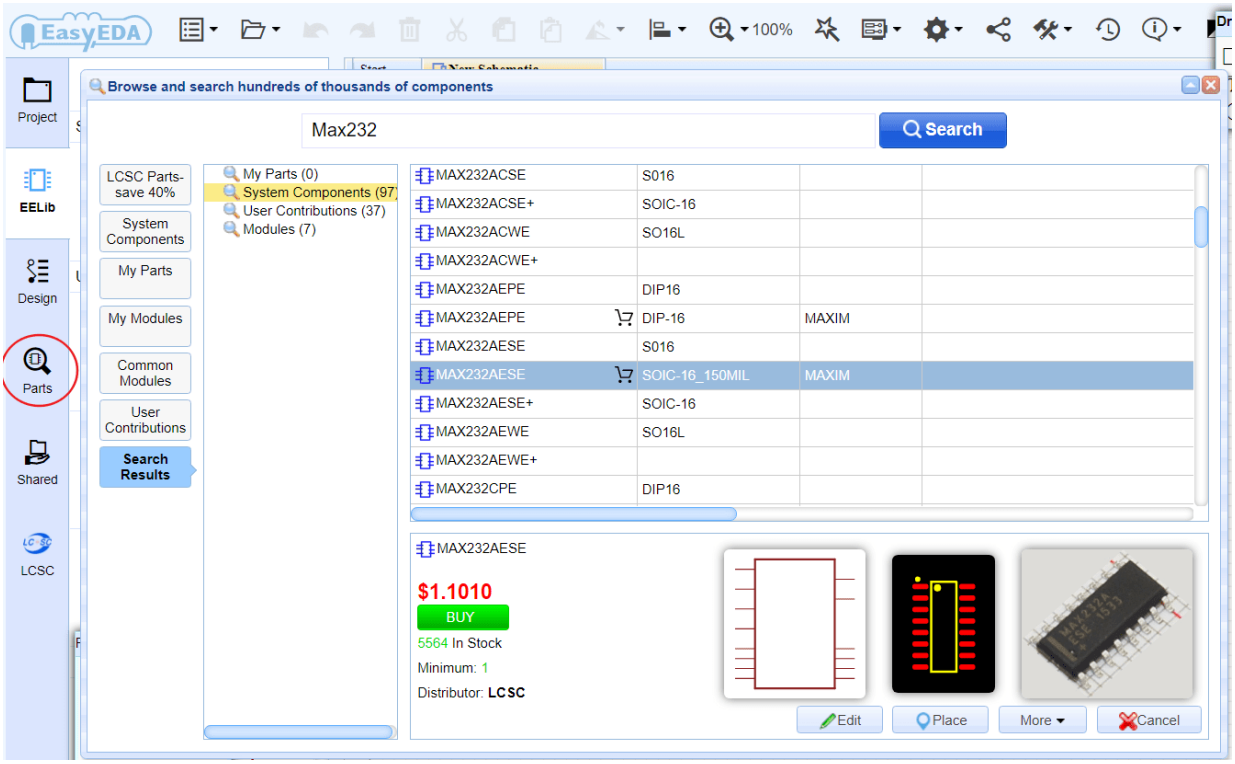


Many of these components have optional US and EU style symbols, we split them, so you can select those you like. Click on the drop down list or right click to popup the context menu, it contains many packages or parameters. EasyEDA will remember your choices for the next time.

Don't forget to use Filter to locate a component fast. For example, you just need to type `res` to find all of resistors:



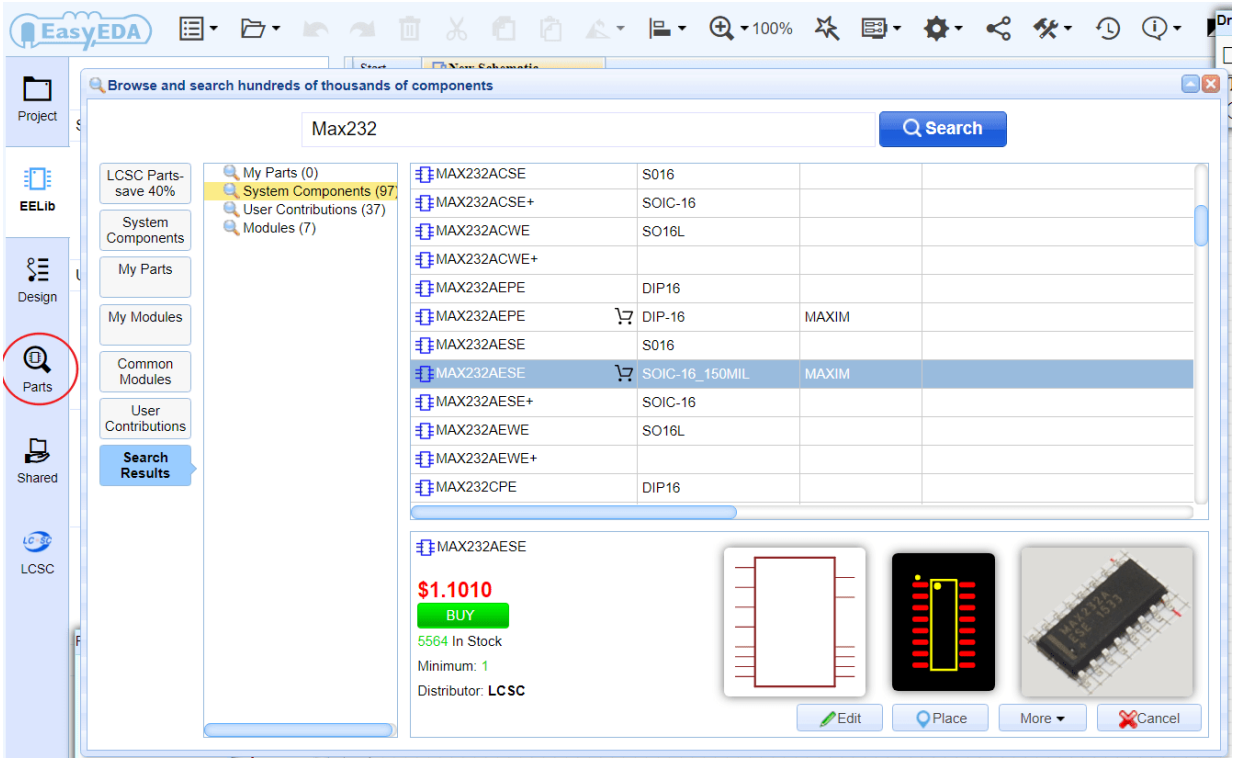
2) **Parts**, or press the hotkey combination `Shift+F`.
then you will see a dialog as shown in the image below. Simply type your part number or symbol's name to Search.



and then click the "Table of contents" to open the categories list to choose your components.

From there you can scroll up and down to browse parts from each category.

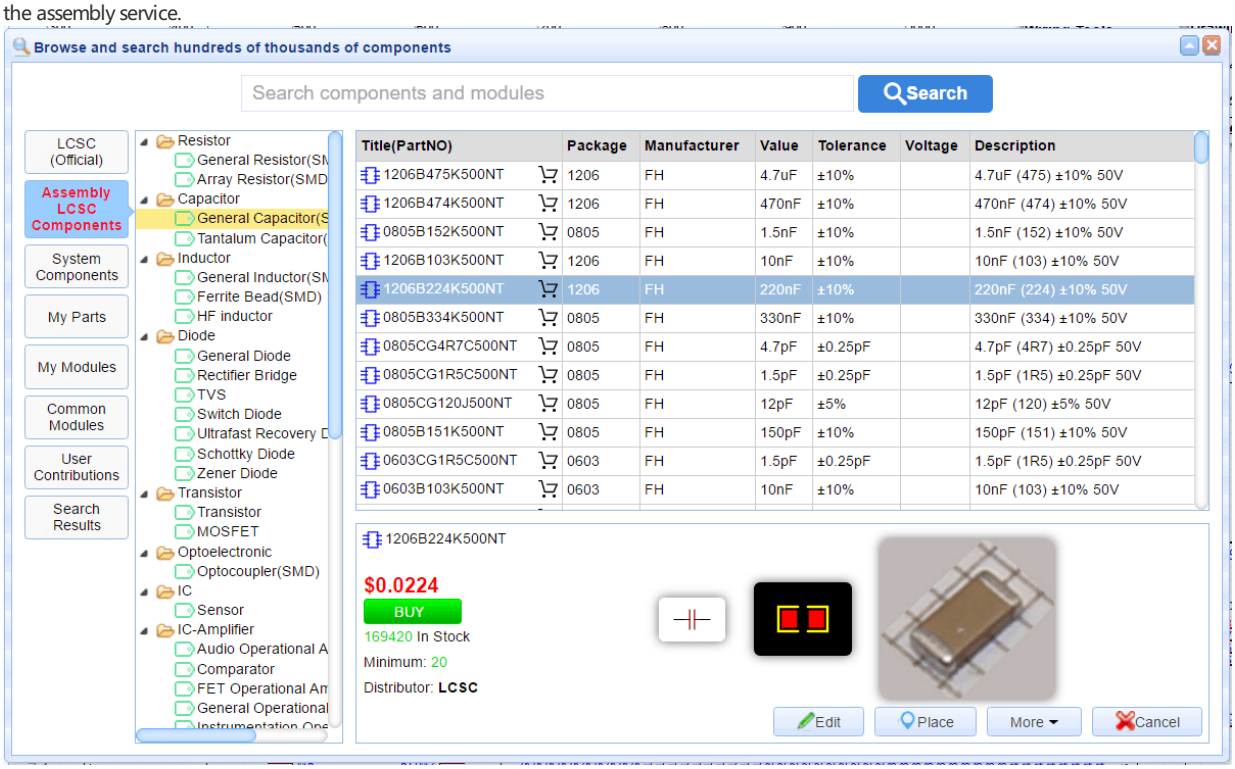
Suppose you wanted the find the **MAX232** (which converts signals from an RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits). Simply type **Max232** into the Search box and press Enter:



When you hover the mouse over the picture of the Schematic symbol or PCB footprint, you will find a toolbar with "Edit", "Place", "More" buttons.

LCSC Assembly Components

We add an LCSC Assembly Components option of the Parts, It's easy to choose which component can be assembled by LCSC. Yes, We provide the assembly service.



Place: For parts you use infrequently, you don't need to Favorite them; just Place it into your canvas directly.

Note:

- *EasyEDA supports multi-documents so please make sure that you are placing the part into the right (active) document. The active document is the one with the highlighted tab.*
- *You can't place a Schematic symbol into a PCB file, or a PCB Footprint into a schematic.*

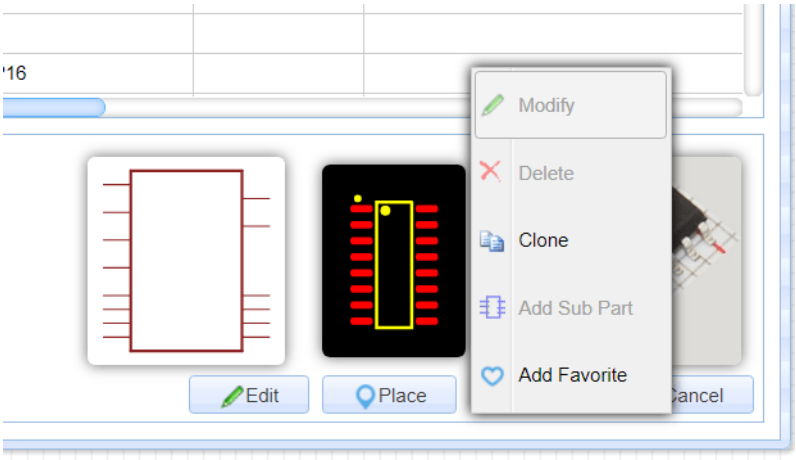
Edit: If you want to create your own version of a symbol or footprint then you can open an existing part from the library to use as a template, edit it and then save it to your local **My Parts** library in **Parts** of the Navigation Panel.

More: We can't promise that every component in the library is free of errors so please check all symbols and footprints carefully before you commit to a PCB order.

If you do find a mistake in a component, please let us know(mail to support@easyeda.com) so that we can fix it.

Components with sub parts (multi-device packages).

When you find a component with sub-parts, you can't Place or Edit it, but you can Favorite and Clone it as your own part, which you can then edit.

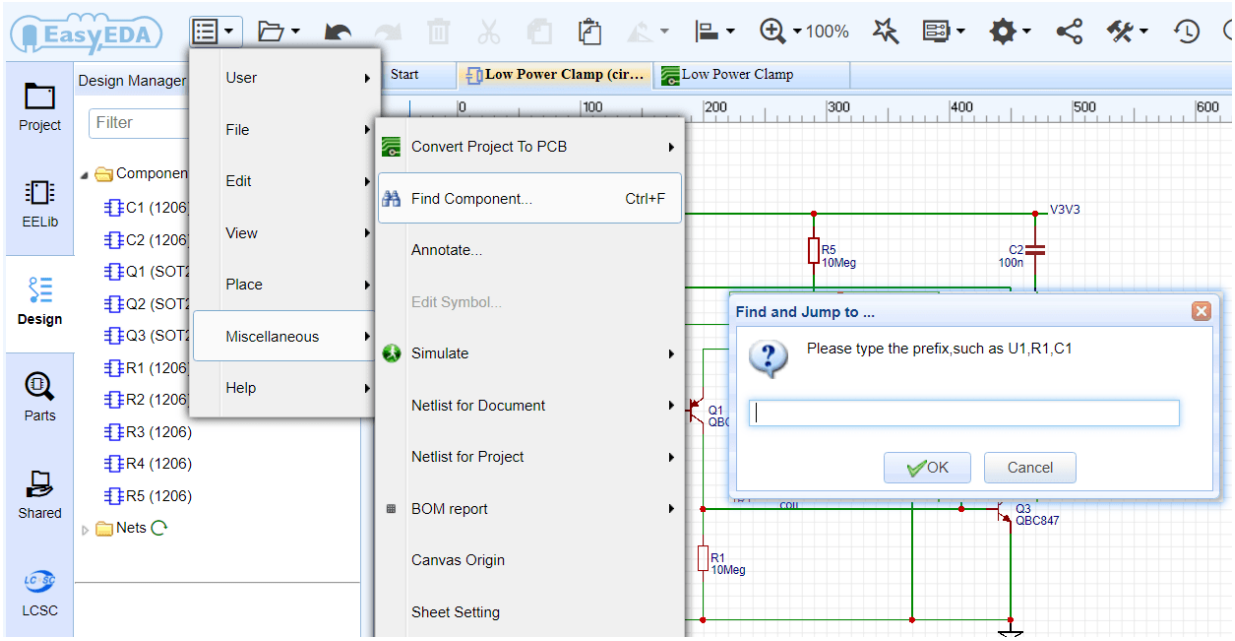


Find Components In The Schematic

Finding individual **components** in a dense schematic can be very time consuming. EasyEDA has an easy way to find and jump to components:

Super Menu > Miscellaneous > Find Component...

(or `Ctrl+F`)



Note: *You have to click OK in this dialog or use the Enter key.*

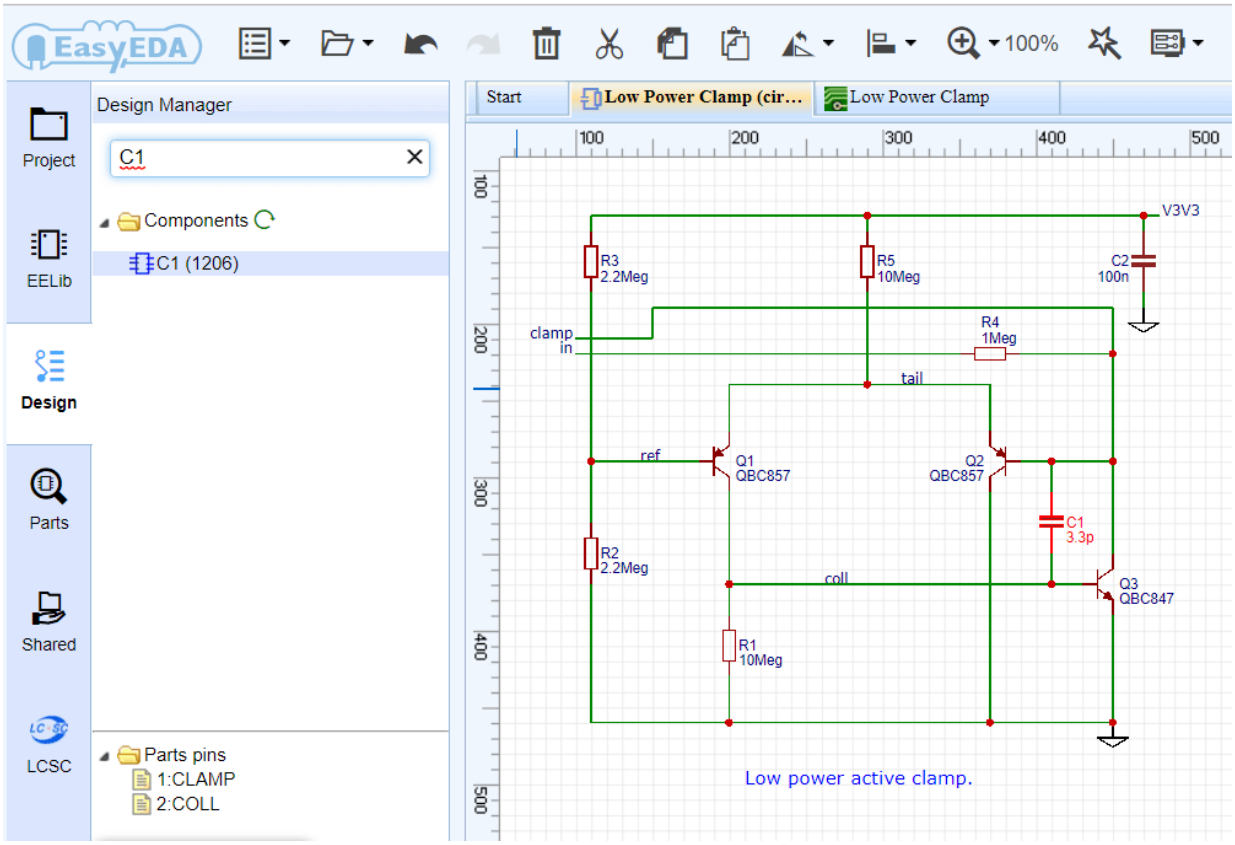
This feature will find, highlight and center in the window, parts by their Prefix (or reference designator). However, it cannot be used to find net names or other text in a schematic.

This is where the Design Manager comes in.

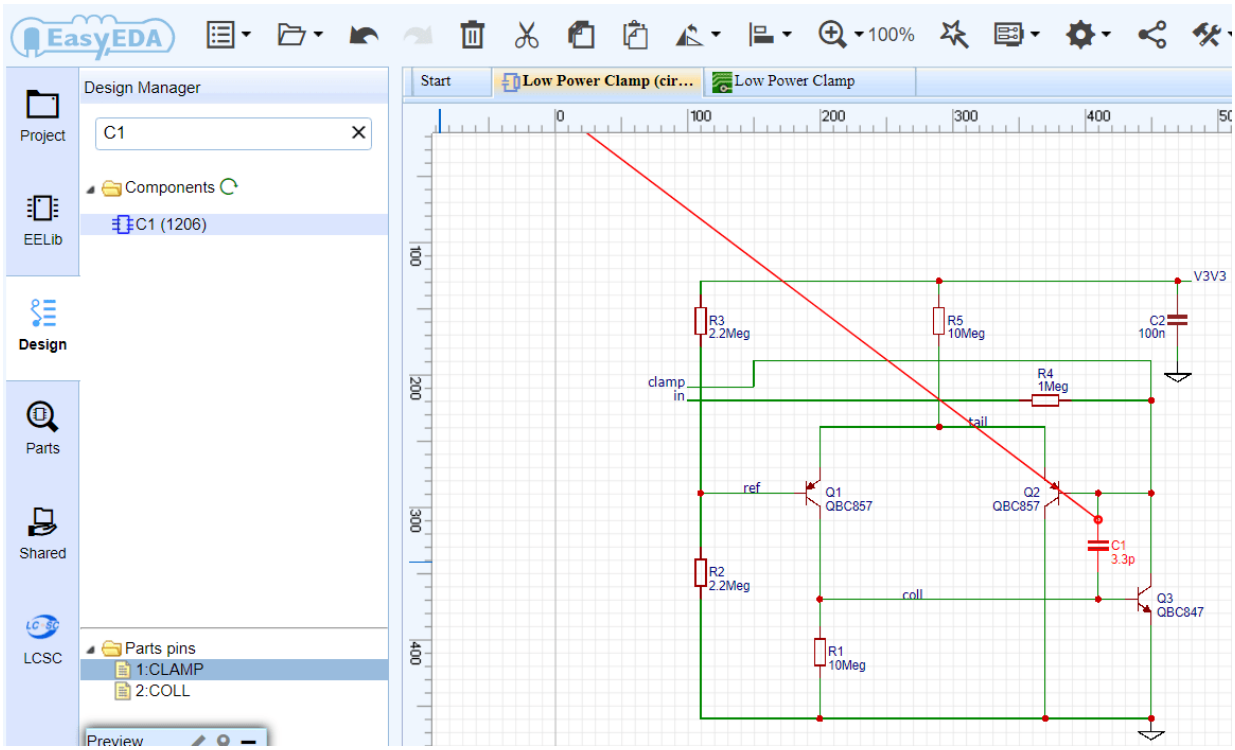
Left Navigation Panel > Design, or use hotkey `ctrl+D`.

The Schematic Design Manager is a very powerful tool for finding **components**, **nets** and **pins**.

Clicking on a Component item **highlights** the component and pans it to the center of the window.



Clicking on a Part pins item brings up a temporary pointer:



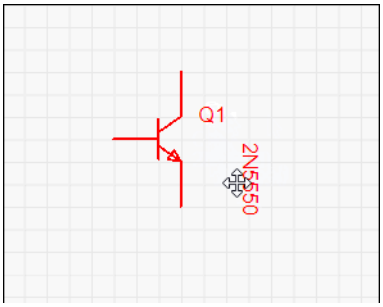
Placing Components

Find the component which you plan to place to your schematic, then move your mouse to the canvas and left click. If you want to add more, just left click again. To end the current sequence of placements, right click once or press `ESC`.

Don't try to Drag and Drop a component to the canvas: EasyEDA team thinks that Click-Click to place components will be easier to use than a Click-Drag mode.

Rotating the Prefix and Value (Name) of components

The default Prefix and Value (or name) of EasyEDA components are horizontal. To change them to vertical like this...

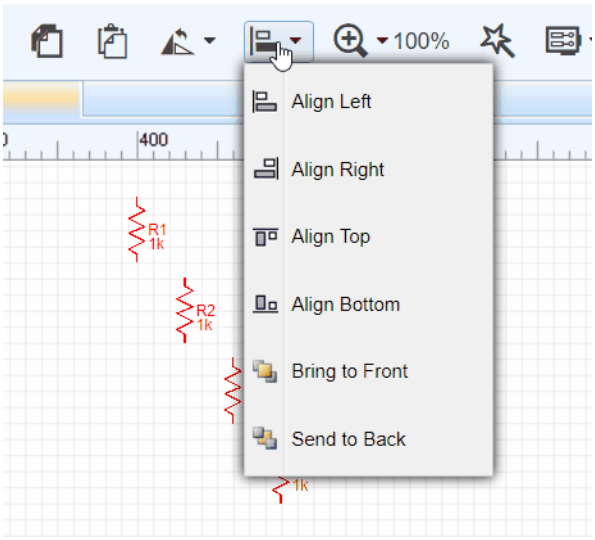


...Left click the prefix or value and when it is highlighted in **red** color, then press the **rotation** hotkey **Space** and you're done.

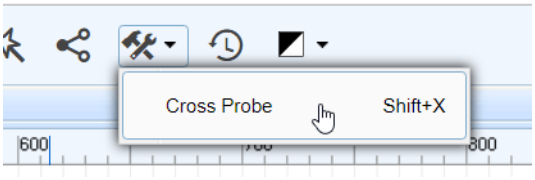
Adjusting Components

About adjusting components you can:

1. Move components with your mouse
2. Move components with the arrow keys.
3. Find components with the Design Manager via the `CTRL+D` hotkey: select the component in the Design Manager to pan it to the centre of the canvas and then move it with your mouse.
4. Align the components:

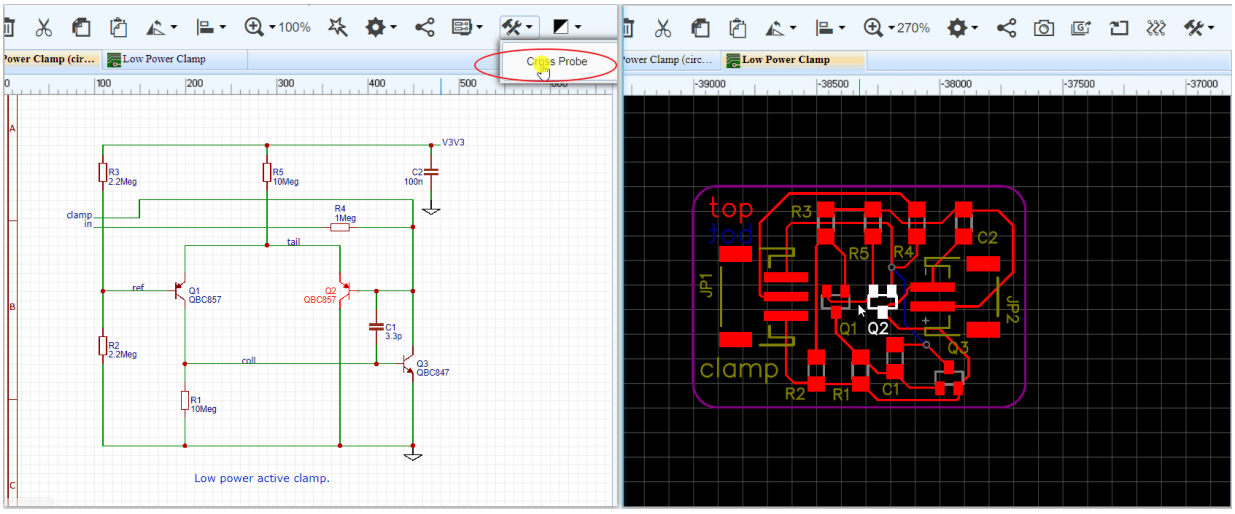


Cross Probe



This tool is used to cross probe from chosen objects on the current schematic to its corresponding counterparts in the PCB, or from PCB Footprints to corresponding counterparts in the schematic.

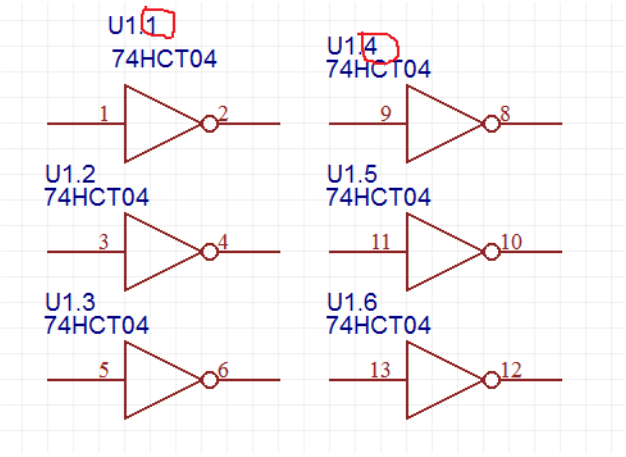
Note: *You don't need to open PCB first before using cross probe in the schematic. Editor will open the PCB automatically.*
And don't forget to use the hotkey `SHIFT+X`.



Multi-part Components

The number of pins on some components can be quite large. That's why it's easier to divide such a component into several parts or functional blocks. As a simple example, there are six gates in the 74HC04 Hex Inverter component. To avoid clutter in the schematic, GND and VCC pins of such components are usually served by a separate part of the component. This is really convenient as it doesn't interfere the working process with logical parts. The NetLabel names of VCC and GND Pin are usually hidden.

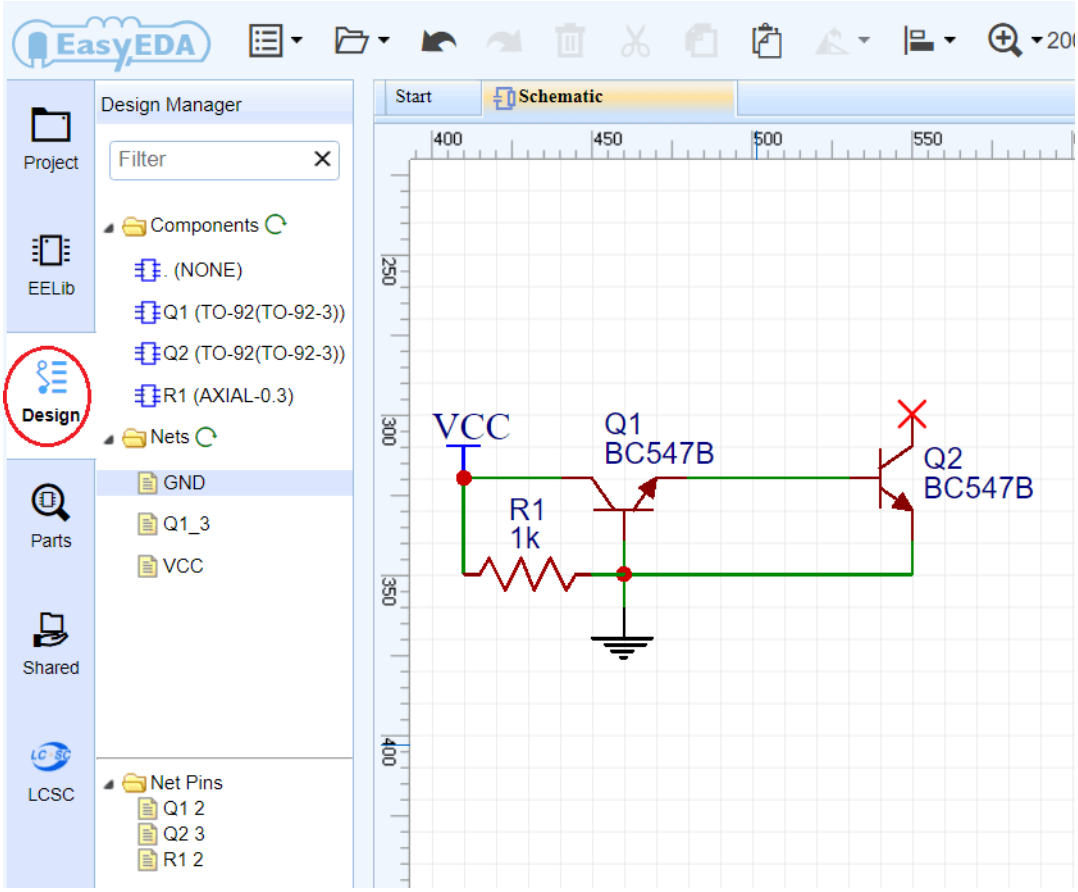
When placing the 74HC04 on a schematic, it will look like the screenshot below.
Note: *The component Prefix will be in form of: U?.1, U?.2 etc.*



Design Manager

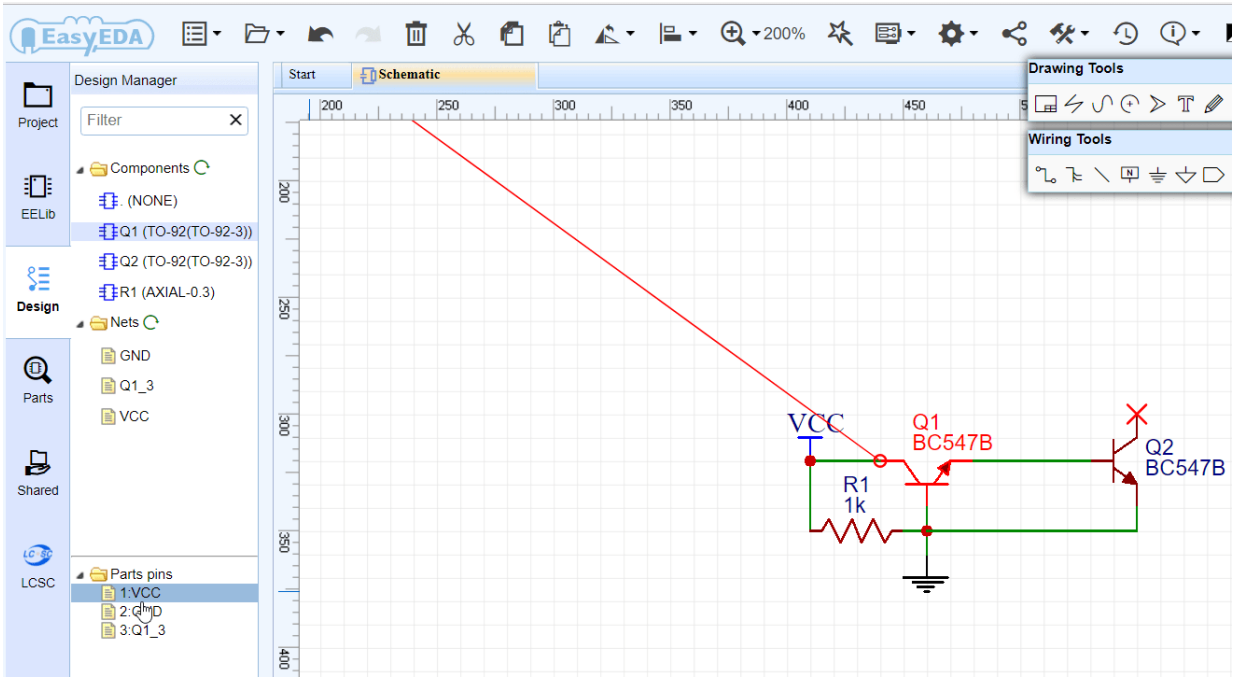
With large schematics it can be hard to find the components quickly. Sometimes, you may make a mistake such as wiring to a wrong component pin. So you need a tool to help you out. **Design Manager** is just the tool.

Just press the CTRL+D hotkey to open the Design Manager. or click it via on the left navigation panel:



1. **Filter**: You can find your components or net name easily: for example, if you want to find all capacitances, you just need to type c;
2. **Components**: Lists all the components in this schematic. Clicking on a Component item highlights that component and pans it to the center of the window.
3. **Nets**: Lists all the nets in this schematic. A net must connect at least two Pins, or the net name will be marked as a red error.
4. **Net Pins/Parts Pins**: Lists all the pins of the selected net name or components.

If you click the **Q1** Pin **1:VCC**, EasyEDA will show you where it is with a temporary marker from the top left of the canvas:

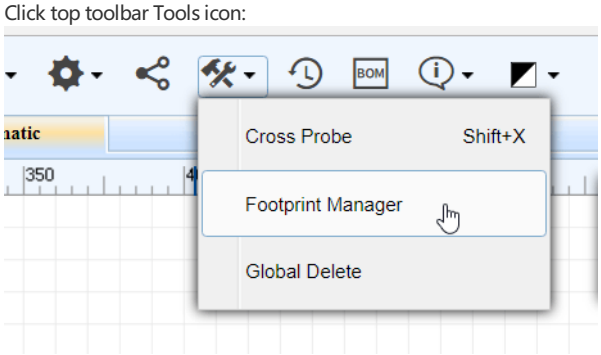


Footprint Manager

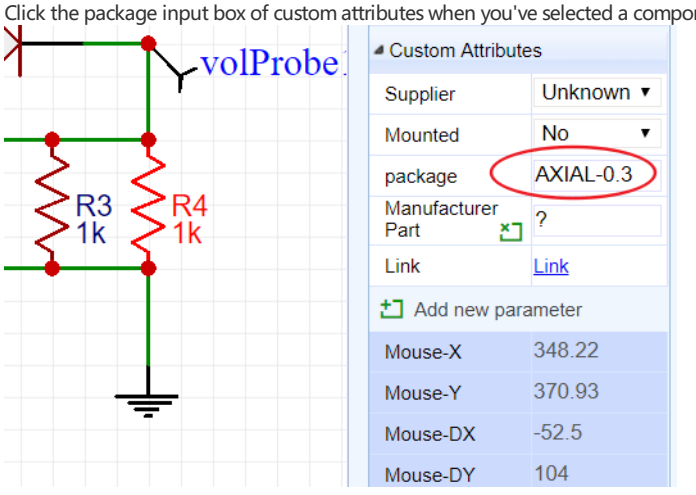
Want to batch modify components? Can't identify the corresponding relationship between component pins and footprint pins? Don't worry, EasyEDA can do this.

There are two ways to open the footprint manager:

- Click top toolbar Tools icon:



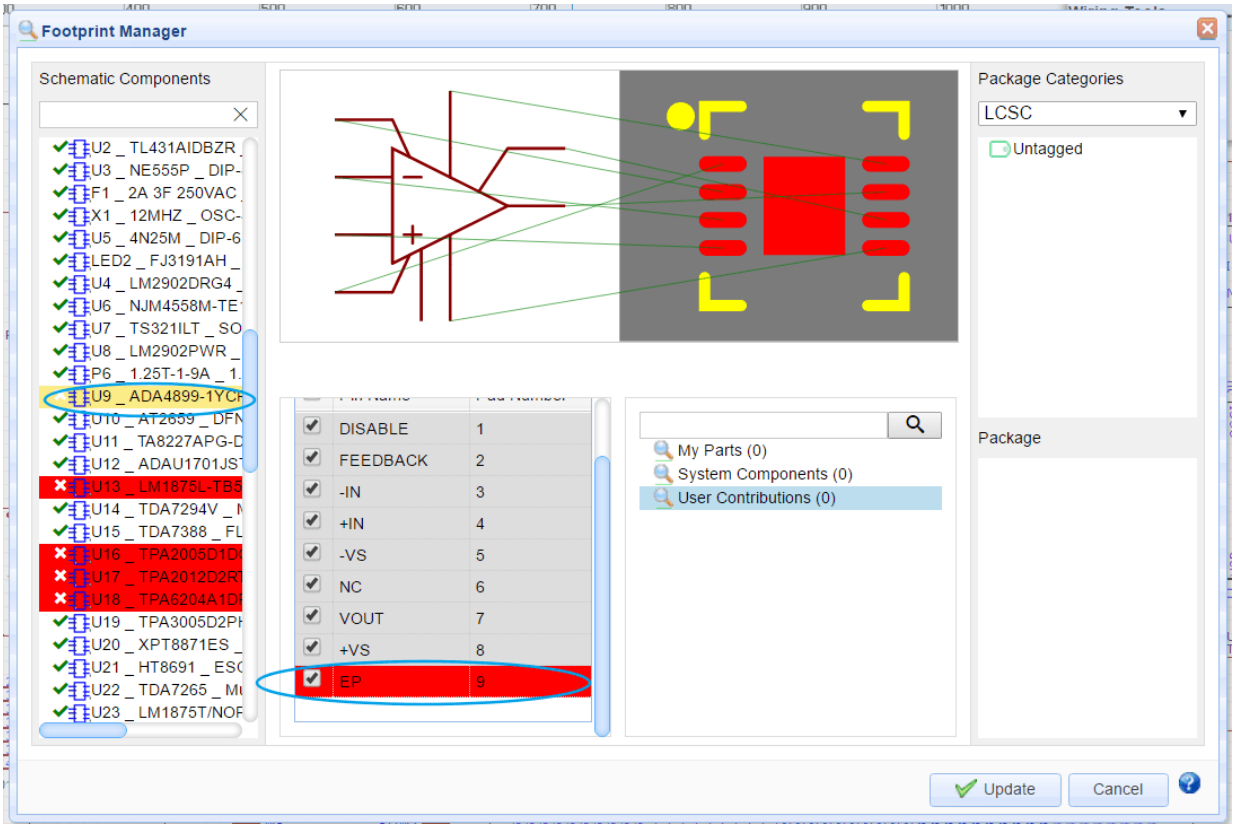
- Click the package input box of custom attributes when you've selected a component:



1.Footprint manager will check your parts package correct or not automatically. If the part without the package or this package doesn't exist in EasyEDA Libraries, or if the part's Pins doesn't correspond the package's Pads correctly, the footprint manager will show the red alert.
For example, If your part U1 has 2 pins, pin number are 1 and 2, pin name is A and B, but you assigned a footprint has 2 pads, pad number are A and B, but the part's pin number doesn't match the pad number, so the the footprint manager will alert red, in order to solve this:

- method 1: change part's pin number as A and B.
- method 2: change package's pad number as 1 and 2.
- method 3: find an other package and update.

2.In the preview area, you can zoom in, zoom out and pan with mouse.



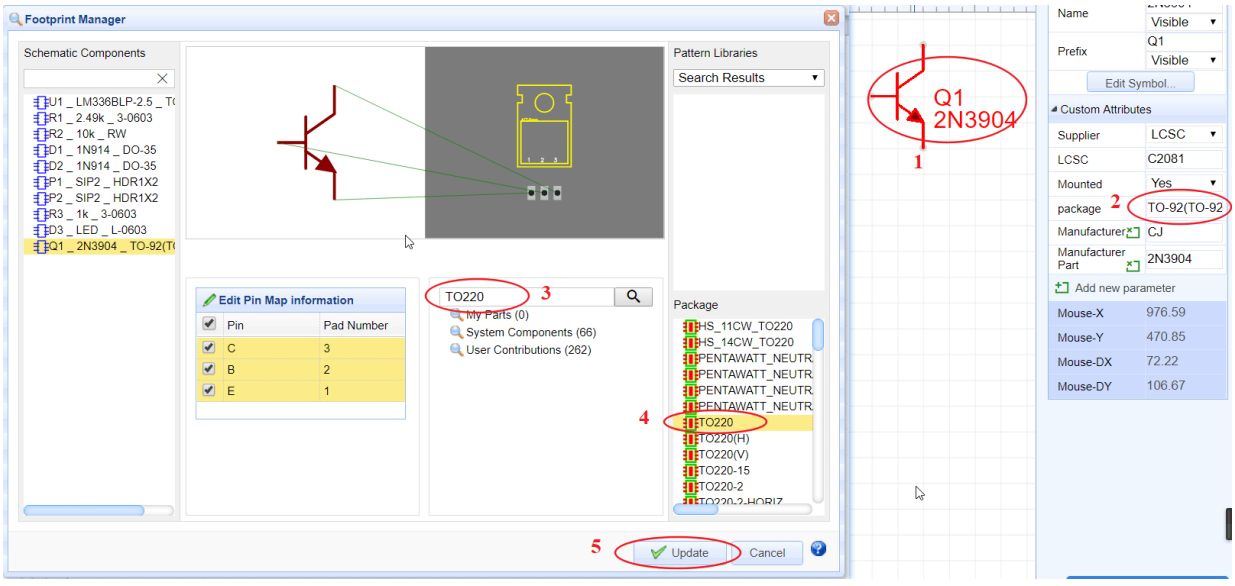
Update Package

If you want to change the Package, for example, select a component such as Q1, from **TO-92** TO **TO220**, you just need to click in the package input box. EasyEDA will popup the footprint manager dialog. You can follow the instructions.

- Type **TO220** into the search box and search,
- Select the **TO220** package,
- Verify it in the preview box,
- then press the **Update** button.

After that you will find you have changed the package to **TO220**.

Note: *To ensure that you use a package type that is already in the EasyEDA libraries, it is recommended that you use this technique to change component packages rather than just typing a package type directly into the package text box.*



Batch Update/Modify Packages

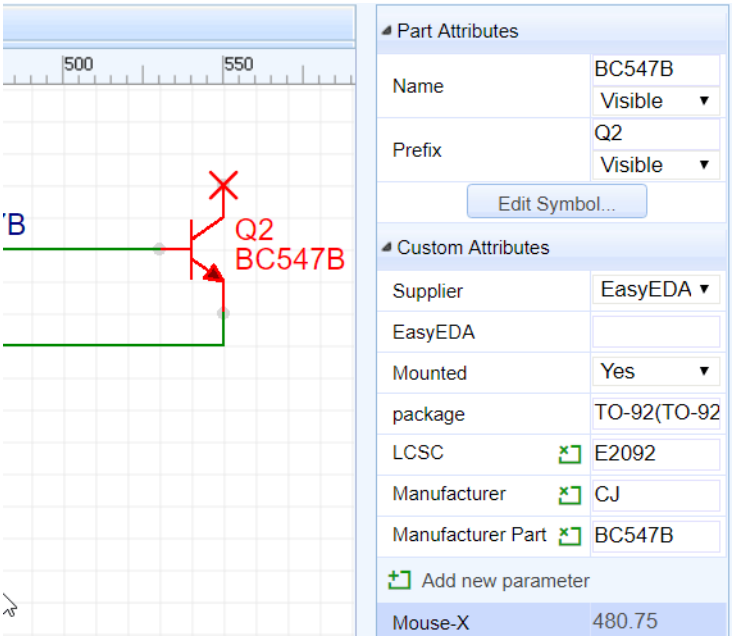
If you want to batch modify components' packages, in the footprint manager dialog you can press `CTRL` and select all components you want. If your schematic has many components, you should filter them first with package name. Such as in the below .gif which will show you how to batch modify resistors' packages from `AXIAL-0.3` to `0603`.
If you want to use your own packages, you can select **My Parts** on Pattern Libraries area.

Modify Pin Map Information

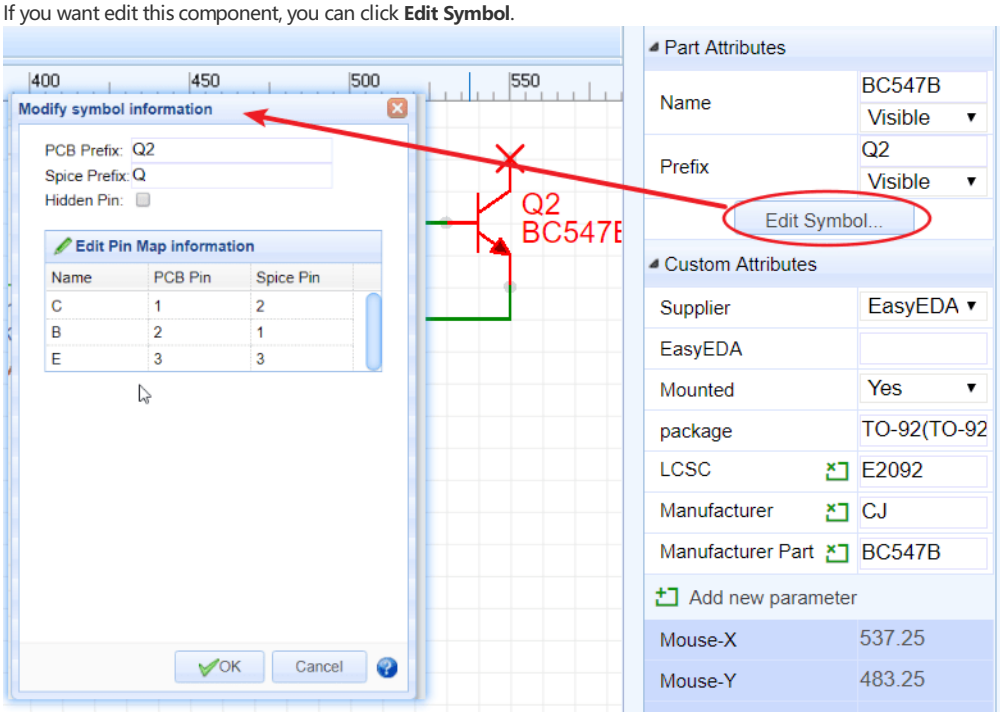
And you can modify component's pin map information in here.

Component Attributes

After selecting a component, you can find the component's attributes in the right hand Properties panel.



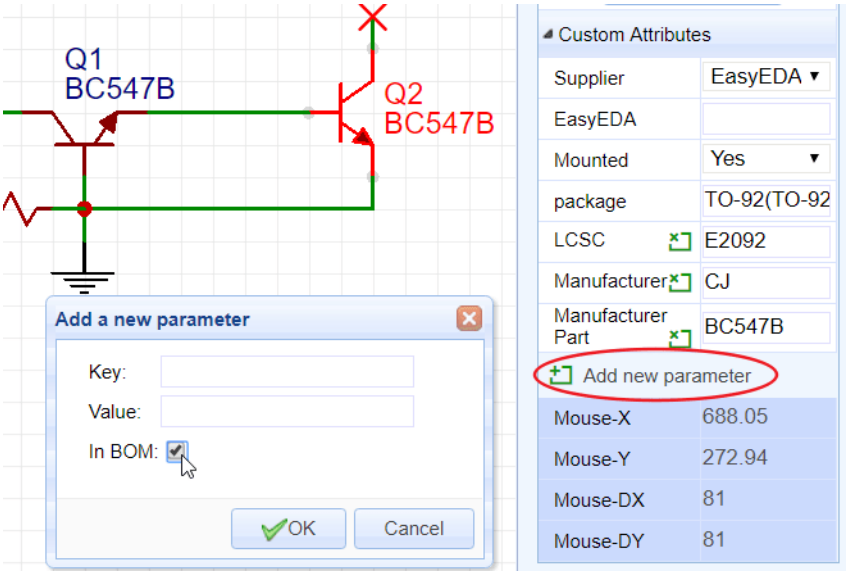
1.Part Attributes: You can change the **Prefix** and **Name** here，And make them **visible** or **invisible**.
If you want edit this component, you can click **Edit Symbol**.



2.Custom Attributes: You can change *component's supplier, mounted or not, change package*, and *add new parameter*.

Define BOM Parameters

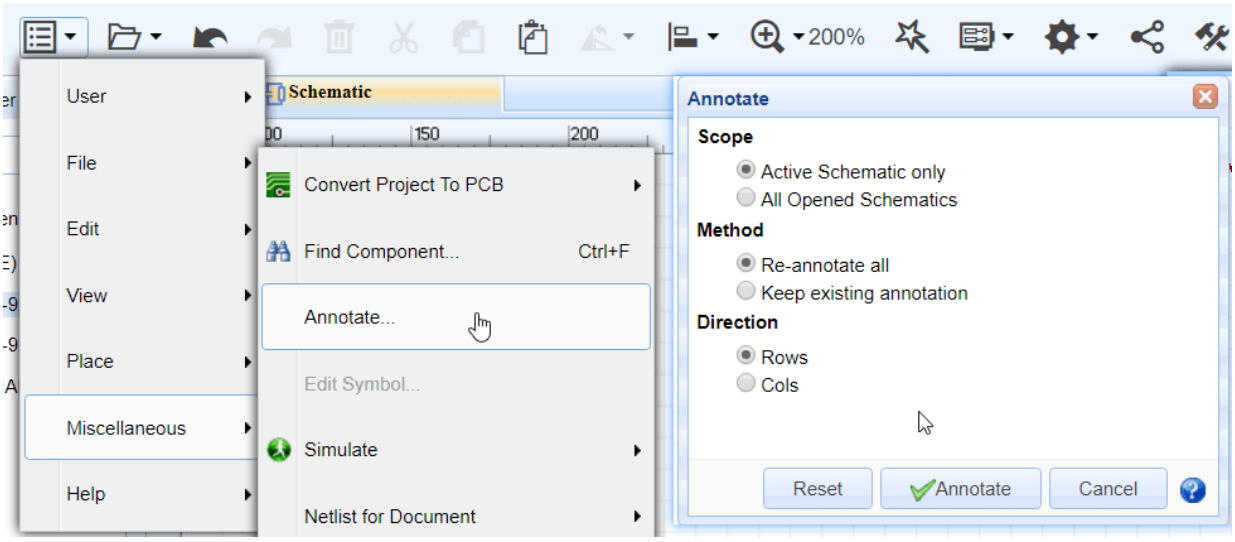
After selected a schematic symbol, you can add a parameter, and you can mark it as `In BOM`, when you export a BOM file, you can find this parameter in CSV file.



Annotate

After creating a schematic, it is quite likely that you have component Prefixes (reference designators) that are in no particular order on the canvas. You may also have duplicates. You can automatically renumber all the components using the **Annotate** function.

Super menu > Miscellaneous > Annotate



Various Annotate possibilities are available:

- **Active Schematic only**: applies annotation actions to the current schematic only.
- **All Opened Schematics**: applies annotation actions to all Schematics that are open in EasyEDA.

Note: *This option applies even if the opened schematics are from different Projects! If the project that you want to annotate has more than one schematic, you should open all of them and close any schematics that are open from other Projects.*

- **Re-annotate all**: resets all existing annotation and then annotates all components again from scratch;
- **Keep existing annotation**: annotates new components only (i.e. those whose reference designator finishes with ? like R? or U?).
- **Direction**: Rows annotates across the schematic in a raster pattern from top left to bottom right; Cols annotates down the schematic in a raster pattern from top left to bottom right.
- **Annotate**: applies the selected annotation actions.

Note: *Annotation cannot be undone! if you do not accept the result: close all of the affected schematics without saving. If you do accept the result: make sure you save all of the affected schematics.*

- **Reset**: if you want to reset all the reference designators to end with '?', just click the Reset button. After that, R1 will be R?, U1 will be U? etc.

Note: *Reset does not reset annotation back to where it was before pressing the Annotate button.*

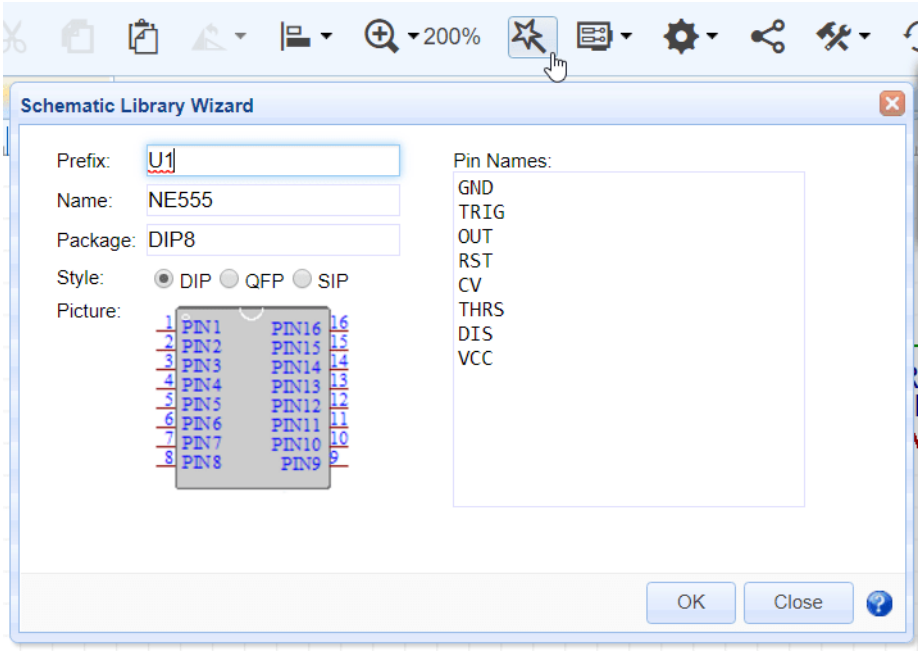
Symbol Wizard

How many times have you hit a schematic capture roadblock because you couldn't find a component symbol?

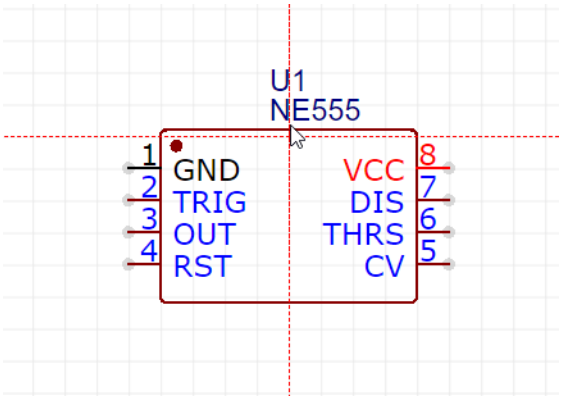
Well, in EasyEDA that would be never because the **Symbol Wizard** provides a quick and easy way to create a general schematic library symbol.

The **Symbol Wizard...** command can be found in the top toolbar.

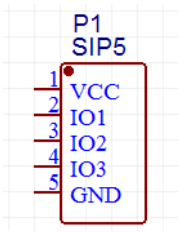
Or **Super Menu > Miscellaneous > Schematic Library Wizard** in a new schematic lib document.



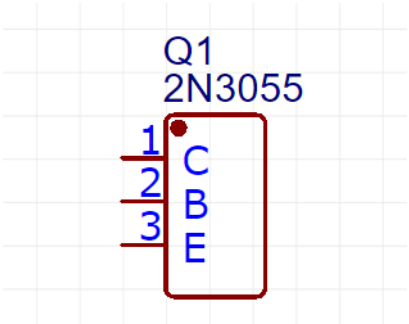
Using the **NE555** timer as an example: this device is available in a **DIP8** package so select **DIP**. Then enter the NE555 pin names into the **Pin Names** text box separated by new line or space, Then press OK. Abracadabra! As if by magic, you will find a perfectly formed dual in line 8 pin symbol for the NE555 attached to your mouse cursor, ready to be placed! You just need a few seconds to build a NE555 symbol, quickly and easily.



The EasyEDA Schematic Symbol Wizard allows you to create DIP, QPF or SIP styles symbols. If you are designing Arduino Shields then you will need lots of SIP symbol, so you can create a SIP symbol like the one shown below in a few seconds.



If you are not too worried that the symbols may not look quite the way people might expect and that they may not look anything like the **Package** type you enter, then of course you can use the wizard to create symbols for any component:

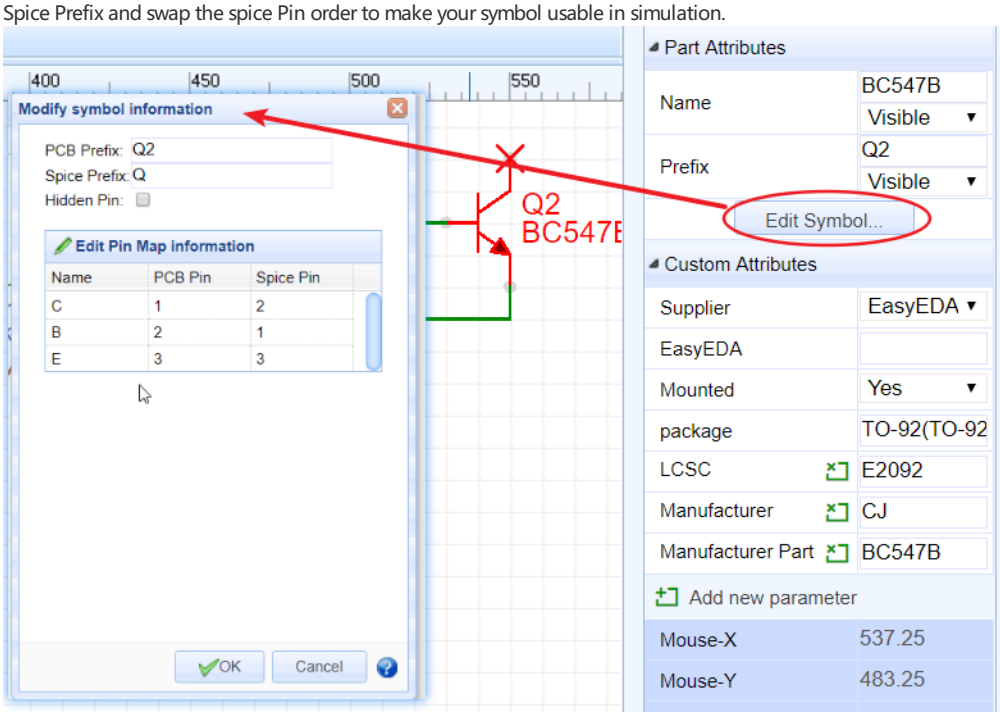


Pinmap Modify symbol information

When you select a component, for opening the Modify symbol information dialog, you can do:

1.Super menu > Miscellaneous > Edit Symbol...;
2.Or press the `I` hotkey;
3.Or click the Edit Symbol on the Parts Attributes on the left panel.

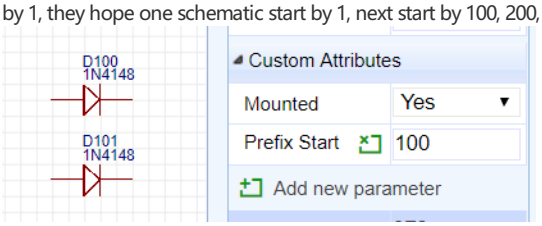
Using this dialog you can edit the pin names and numbers, for example, to suit a different package or device variant. You can also enter a Spice Prefix and swap the spice Pin order to make your symbol usable in simulation.



More detailed description of PCB and Spice Prefixes and pin numbers at next section.

Prefix Start

Every **NEW** schematic file has a `Prefix Start` custom parameter, some users would like use **multi-sheet designs**, but they hate every prefix start by 1, they hope one schematic start by 1, next start by 100, 200, 300. So you can use this solution.



Prefixes And Pin Numbers

Device and subcircuit (or hierarchical block) symbols created for use in schematics that are intended to be run as spice simulations, in addition to having a PCB Prefix that is used for the reference designator in the schematic, also have a **Spice Prefix**. They also have two sets of pin numbers: PCB pins and Spice pins.

PCB And Spice Prefix

The rules on the assignment of the PCB Prefix or reference designator of a schematic symbol are somewhat dependent on the EDA tool and on the user's preferences. Depending on how a device is graphically represented by its schematic symbol it may have a different **PCB Prefix** or** reference designator**. For example, a single discrete MOSFET device may have a PCB Prefix of Q, M or perhaps TR, whereas if it is part of a monolithic multiple transistor array it may have a PCB Prefix of U or IC.

The rules on the assignment of the **Spice Prefix** of a schematic symbol are strict. This is because the Spice Prefix is used to tell the simulator which circuit element the symbol represents and therefore which simulation model it is to use.

Simulation models for most of the spice circuit elements are in the form of a single-line .model statement however some of them may be in the form of a multi-line **.subckt** subcircuit definition. For example, some MOSFETs may be described by a .model statement in which case their Spice Prefix is **M** but many MOSFETs are described by a .subckt and so their Spice Prefix is **X**.

Therefore, irrespective of the PCB Prefix chosen for a schematic symbol, the Spice Prefix for a schematic symbol representing a given circuit element must match the type of model required to simulate that instance of that circuit element in your schematic.

For example, if you have two different n-channel MOSFETs in a schematic; Q1, a BSS123 which is modelled by a .model statement:

BSS123*SRC=BSS123;DI_BSS123;MOSFETs N;Enh;100V 0.170A 1.00ohms Diodes Inc.

MOSFET

```
.MODEL DI_BSS123  NMOS( LEVEL=1 VTO=1.00 KP=6.37m  GAMMA=1.24
+ PHI=.75  LAMBDA=625u RD=0.140 RS=0.140
+ IS=85.0f  PB=0.800 MJ=0.460 CBD=19.8p
+ CBS=23.7p  CGSO=36.0n CGDO=30.0n CGBO=124n  )
*   -- Assumes default L=100U W=100U --
and Q2, a BSS127S which is modelled by a .subckt:
BSS127S*---------- BSS127S Spice Model ----------
.SUBCKT BSS127S 10 20 30
*     TERMINALS:  D  G  S
M1 1 2 3 3 NMOS L = 1E-006 W = 1E-006
```

```
RD 10 1 84.22
RS 30 3 0.001
RG 20 2 29
CGS 2 3 1.958E-011
EGD 12 0 2 1 1
VFB 14 0 0
FFB 2 1 VFB 1
CGD 13 14 2E-011
R1 13 0 1
D1 12 13 DLIM
DDG 15 14 DCGD
R2 12 15 1
D2 15 0 DLIM
DSD 3 10 DSUB
.MODEL NMOS NMOS LEVEL = 3 VMAX = 8E+005 ETA = 1E-012 VTO = 3.419
+ TOX = 6E-008 NSUB = 1E+016 KP = 0.127 U0 = 400 KAPPA = 1.044E-015
.MODEL DCGD D CJO = 1.135E-011 VJ = 0.9232 M = 0.9816
.MODEL DSUB D IS = 2.294E-010 N = 1.601 RS = 0.1079 BV = 65
+ CJO = 1.956E-011 VJ = 1.514 M = 0.8171
.MODEL DLIM D IS = 0.0001
.ENDS
*Diodes BSS127S Spice Model v1.0 Last Revised 2012/6/6
```

then even though both have the same PCB Prefix of **Q**: Q1 must have a Spice Prefix of M and **Q2** must have a Spice Prefix of **X**.

A list of Spice Prefixes and their associated circuit elements is given in the table below.

Spice Prefix	Element description	Comment
A	XSPICE code model	analogue, digital, mixed signal
B	Behavioral (arbitrary) source	
C	Capacitor	
D	Diode	
E	Voltage-controlled voltage source (VCVS)	linear, non-linear
F	Current-controlled current source (CCCS)	linear
G	Voltage-controlled current source (VCCS)	linear, non-linear
H	Current-controlled voltage source (CCVS)	linear
I	Current source	
J	Junction field effect transistor (JFET)	
K	Coupled (Mutual) Inductors	
L	Inductor	
M	Metal oxide field effect transistor (MOSFET)	
N	Numerical device for GSS	
O	Lossy transmission line	
P	Coupled multiconductor line (CPL)	
Q	Bipolar junction transistor (BJT)	
R	Resistor	
S	Switch (voltage-controlled)	
T	Lossless transmission line	
U	Uniformly distributed RC line	
V	Voltage source	
W	Switch (current-controlled)	
X	Subcircuit	
Y	Single lossy transmission line (TXL)	
Z	Metal semiconductor field effect transistor (MESFET)	

For more information on circuit elements in Ngspice, please refer to:

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.2.1.2

PCB and Spice pin numbers

The two sets of pin numbers are:

- **PCB pin number:** these are the numbers for the real, physical device pins in its package. They are required so that the pins of a device symbol in a schematic can be mapped onto the physical pins of a PCB footprint. In other words, so that the connections shown in the schematic, end up connected properly by copper on the PCB.
- ** Spice pin number or pin order:** these are the numbers that map the pins on the symbol to their respective functions in the spice model or subcircuit.

Actually the spice pin ordering has a slightly deeper meaning.

Spice has no concept of component symbols: they are a construct of the schematic editor.

When a spice netlist is generated, the symbol in the schematic editor is either - in the case of model defined devices such as resistors, capacitors, inductors, diodes, transistors and sources - mapped directly to the relevant models (defined by the device prefix such as R, C, L, D, Q and so on), or in the case of a subcircuit, converted into a subcircuit call statement.

The spice pin ordering for the majority of built-in models such as resistors, capacitors, inductors, diodes, transistors and sources are defined and generally taken care of by the schematic editor, more care has to be taken with the spice pin ordering of subcircuits.

This can be illustrated by a simple opamp with 5 pins: inverting and non-inverting inputs; output and positive and negative supply pins but the principle applies to all spice subcircuits.

The subcircuit call for this opamp might look like this in the spice netlist:

X1 input feedback vpos vneg output opamp_ANF01

where:

X1 is the name of the subcircuit in the top level (i.e. the calling) circuit;

input feedback vpos vneg output are the netnames in the circuit calling (i.e. containing) the subcircuit and

opamp_ANF01 is the name of the subcircuit being called.

Pay special attention to the order of the netnames in the subcircuit call.

The spice pin ordering for the majority of opamp subcircuits is like that shown

in the example below:

```
*

* opamp_ANF01

*

* Simplified behavioural opamp

*

* Node assignments

*                    noninverting input

*                    |   inverting input

*                    |   |   positive supply

*                    |   |   |   negative supply

*                    |   |   |   |   output

*                    |   |   |   |   |

* spice pin order:   1   2   3   4   5

*                    |   |   |   |   |

.subckt opamp_ANF01 inp inn vcc vee out ; these are the netnames

*                                       used internally to the

*                                       subcircuit.

B1 out 0

+ V=(TANH((V(inp)-V(inn))*{Avol}*2/(V(vcc)-V(vee)))*(V(vcc)-V(vee))

+ +(V(vcc)+V(vee)))/2

*

.ends opamp_ANF01

*
```

Note: *The spice pin order of the subcircuit call is in exactly the same order as that of the subcircuit.*

Although the physical pin numbering of any device is critical for successfully mapping the pins on a schematic symbol onto a physical package footprint when laying out the PCB, because spice only knows about single devices and does not care about how they are physically packaged, each instance of any device in a spice schematic has to be mapped onto its own copy of the spice model or subcircuit, irrespective of where it is in any physical package.

Therefore, for the physical, package pin numbering of the four opamps in a quad opamp in say, a **SOIC14** or a **DIP14** package, as shown below, to work with the example subcircuit above, the spice pin ordering would be:

```
Opamp A         pin number         spice pin order

OUT               1                 5
```

```
IN-                2                2

IN+                3                1

V+                4                3

V-                11                4

Opamp B          pin number         spice pin order

OUT                7                5

IN-                6                2

IN+                5                1

V+                4                3

V-                11                4

Opamp C          pin number         spice pin order

OUT                8                5

IN-                9                2

IN+                10                1

V+                4                3

V-                11                4

Opamp D          pin number         spice pin order

OUT                14                5

IN-                13                2

IN+                12                1

V+                4                3

V-                11                4
```

The physical package pin numbering reflects that of each opamp in the package.

The spice pin ordering is the same for each instance of the individual opamps.

Of course there is only one physical instance of each supply pin on the schematic symbol for the quad opamp in this example but each spice subcircuit must have the supply pins explicitly defined.

Exactly how this is handled is at the schematic symbol level depends on how the schematic capture package handles symbols for multiple devices with shared supply pins but the generation of a spice netlist from the schematic will always generate the complete set of pins required in the subcircuit calls.

In cases where the subcircuit is built by the user as opposed to where it is supplied by a vendor for a particular device, exactly the same rules apply except that it is up to the user to specify the subcircuit pin order and to construct the symbol appropriately.

Although as described earlier, built in spice models usually have defined spice pin orders, not all subcircuits have the same spice pin numbering. Therefore if your spice circuit throws errors - especially if there are warnings about pin numbers or pin names - it is worth remembering to check that the pin order of the symbol that is netlisted to form the calling statement matches that of the subcircuit that is being called!

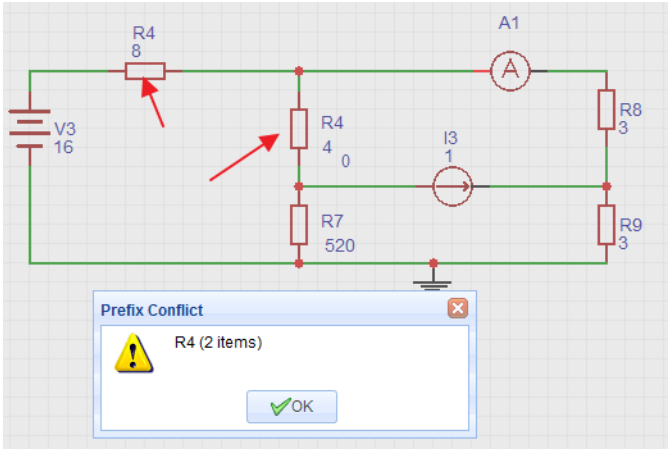
---

[8] In Debian based distributions gerbv is listed under Electronics in the package management system. The version in the repositories may be an earlier version but some users may find it easier to install than the SourceForge archive.

[9] As is the opamp_ANF01 example above

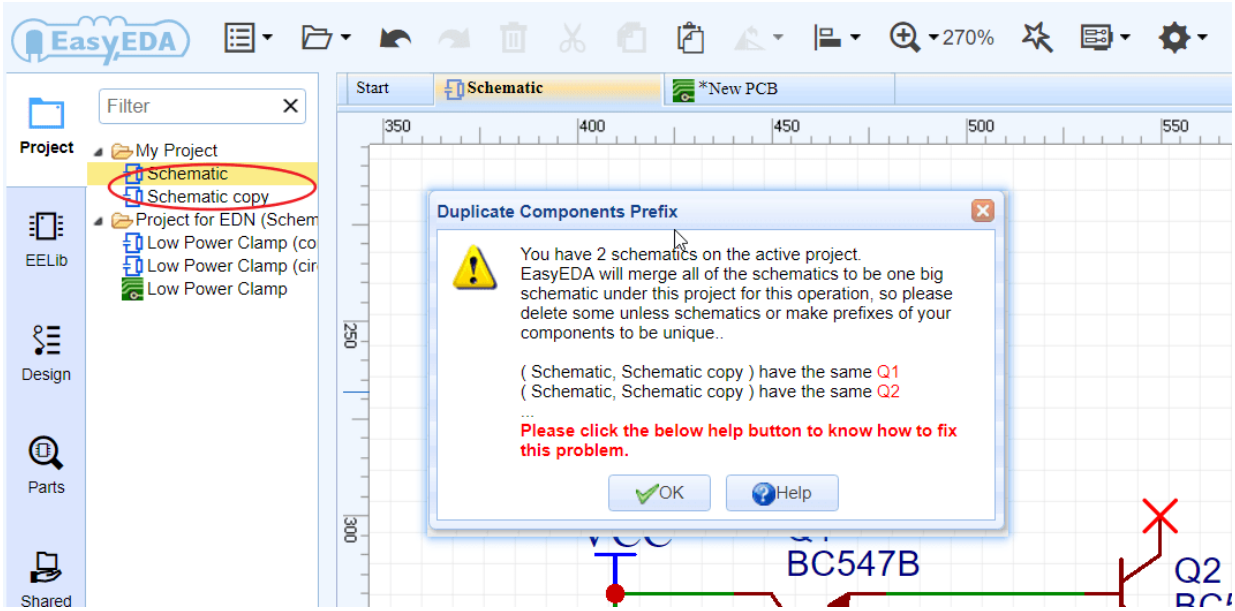
Prefix Conflict Error

Sometimes, when you convert a project to PCB, open the Design manager or run a simulation, you will get a Prefix Conflict error message.



In this schematic, you will find two components with the R4 reference designator, so you just need to change one to Rx where x is a unique number in that schematic.

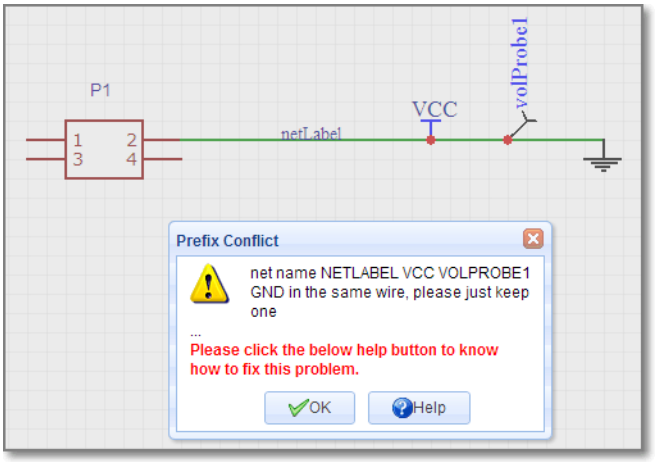
It may be tempting to backup a schematic into the same project as the original, however, if an attempt is then made to do Convert Project to PCB, you will get the Prefix Conflict error for every component.



In the above image, you can find the two identical copies of the same schematic, which when you Convert Project to PCB, EasyEDA will try to merge into a single schematic, so every item will have 2 copies.
To fix this, you just need to create a backup project and remove or better still save backup copies of your schematics to that project.

Net Name Conflict Error

Sometimes, when you convert project to PCB, open the Design manager or run a simulation, you will get a **Net Name Conflict** error message.



In this schematic, you will find four net label/net flag(EasyEDA takes volprobe, GND VCC as netlabel too) in the same wire, So you must remove the others.

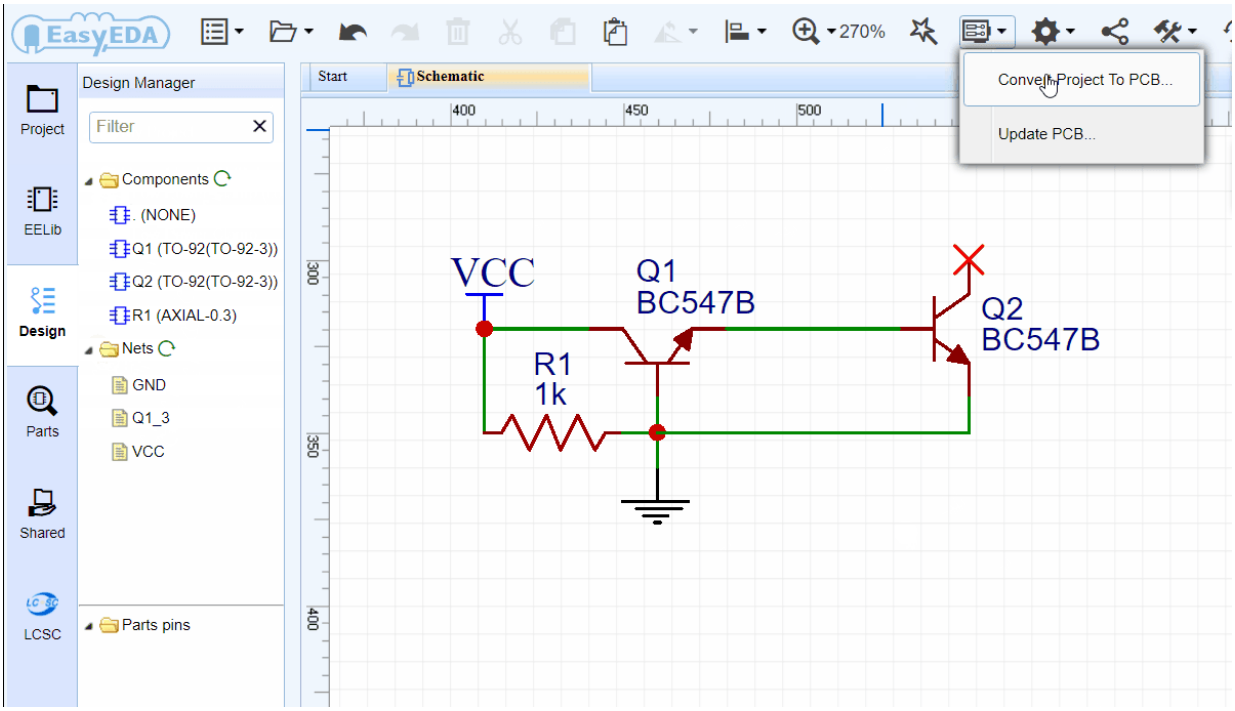
If you would like to probe a GND, you can use Probe command

Converting Schematics To PCB

Before using "Convert to PCB", "Update PCB" in Schematic and "Import Changes" in PCB, please read Essential Check section.

Most of the time, schematics are created with the aim of producing a PCB. So how do you convert your schematic to a PCB in EasyEDA? You just need to to click the PCB icon on the toolbar with the title **Convert project to PCB**.

Note: *Before converting, you need to use the Design Manager and Footprint Manager to check all the components, nets(connection) and packages/footprints to ensure no errors exist.*



PCB Libs search order

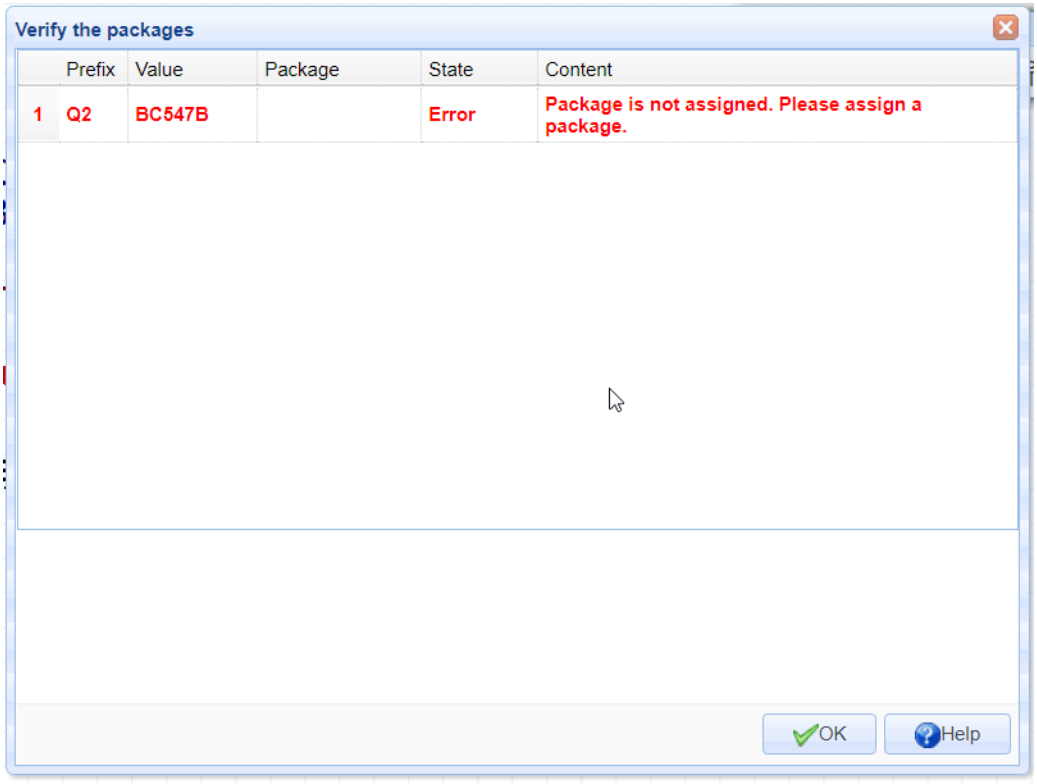
To find PCB footprints to match the package type attributes of your Schematic symbols, EasyEDA will search the available PCB libraries.

EasyEDA will search your own PCB Libs from the **My Parts** section first. If it doesn't find a matching footprint there then it will search in the System PCB Libs.

So, for example, if your symbol calls up a "**TO220**" package, you have a **TO220** package in your My Parts section and there is a "**TO220**" package in the system PCB Libs, then EasyEDA will use the "**TO220**" package in your My Parts and ignore the system PCB Lib.

Verify Packages and Build PCB

After clicking the **Convert project to PCB** button, if the project has errors the following dialog will open:

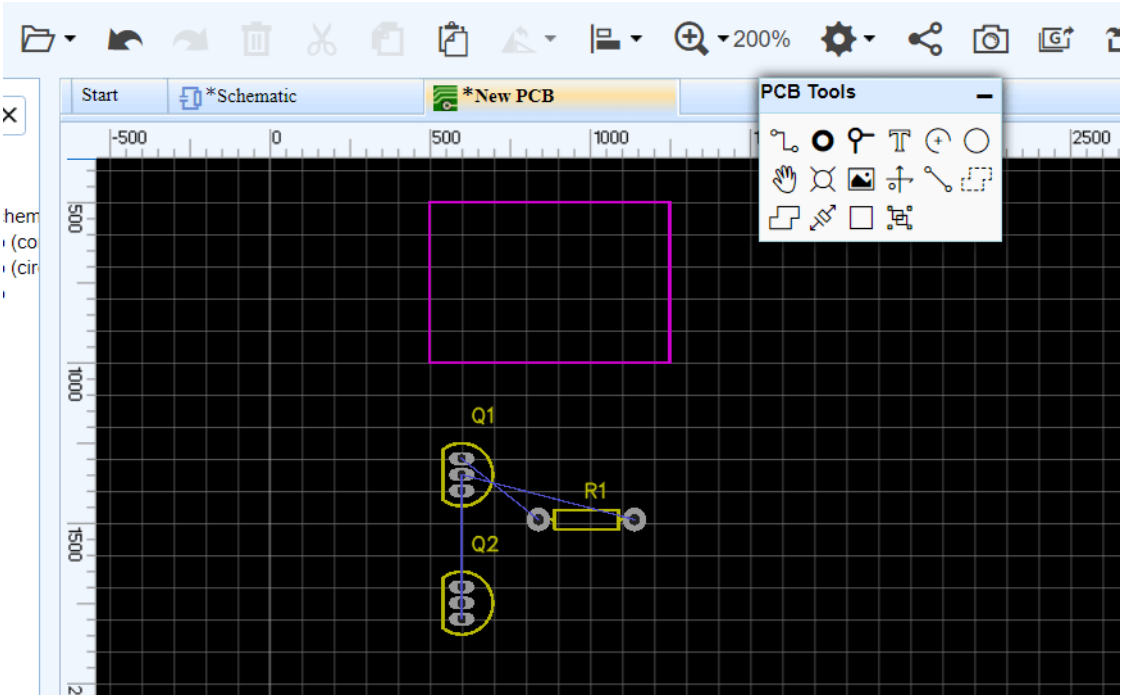


The row in red indicates that EasyEDA can't find a PCB footprint matching the Package that the schematic symbol is calling for.

This could be because you have made an error entering the package attribute in the symbol's Properties or maybe you haven't yet created a PCB footprint for the package that your symbol is calling for.

In this case the package should have been **TO-92(TO-92-3)** but instead it is empty. To correct it you can click on the row and change it to **TO-92(TO-92-3)**.

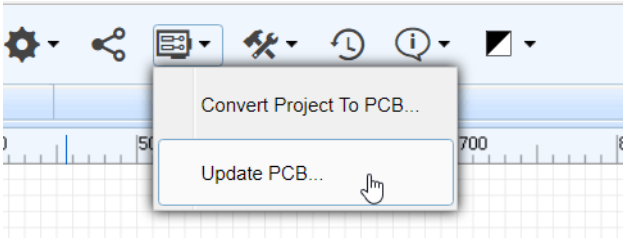
After making any necessary corrections, click the **Convert project to PCB** button and EasyEDA will automatically load all the package PCB footprints into the PCB editor as shown in the image below.



This shows the footprints placed in arbitrary positions with the connections between them shown as blue Rat lines.

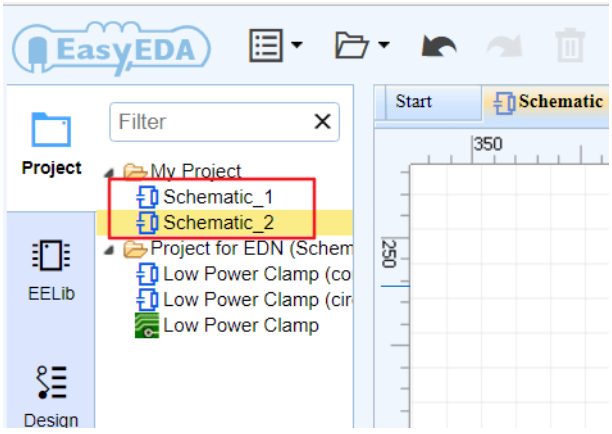
Update PCB

Converting a schematic to PCB can be done using the `Convert Project to PCB...`, but if you do modifications to the schematic, by using the `Update PCB` button you can immediately be passed forward to update the selected PCB without having the PCB editor window already open or without creating a new PCB file.

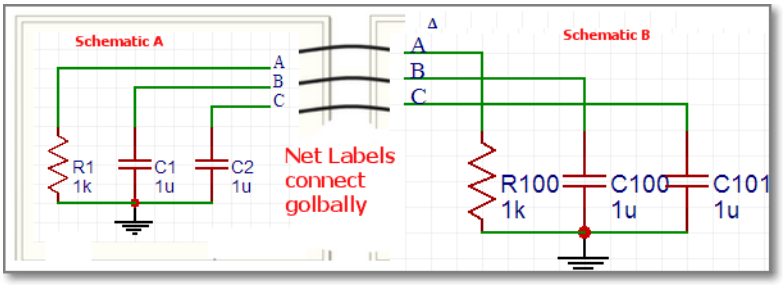


Hierarchy

EasyEDA does not support true hierarchical designs but it does support **multi-sheet designs**. You can put several schematics in one project with connections between made by netlabels. All nets in EasyEDA are global so if you create a netlabel DATA0 in schematic A and then create a netlabel DATA0 in schematic B, when Schematic A and schematic B are in the same project, they will be connected.

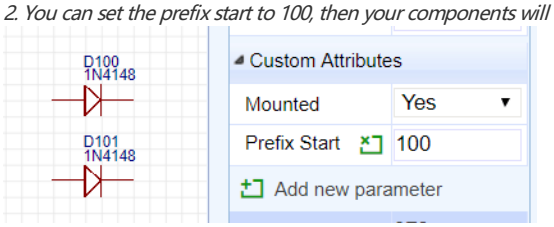


Multi-sheet designs(equivalent to a circuit spread over several pieces of paper), all schematics under the same project will be merged into one when be converted to PCB connecting in **netlabe**, **netflag**.



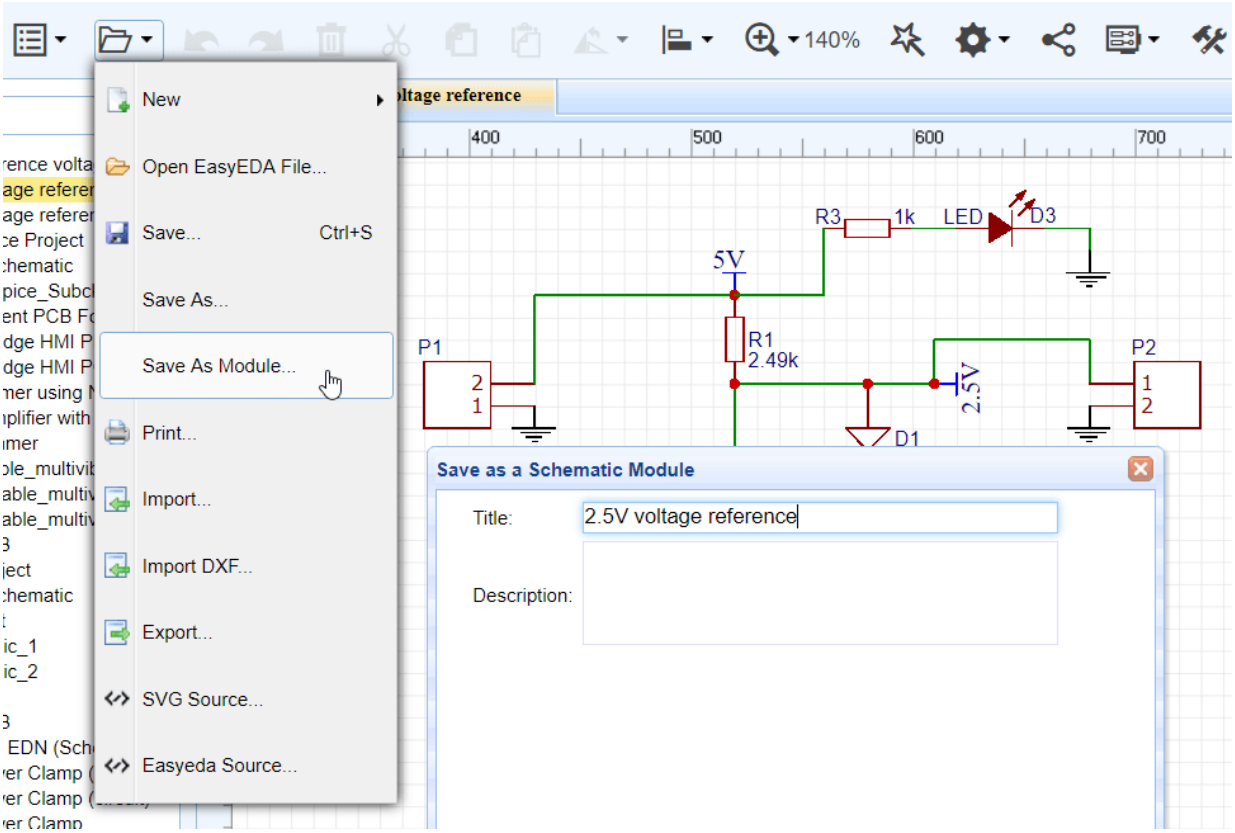
Note:
Please make all of the prefixes unique, if the Schematic A has a R1, and the Schematic B has a R1, then you will get a Prefix Conflict Error on above section.

Tip:
Every schematic's prefix will start from 1, such as R1, C1, U1 etc.
1. you can use Annotate to fix prefix.
2. You can set the prefix start to 100, then your components will start from R100, C100.

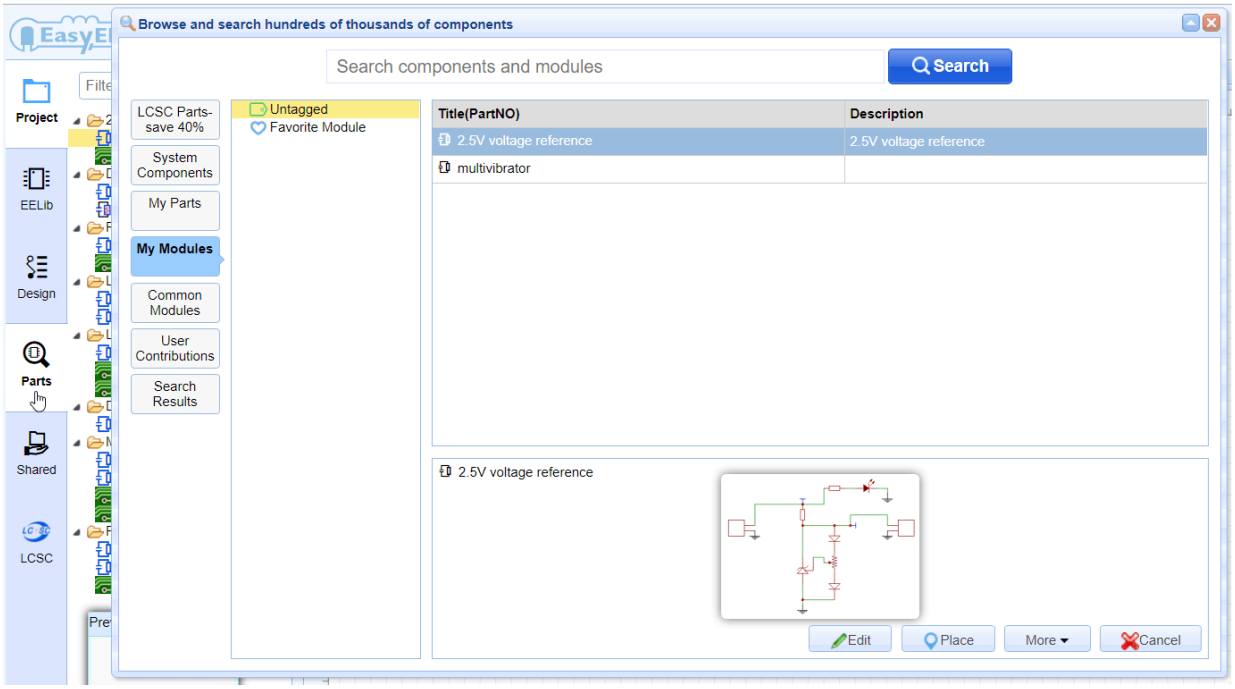


Modules

Copying codes is an easy job for coders, now copying and reusing a schematic or PCB is easy. Take a power supply unit for example, you can save this unit as a schematic module.



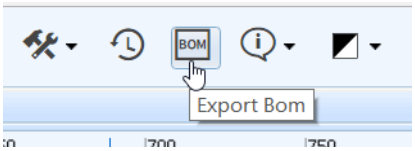
After saving, you can find it at the **Parts** > **My Modules** section. And you can place the whole block to your schematic.



BOM/Netlist

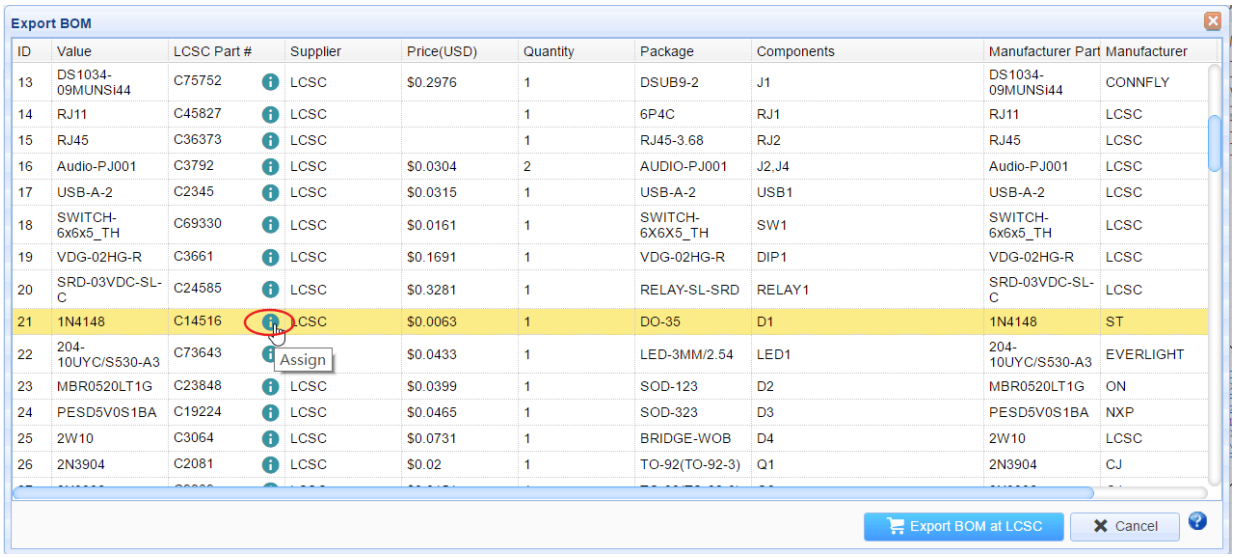
Export BOM

You can **export** the Bill of Materials (BOM) for the active schematic (Document) and PCB or for the active project (i.e. the BOM for all the sheets in the project) as shown below，click the top toolbar `BOM` icon:

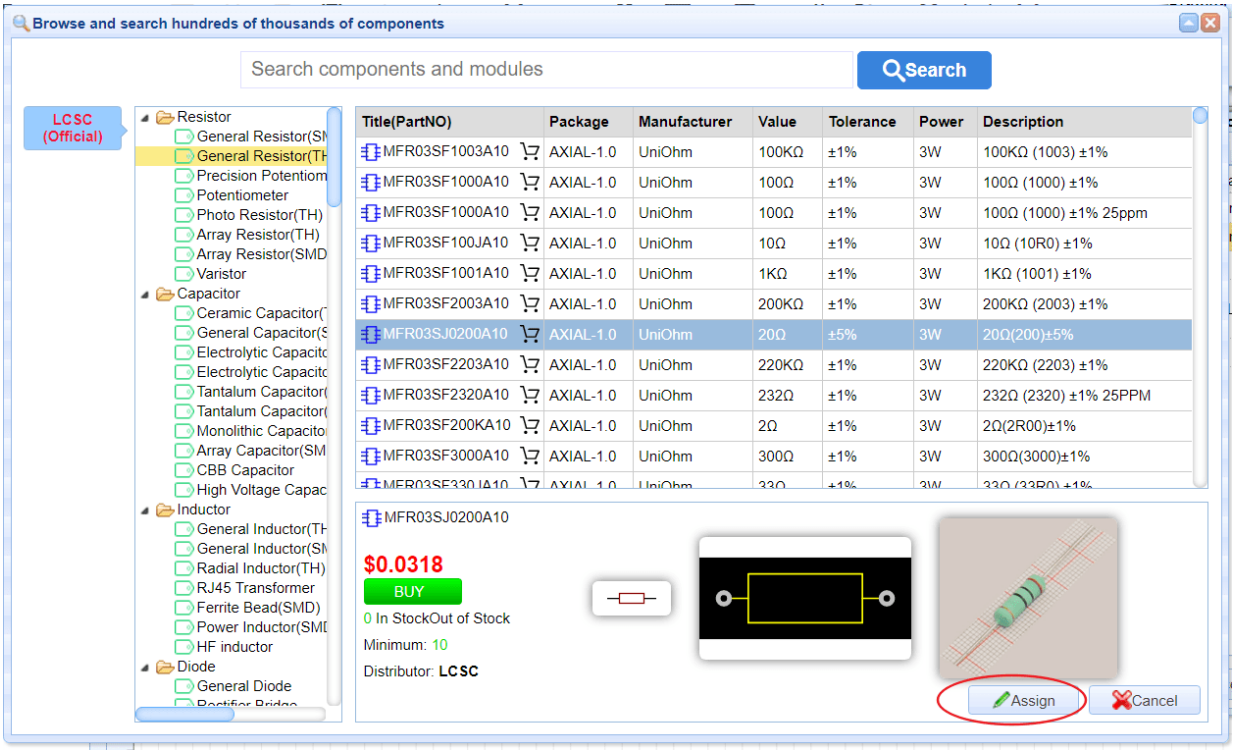


After clicking the BOM export option, the dialog below will open.

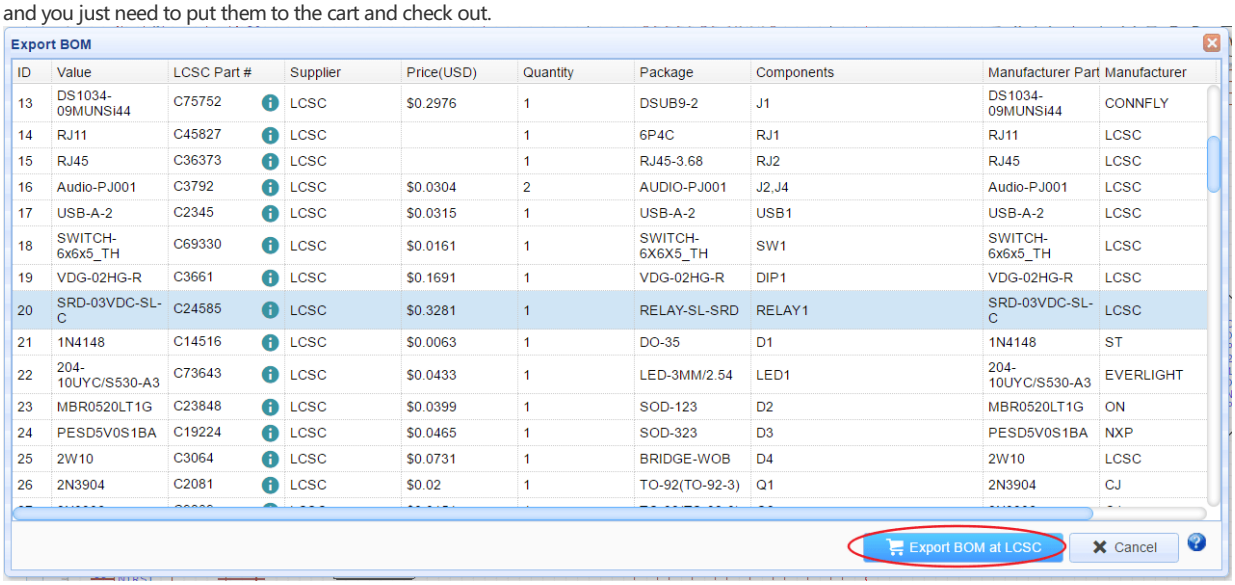
In this dialog，you can assign LCSC part's order code for your components.

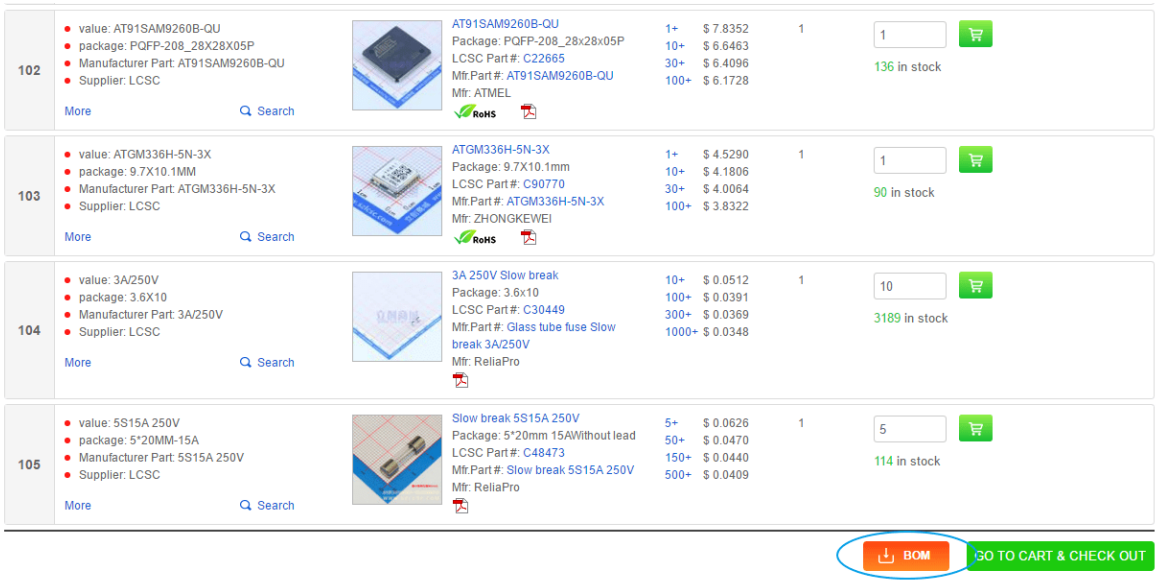


After clicking on the assign icon，the components and packages search dialog will pop up, and you can choose which component you want to assign.



When you click "Export BOM at LCSC", we will help you to list all the components of your BOM, If you want to buy the components form LCSC, and you just need to put them to the cart and check out.





And Click the "BOM" button to download the BOM file. You can open it in any text editor or spreadsheet.

	A	B	C	D	E	F	G	H	I	J
1	id	value	quantity	package	components	Manufacturer Part	Manufacturer	Supplier	LCSC	price
2	1	150	2	AXIAL-0.3	R1,R4	25121WJ020KT4F	UniOhm	LCSC	C45278	$0.02
3	2	22k	2	AXIAL-0.3	R2,R3	25121WF300LT4F	UniOhm	LCSC	C16074	$0.03
4	3	22u	2	CAP-D3.0XF1.5	C1,C2	1812B225K500NT	FH	LCSC	C28503	$0.28
5	4	204-10UYC/S53	2	LED-3MM/2.54	LED1,LED2	67-21S/KK3C-H2727QAR3LED	EVERLIGHT	LCSC	C73540	$0.04
6	5	2N3904	2	TO-92(TO-92-3)	Q1,Q2	MURA220T3G	ON	LCSC	C37995	$0.17
7										

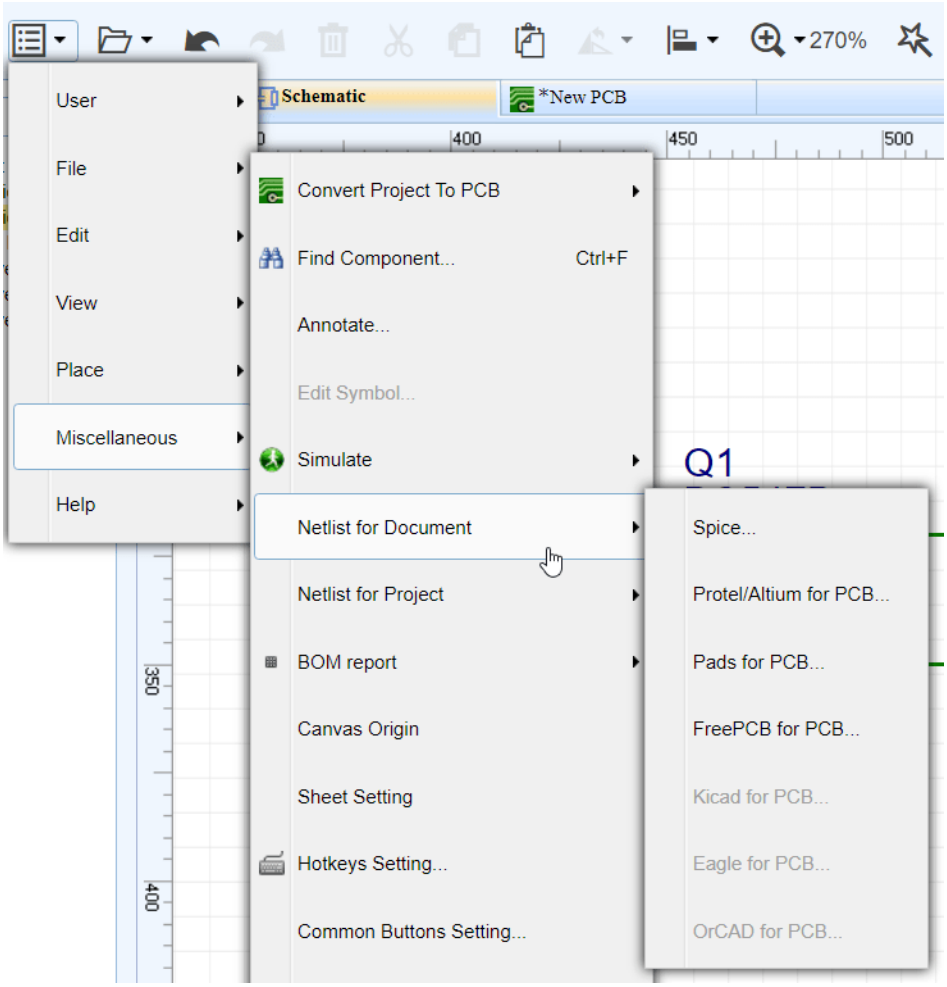
Export Netlist

EasyEDA can export the netlist for the active schematic (Document) and/or for the whole active project:

Super menu > Miscellaneous > Netlist for Document or Netlist for Project

EasyEDA can export a netlist in a variety of formats:

- **Spice**: this is a Spice3f5 compatible netlist generated by the simulation engine of EasyEDA, Ngspice. It is not normally used as the basis for as a PCB layout.
- KiCad: a PCB netlist in a format that can be imported straight into Pcbnew, the PCB layout tool part of the free, open source cross-platform EDA suite.
- **Altium Designer**: a PCB netlist in a format that can be imported straight into Altium Designer and it's predecessor, Protel.
- **Pads**: a PCB netlist in a format that can be imported straight into Pads PCB layout tools.
- FreePCB: a PCB netlist in a format that can be imported straight into FreePCB, a free, open source PCB editor for Windows.

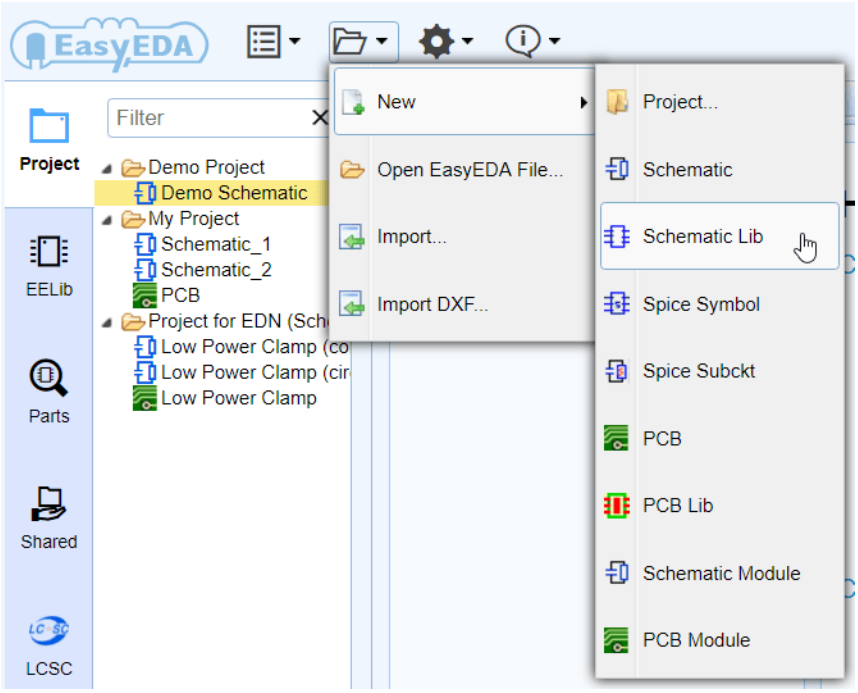


Creating the Schematic Libs

Using **Symbol Wizard** and **Group/Ungroup...** is a quick way to create schematic symbols but they are placed directly into the schematic that they are built in. It is possible to reuse them by copying them (`CTRL+C` hotkeys) from the schematic they were created in and then cross-document-pasting them (`CTRL+SHIFT+V` hotkeys) into a different schematic but this quickly gets messy if you need to copy symbols that were created in several different schematics. OK, you could keep copying new symbols into a dedicated "symbol library" schematic sheet to save searching for them but EasyEDA offers you an easier way to create and manage your symbols in a library.

Start a new Schematic Lib as shown below or by doing:

Document > New > Schematic Lib



This opens the New Schematic Lib symbol editor.

You can now create a symbol using Symbol Wizard as before or draw it using the Drawing Tools palette and add pins using the `P` hotkey (except that you no longer need to use **Group/Ungroup...**).

Then you can edit the pin map using:

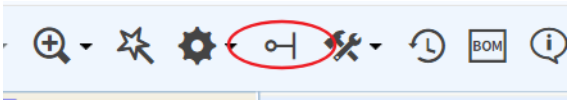
Super menu > Miscellaneous > Pin Map...

Note the Origin Point. To simplify rotating and flipping your symbols when they are placed into a schematic, make sure all of your symbols are created as near as possible centered around that point.

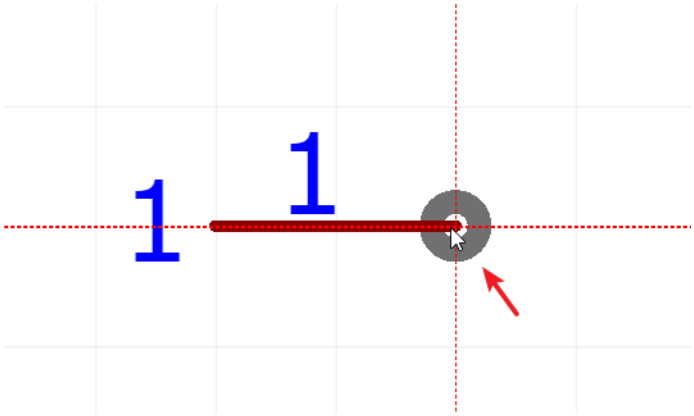
Pins

Symbols pins are the most important part of any Schematic Lib symbol. They are the things that allow wires to be attached to symbols to connect up your circuit.

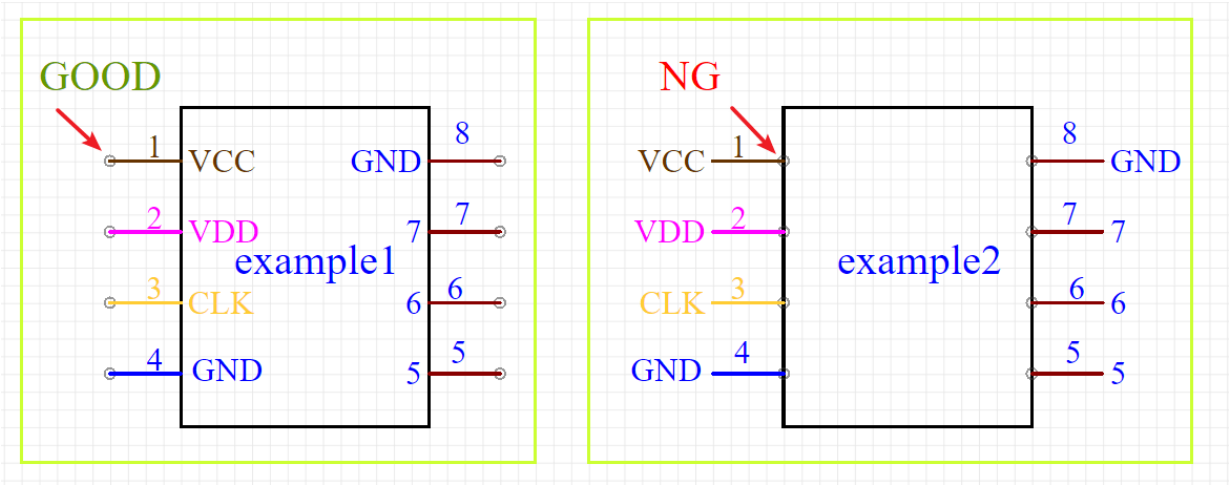
You can use the **P** hotkey to add a Pin or from the toolbar:



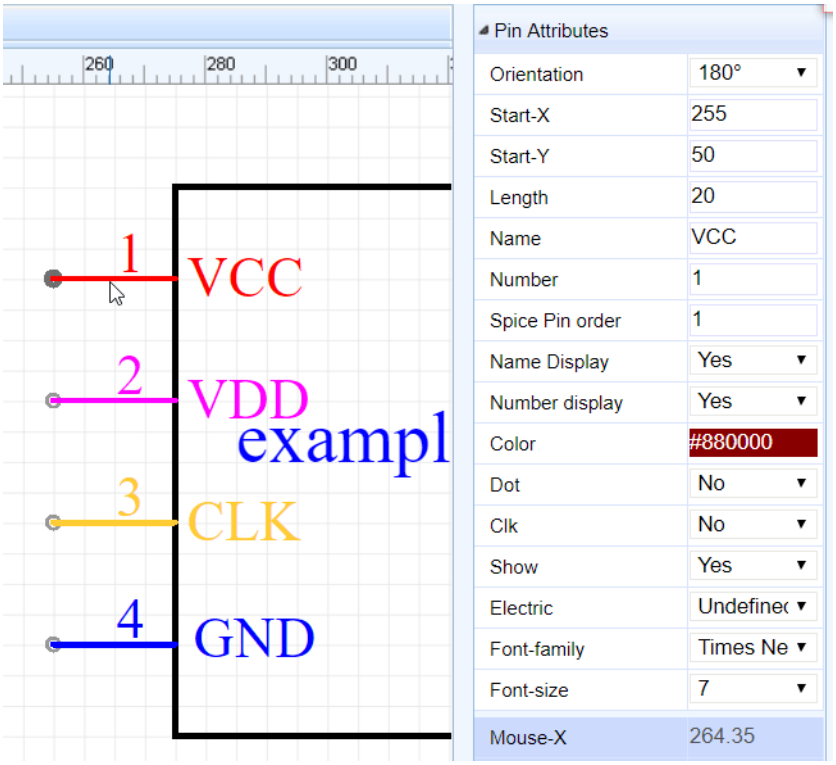
Before placing it on the canvas, you can use the rotation hotkey or rotate and flip from the menu to rotate it to the right orientation. Make sure the **Pin Dot(black dot)** is in the right position. The **Pin Dot** will be used to connect your wires or netlabels. Whenever a PIN is either placed as directly onto the canvas or as part of a symbol, the mouse has to point to the **Pin Dot** position to automatically start the Wire mode or to join a wire to it.



Whenever a Pin is placed as part of a symbol, the **Pin dot** should be **outside** of — and pointing away from — the symbol like in example 1(correct position), inside or pointing towards the symbol as shown in example 2(wrong position).



When you select a single Pin, the **Pin attributes** will be shown in the right hand **Properties** panel:



Orientation: 0°,90°, 180° and 270°。If you want to create a 45° pin, you need to set it length as 0, and draw a line with 45°。

Start-X and Start-Y: The pindot position. Sometimes it may be difficult to move the pin to the desired position using the mouse, so you can move the pin via Start-X and Start-Y.

Length: Pin length.

Name: In this example, *VCC* is the name of the Pin.

Number: In this example, *1* is the number of the Pin. This number is the pin number of the device in a physical package and so will be the pin number used in the device footprint for that device in that package in your PCB lib.

Note that you can use alphanumeric identifiers such as; A1, B1, C1, A2, B2 and so on as the Number.

Spice Pin order: These are the pin numbers used to connect your symbol to the corresponding pins defined by the .model or .subckt used to simulate your device. The pin numbers of the simulation model may be different from the physical package pin numbers and - unless the model is specifically created to model multiple devices in a single package - do not change for different instances of a device in a multi-device package. The Spice Pin order must be **numerals** only.

For more information about Spice Pin order please see the section on Prefixes And Pin Numbers.

Name Display: If you don't want to show *VCC*, switch it to NO.

Number Display: If you don't want to show *1*, switch it to NO.

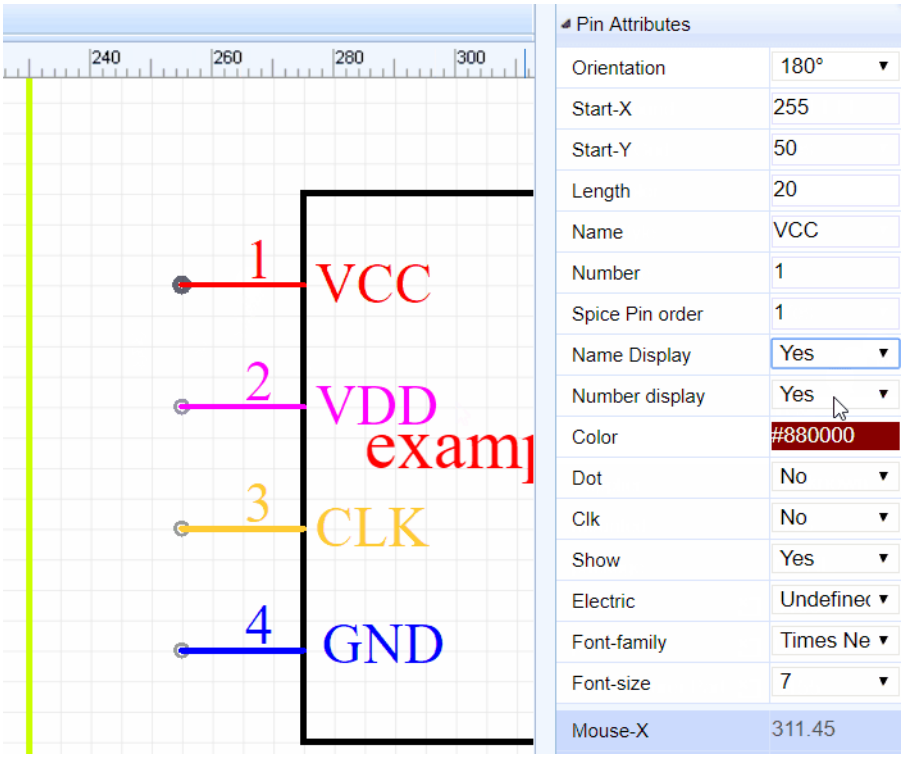
You can adjust the Name or Number position using your mouse but note that rotate and flip applies to the whole pin including the name and pin number; these items cannot be rotated and flipped independently of the pin itself.

Note also that rotate and flip actions do not result in upside down or mirrored pin number or names.

Color: You can set the Pin to different colours, such as *PIN3:CLK* as orange and *PIN4:GND* as blue. In this example, the PIN1 is set as color `#880000`, but it shows as red, because it is selected. After deselecting it, the pin will appear color `#880000`.

Dot: adds a circle to the inside end of the pin to indicate logical (or analogue) inversion.

Clk: adds a `>` to the inside end of the pin to indicate that the pin is logical clock input.



Show: YES/NO. Allows you to hide the pin. When set it to NO, this Pin will be hidden when the symbol is placed on the schematic editor canvas.

Note that the pin is not hidden here in the Schematic Lib symbol editor canvas because if it was, it would disappear from view and so how would you find it to make it visible again? For the same reason this option has no effect in symbols made using Group/Ungroup…

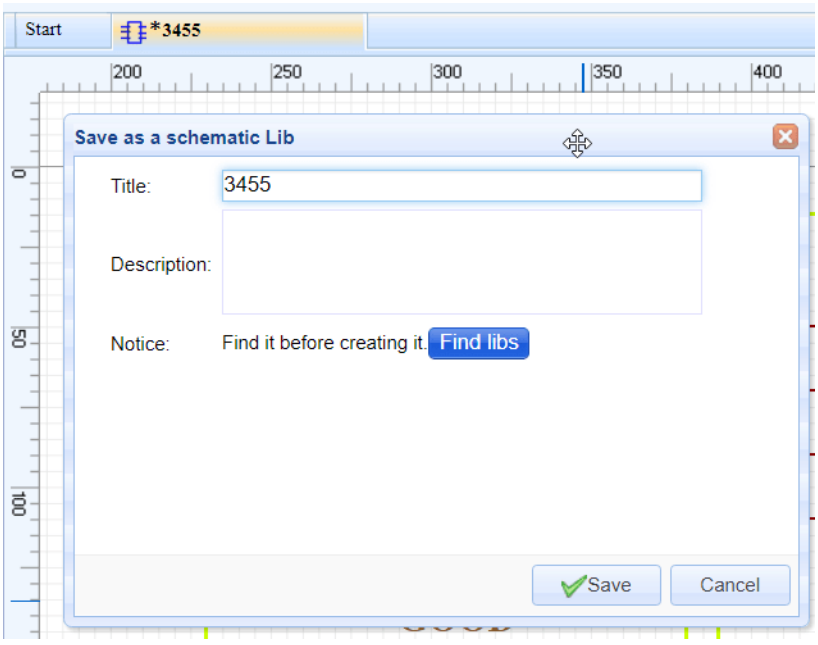
We may not have thought of everything in EasyEDA but we do try. ;)

Electric: [Undefined, Input, Output, I/O, Power]

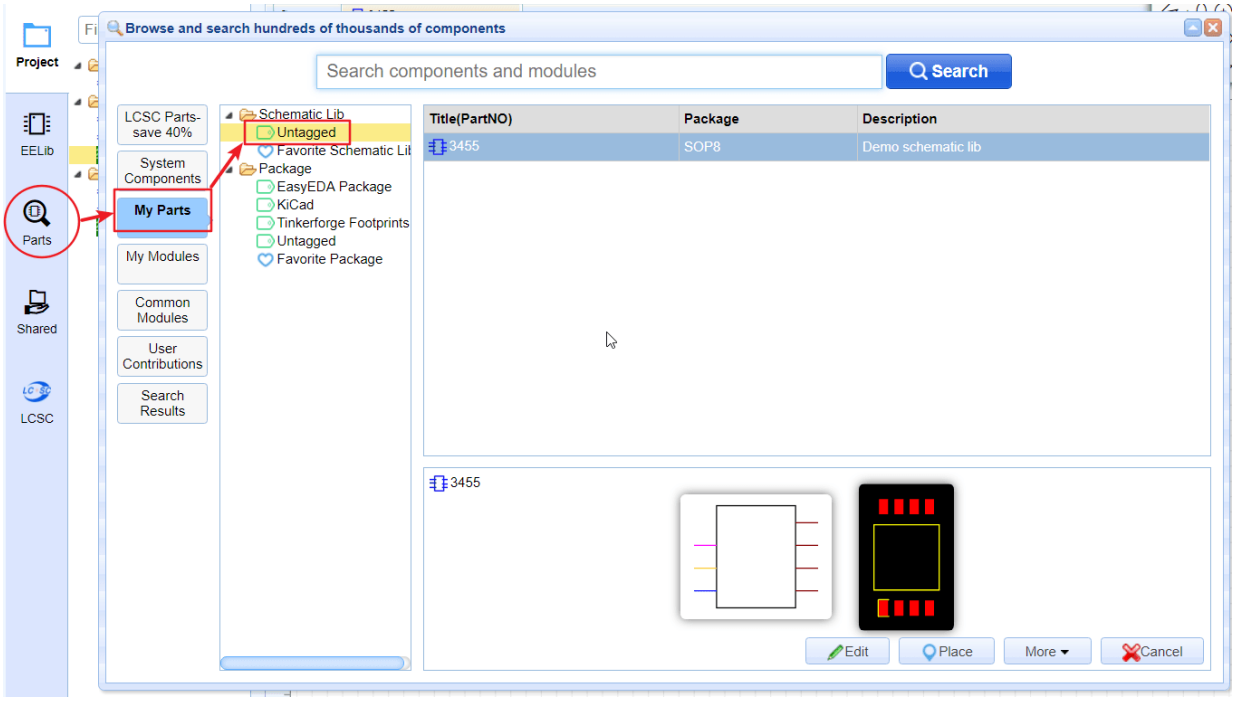
EasyEDA provides Electrical Rules Checking (ERC) right now, But you still need to set electric of your Schematic libs.

If you set the PIN as Power and set the pin to be hidden, then the Pin will be connected by Name which is the NetLabel. If the Name is VCC, it will be connected to the net in your circuit with the NetLabel or NetFlag VCC. This is helps to keep the schematic clear and uncluttered when using Multi-part Components.

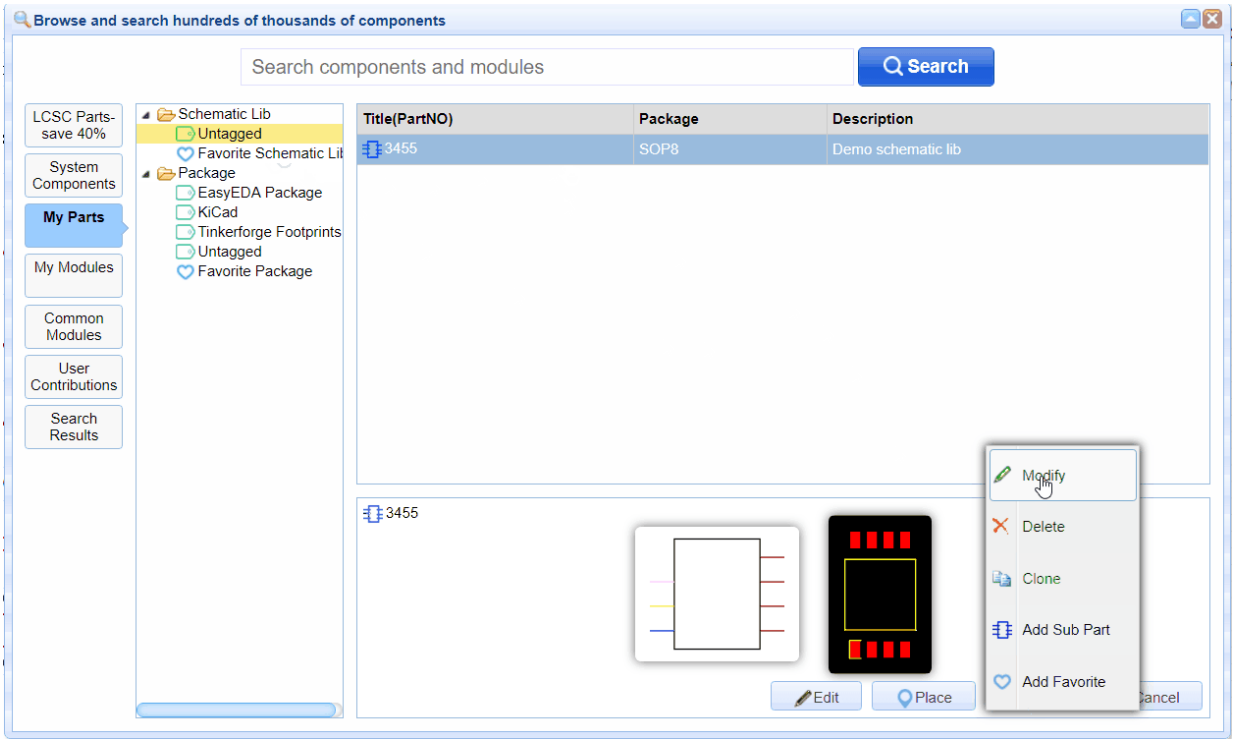
After created the Lib, use `CTRL+S` will open the save dialog:



After clicking **Save**, you will see it appears in **Parts** > **My Parts** > **Schematic Lib** of the left hand Navigation panel.



You can add a tag for your new symbol: **Parts** > **My Parts** > **Schematic Lib** > **Select New Lib** > **More** > **Modify**, otherwise it will appears on **Untagged**.



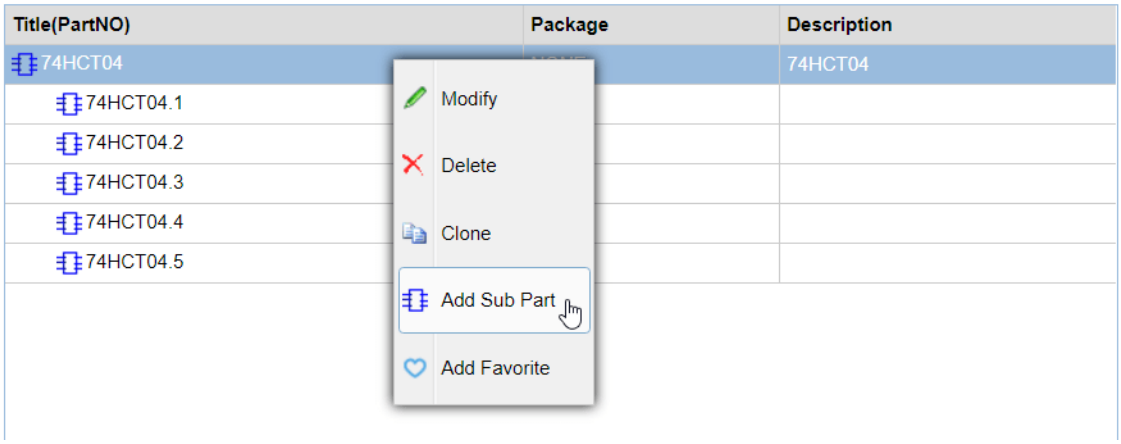
Subparts

We have already touched on how EasyEDA can support **Multi-part Components** but how do you create **multi-part components**?

EasyEDA provides a sub parts facility to do this.

After creating a part, you can right click the part in the My Parts section to pop up the content menu.

Suppose you have created your own symbol for a 74HCT04 hex inverter.



Right Click **Add sub part** and that will add 74HCT04.*1*,

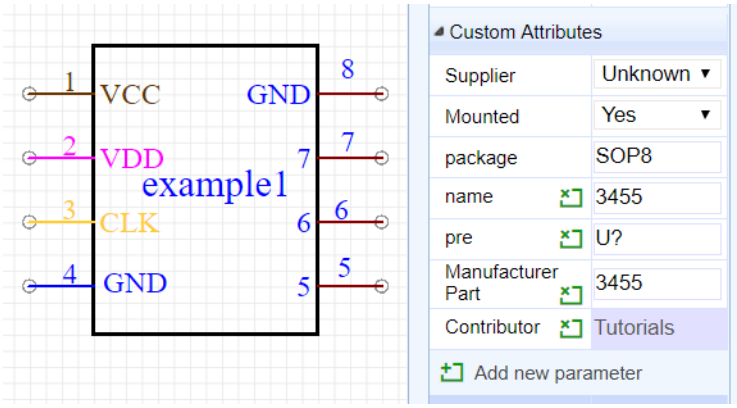
Click again to add 74HCT04.*2*, up to 74HCT04.*6*.

Then double click on each sub part in turn to modify the Pin Name and Number attributes.

Easy or what?

Custom Attributes

In the Schematic Lib editor's canvas Properties panel, you will find a **Custom Attributes** section:



Mounted

You can control this part mounted or not on the PCB. If you choose No, this component will not appear in the BOM report.

Package

If you would like to built a PCB, you need to assign a package for your schematic Lib symbol. Although there are other ways to do this in EasyEDA, here is the right place to do it. When you set a package , the package's pad numbers must match the schematic Lib's pin number, otherwise, when you convert the schematic to PCB , there will miss several nets.

Click in the **Package** input box, and the **Footprint Manager** dialog will open as used to do this task in the Schematic Editor.

Prefix

The default Schematic symbol Prefix is **U?** If you create a resistor, you can set the Prefix to **R?**

Name

You can change the schematic lib's name here, it is can be different from the part's file name.

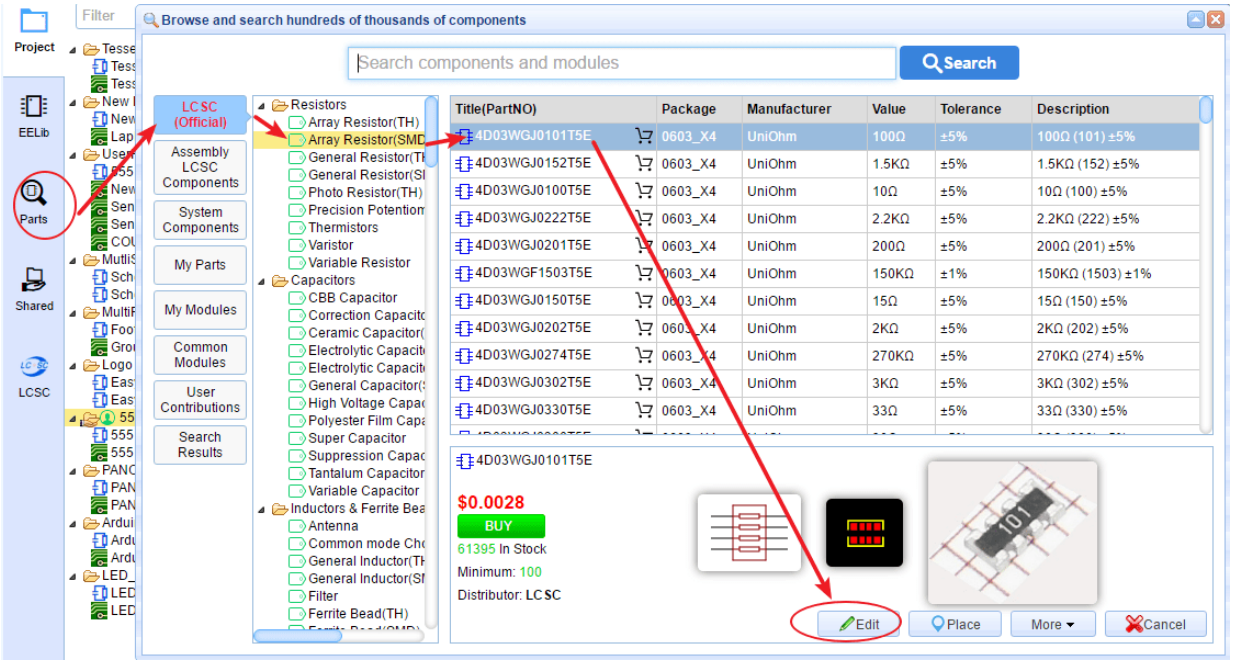
Contributor

This is your registered user name. Other EasyEDA's users will remember your contributions!

Edit SchematicLibs

When you feel the Schematic Libs can not be satisfy for you, you can edit it.

Via **"Parts" > "Search Part/My Parts/LCSC Parts/System Components/User Contributions" > Select Schematic Lib > Edit**



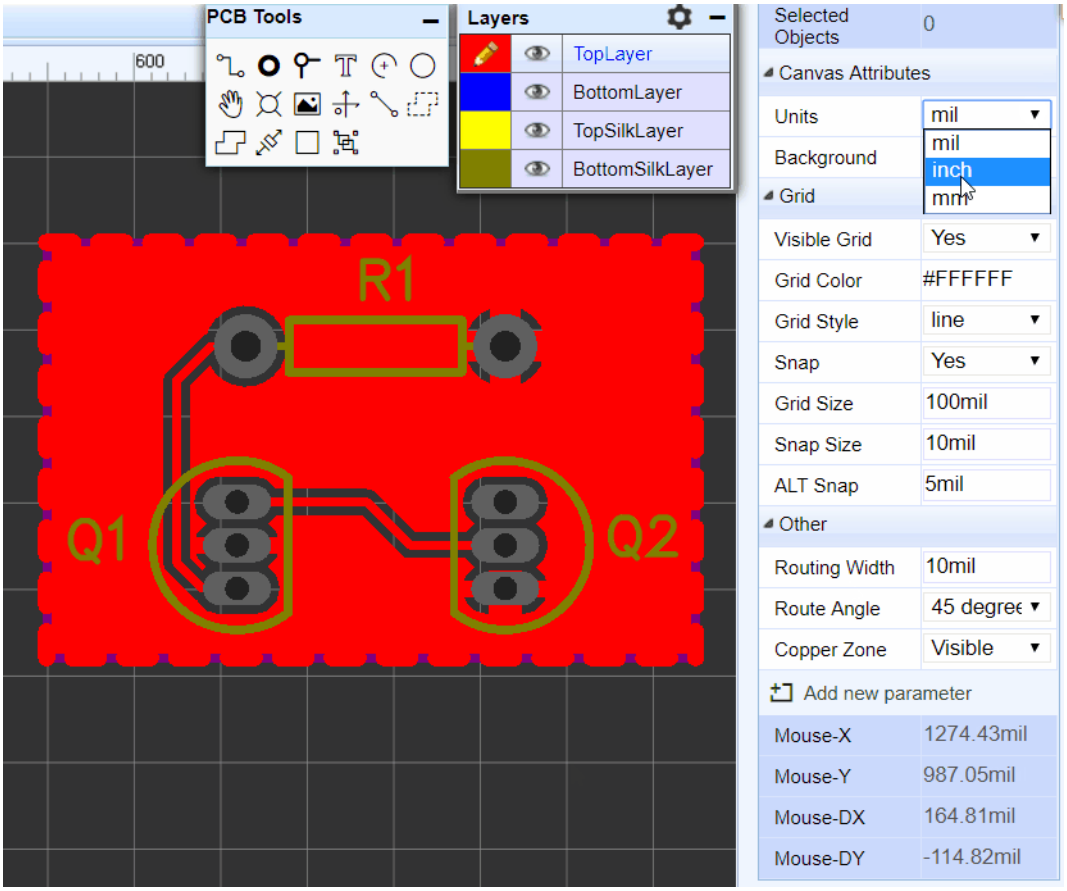
when you finish and save , it will be saved to your personal libraries "My Parts" and become your personal libraries.

PCB Design Editor

After the initial conversion of a schematic to PCB, it is time to learn how to manage EasyEDA's PCB Design Editor.

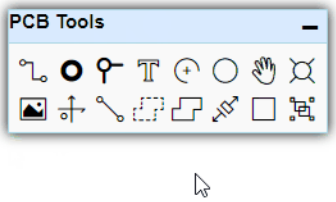
Canvas

Lots of PCB canvas attributes are the same as Schematic canvas attributes. The key is that you can set **units** in PCB canvas attributes.



PCB Tools

PCB tools provide many function to fulfill your PCB design requirement. Such as: Track, Pad, Via, Text, Arc, Circle, Move, Hole, Image, Canvas Origin, Connect Pad to Pad, Copper Area, Solid Region, Measure/Dimension, Rect, Group/Ungroup. etc.



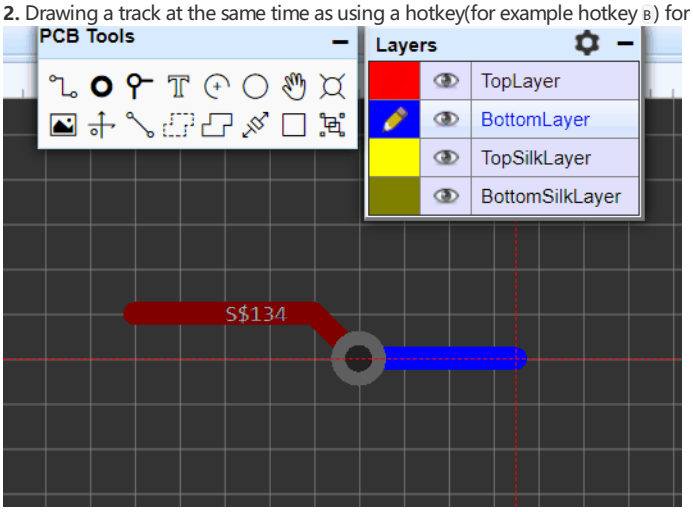
Track

In the schematic editor, we use Wire or the `W` Hotkey to connect Pins, in a similar way in the PCB editor, we use Track to connect Pads. Track allows you to draw PCB tracks and can be found on the PCB Tools palette or using the `W` Hotkey (not T: see above!).

Some Tips about Track.

1. Single click to start drawing a track. Single click again to pin the track to the canvas and continue on from that point. Right click to end a track. Double right-click to exit track mode.

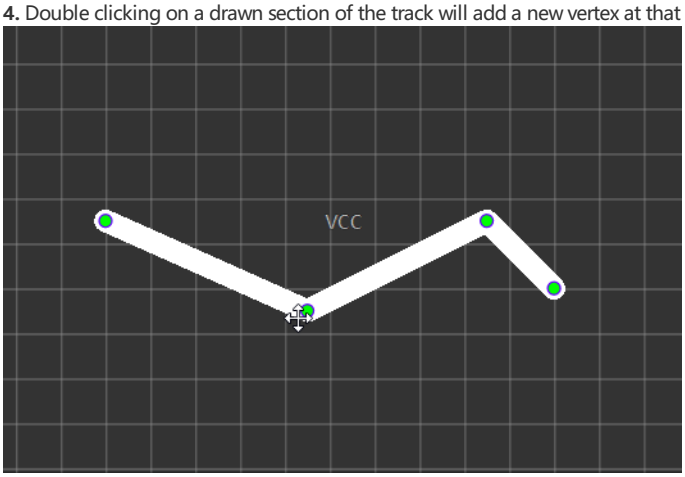
2. Drawing a track at the same time as using a hotkey(for example hotkey `B`) for changing the active layer will automatically insert a Via:



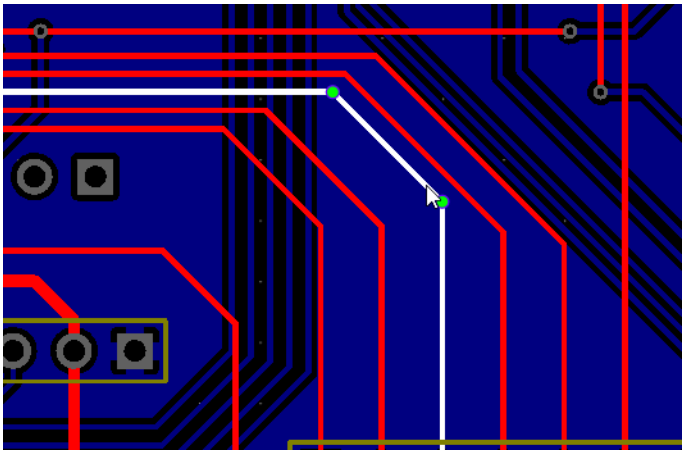
If you start drawing a track on the top layer - you will see it drawn in red - then press the B key to change to bottom layer and you will see EasyEDA insert a grey via and then the track will continue being drawn but now on the bottom layer in blue.

3. Pressing the `+` or `-` Hotkeys when drawing the track will change the width of the track on the fly.

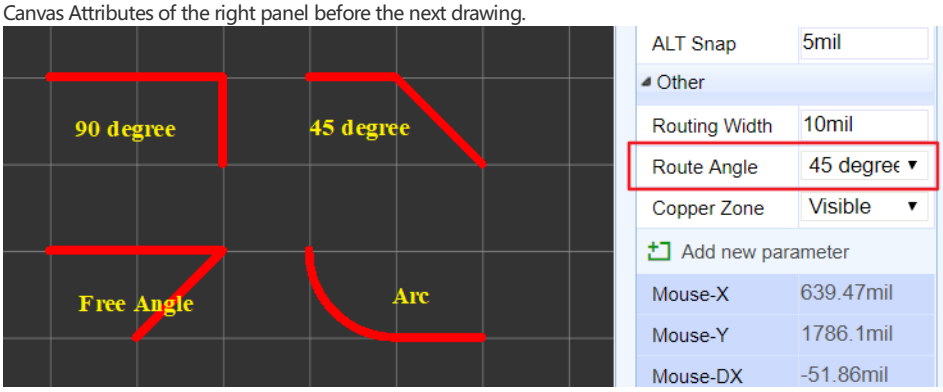
4. Double clicking on a drawn section of the track will add a new vertex at that point. You can drag the vertex to form a new corner.



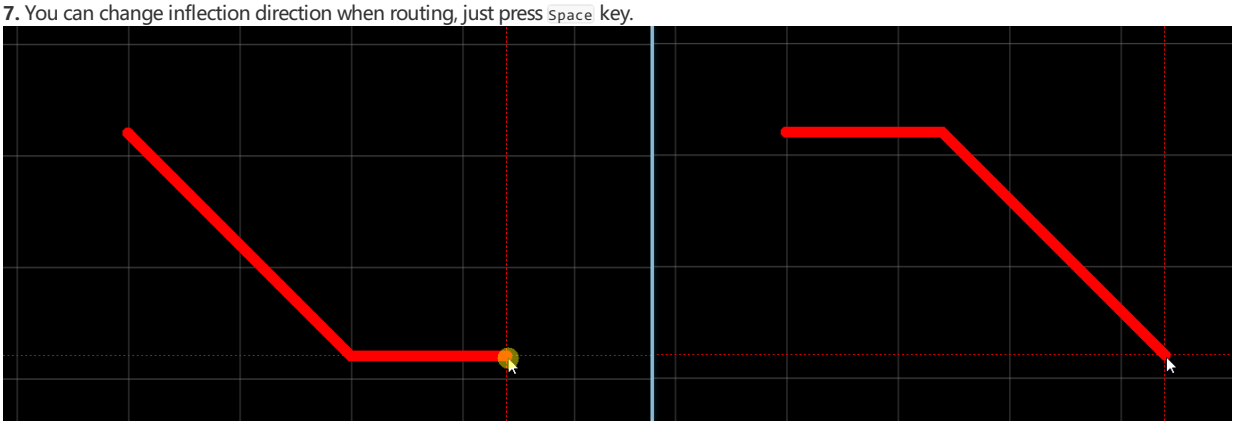
5. Click to select the track and then Click and Drag on a segment of the track to adjust the segment between vertices.



6. Pressing the `L` Hotkey when drawing the track will change the track's Route Angle on the fly. And you can change Route Angle on the Canvas Attributes of the right panel before the next drawing.

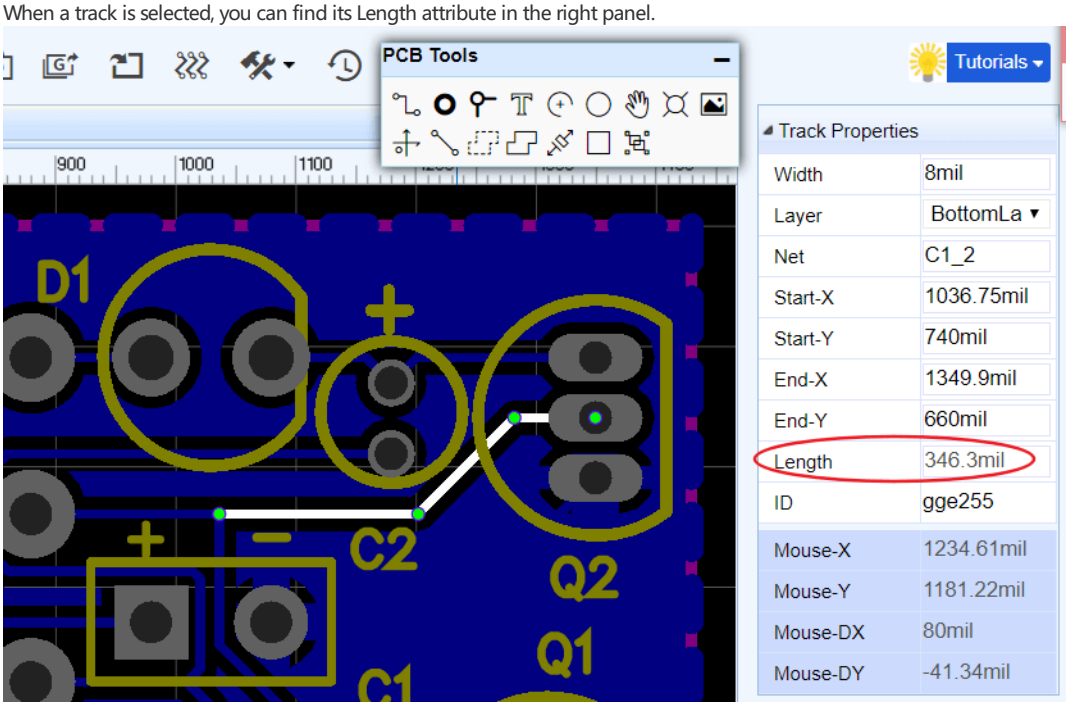


7. You can change inflection direction when routing, just press `Space` key.



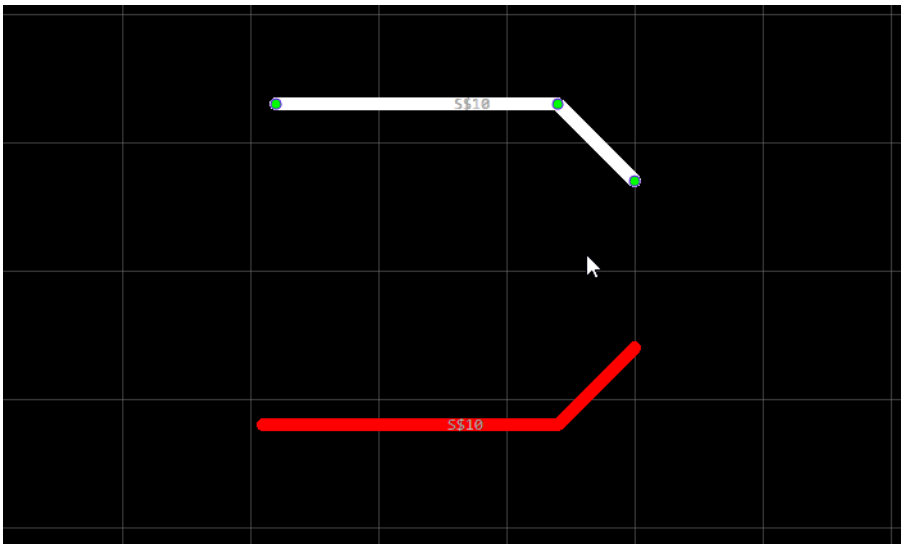
Track Length

When a track is selected, you can find its Length attribute in the right panel.



Delete a Segment from a Track

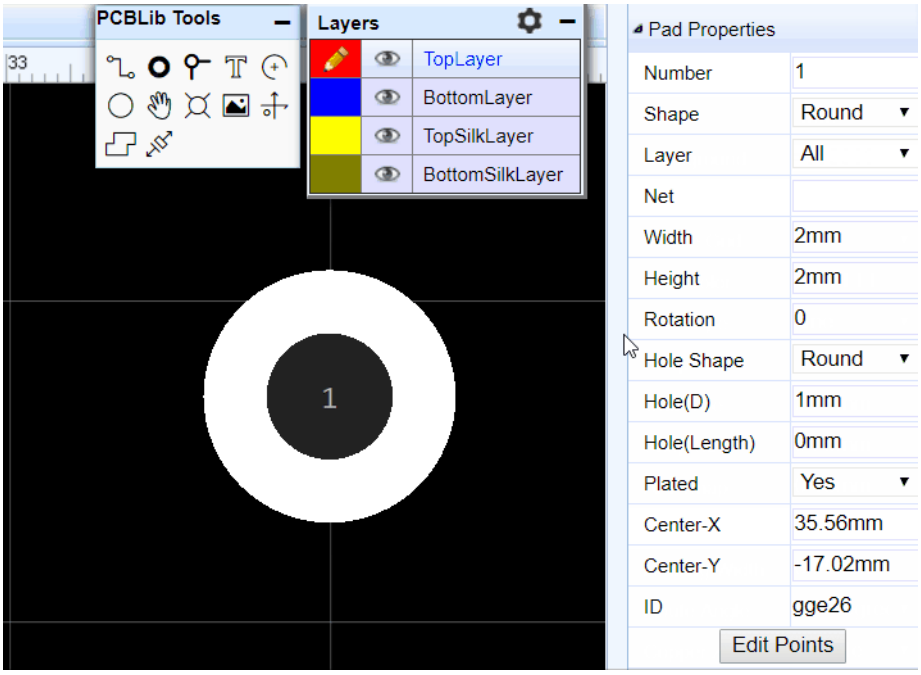
In lots of other EDA tools, the track is segment line, but in EasyEDA, the track is polyline. Sometimes, if we want to delete a segment, we must delete the whole track and route again. Now we provide a better way to do this. Move your mouse to the segment which you want to delete, click it, then hold `SHIFT` and **double click it**. the segment will be removed.



Pad

You can add pads using the Pads button from the PCBLib Tools palette or using the `P` hotkey.

After selecting one of the pads, you can view and adjust its attributes in the right hand Properties panel.



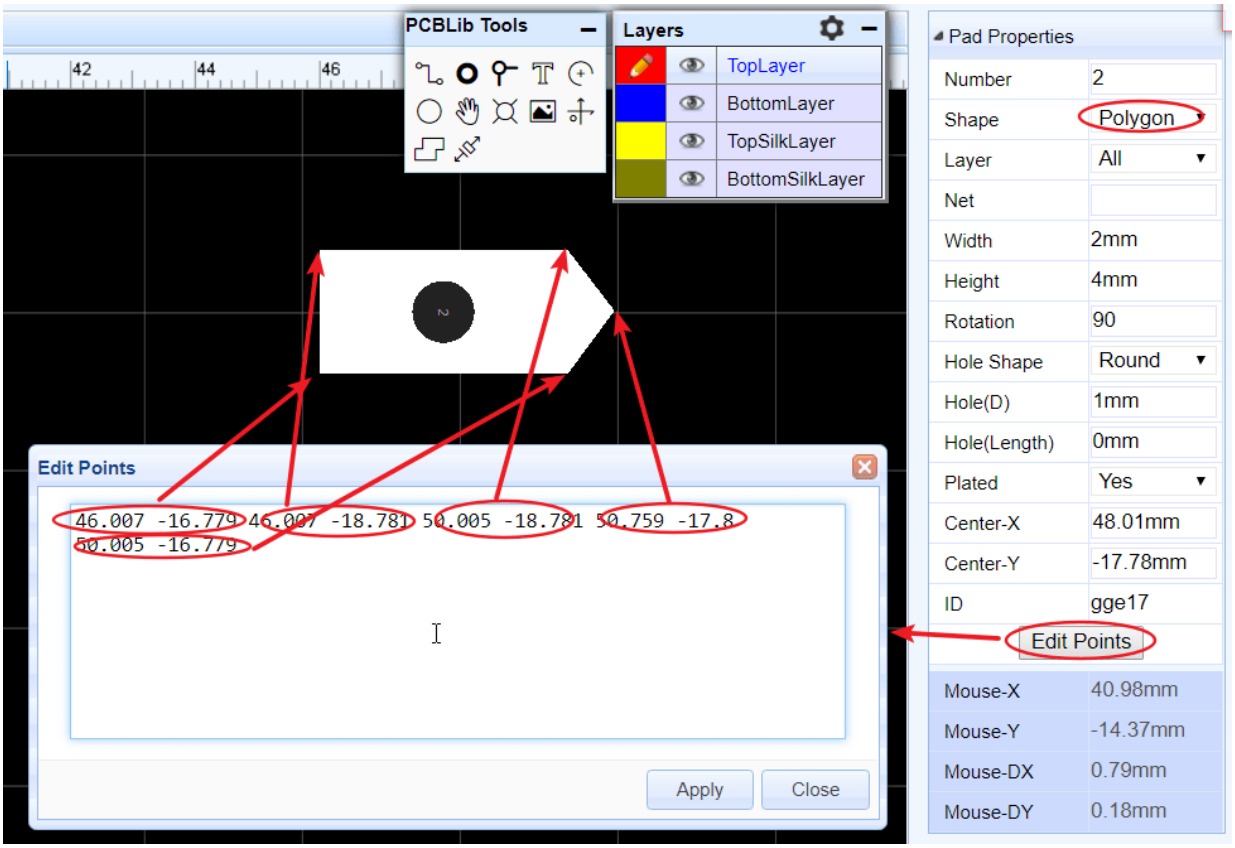
Number: Remembering the pin numbers you set in the schematic symbol in your Schematic Lib: to connect those schematic symbol pins to the pads in your PCB footprint, the pad numbers you set here in the PCB Lib footprint must be the same.

Shape: Round , Rectangular , Oval and Polygon.

EasyEDA supports four shapes: `Round` , `Rectangular` , `OVAL` and `POLYGON`.

- `OVAL` PAD will give your more space.
- `POLYGON` PAD will let you to create some strange pad.

Like in the image below, you can edit the PADs points when you select a `POLYGON` PAD



Layer: If the pads are part of a **SMD** footprint, you can set it to **Top layer** or **Bottom layer**. For through hole components you should set it to **All**.

Net: You don't need to enter anything here because at present this footprint is not connected to anything in a circuit.

Width and Height: When the shape is set to Round, Width will equal Height.

Rotation: Here you can set the Pad's rotation as you want.

Hole(D): This is the drill hole **diameter** for a through hole pad. For a SMD Pad, set this to **zero**.

Center-X and Center-Y: using these two attributes, you can set the pad's position with more precision, compared to using the mouse.

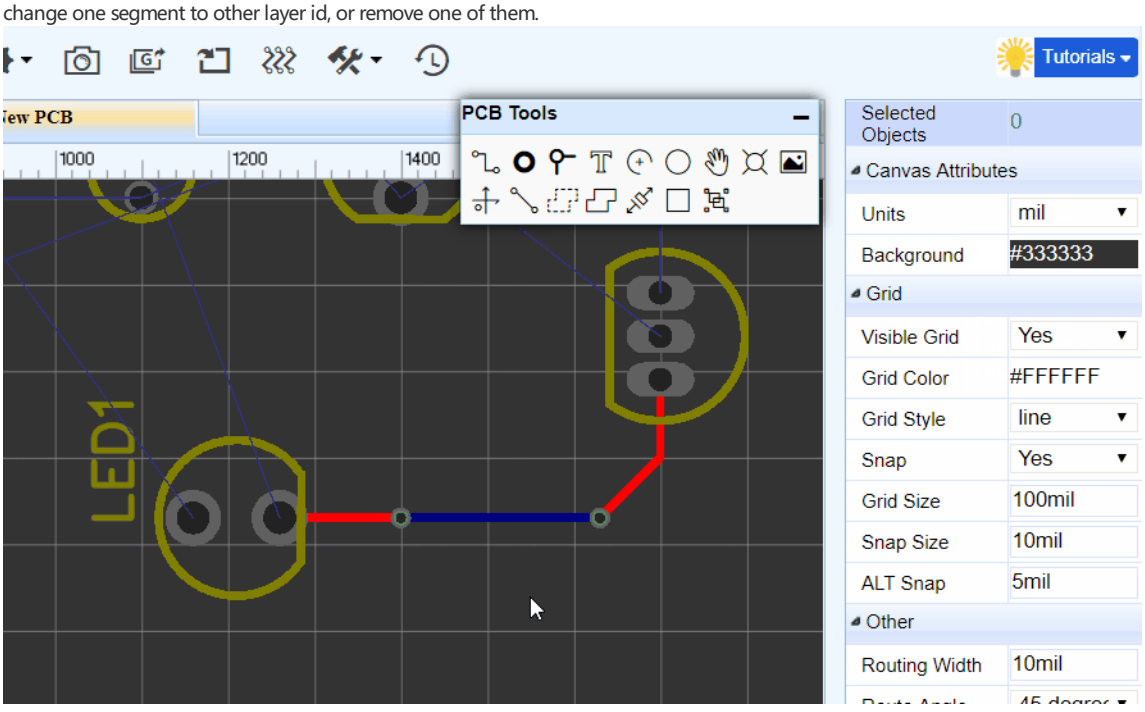
Plated : Yes or No.

Via

When you want to lay a multilayer PCB, you need to add Vias for nets getting through layer and layer.

Place a Via On a Track

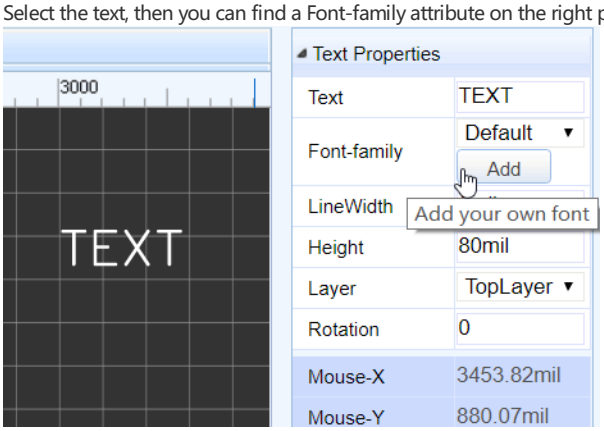
When placing a `via` on a track, the track will be cut to two segments. Placing two vias on a tracks, you will get three segments, then you can change one segment to other layer id, or remove one of them.



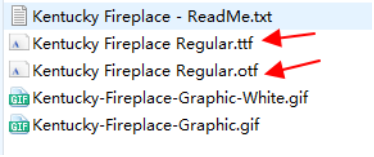
Text

You can add more fonts from your computer or download some free fonts:www.1001freefonts.com .

Select the text, then you can find a Font-family attribute on the right panel like in the image below.



Click the add button, then choose the font, the font file must be `ttf` or `otf`.

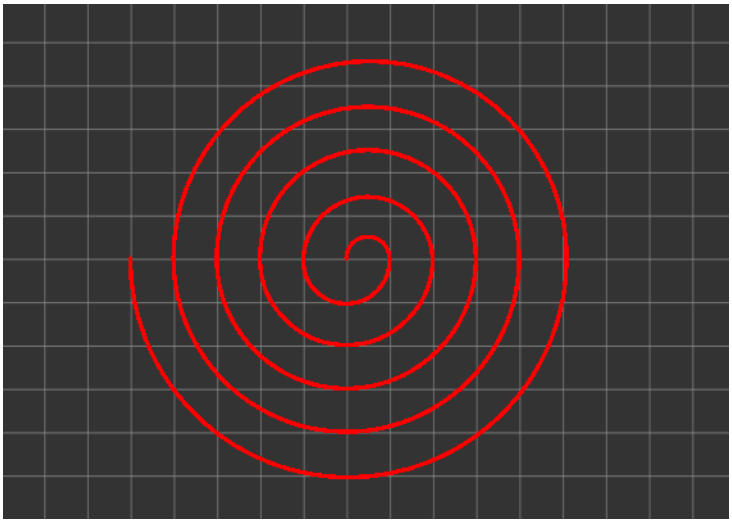


So you can add any fonts by yourself. EasyEDA doesn't cache the font on our server, so if you close the editor, you need to add the font again by yourself.

Note: *If you use the other font, the `LineWidth` attribute is useless, because it will be automatically set by changing the `Height`.*

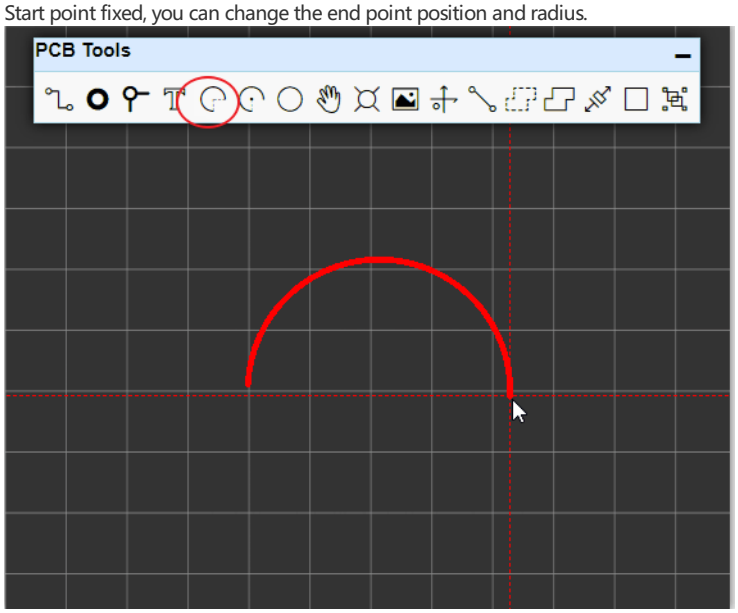
Arc

You can draw many Arcs with different sizes, it's easy to create a pretty cool PCB as you like.

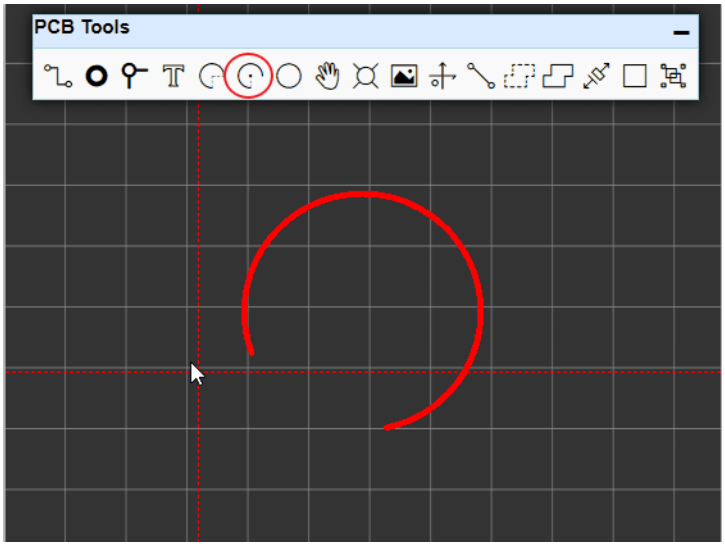


EasyEDA provides two Arc tools:

- Start point fixed, you can change the end point position and radius.



- Center point fixed, you can change the radius.



Circle

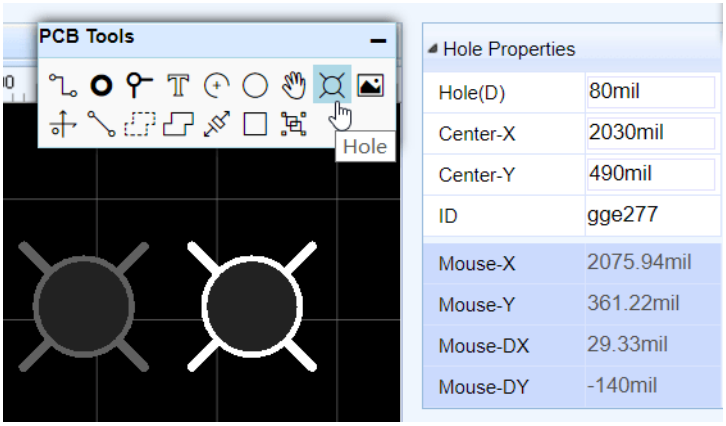
You can draw a circle in PCB , but it can only be drawn at SilkLayer and Document Layer. If you want to draw a circle at TopLayer or BottomLayer, please use Arc.

Move

This option is same as schematic's drag.

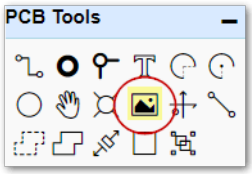
Hole

There were lots of users that didn't know how to use PAD or VIA as a HOLE, they asked EasyEDA for help, so EasyEDA added a HOLE TOOL in the PCB toolbar.

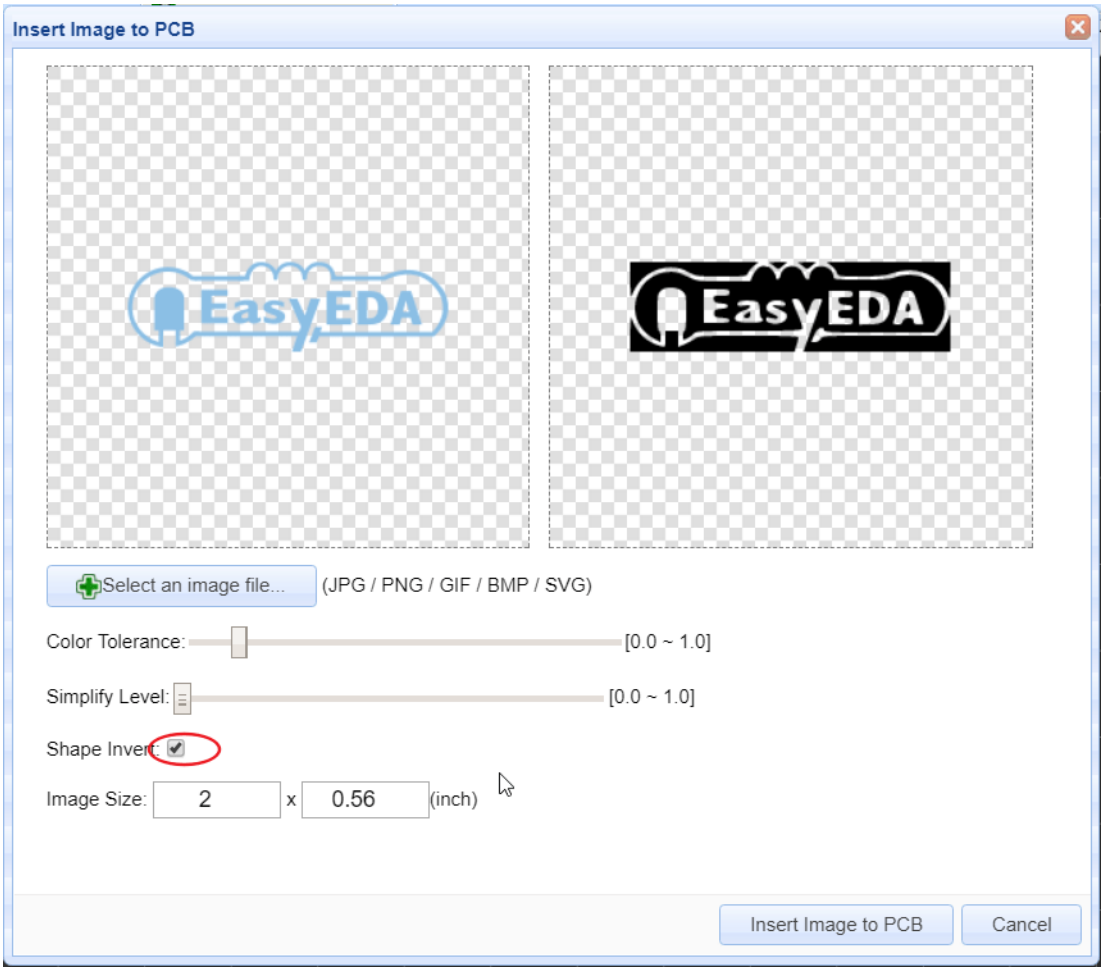


Image

On PCB and PCB Lib editor, there is a nice feature on the PCB Tools bar.



After clicking on the image icon, you will see the Insert Image window as below.

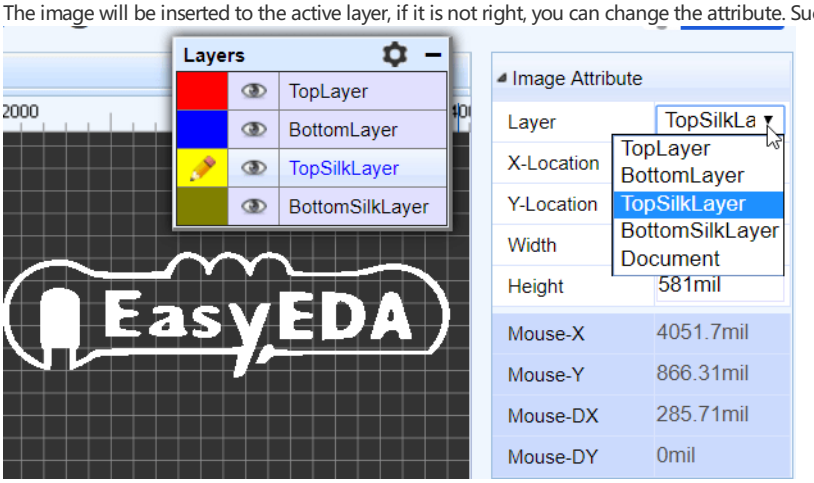


In this dialog, you can choose your favorite image, EasyEDA support `JPG`, `BMP`, `PNG`, `GIF`, and `SVG`. Unlike some other EDA tools which only support a Monochrome Bitmap image, EasyEDA supports full color, but Monochrome Bitmap is welcome.

You can adjust the color tolerance, simplify level and reset the image size there.

And you can select shape invert.

The image will be inserted to the active layer, if it is not right, you can change the attribute. Such as TopSilkLayer.

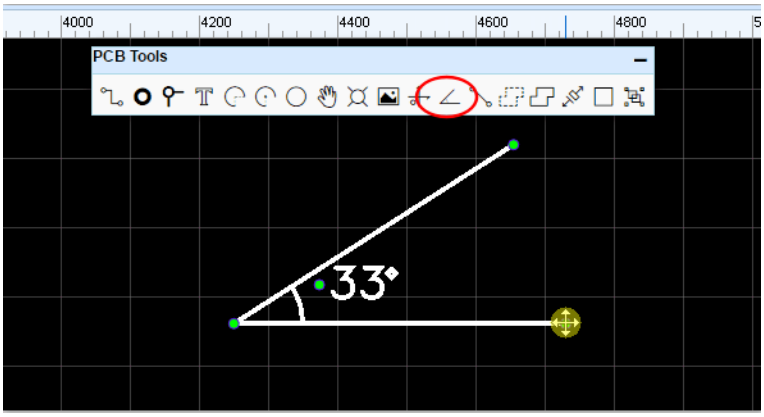


Canvas Origin

This option is the same as schematic's Canvas Origin.

Protractor

We provide a protractor for PCB tools.

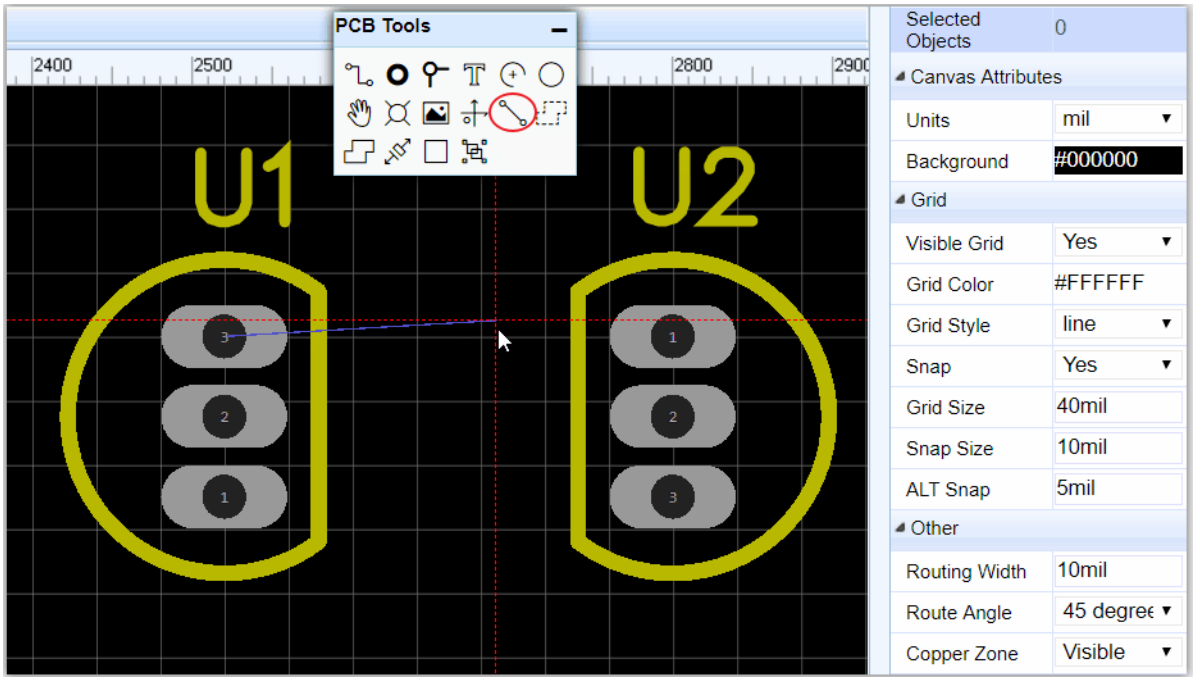


Connect Pad to Pad

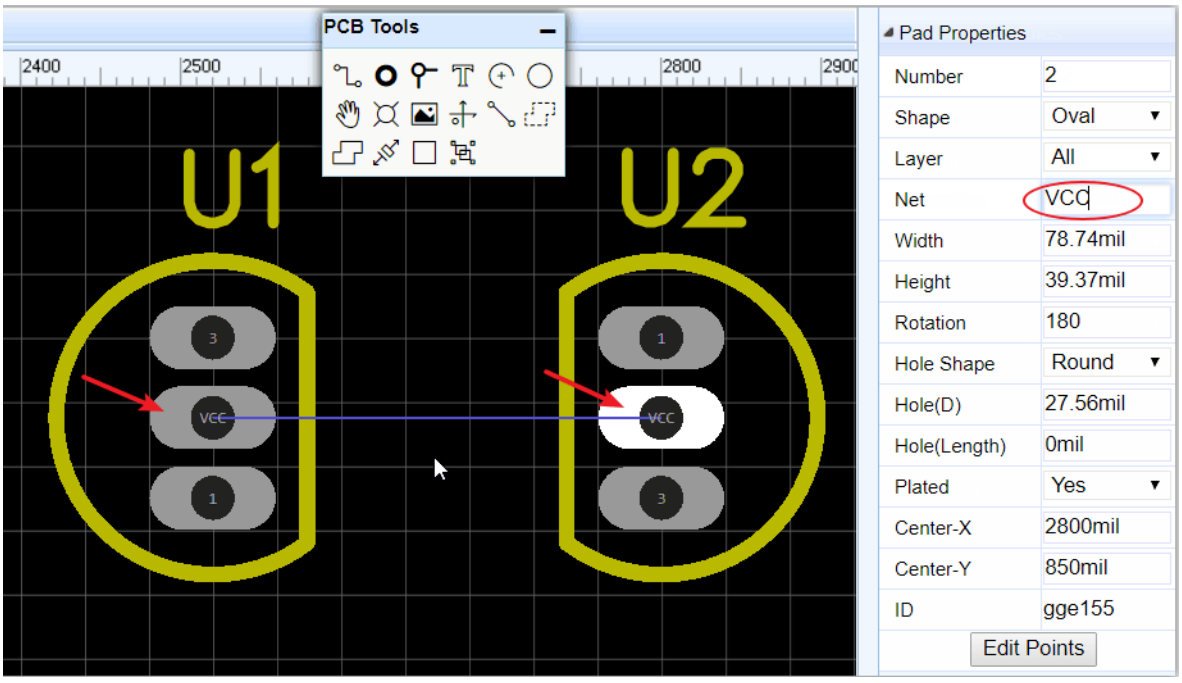
When creating a PCB without a Schematic, none of the pads on the Footprints have nets connecting them so there will be no ratlines.

Rather than try to track the pads from scratch, it is a good idea to connect them up by hand first using `Connect Pad to Pad` from the PCB Tools palette. This will help you to remember to track the pads correctly with fewer mistakes.

You could also do this by setting net names for all the pads: if the two pads are given the same net name then EasyEDA will understand that they are connected together and will automatically create a ratline between them.



Or you can set these two pads with the same net name at the right panel Pad Properties after you click the pad.

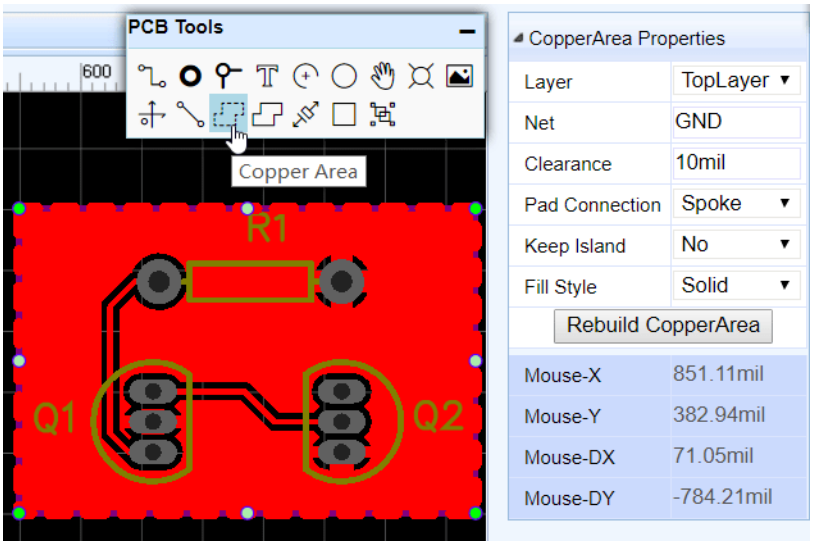


For more information about Ratline you can refer to the Ratline section.

Copper Area

Sometimes you will want to fill in or flood an area with copper. Usually this copper area will be connected to a net such as **GND** or a supply rail. You can draw the outline of a flood using the **Copper Area** button from the PCB Tools palette.

When selecting a copper area, you can find its attributes from the right hand **Properties** panels.



Layer: Bottom, Top, Inner1, Inner2, Inner3, Inner4;

Net: the net that the copper area is connected to;

Clearance: clearance of the copper area from other nets and floods;

Pad Connection: direct or spoke (i.e. a cross shaped heat shunt);

Keep Island: Yes/No. This keeps or removes any isolated areas of copper created as part of the flooding process. It is usually good practice to removes these unless you really need them to maintain a more even spread of copper (copper balance) on your PCB;

Fill Style: No/Filled. No removes the fill so that you can see the tracking more clearly;

After drawing the copper area, set the net it is to be connected to (floating copper areas are not recommended because they can cause EMC and Signal Integrity (SI) problems).

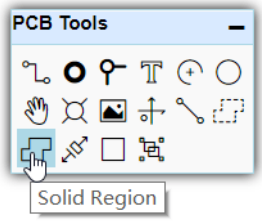
Lastly, don't forget to click the button Rebuild Copper Area to **rebuild** the flood.

Two Tips:

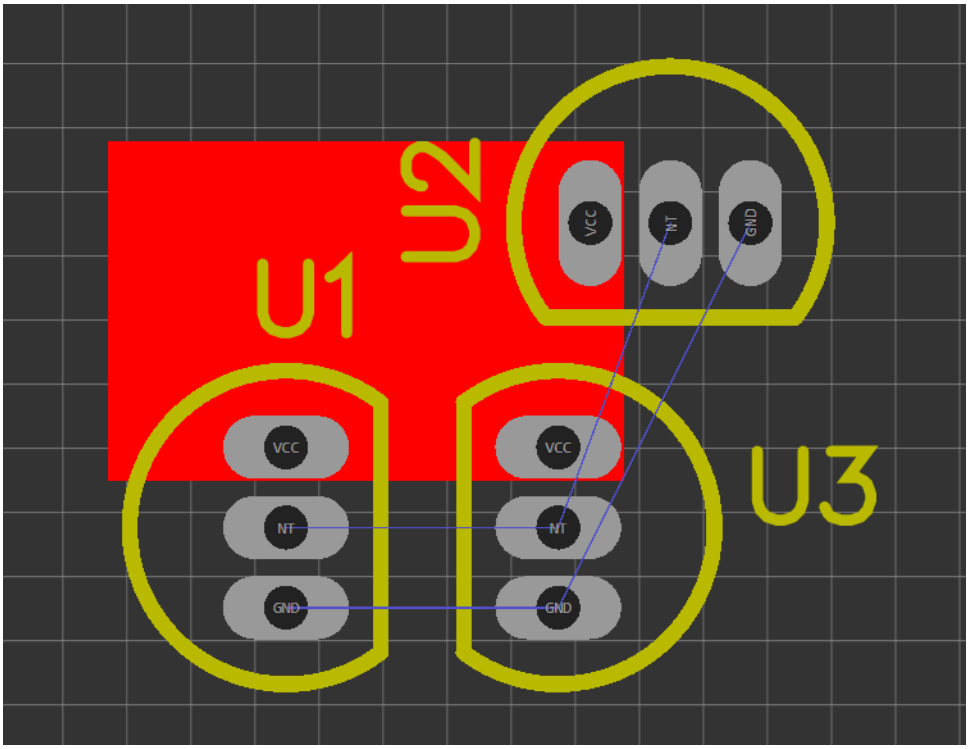
1. Hotkey `Shift+B` to build all of the copper areas.
2. Hotkey `Shift+M` to clear all of the copper areas.

Solid Region

EasyEDA has added a new tool Solid Region for PCB design

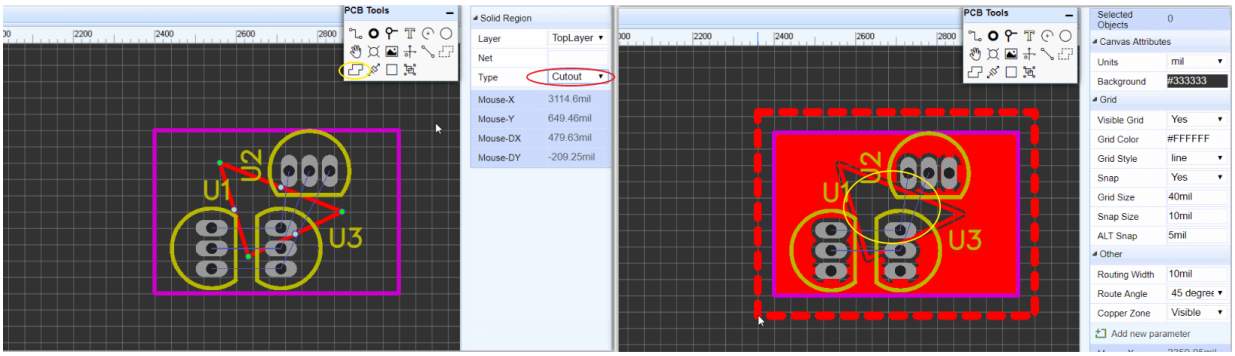


This is a very useful, quick way to connect Pads. You can draw a Solid Region to include all of these pads with same net name, then set the region to the same net name as the pads. It is like Copper Area but easier to use for small areas. To use Solid Region like this, set the Type attribute (in the right hand Properties panel) to Solid.



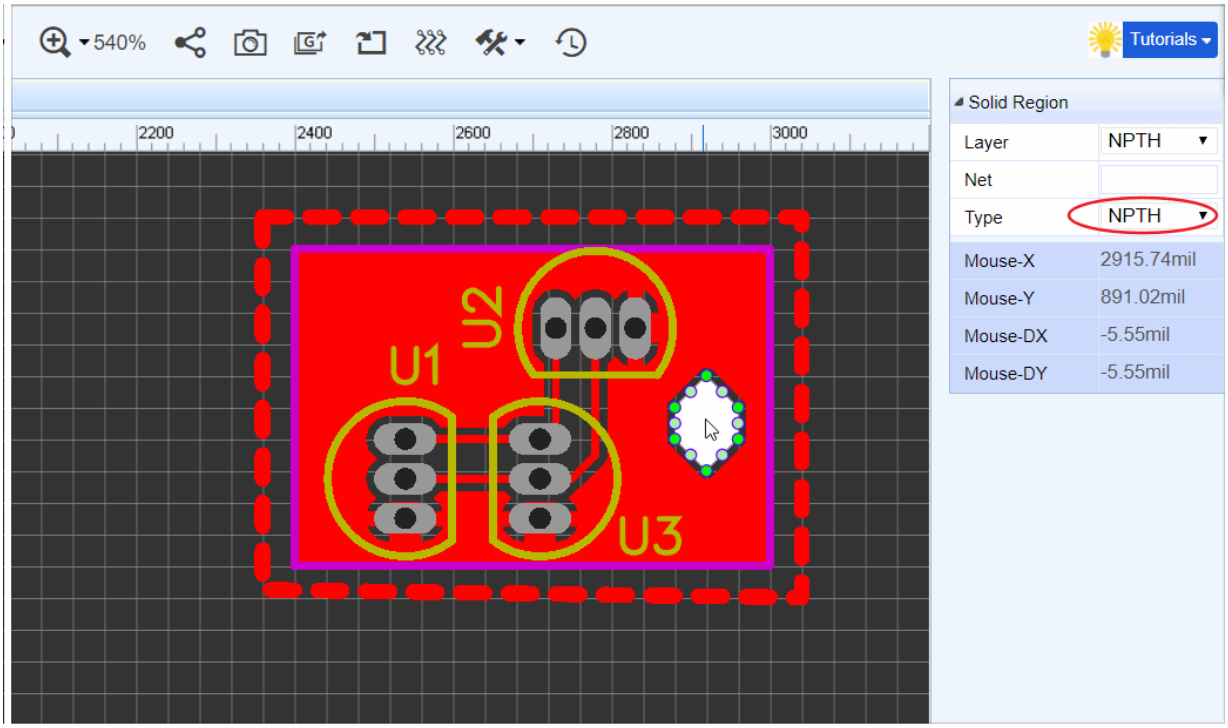
The Solid Region can also be used to create a cutout in a copper area.

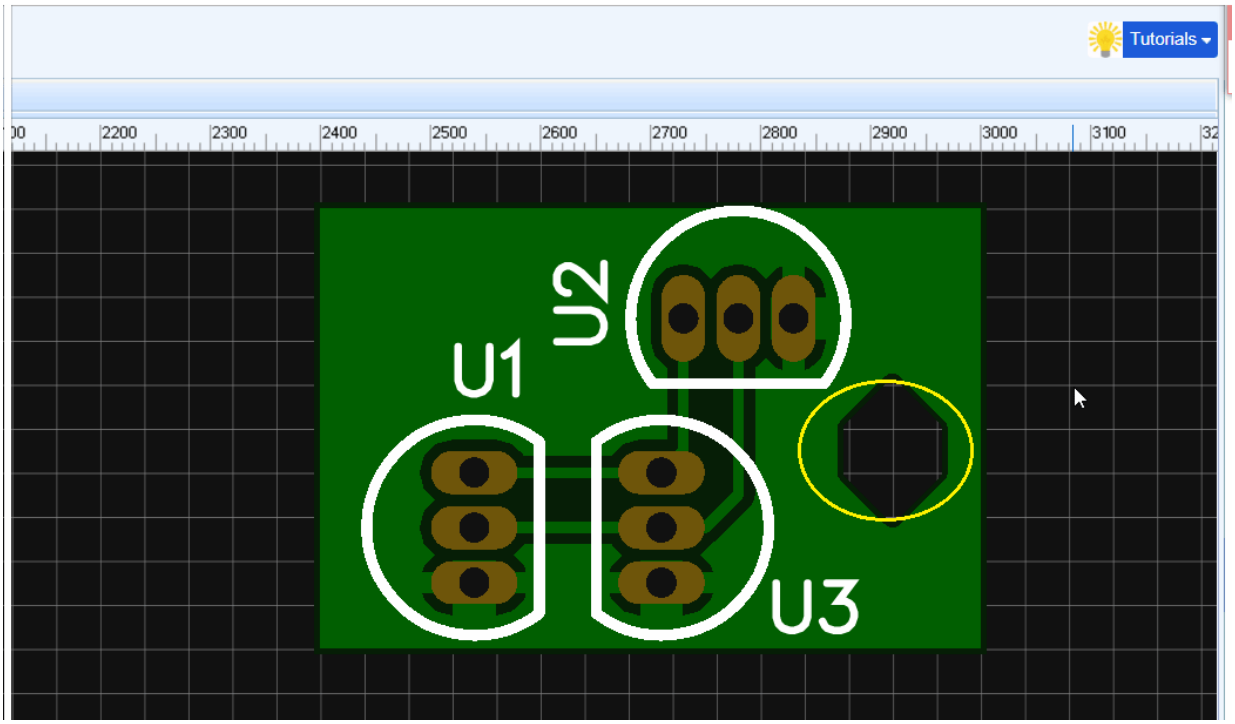
If you have a copper area but need an area inside it to not be filled then you can draw a Solid Region and set the Type attribute (in the right hand Properties panel) to Cutout , then this area will be free of copper, as shown in the image below:



Lastly, by setting the Type attribute (in the right hand Properties panel) to NPTH(Non Plated Through Hole), Solid Region can be used to create a *Non Plated Through Hole* of an arbitrary shape.

When the Gerber files are generated, an area defined by a Solid Region set to a Type NPTH in the PCB editor will create an area defined to be a NPTH hole and you can see it in the PCB photo view as below:

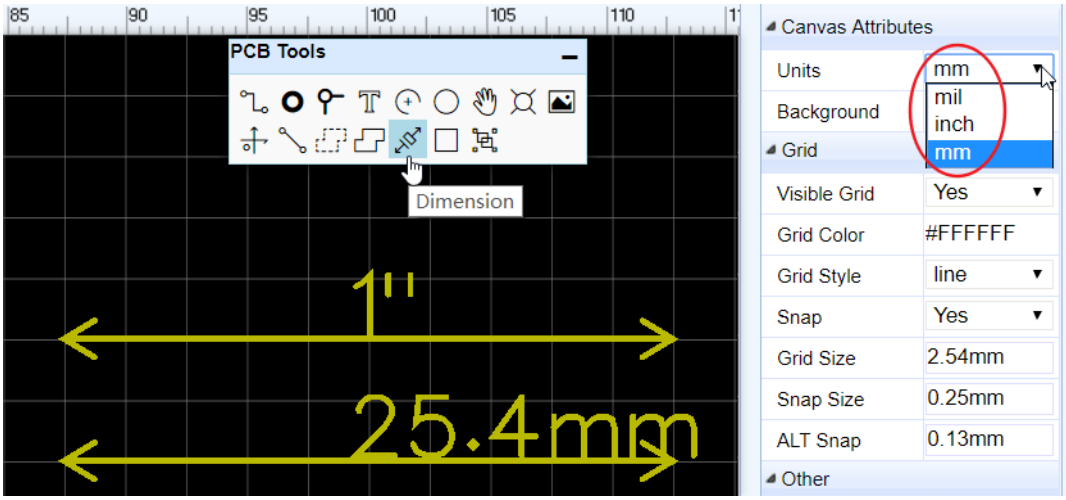




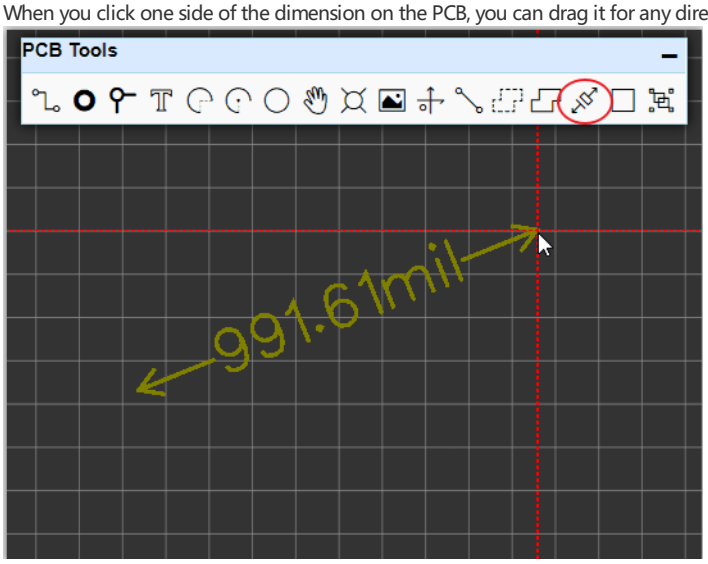
Measure/Dimension

Making and adding measurements is useful in PCB design. EasyEDA provides two methods to do this.

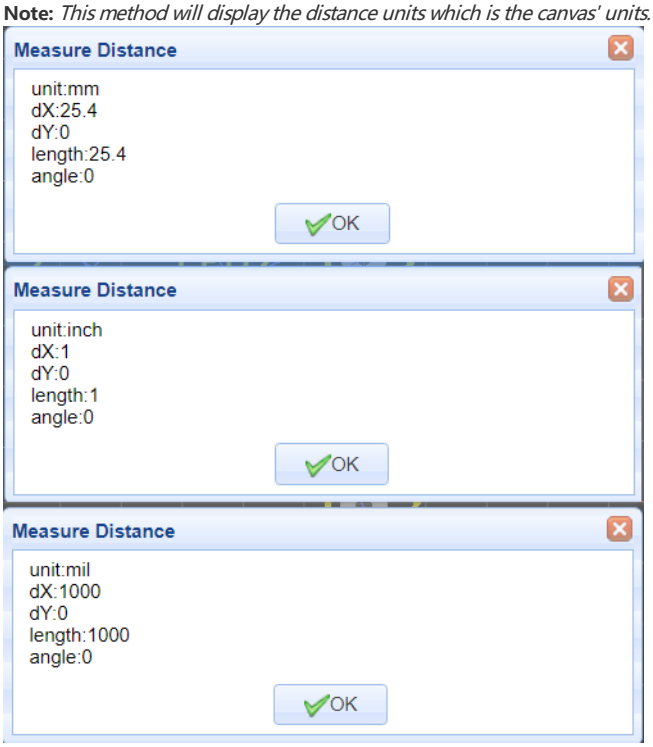
1. Dimension tool in the PCB Tools palette:
   This tool can show three units on the canvas, milliliter, inch and millimeter.



When you click one side of the dimension on the PCB, you can drag it for any directions or change its length.

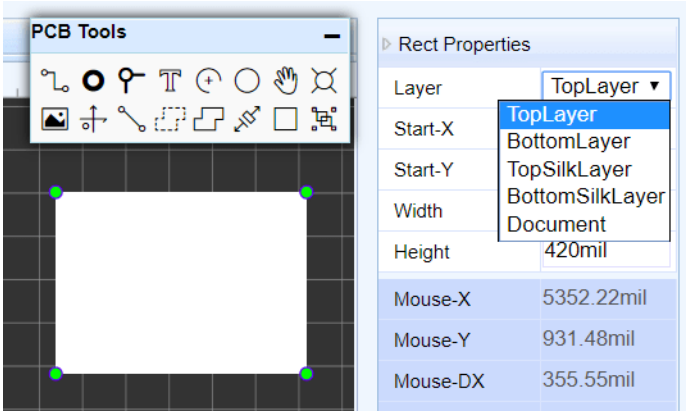


1. Measure a distance using **M** Hotkey: press **M**, Or Via: **Super menu** > **Miscellaneous** > **Measure Distance**, then click the two points which you would like to measure.
   Note: *This method will display the distance units which is the canvas' units.*



Rect

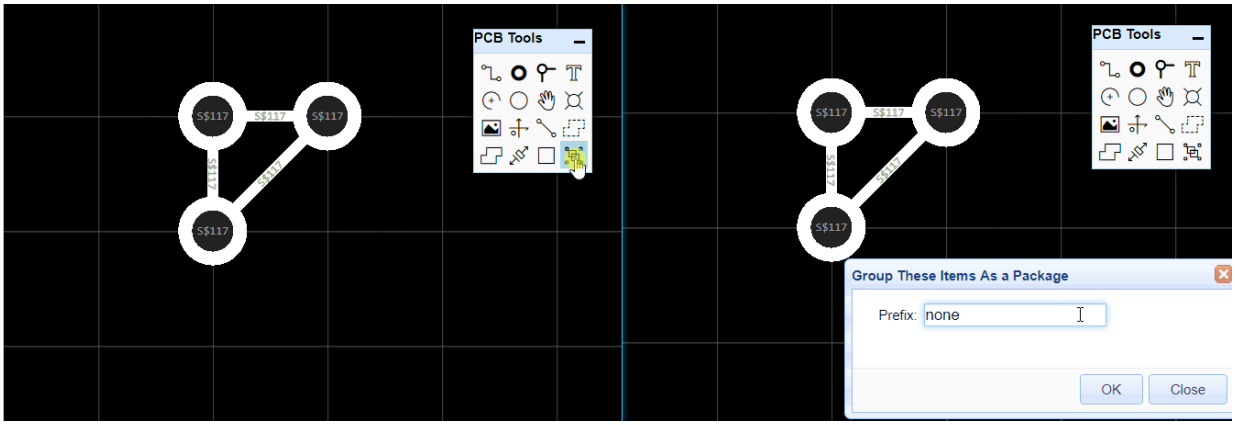
It looks like a Solid Region, but it can't be set Nets and you can't set the Layer as NTPH.



Group/Ungroup

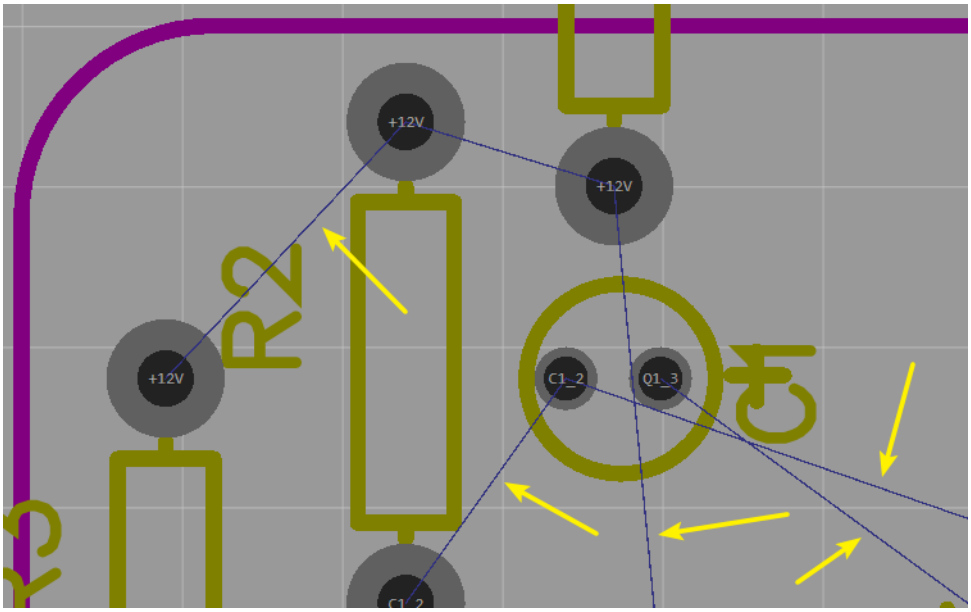
Just like Group/Ungroup in the Schematic Editor can be used to create a schematic lib symbol, you can use Group/Ungroup from the PCB Tools palette to create a PCB Lib footprint in the PCB editor.

For example, place Tracks and Pads on the canvas, then select all of them and click **Group/Ungroup** to group them like in the image below:



Ratline

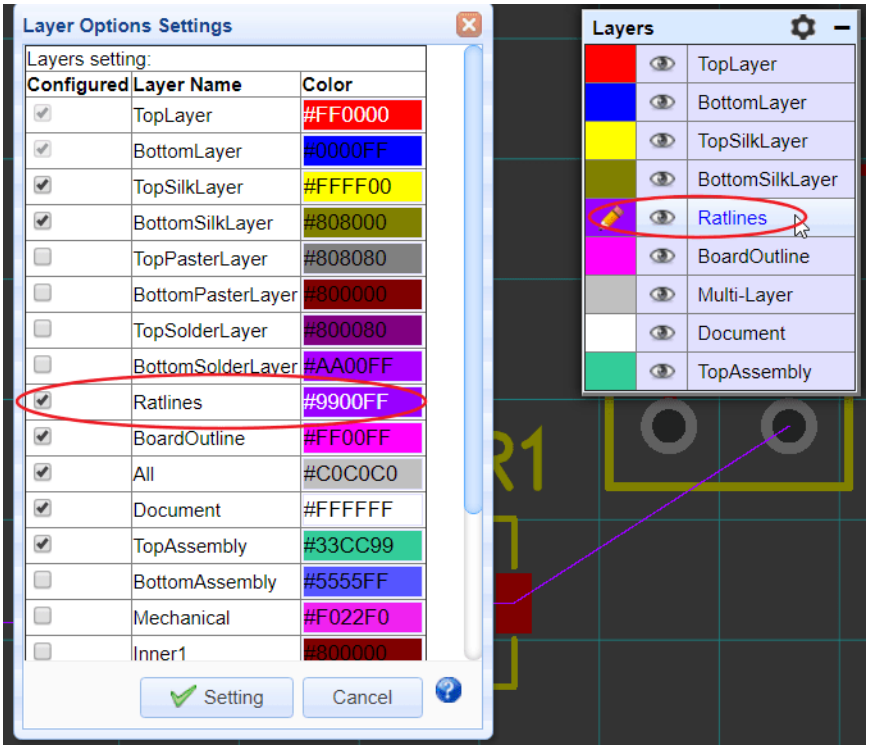
When you lay the track in the PCB, Between PIN and PIN as they have the same net name, a Ratline will be automatically shown among them to reveal that they can be connected with a track.



If you want some type of ratline to not show on the PCB editor, you can untick the net you like in the design manager, as below deselect `+12V`: If you still draw a track in `+12V` after deselecting, canvas will not display this track , but it will show a text with `+12V` as below.

Based on this skill， you don't need to lay GND net before copper area in the PCB.

If you want to check the ratlines with highlight, you can click the pencil on the Ratlines Layer as below, and you can change the ratline's color.

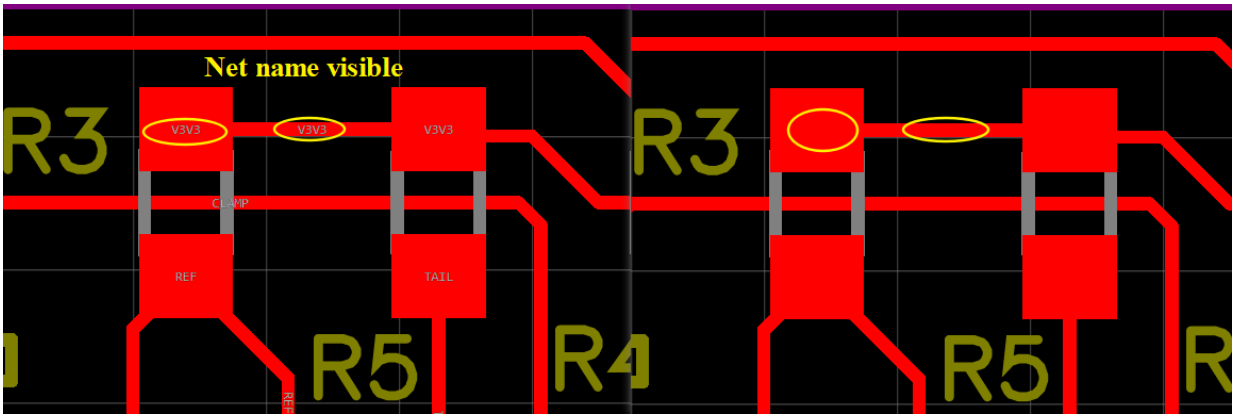


PCB Net

Net Name Visible

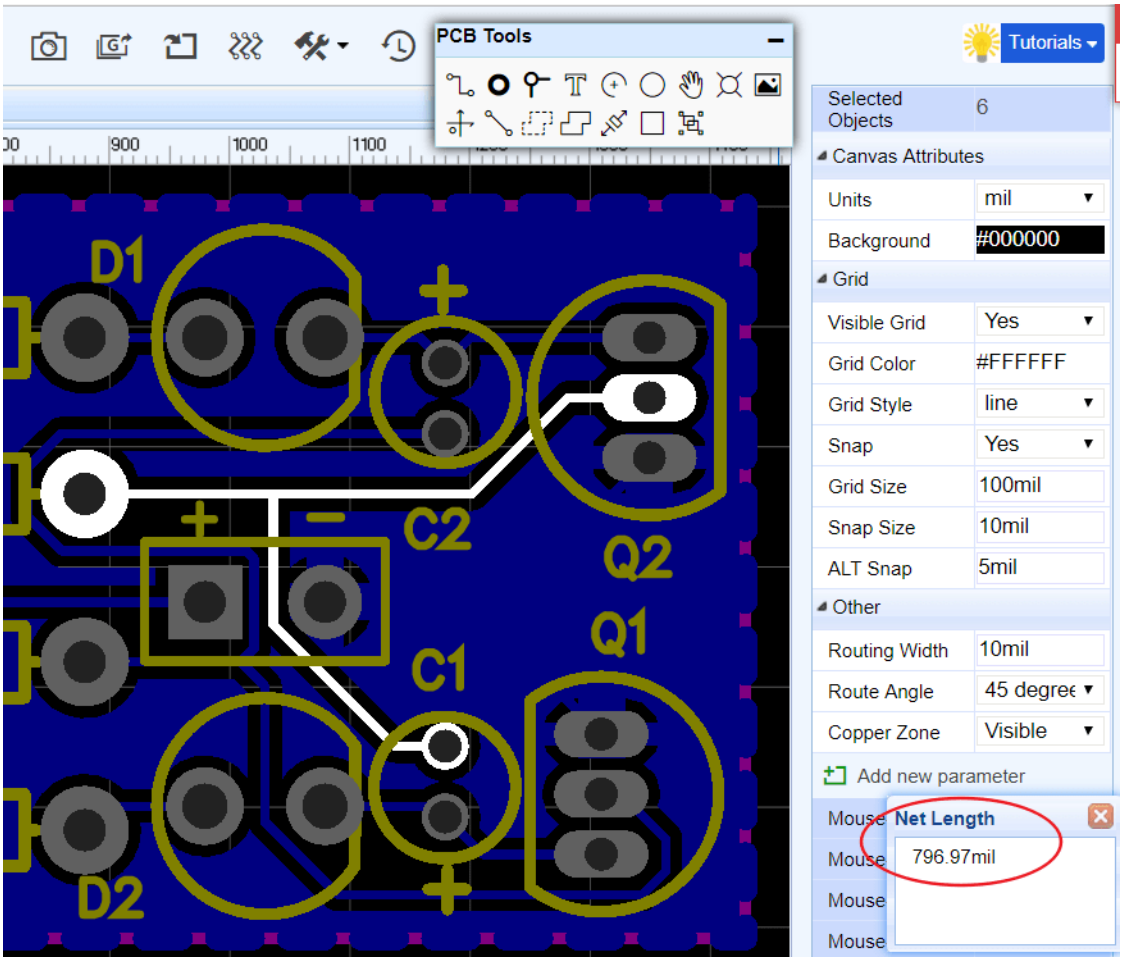
PCB editor can display net name in the track or Pads, if you don't need this feature, just need to turn it off via：

Super menu > **View** > **PCB Net Visible**, or press hotkey `Q` .



Net Length

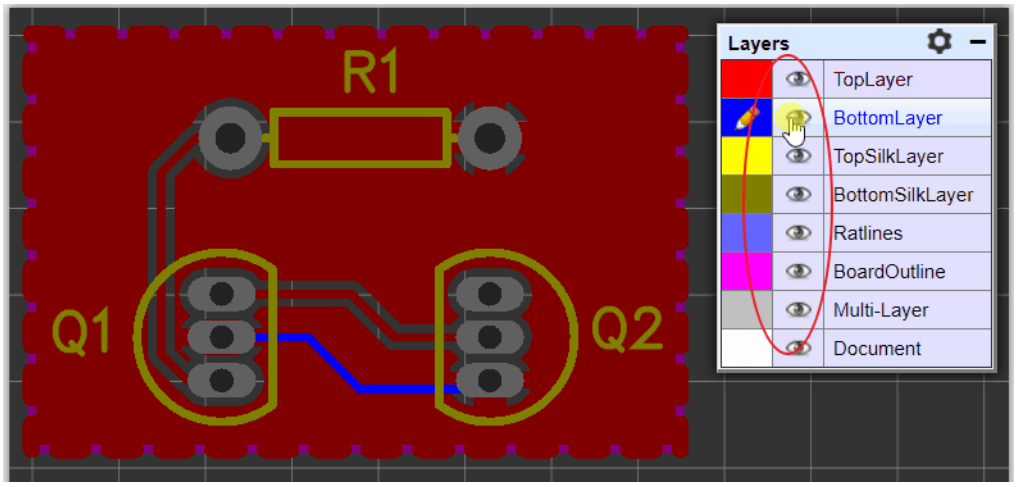
After selecting a track, and then pressing `H` key, EasyEDA will highlight the whole net and pop a message box to tell you the whole net's length. like in the image below



Layers Tool

Active Layer: The colours of the layers in the **Layers Tool** are defined in the Layer Options Settings. To work on a layer then you must make it the Active layer. To do this; click on the coloured rectangle representing the required layer. The pencil icon in the coloured rectangle indicates that this is the active layer.

Show/Hide layers: click on the eye icons to show/hide layers.



HotKeys for layer activation:

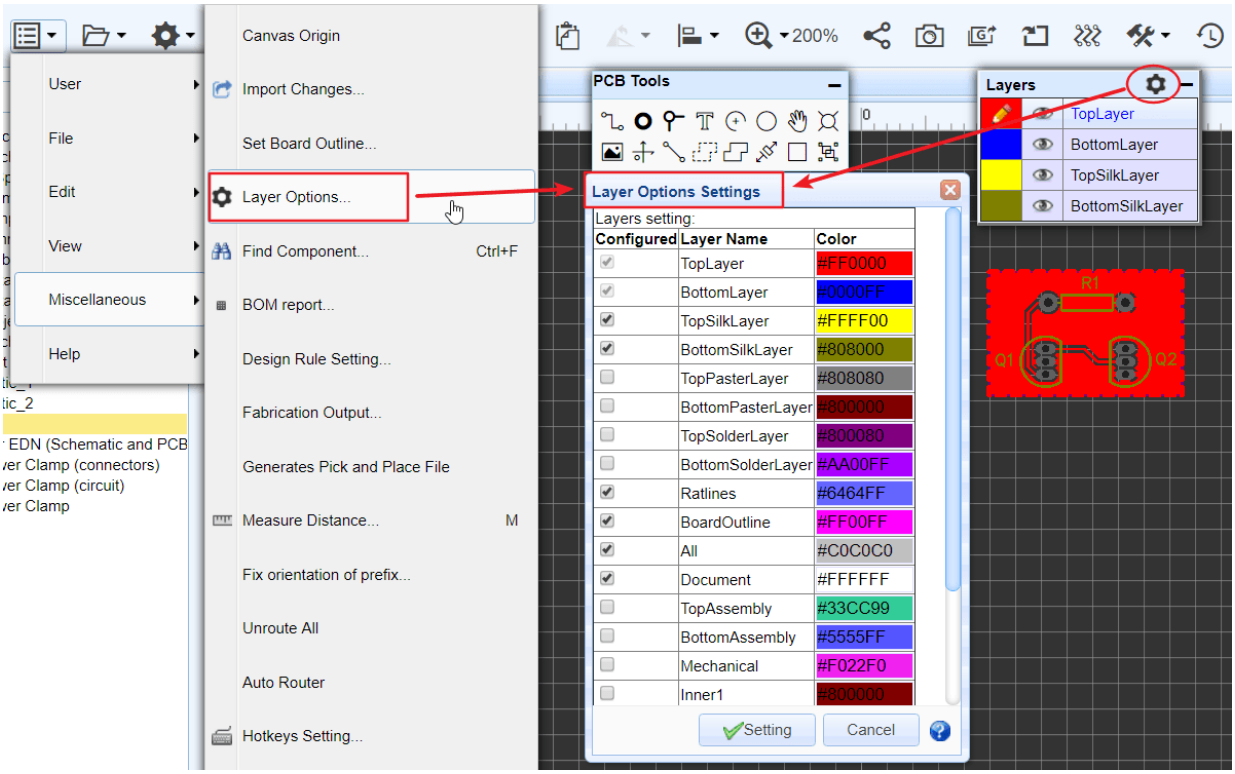
- **T:** Top Layer is active
- **B:** Bottom Layer
- **1:** Inner1 Layer
- **2:** Inner2 Layer
- **3:** Inner3 Layer
- **4:** Inner4 Layer

Layer Setting

Via **Super menu** > **Miscellaneous** > **Layer Options...**, Or Click **Layers**' gear icon.

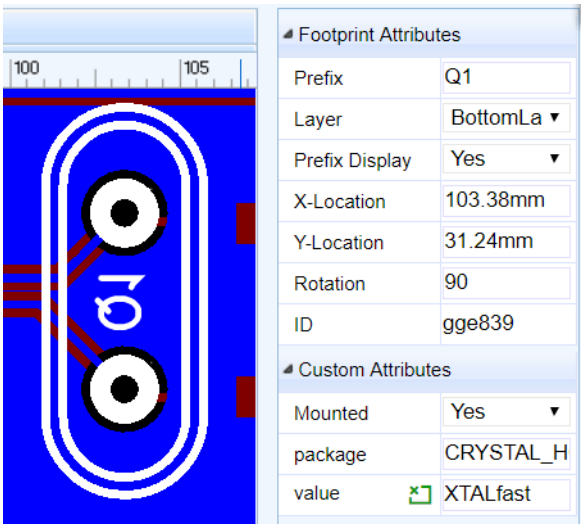
You can find the Layer Options Settings dialog.

In this dialog, you can change the layer's Color and configure which layers are shown in the Layers Tool. If you plan to design a PCB with more than 2 layers, then you must tick Inner1 and Inner2 for a 4 layer PCB plus Inner3 and Inner4 for a 6 layer PCB.



Footprint attributes

When selecting a Footprint, you can find its attributes at the right hand Properties panel.



Layer: You can set a footrpint to be on the TopLayer or BottomLayer.
Note: The footprint mirrors when swapping layers.

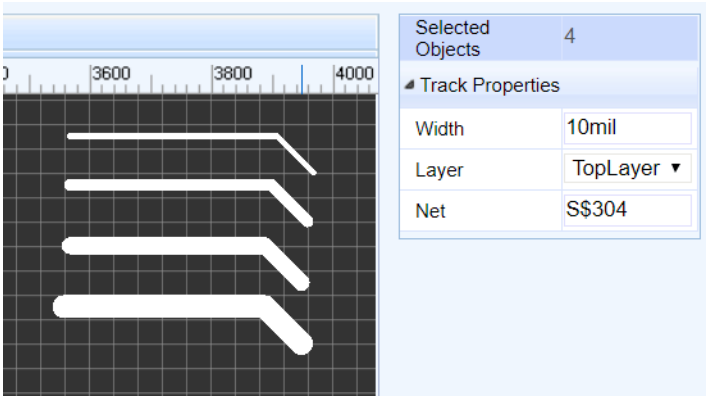
X-Location and Y-Location: Moves the origin of the footprint to a precise position.

Rotation: Rotates the footprint about its origin over the range from 0o to any angle in 1o steps (visually of course multiples of 360o will appear identical).

ID: EasyEDA will assign a unique ID for each footprint automatically, you can't modify it.

Change Attributes in Batch on PCB Editor

Sometimes, we need to change some attributes of multiple objects together, such as the track width, hole size and font size.
Now, you can select them and do some changes. Taking the track for an example. If you select 3 tracks, now you can change their `Width`, `Layer`, `Net` together.



You could also use it with other items such as `Pad`, `Via`, and `TEXT`.

Layout A PCB Without Schematic

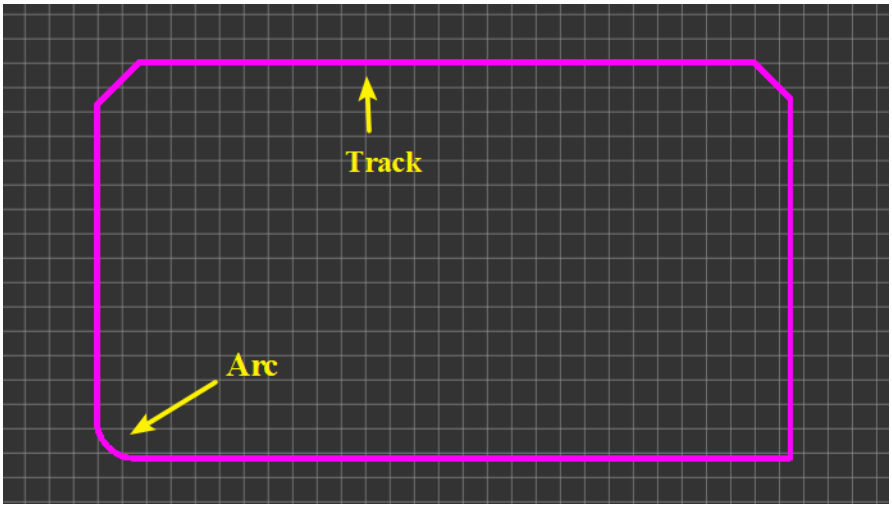
For some small PCB projects, maybe you don't need a schematic. EasyEDA allows you to lay the PCB directly from the PCB Editor.

Start a new PCB and you can add footprints directly from the PCB Libs from Left Navigation Panel **Parts** and then just track them.

For setting pad to pad connections, you can check the above section : Connect Pad to Pad

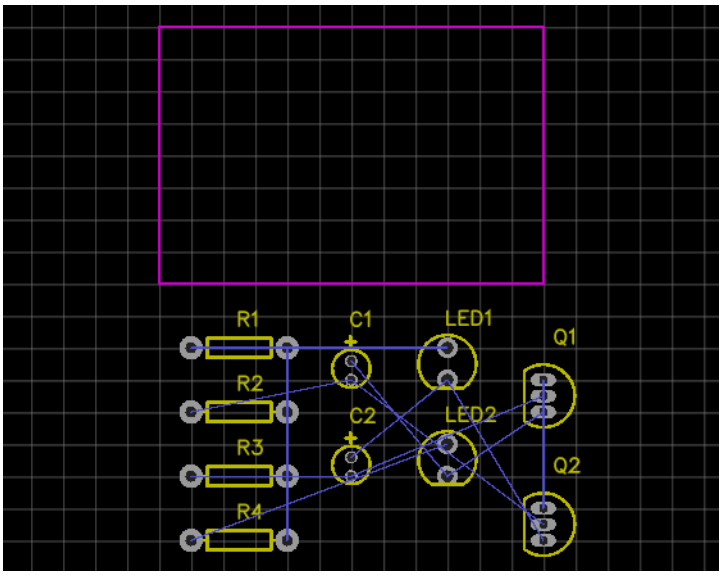
Board Outline

Before placing footprints we need to create a board outline. The board outline must be drawn on the **BoardOutLine** layer. So first, set **BoardOutLine** as the active layer, then draw the board outline using **Track** and **Arc** from the PCB Tools palette.



When converting a Schematic to PCB, EasyEDA will try to create a board outline for you.

The area of the default board outline area is 1.5 times the sum of the area of all of your footprints, so you can place all of your footprints into this board outline with some allowance for tracking. If you do not like the board outline, you can remove the elements it is made up from and draw your own.

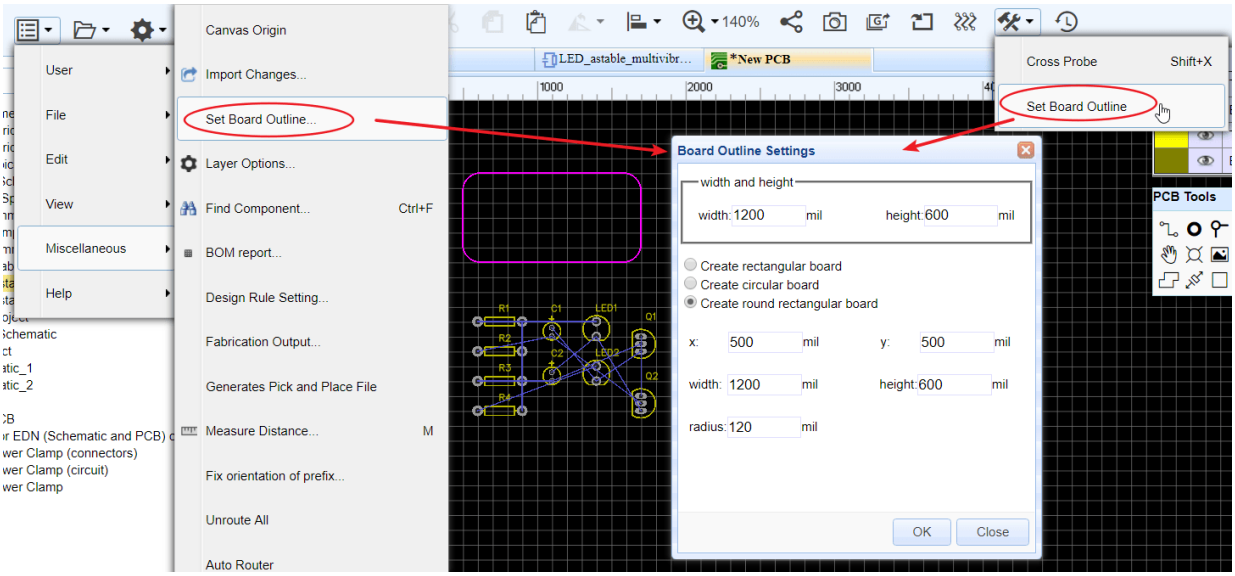


To create a simple rectangular board outline, this arc can be removed and the line X and Y end points edited - either directly in the Properties panel or by dragging the line ends - to close the rectangle.

Alternatively, an outline with more rounded corners can be created by copying the arc and rotating it in 90 degree steps to position it over the

desired right angle corners and then editing the line X and Y end points - either by dragging the line ends or directly in the Properties panel - to overlap the arc end points (also shown but not editable in the Properties panel).

And EasyEDA provides a **Board outline wizard**, so it is very easy to create a board outline. Via: **Super menu** > **Miscellaneous** > **Set Board Outline**, Or find it on the toolbar.



In this dialog, there's a choice of 3 types of board outlines, Rectangular , Circular, Round Rect. If you need a different more complex board outline, you need to import a DXF file.

Design Manager

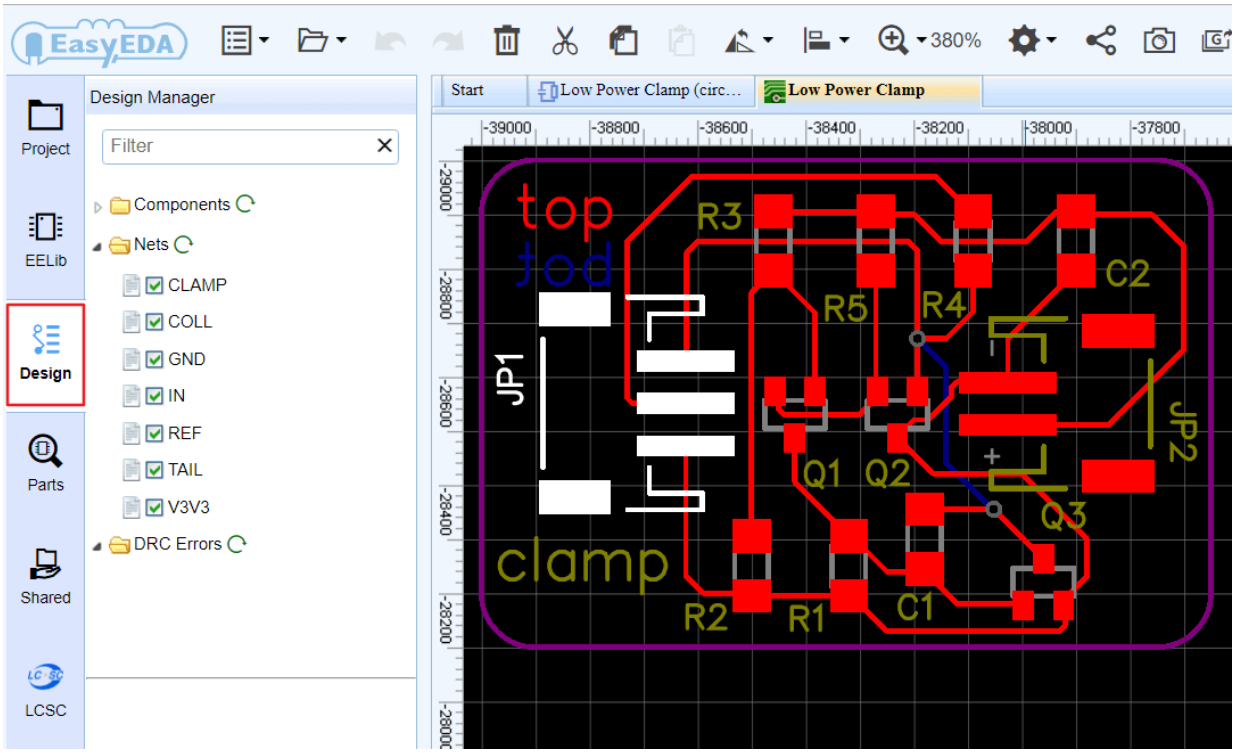
Just like Schematic's Design Manager, PCB's Design Manager can be found via:

Left Navigation panel > Design

or just press the `CTRL+D` hotkey to open the Design Manager dialog.

In this dialog, you can:

1. Click a component to highlight it.
2. Check/uncheck a component to show/hide it.
3. Filter to find a component or net.
4. Click a net to highlight the tracks/vias with the same net.
5. Check/uncheck the net to show/hide the net. For example, very often you may want to use this to hide a GND or supply net which has had a copper flood added to turn it into a plane and then show it again later.
6. Double click the net to remove all of the tracks and vias with the net name. If you want to reroute a net, this is the recommended method to use to un-route it first.



Import Changes

Before using "Convert to PCB", "Update PCB" in Schematic and "Import Changes" in PCB, please read Essential Check Before Clicking "Convert to PCB" or "Update PCB" or "Import Changes" section.

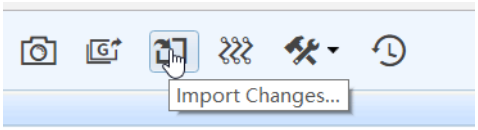
Sometimes, while working on a project, you need to make changes to the schematic and then update your board, to incorporate them.

It's easy to do this with EasyEDA.

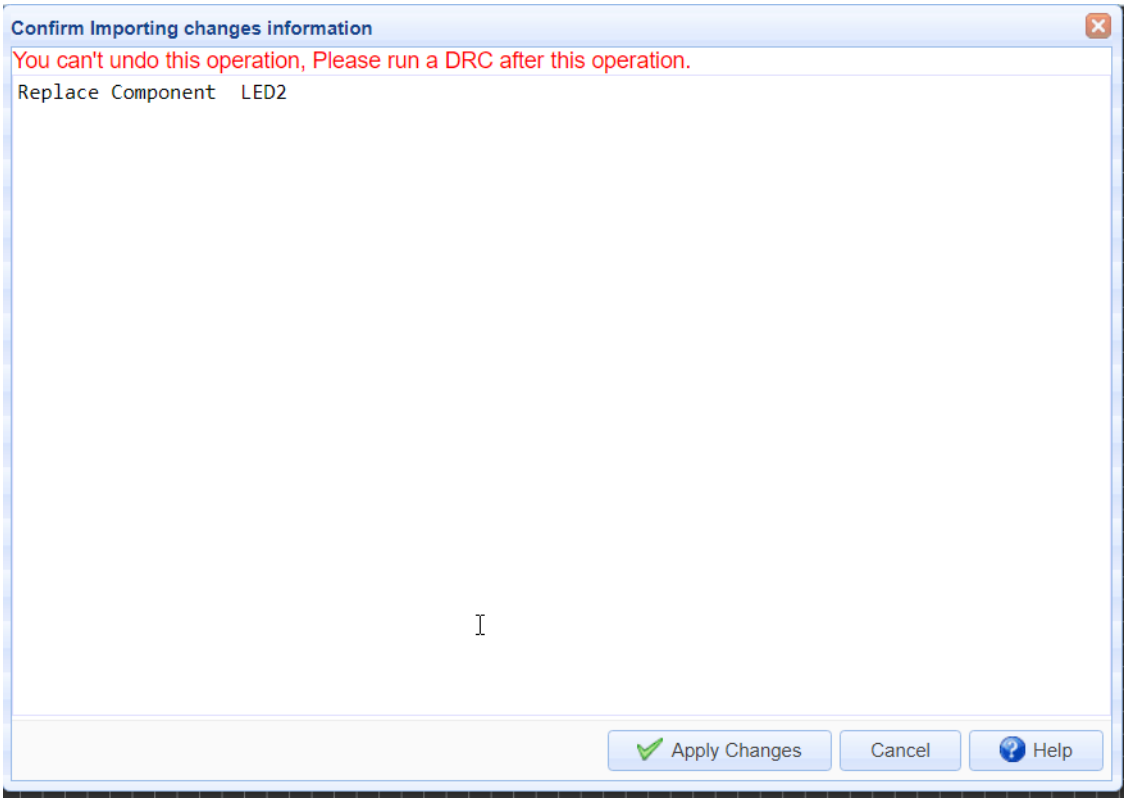
Go to the **PCB Editor**,

Super menu > Miscellaneous > Import Changes

Or click that button at the tool bar



You will get a Confirm Importing changes information dialog:

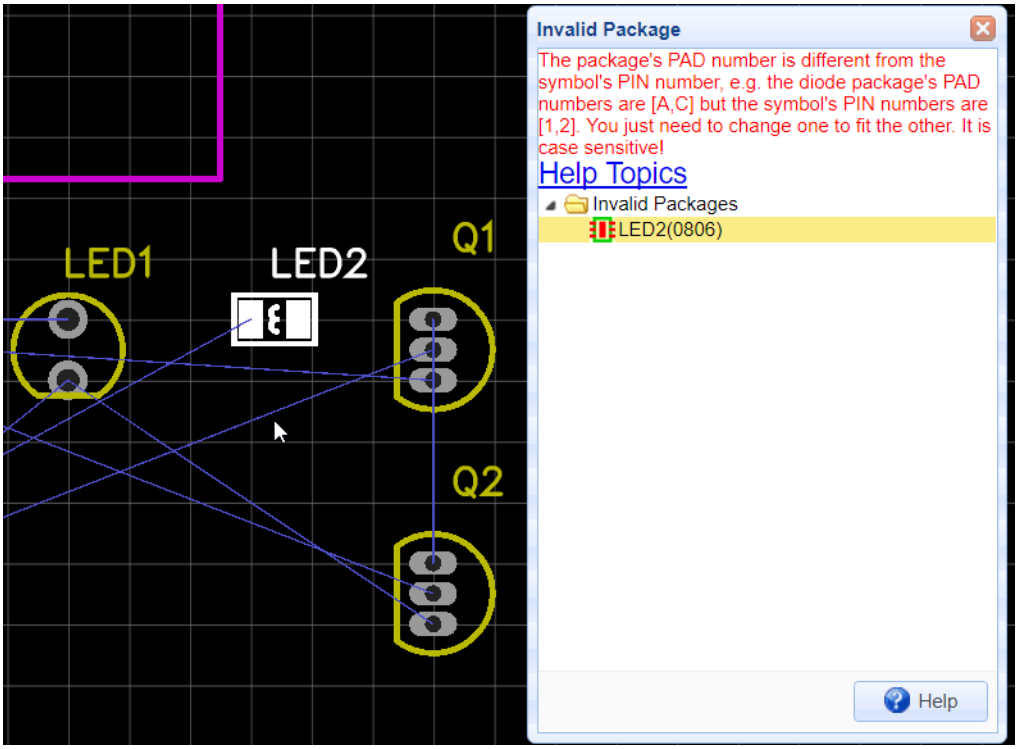


If you are happy with your changes, just click the Apply Change button.

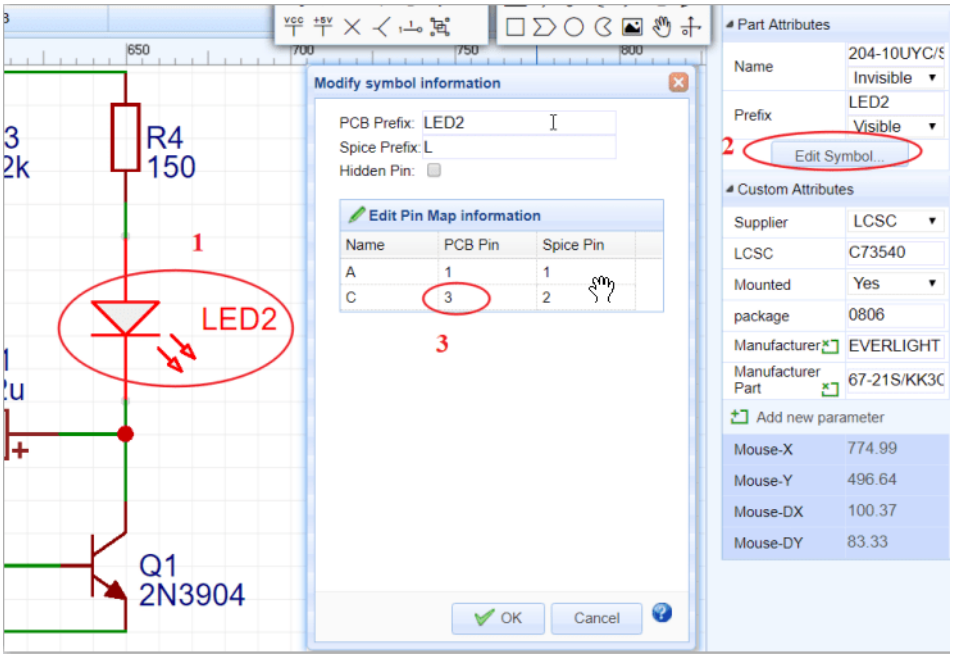
The changes will then be passed into the PCB layout and you can then adjust the tracking to suit.

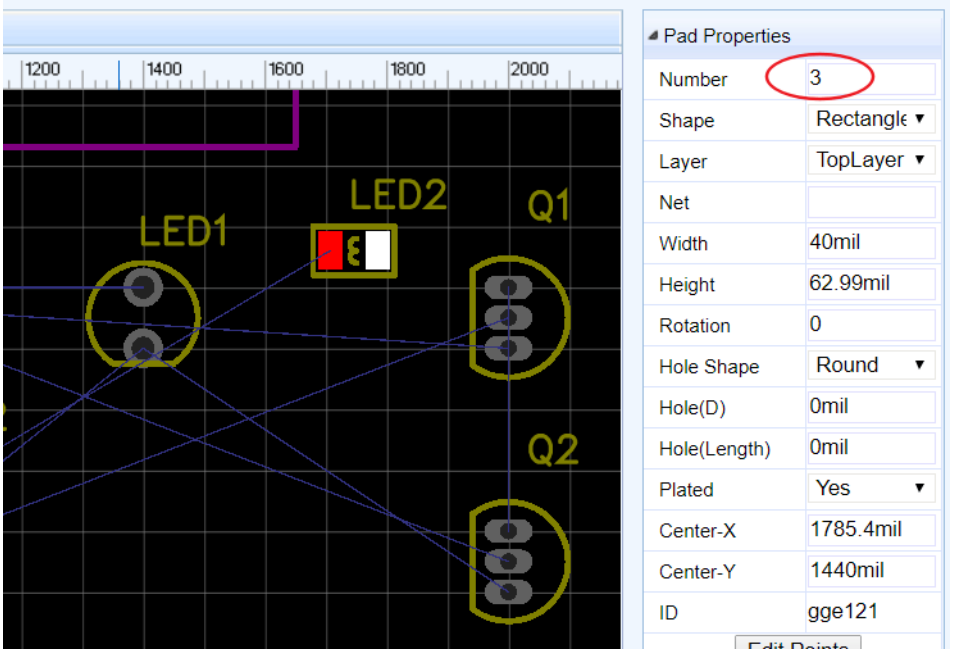
Invalid Packages

Sometimes, when you try to convert a schematic to a PCB, you will get an error message dialog like below. Don't worry, it is easy to fix this problem.



From the error message, you will find that the symbol's PIN number is different from PAD number. What caused that? Check the image below,





From the image, we can get the PIN number in the schematic symbol is set as `3`, but the PAD Number in the PCB Footprint is set as `2`. Now that

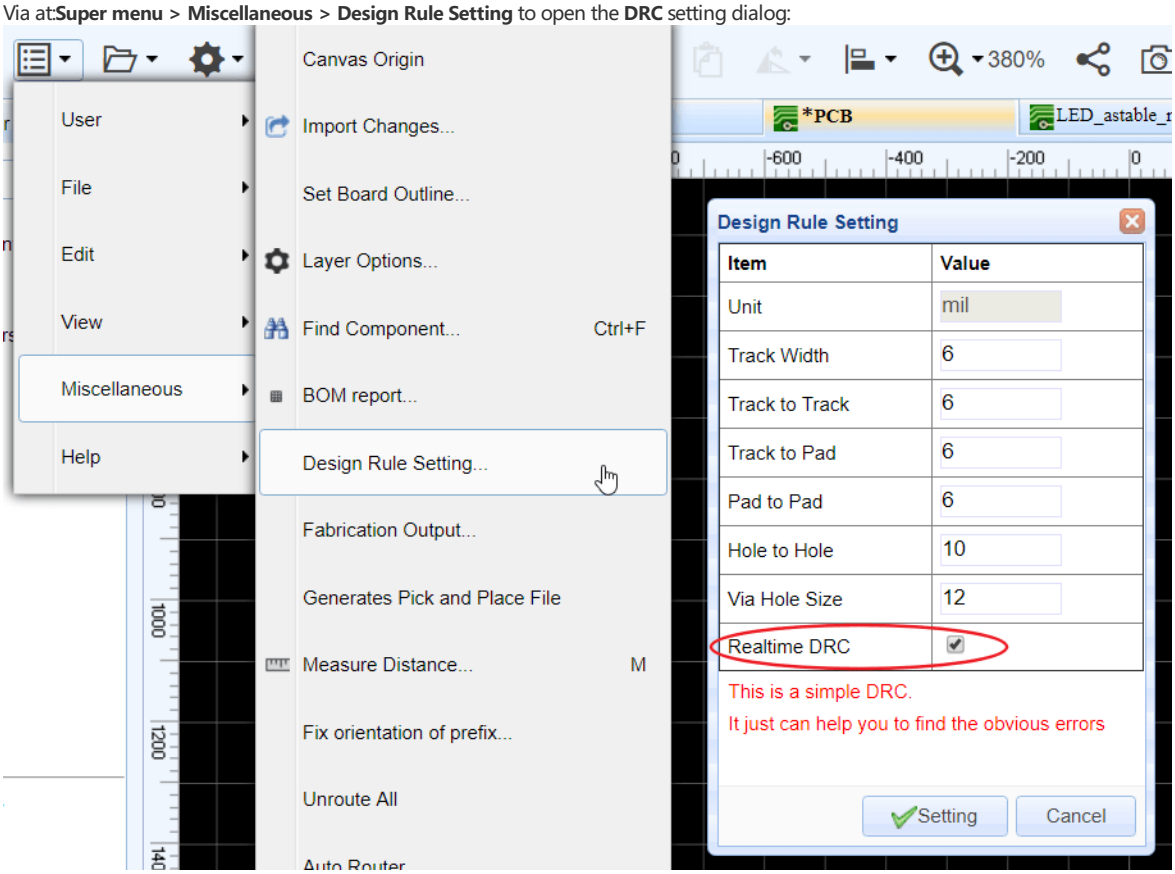
we've found the problem, how to fix this?

- Solution One: Change the schematic symbol. Using PinMap function. Change the PCB PIN from `3` to `2`. And save your schematic , and update PCB.
- Solution Two: Modify the Footprint. Edit the Footprint, change the PAD from `2` to `3`. And set this PAD net name to be the same as LED2 net name in the schematic.
  So, we should be aware that PIN number should be the same as Pad number.

Design Rule Check

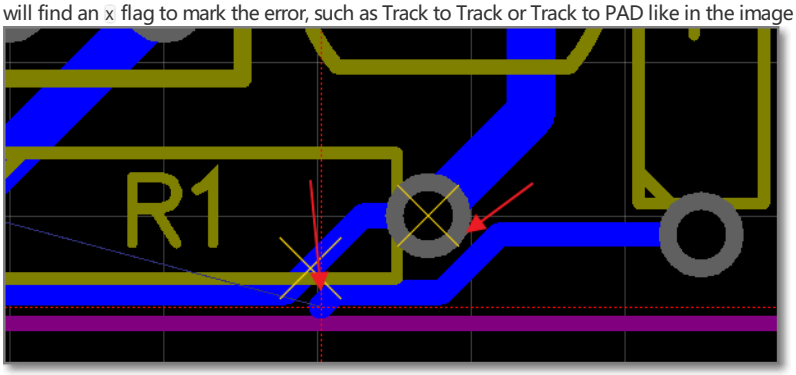
EasyEDA provides a powerful real time DRC(Design Rule Check) function.

Via at:**Super menu** > **Miscellaneous** > **Design Rule Setting** to open the **DRC** setting dialog:



Note: *When you convert a schematic to PCB, the real time DRC is open. But in the old PCB, the real time DRC is closed. you can open it as in the image above.*

This is a big feature of EasyEDA. It is hard to fix DRC errors after laying out the PCB. Now EasyEDA will let you know the error in routing. You will find an `x` flag to mark the error, such as Track to Track or Track to PAD like in the image below

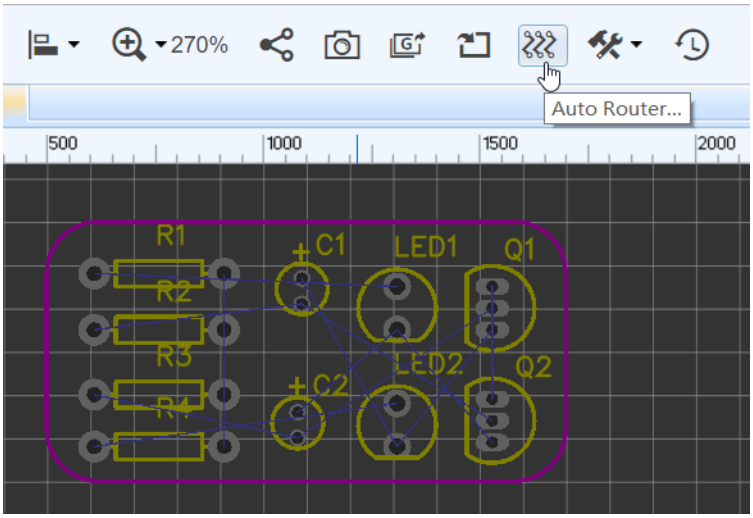


Auto Router

For some simple or prototype PCBs, you may want to use the auto router function to save time. Layout is a time costly and dull job. EasyEDA spends lots of time to provide such a feature and it is loved by our users. Before using the auto router, you need to set the board outline for the PCB.

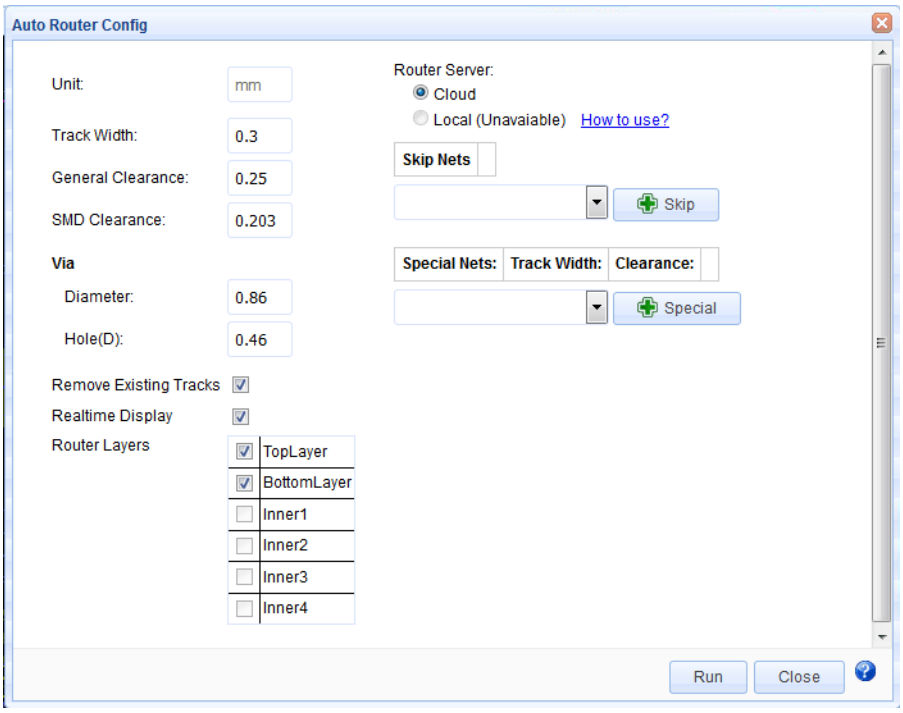
Steps:

1 Click the the auto router button from the toolbar or **"Super Menu > Miscellaneous > Auto Router"**



2 Config the auto router

After you click that button, you will get a config dialog like in the image below.

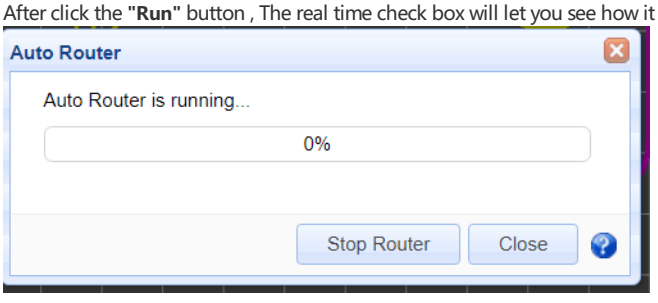


In the config dialog, you can set some rules to make the auto router result professional. These rule must equalize or more than DRC setting.

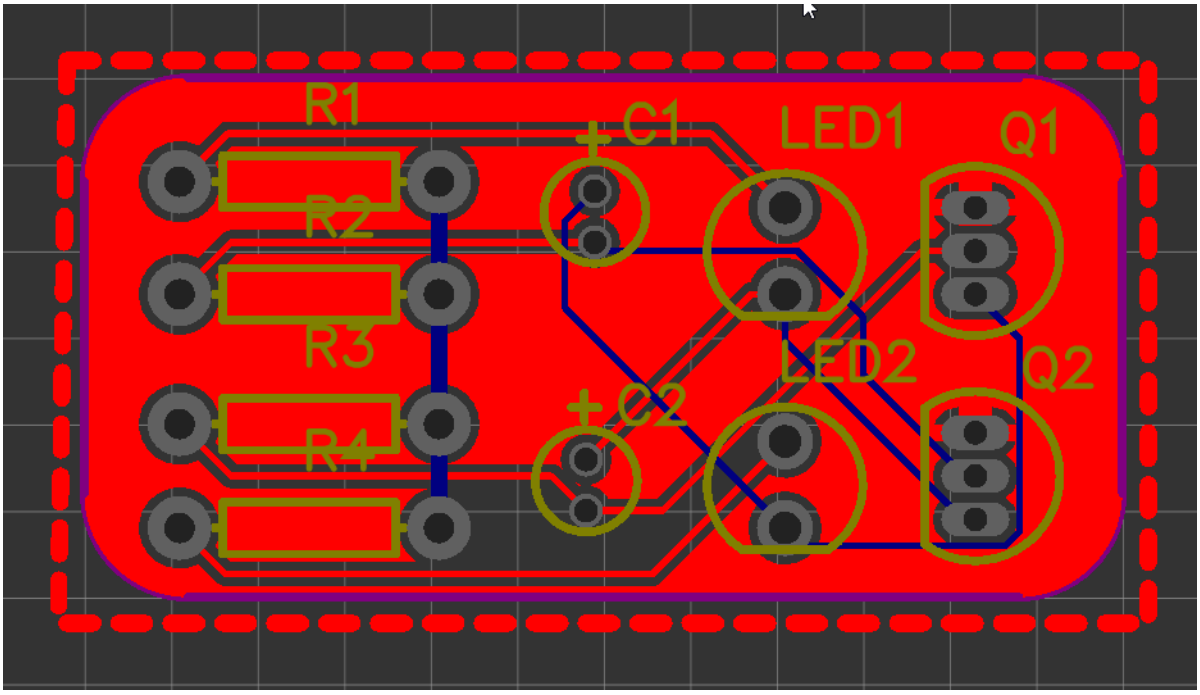
- **Remove Existing Tracks:** If you want to reserve the routed track, you need to deselect it.
- **Realtime Display:** when you select it , the real time routing status will show on.
- **Router Layers:** If you want to route inner layer, you have to enable the inner layer first at Layers Setting.
- **Router Server:**
  - **Cloud:** Using EasyEDA online server.
  - **Local:** Using the local auto router server, when you click the Auto Router icon, the editor will check the local router server available or not automatically. How to use please see as below.
- **Skip Nets:** If you like to keep the a net with no route, you can skip it. For example, if you want to use copper area to connect `GND` net, you can skip the `GND` net.
- **Special Nets:** For the power supply track, you may want it to be bigger, so you can add some special rules.

3 Run it

After click the **"Run"** button , The real time check box will let you see how it is going, but it will make the process a little bit slow.



Waiting for a few minutes, after adding bottom and top copper area, you will get a finished PCB board like in the image below.



Local Auto Router

EasyEDA suggest that using local auto router rather than using the cloud server, because when many users using cloud server, the cloud auto router will fail.

The local auto router server need to download and unzip it to the Non-System folder.

Download Address1(Google Drive): EasyEDA Router.7z
Download Address2(GitHub): EasyEDA Router.7z
Download Address3(Baidu): EasyEDA Router.7z

You need to configure the browser and execute the script first before click the **Auto Router** icon at editor.

Supported OS:

- Windows7(x64) or later 64bit Windows
- Ubuntu17.04(x64) or other 64bit Linux

Start local Auto Router:

- Double click win64.bat in Windows.
- Run "sh lin64.sh" on command terminal in Linux.

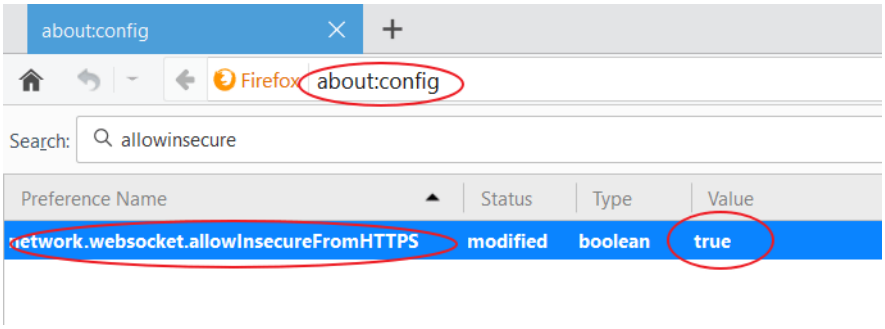
Notice: *Please use the latest Chrome or Firefox !!!*

1)Chrome

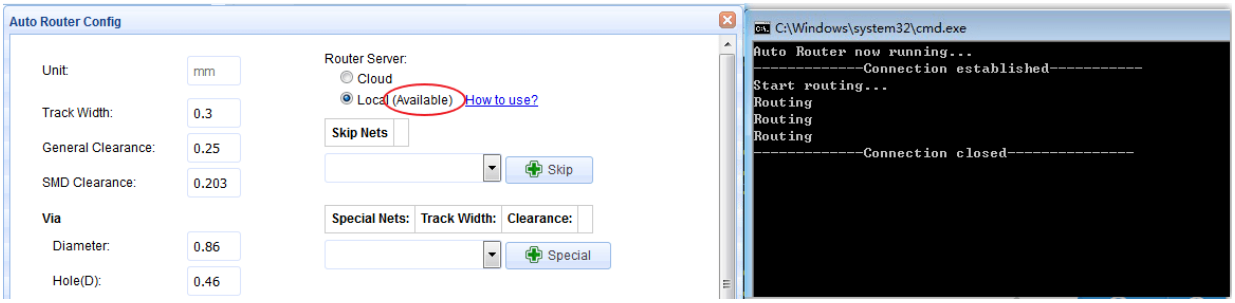
The Crome Browser don't need to be configure, If the local auto router is unavailable, you have to upgrade Chrome to version 60.0.3112.78 or later.

2)Firefox

1. Type "about:config" into the address bar then press enter
2. Search and double click the options as below (change the values to "true"):
   `network.websocket.allowInsecureFromHTTPS`
   `security.mixed_content.block_active_content`
3. Re-open Firefox and try again.



If the local router server is available, the dialog will tell you. Click the **Run** button, the dialog will show the process as below:



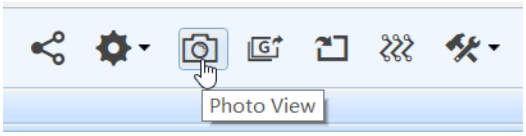
Sometimes, if you can't get it done, try the tips below.

- Skip the GND nets, add copper area to GND net.
- Use small tracks and small clearance, but make sure the value is more than 6mil.
- Route some key tracks manually before auto routing.
- Add more layers, 4 layers or 6 layers
- Use local auto router rather than cloud server.
- Don't use the special characters for the net name, such as <> () # & and space.
- Tell the error detail to us and send your PCB file as EasyEDA Source file to support@easyeda.com.

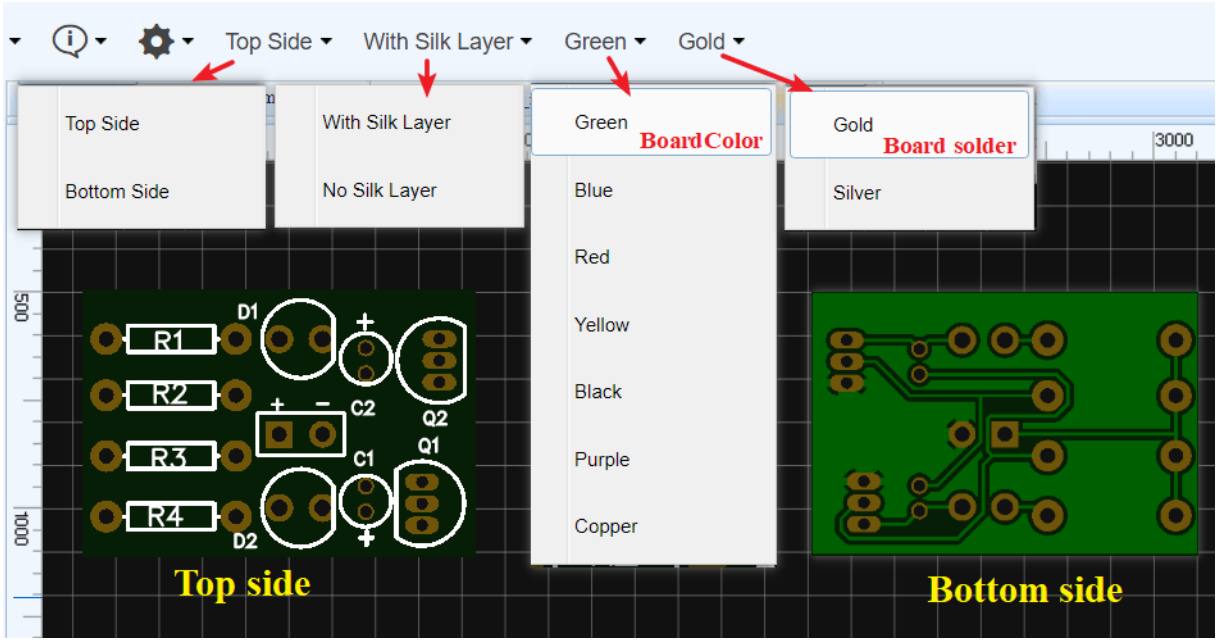
Some professional people don't like the auto router, because they think auto router is not professional, but you can use the auto router to check your placement. to check the density of your PCB.

Photo View

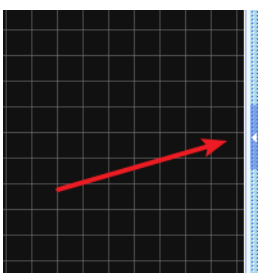
EasyEDA has no 3D View at present, but we provide a nice Photo View to help you to check the PCB. There is a `PhotoView` button on the PCB document toolbar, like in the image below. If you can't see this button, try to **reload** the PCB again.



After converting the PCB to Photo View, you can see the result as in the image below.

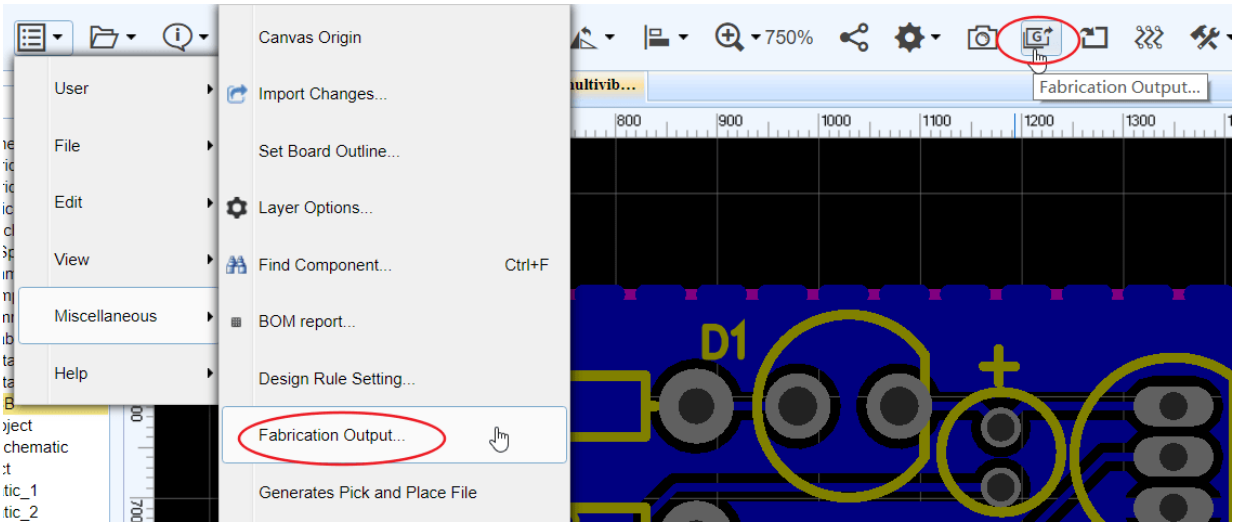


The photo view background default set as black and the right panel was hidden , you can popup up the right attribute panel and modify it.

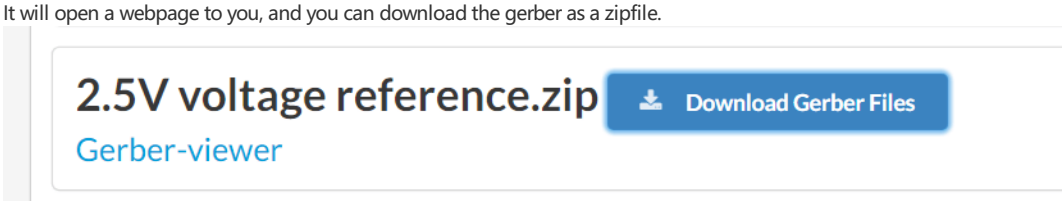


Getting Fabrication Files

When you finish your PCB, you can output the Fabrication Files(gerber file) via :
Super menu > **Miscellaneous** > **Fabrication Output** , or by clicking the Fabrication Output button from the toolbar.



It will open a webpage to you, and you can download the gerber as a zipfile.

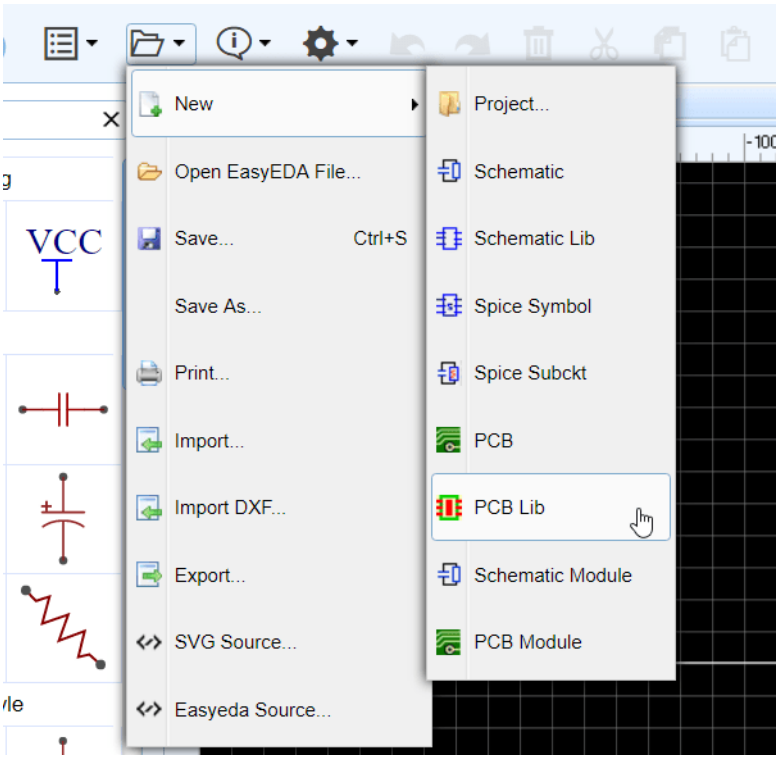


Creating The PCB Libs

There will be times when you will need a PCB footprint that is not already in the EasyEDA libraries.

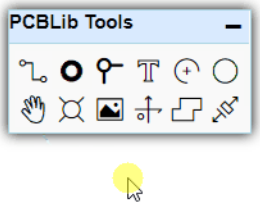
The process of creating your own PCB Libs is very similar to how you make symbols for your own Schematic Libs.

You can start a new PCB lib as shown below:



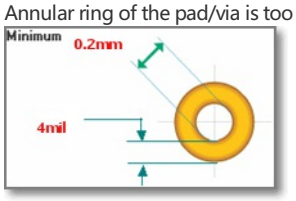
PCBLib Tools

PCBLib Tools almost are the same as PCB tools, just lacking some of the functions.



Others

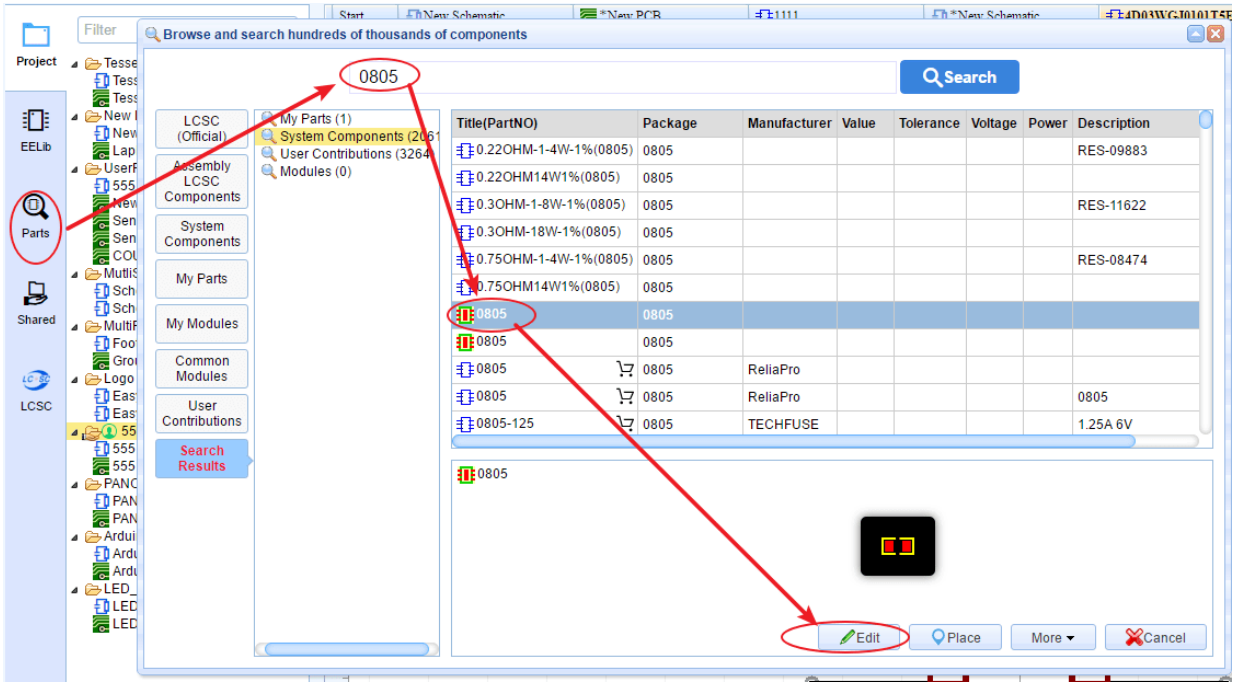
1. It is important to set the right Snap and Grid sizes to ensure that the pads on the finished footprint snap exactly to the grid and so connect the nets. For example, if you are creating a DIP package, set the Grid size to 100mil.
2. Keep all other shapes such as component outlines and any associated pin identification marks or text on the TopSilkLayer. EasyEDA will automatically take care
3. of the actual layer assignment when you place the footprint on the PCB.
4. `CTRL+S` to save your footprint designs and you will find them saved into the **Parts** > **My Parts** > **Packages** section of the left Navigation panel.
5. Annular ring of the pad/via is too small, keep the annular ring >= 4mil. In this case, you can add a `Hole`



Edit PCBLibs

When you feel the PCB Libs(footprint) can not be satisfy for you, you can edit it.

Via **"Parts"** > **"Search Part/My Parts/LCSC Parts/System Components/User Contributions"** > **Select Package** > **Edit**



when you finish and save , it will be saved to your personal libraries "My Parts" and become your personal libraries.

Import

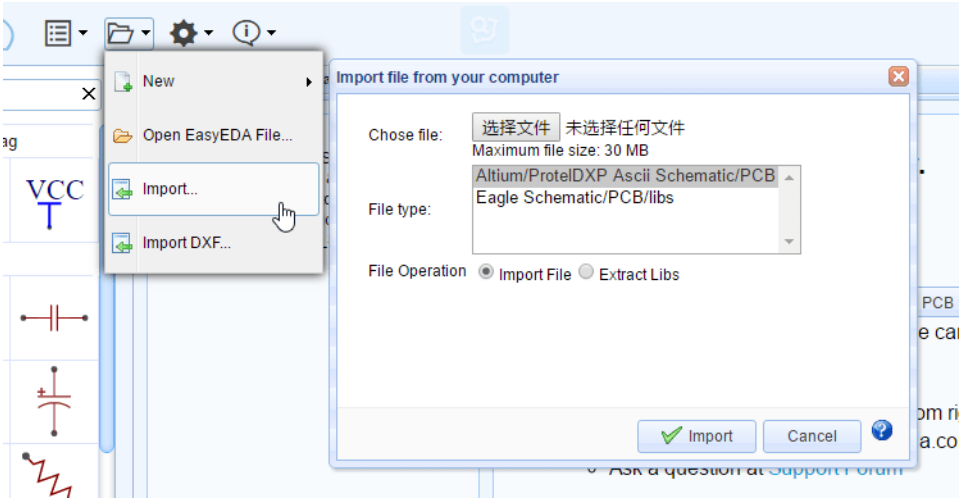
EasyEDA provides importing from:

- Altium/ProtelDXP Ascii Schematic/PCB
- Eagle Schematic/PCB/libs

You can find the import menu from the Document menu:

Document > **Import...**

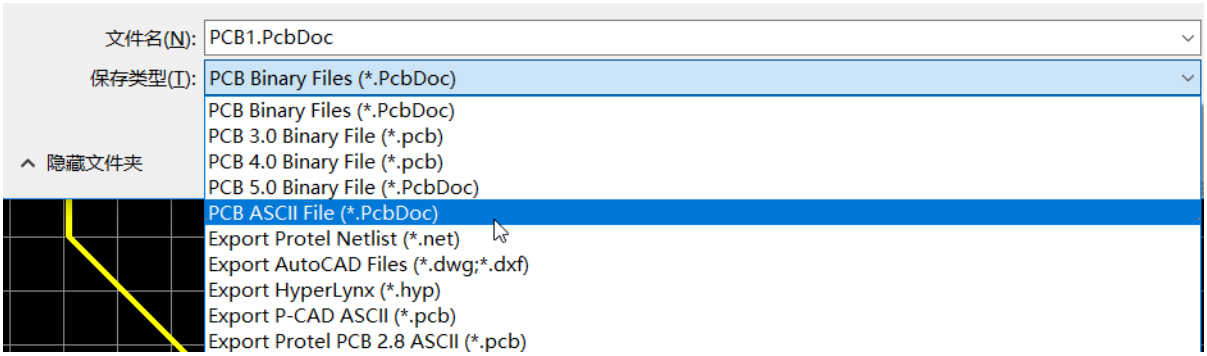
which opens the Import file from your computer dialog:



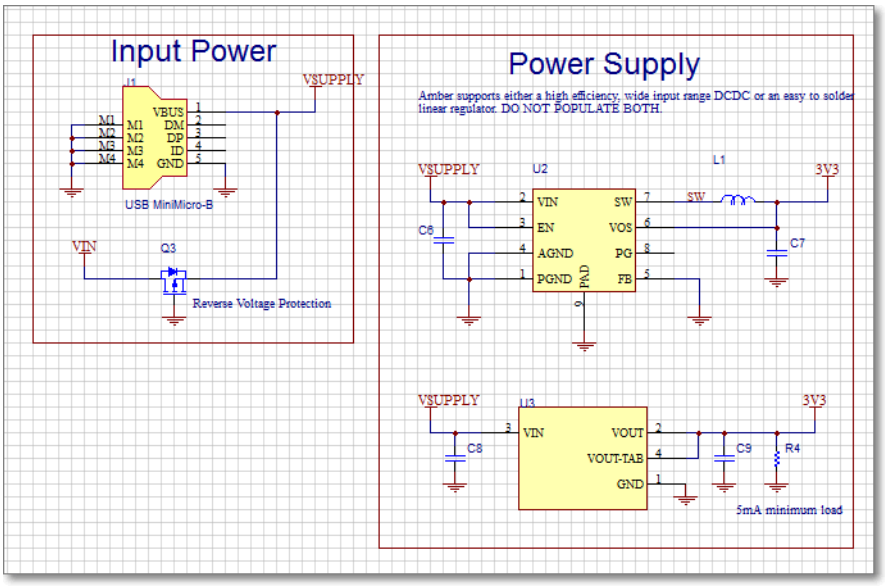
Please note that in File Operation: the Extract Libs option is only supported when importing Altium Designer and Eagle libraries.

Import Altium Designer

You can import Altium Designer's Schematic and PCB files into EasyEDA but only from **ASCII** files, so you need to save the designs as Ascii files like this.



EasyEDA offers an excellent experience in importing Alitum Designer's Schematic and PCB: as you can see from the image below of a schematic imported from Altium Designer:



If your schematic and PCB are Protel 99se format files, please open at Altium Designer and save as ASCII format, and then import them.

Altium Designer's Schematic and PCB libraries are not available as **ASCII** files, so how can you import them?

In the Import file from your computer dialog to the right of File Operation; tick the Extract Libs option and EasyEDA will extract all of the libs from the Schematic files or PCB Files. So, if you want to import Altium Designer's Libs, you can add them to your Altium Designer Schematic or PCB and then extract them again into your EasyEDA library.

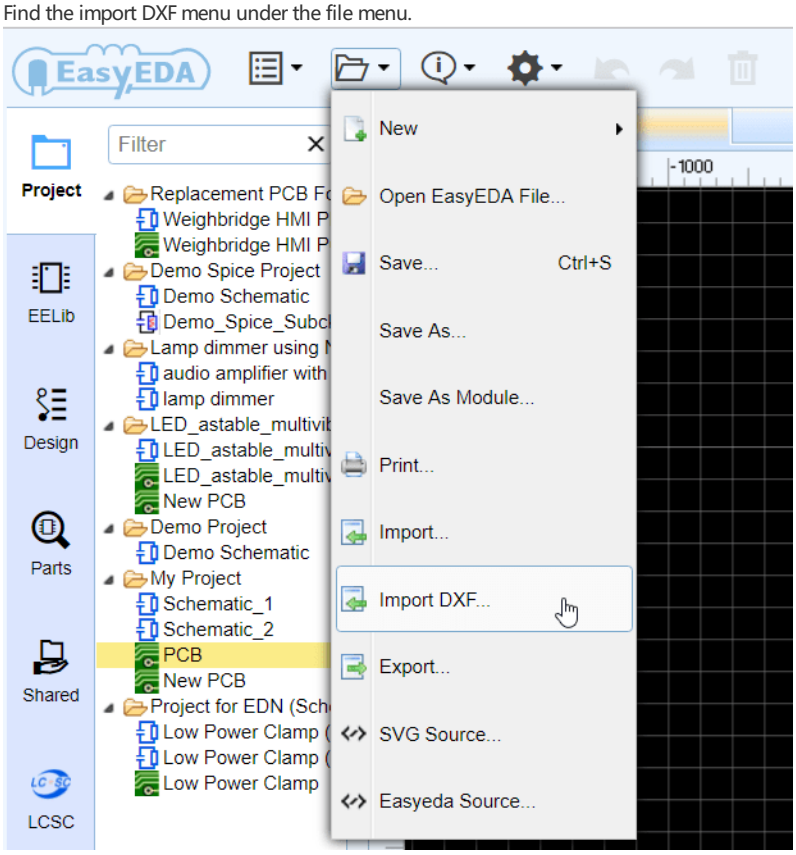
Import Eagle

Eagle Schematic/PCB/libs can be imported, but EasyEDA can only support version 6 and later (6+) because that was when Version 6 Eagle adopted an **ASCII XML** data structure as their native file format.

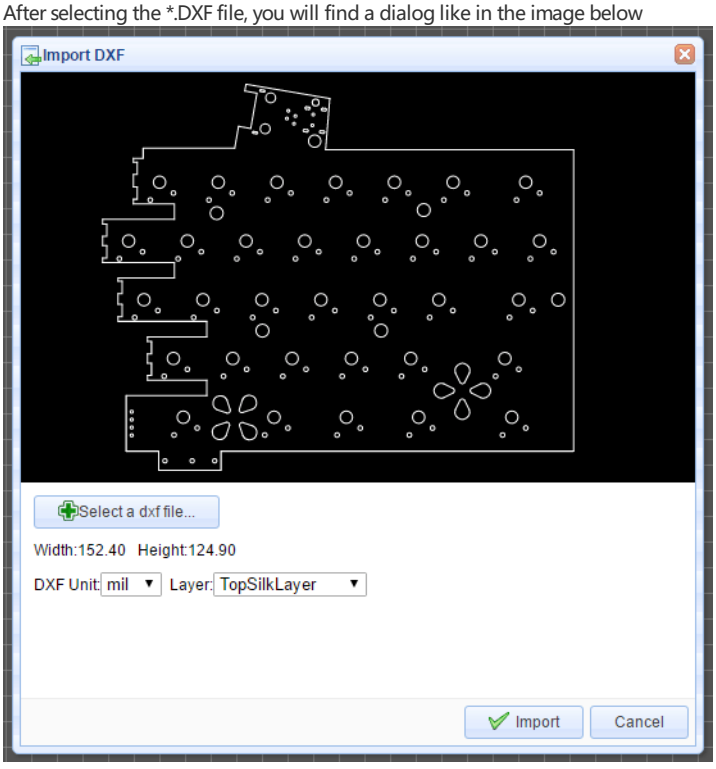
Import DXF File

How to create irregular board outlines or complex board outline in EasyEDA? This is sometimes needed when you are designing a PCB for an enclosure that may have a curved profile, or other unavoidable mechanical features for which one must design.

Find the import DXF menu under the file menu.

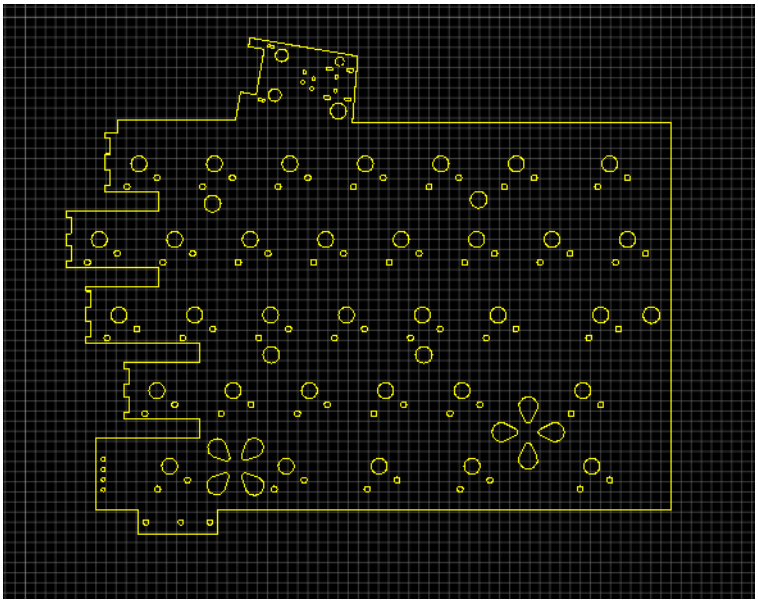


After selecting the *.DXF file, you will find a dialog like in the image below



EasyEDA provides two options, unit(mm or inch), and selection of the layer to which the shapes will be applied.

After clicking the import button, you will find them on your PCB canvas.



You can try this to import this example by yourself. DXF example

Please note:
1. The file must have a *.dxf filename extension
2. The circles will be converted to holes if you choose the layer as board outline.
3. There are some items which are not supported.

Export

For documentation and other purposes, you can export your Schematic and PCB designs for many items.

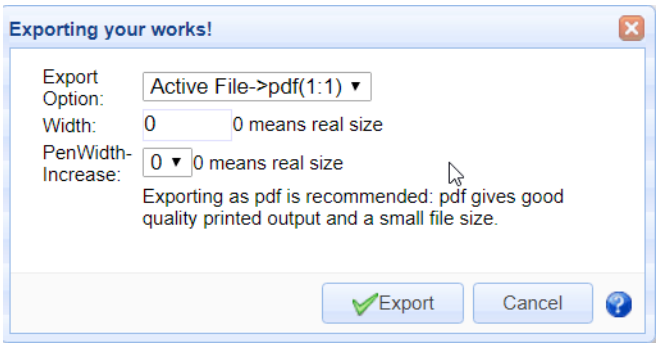
Exporting Schematics

Exporting Schematics In Documentation Formats

Using:

Document > Export...

will open this dialog:

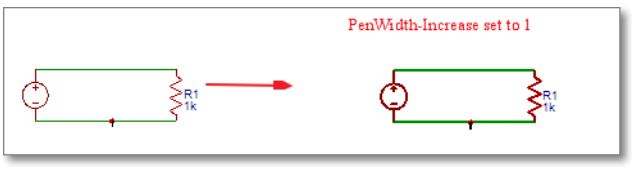


From here you can choose to export your design to SVG, image (.png) and PDF file format.

For all file formats:

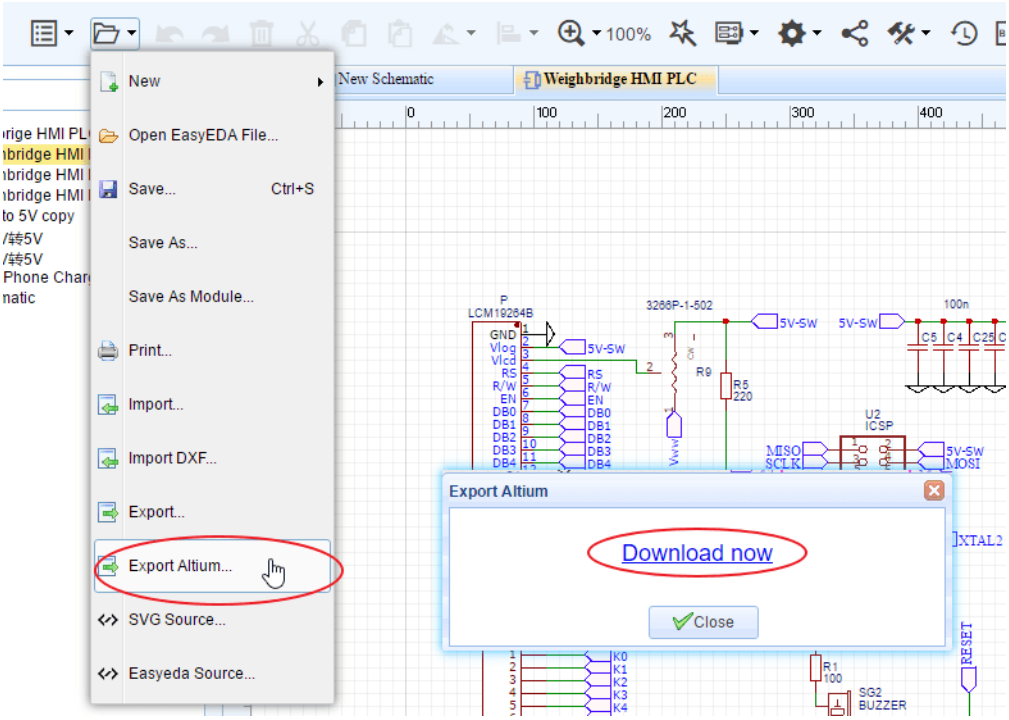
Width: This is images' width , 0 is a 1:1 export of your image, higher numbers scale your image , if you set number as 1024 , the width will be 1024 pixels of the export PNG .

PenWidth-Increase: 0 represents a default line width of 1 pixel; if you set this to 1, the line will be 2 pixels. This is illustrated in the image below.



Exporting Schematics In Altium Designer Format

EasyEDA support exporting the schematics in Altium Designer format. Via **"Documents > Export Altium..."**, and click the **"Download now"**

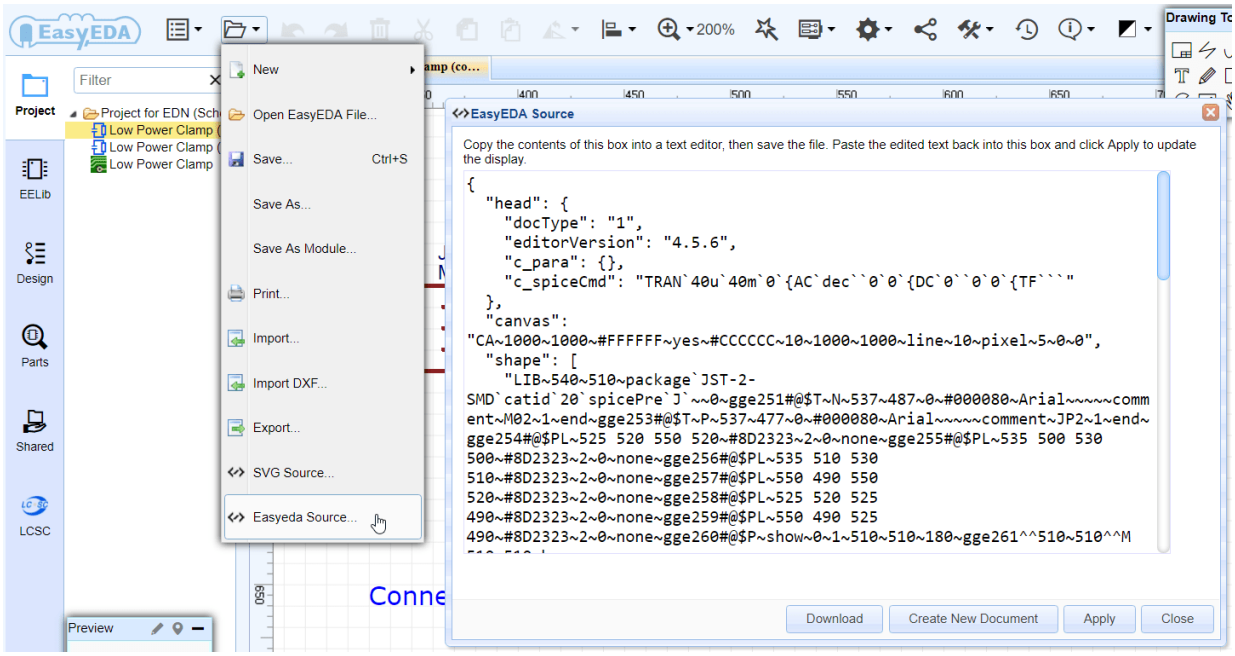


you will get a `.schdoc` file.

Download Schematics

You can download the schematic when it is opening, via:

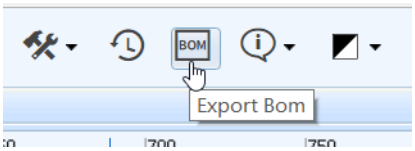
Document > **EasyEDA Source...**, click the download button, you will get a .json file.



Or **Project** > **Right Click** > **Download Project**, you will download a zip file with EasyEDA Source files for Schematics and PCBs.

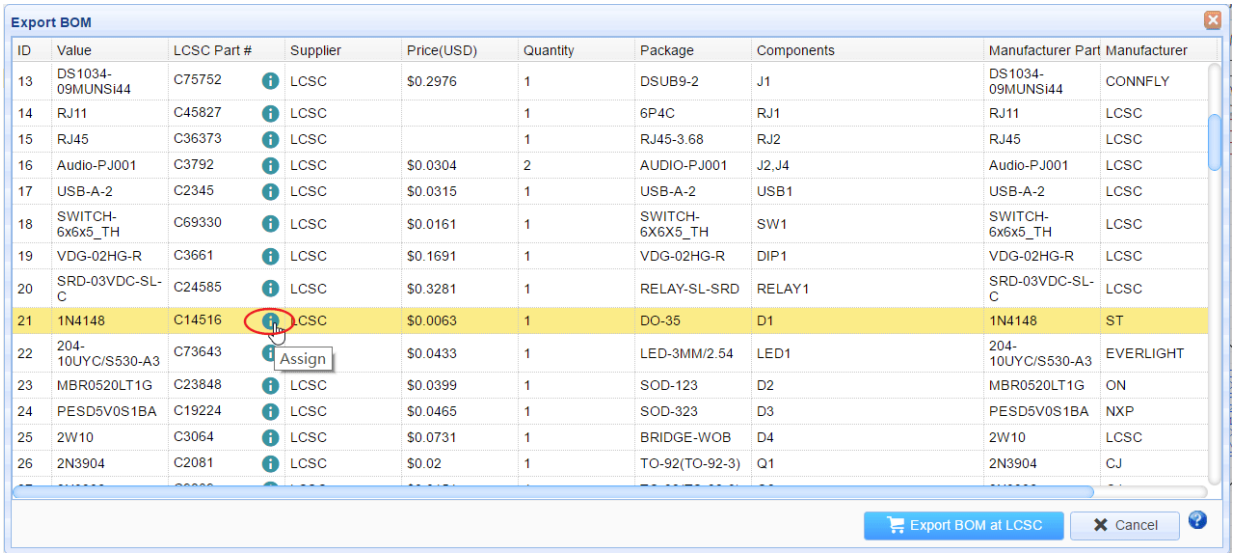
Exporting BOM

You can **export** the Bill of Materials (BOM) for the active schematic (Document) and PCB or for the active project (i.e. the BOM for all the sheets in the project) as shown below , click the top toolbar `BOM` icon:

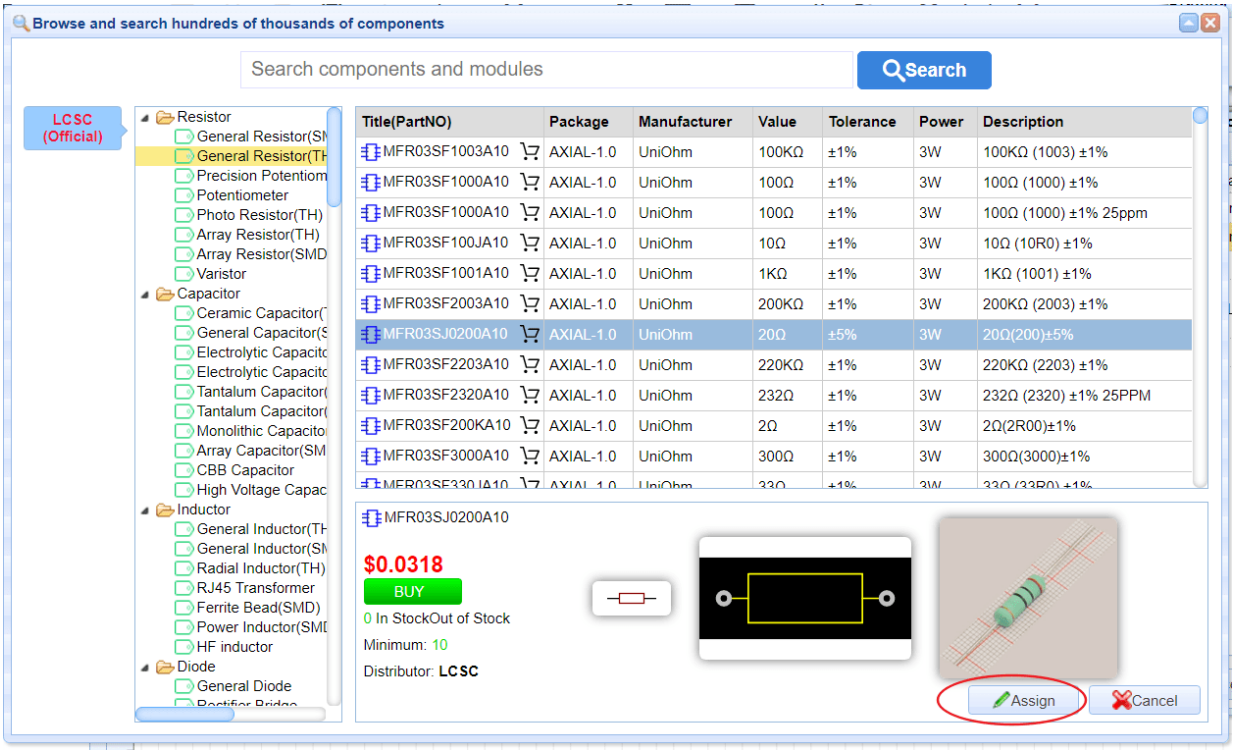


After clicking the BOM export option, the dialog below will open.

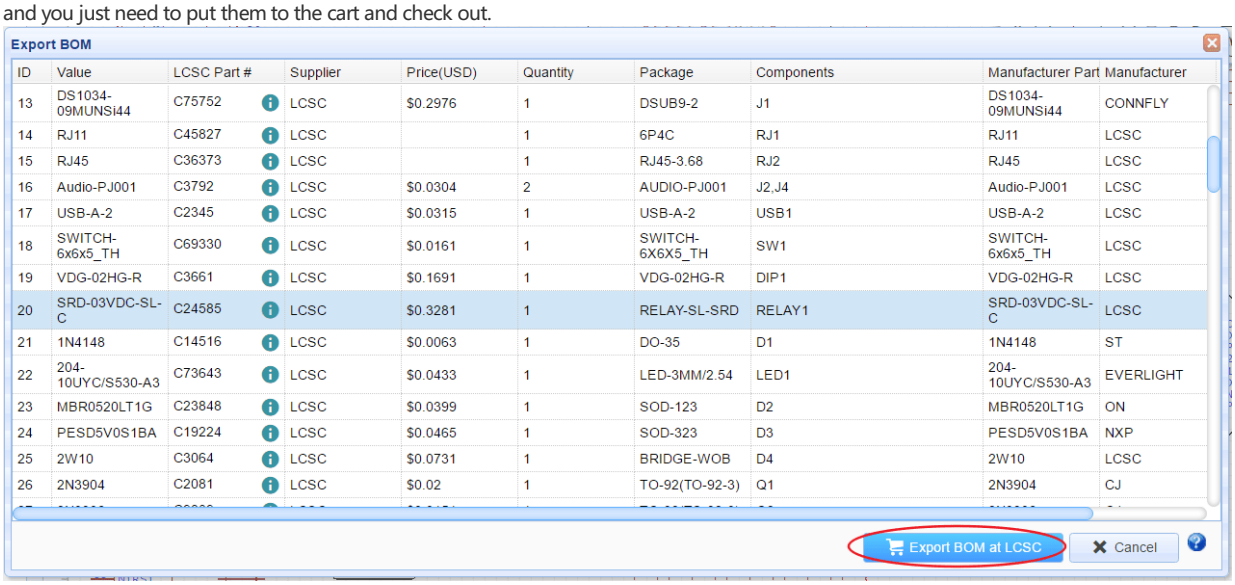
In this dialog , you can assign LCSC part's order code for your components.

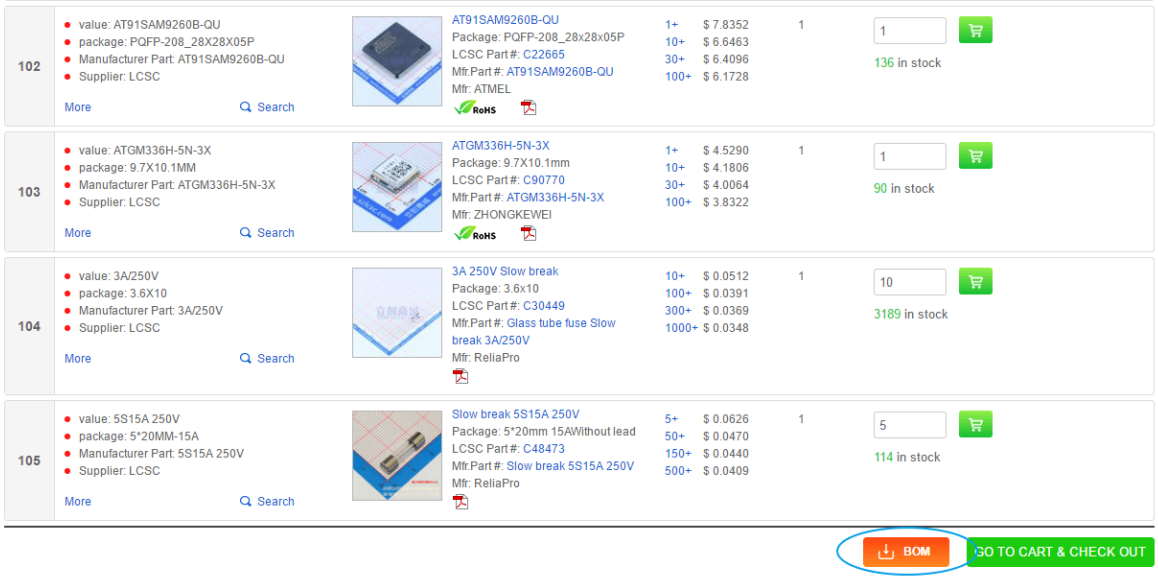


After clicking on the assign icon，the components and packages search dialog will pop up, and you can choose which component you want to assign.



When you click "Export BOM at LCSC", we will help you to list all the components of your BOM, If you want to buy the components form LCSC, and you just need to put them to the cart and check out.





And Click the "BOM" button to download the BOM file. You can open it in any text editor or spreadsheet.

	A	B	C	D	E	F	G	H	I	J
1	id	value	quantity	package	components	Manufacturer Part	Manufacturer	Supplier	LCSC	price
2	1	150	2	AXIAL-0.3	R1,R4	25121WJ020KT4F	UniOhm	LCSC	C45278	$0.02
3	2	22k	2	AXIAL-0.3	R2,R3	25121WF300LT4F	UniOhm	LCSC	C16074	$0.03
4	3	22u	2	CAP-D3.0XF1.5	C1,C2	1812B225K500NT	FH	LCSC	C28503	$0.28
5	4	204-10UYC/S53(	2	LED-3MM/2.54	LED1,LED2	67-21S/KK3C-H2727QAR3LED	EVERLIGHT	LCSC	C73540	$0.04
6	5	2N3904	2	TO-92(TO-92-3)	Q1,Q2	MURA220T3G	ON	LCSC	C37995	$0.17
7										

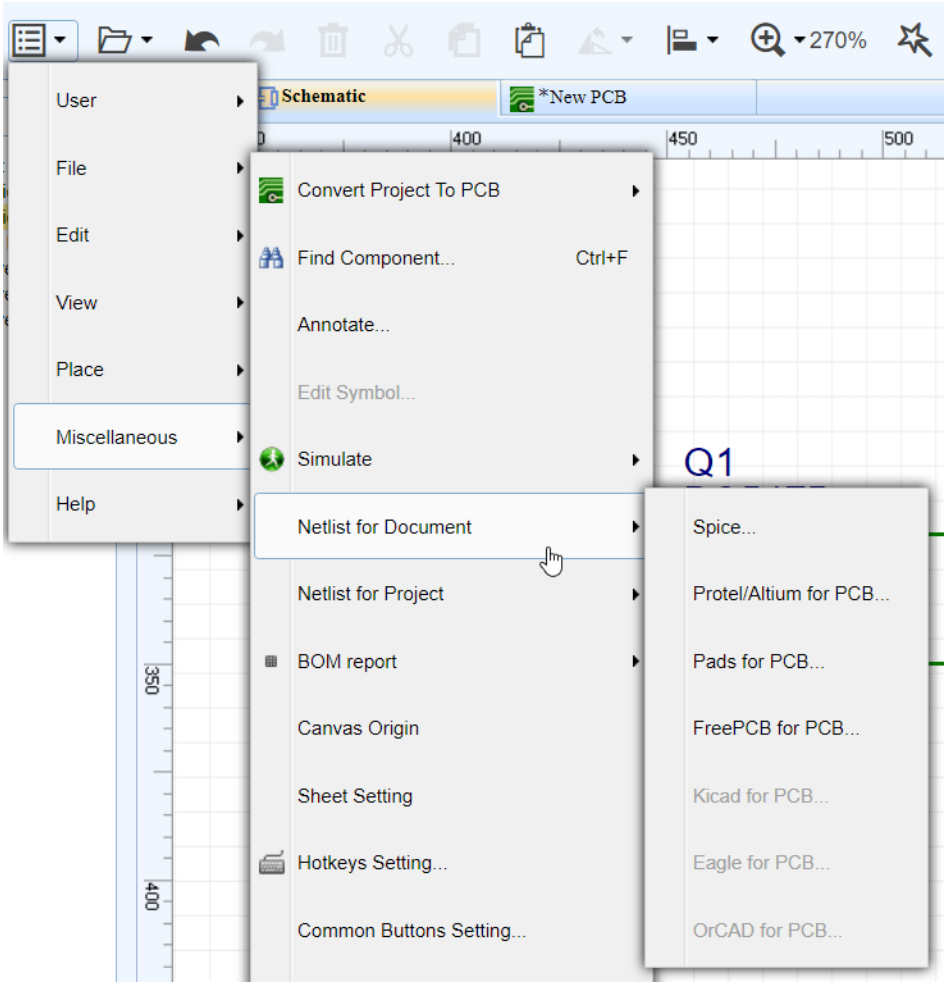
Exporting Netlist

EasyEDA can export the netlist for the active schematic (Document) and/or for the whole active project:

Super menu > Miscellaneous > Netlist for Document or Netlist for Project

EasyEDA can export a netlist in a variety of formats:

- **Spice**: this is a Spice3f5 compatible netlist generated by the simulation engine of EasyEDA, Ngspice. It is not normally used as the basis for as a PCB layout.
- KiCad: a PCB netlist in a format that can be imported straight into Pcbnew, the PCB layout tool part of the free, open source cross-platform EDA suite.
- **Altium Designer**: a PCB netlist in a format that can be imported straight into Altium Designer and it's predecessor, Protel.
- **Pads**: a PCB netlist in a format that can be imported straight into Pads PCB layout tools.
- FreePCB: a PCB netlist in a format that can be imported straight into FreePCB, a free, open source PCB editor for Windows.



Exporting PCB Designs

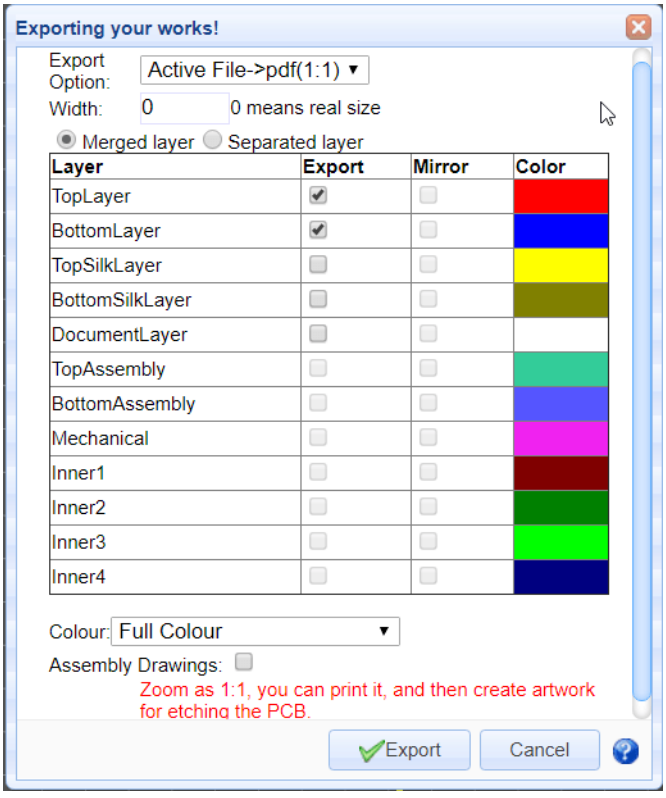
Exporting PCB In Documentation Formats

Exporting a PCB design or footprints from EasyEDA is very similar to exporting a Schematic or a Symbol.

Using:

Document > Export...

you can open this dialog:



You can select to export in PDF, drawing (.PNG) or SVG format.

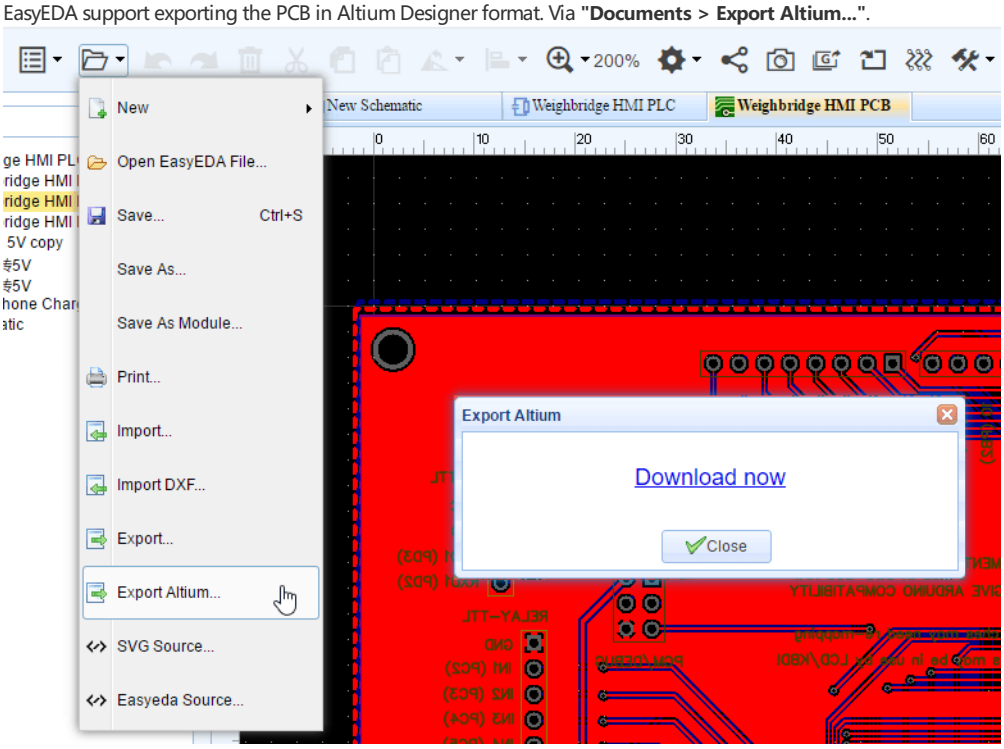
Note: *If you want to print the PCB 1:1 with the paper, you need to choose to export PDF(1:1).*

You can select to print individual layers or selected layers merged into a single file.

It is also possible to mirror selected layers for example to show bottom layers in easily readable orientation.

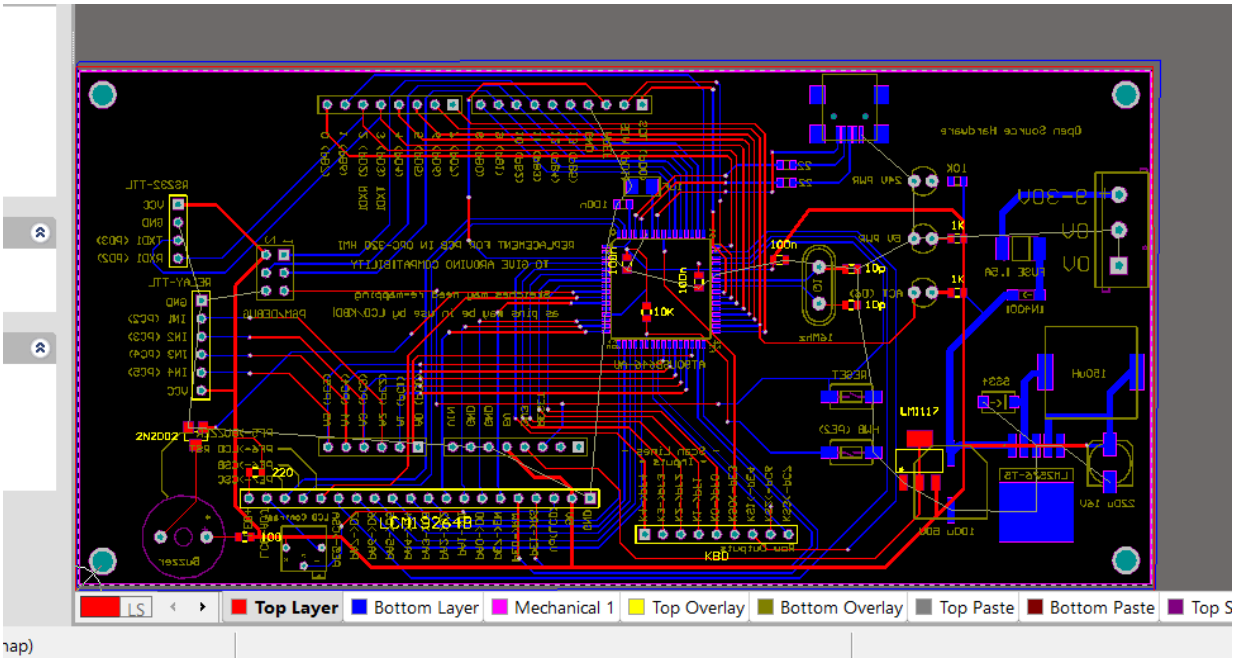
Exporting PCB In Altium Designer Format

EasyEDA support exporting the PCB in Altium Designer format. Via "**Documents** > **Export Altium...**".



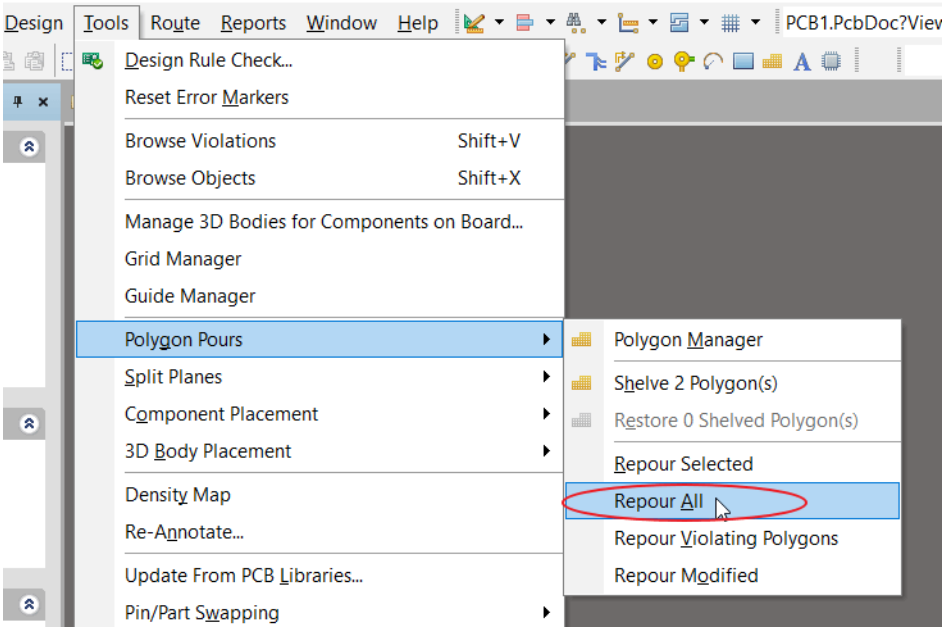
When open the exported PCB file at Altium Designer, there will open a dialog of DXP Import Wizard, don't worry, just cancel it to continue.

And then, you will see the PCB file, which is looks like without copper area as below:

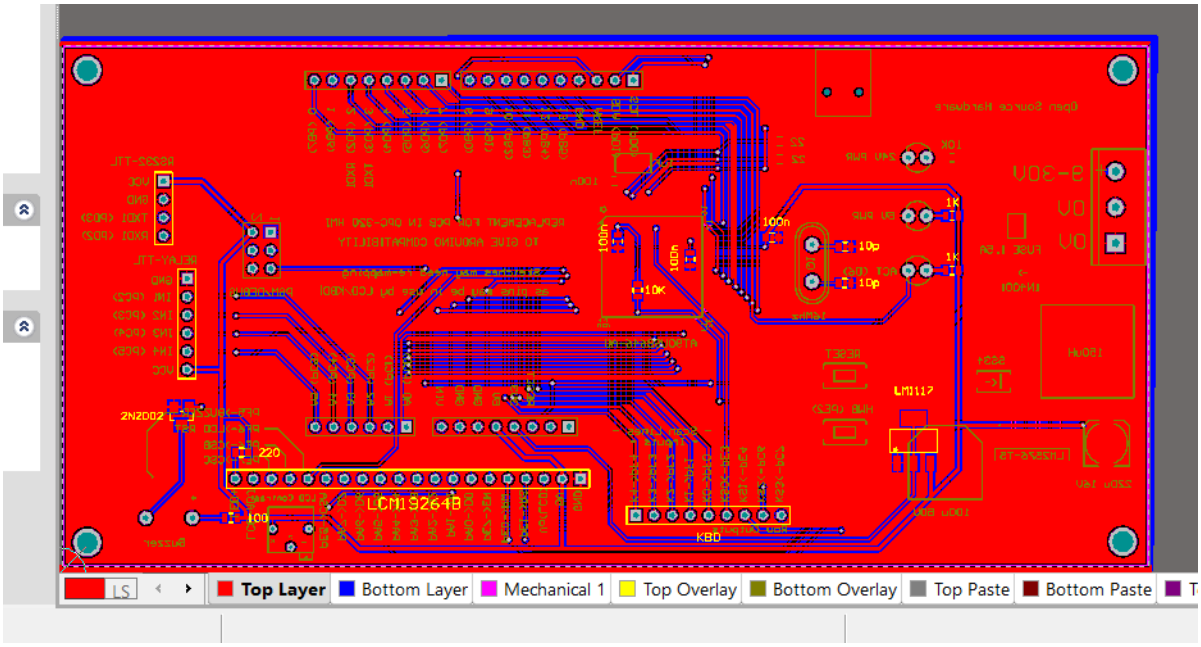


At now, you need to repour all polygons at Altium Designer. Via: **Tools** > **Polygon Pours** > **Repour All**:

ee Documents. Not signed in.

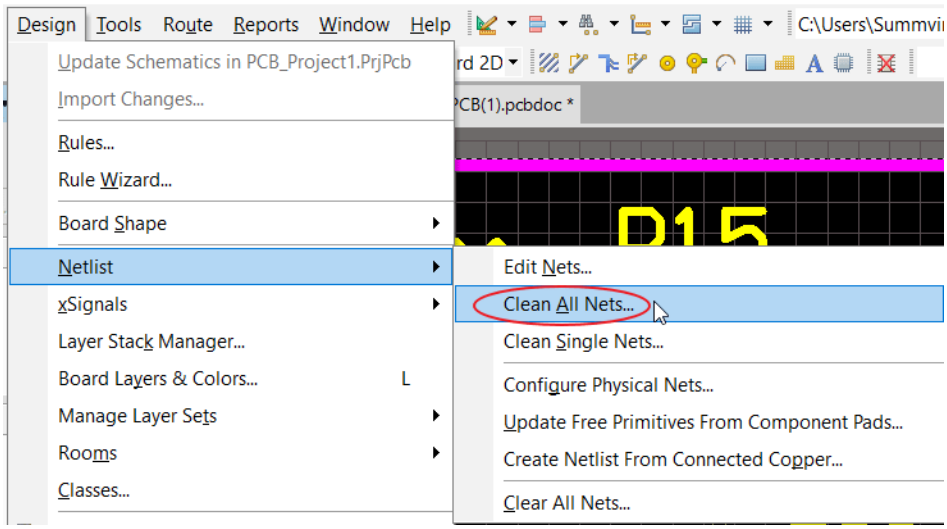


And the last, save it.

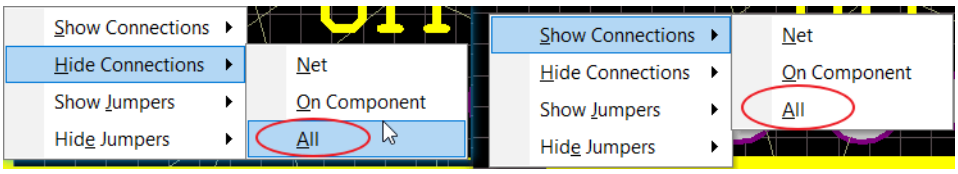


If you export the PCB without tracks, you need to show all connections first before routing :

Via: **Design** > **Netlist** > **Clean All Nets**



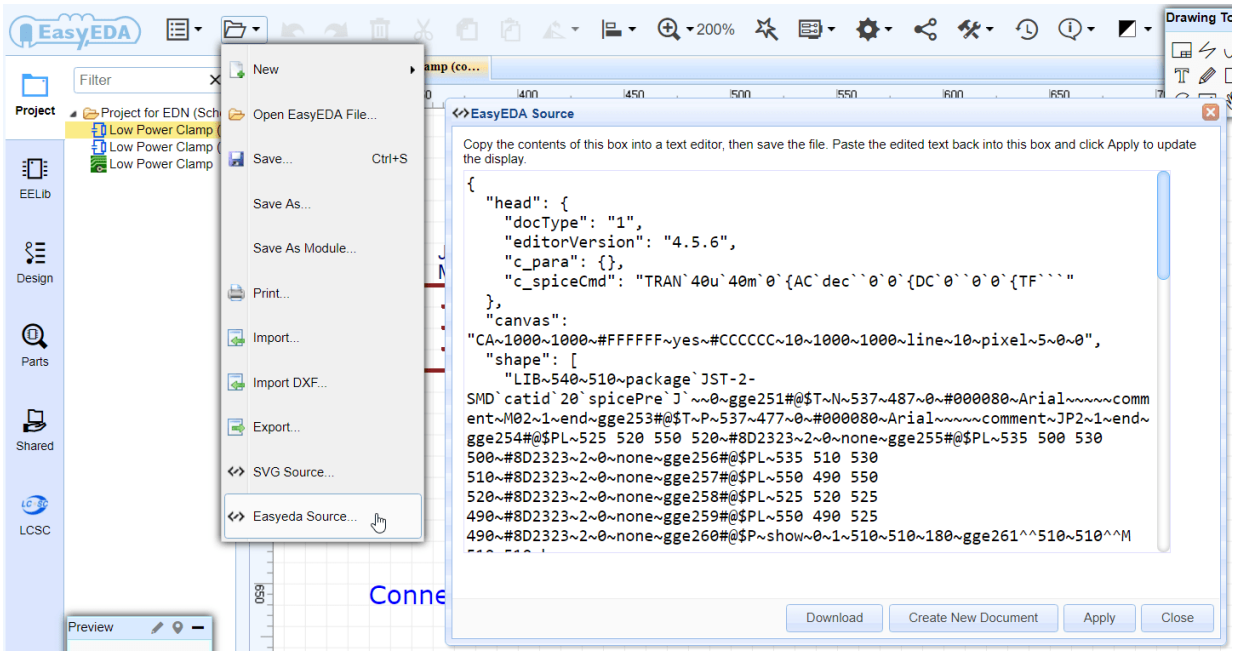
Or use hotkey: **N** > **H** > **A** and then **N** > **S** > **A**:



Download PCB

You can download the PCB when it is opening, via:

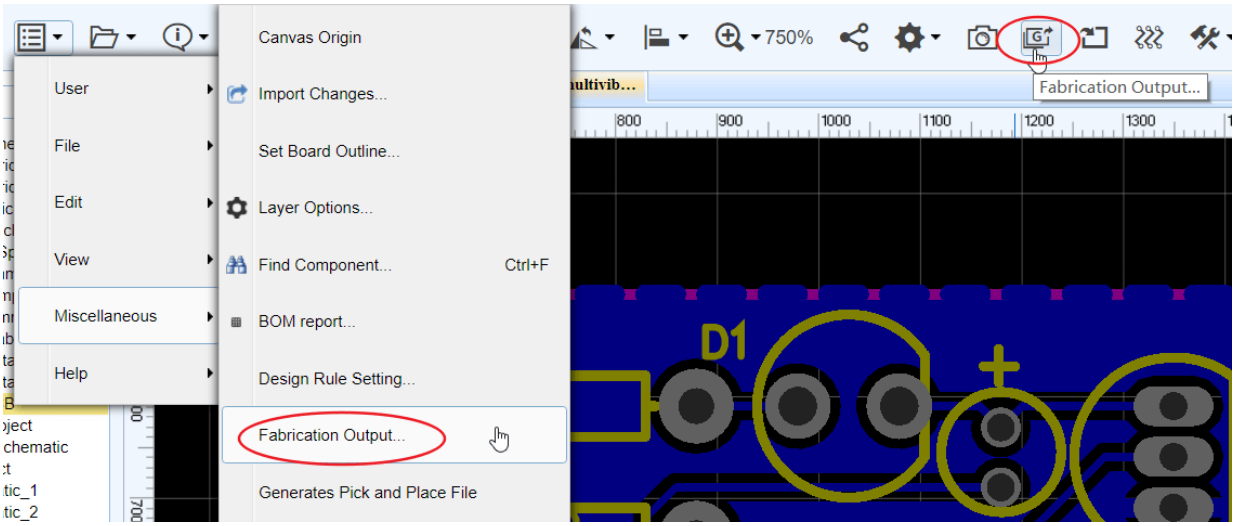
Document > **EasyEDA Source...**, click the download button, you will get a .json file.



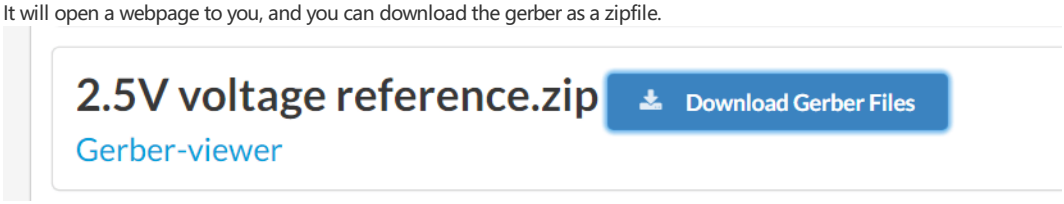
Or **Project** > **Right Click** > **Download Project**, you will download a zip file with EasyEDA Source files for Schematics and PCBs.

Exporting Fabrication Files

When you finish your PCB, you can output the Fabrication Files(gerber file) via :
Super menu > **Miscellaneous** > **Fabrication Output** , or by clicking the Fabrication Output button from the toolbar.



It will open a webpage to you, and you can download the gerber as a zipfile.

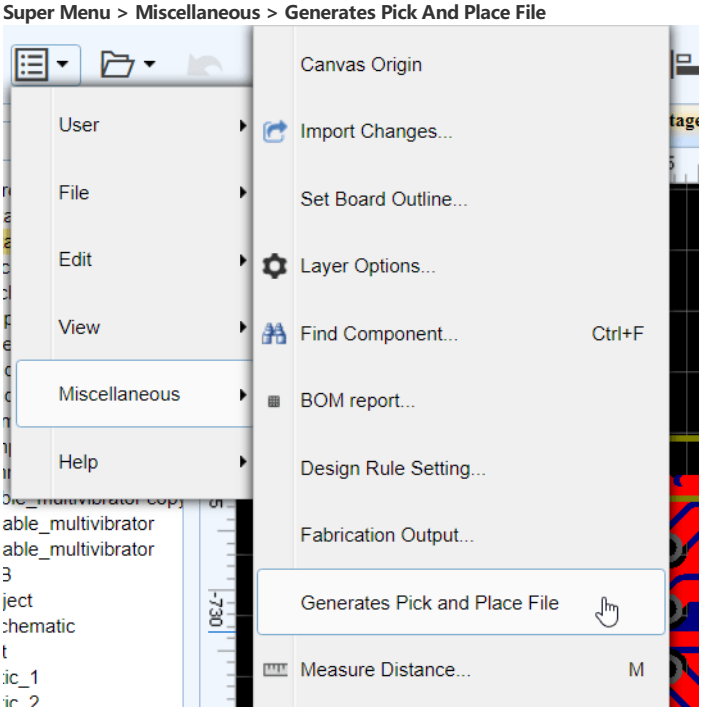


Notice: *Before order your PCB, Please read Essential Check Before Placing a PCB Order section!!*

Exporting Generates Pick And Place File

In PCB editor, if you want to generate Pick And Place as a CSV file, you can via:

Super Menu > Miscellaneous > Generates Pick And Place File



When you open the exported CSV file, you can see:

	A	B	C	D	E	F	G	H	I	J
1	Designator	Footprint	Mid X	Mid Y	Ref X	Ref Y	Pad X	Pad Y	TB	Rotation
2	Q1	SOT23	580mil	430mil	580mil	430mil	617mil	473mil	T	180
3	Q2	SOT23	770mil	430mil	770mil	430mil	807mil	473mil	T	180
4	Q3	SOT23	1040mil	120mil	1040mil	120mil	1003mil	77mil	T	0
5	R1	1206	680mil	150mil	680mil	150mil	680mil	95mil	T	90
6	R2	1206	500mil	150mil	500mil	150mil	500mil	95mil	T	90
7	R3	1206	540mil	750mil	540mil	750mil	540mil	695mil	T	90
8	R4	1206	910mil	750mil	910mil	750mil	910mil	695mil	T	90
9	R5	1206	730mil	750mil	730mil	750mil	730mil	695mil	T	90
10	C1	1206	820mil	200mil	820mil	200mil	820mil	255mil	T	270
11	C2	1206	1100mil	750mil	1100mil	750mil	1100mil	805mil	T	270
12	JP2	JST-2-SMD	1076.5mil	450mil	1120mil	450mil	974mil	489mil	T	270
13	JP1	JST-3-SMD	275.5mil	450.5mil	190mil	450mil	378mil	372mil	T	90
14										

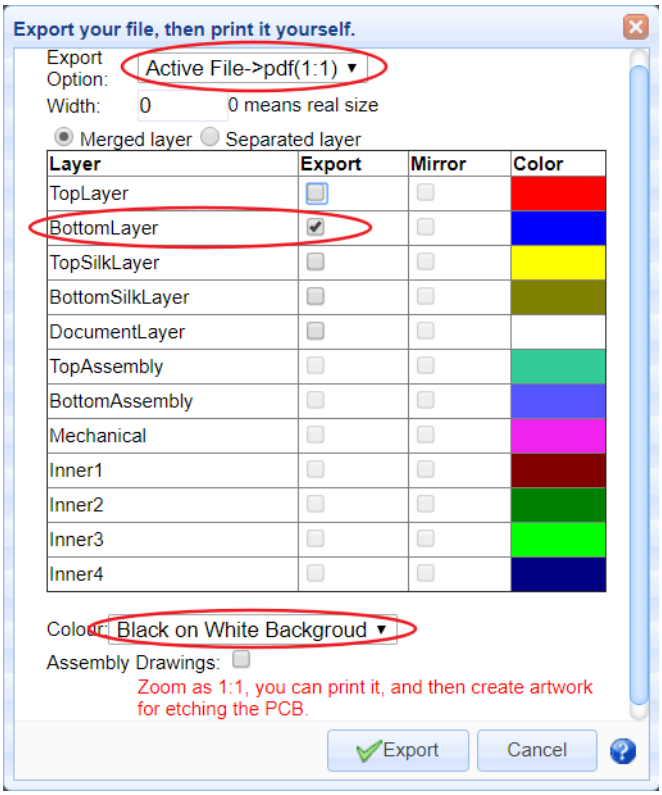
Exporting Print For Etching

If you don't want to order your PCBs from EasyEDA then maybe - for single and double sided PCB designs - you might like to try like using some home made PCB tech:

http://hackaday.com/2012/12/10/10-ways-to-etch-pcbs-at-home/

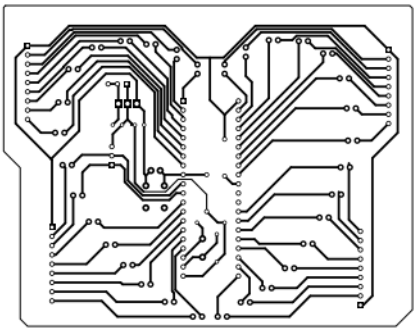
So here's how you can print your PCB layer by layer and then etch it onto a PCB.

Step 1) Export it to PDF, Using: **Document > Export...**, or **Document > Print...**

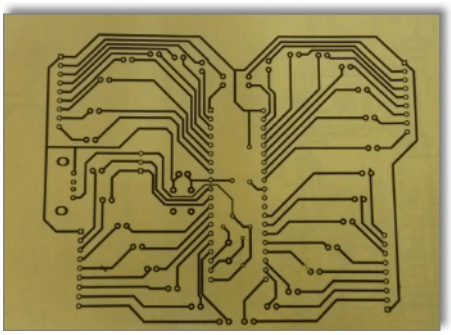


Note: *Make sure the Colour is Black on White Background.*

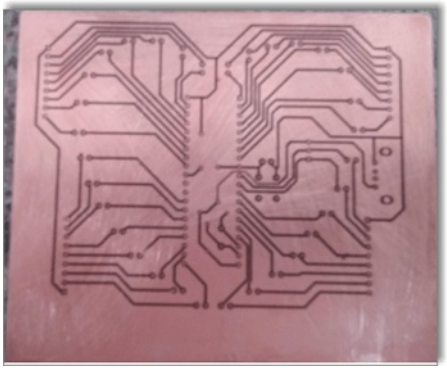
Step 2) Open the pdf file in a viewer



Step 3) Print it to paper

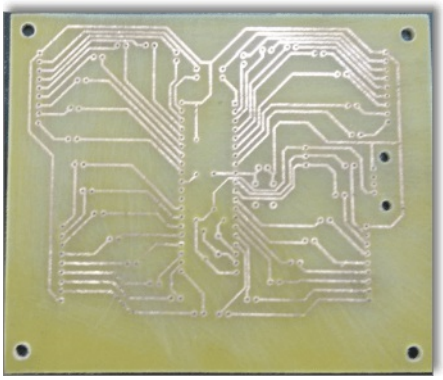


Step 4) Copy it to the copper

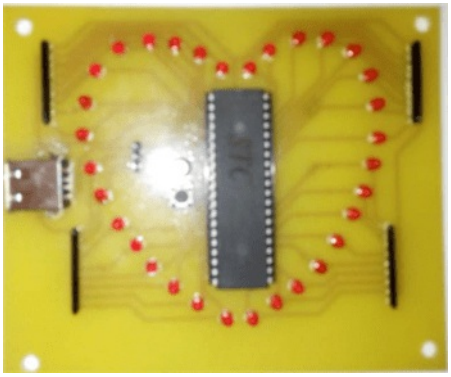


Step 5) Etch it.

Step 6) Drill it ...



Step 7) Get your soldering iron out!



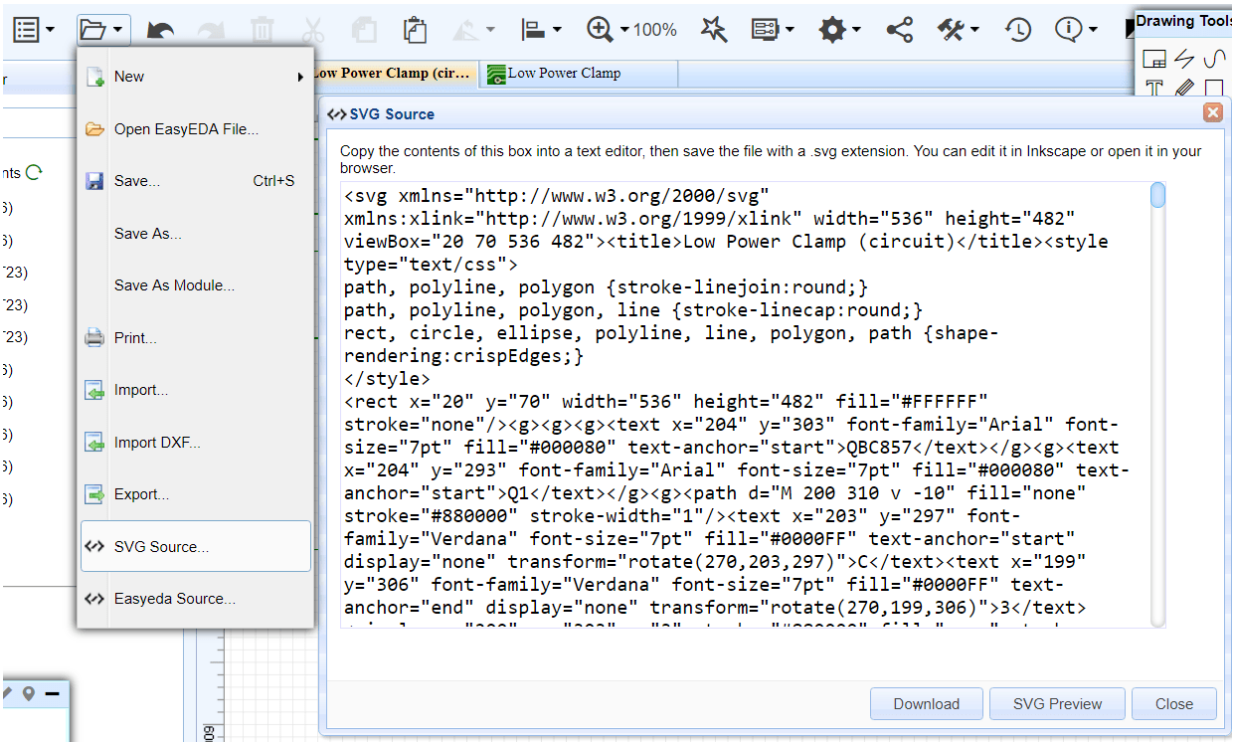
Exporting SVG Source

You can create an SVG sourcefile via:

Document > SVG source...

then copy the contents of this box into a text editor and save the file with a .svg extension. You can edit it in Inkscape or open it in your browser.

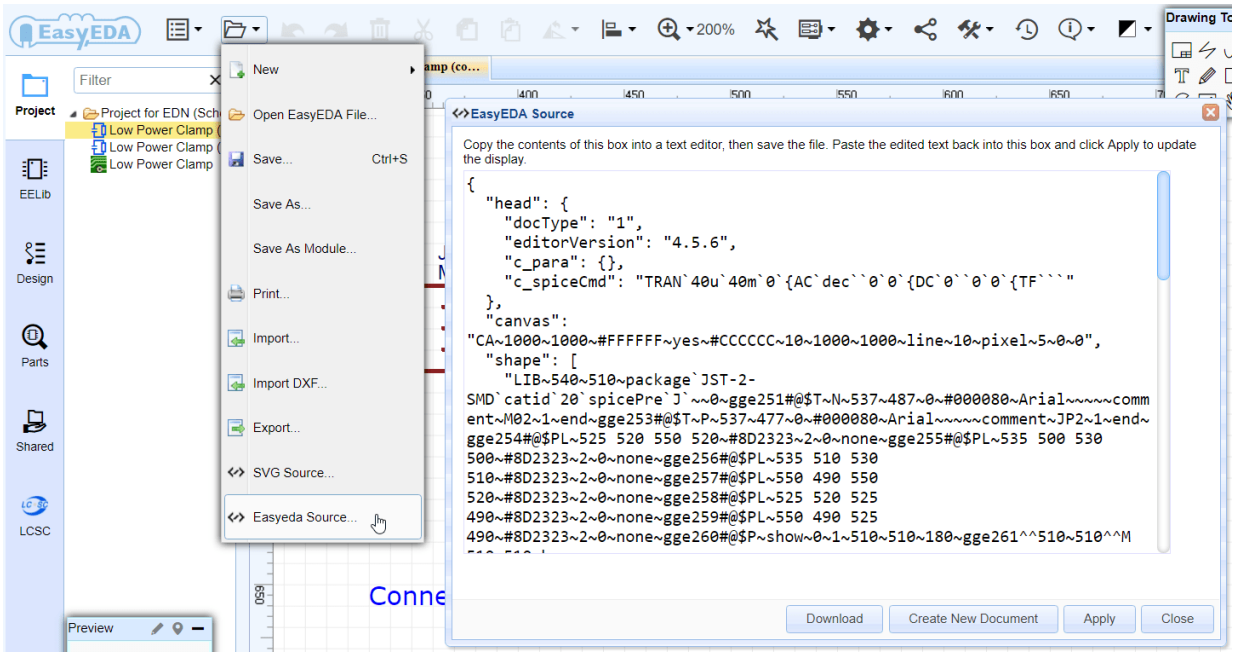
This solution doesn't need an internet connect so if you open EasyEDA offline, you can use it.



Exporting EasyEDA Source

You can create an EasyEDA source file via:

Document > EasyEDA Source...



Or **Project** > **Right Click** > **Download Project**, you will download a zip file with EasyEDA Source files for Schematics and PCBs.

EasyEDA Source is a **JSON** file which can be read by many other programs. Please see:

http://en.wikipedia.org/wiki/JSON

for more information.

The open EasyEDA Source file allows you to work on files at a text level which enables some powerful ways to manipulate schematic and spice files and symbols as well as PCB files and footprints.

Click on the **Download** button or copy the contents of this EasyEDA source into any text editor, then save the file. You can paste the text back into this box and click Apply to update the display. If you have made no changes to the text then the canvas will show your file exactly as if it was saved and reopened from the EasyEDA server.

This is a good way to share/backup your works. Your file doesn't need to be saved to EasyEDA's server. It can be highly compressed in any readily available format such as such as zip or 7z. It can be emailed to anyone who can then open it in EasyEDA without worrying if they have the same libraries as you.

EasyEDA team will provide more details of the EasyEDA Source soon to show how you can edit and even create drawings, schematics, symbols, footprints and PCB layouts in EasyEDA Source. It is also possible to copy and edit symbols straight out of a Schematic and save them as new Schematic Lib or Spice Symbols and even to create a new Spice Subckt from a Schematic.

Sharing

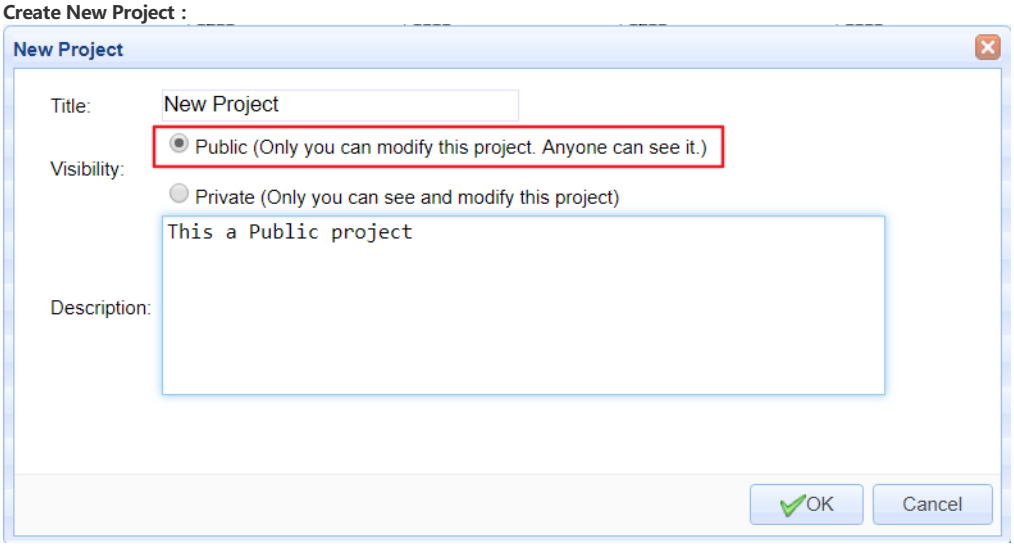
Sharing your work with others is a big feature of web based EDA tools and EasyEDA is no exception in offering you some nice features.

Share to Public

Did you create a really cool project with EasyEDA? Show it off and be super helpful to other EasyEDA users, you just need to set your projects to public, so others can explore your circuits.

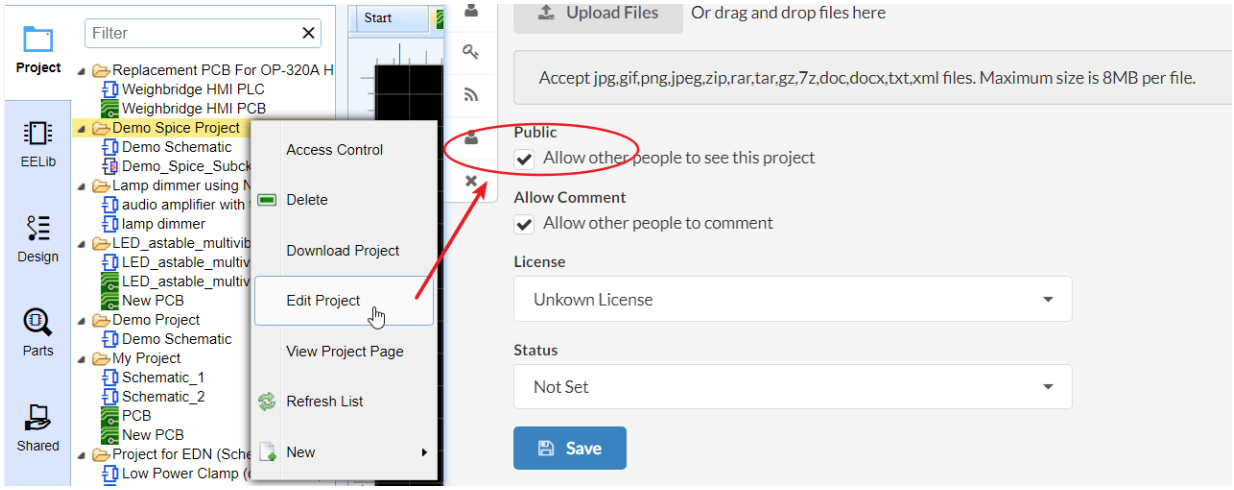
All projects in EasyEDA are set to private by default, your private project can not be shared with anyone.
i.e. to make it public, you should create a new project or right click and edit your existing project to be a Public project:

Create New Project :

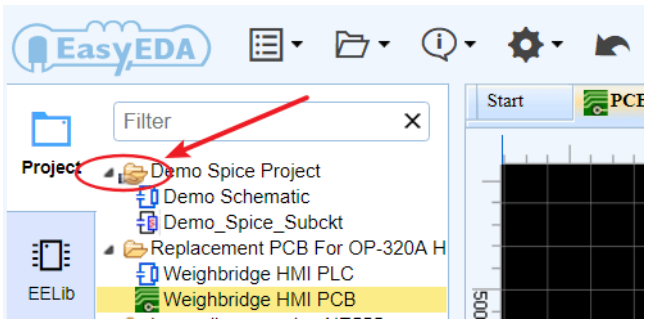


Edit Existing Project:

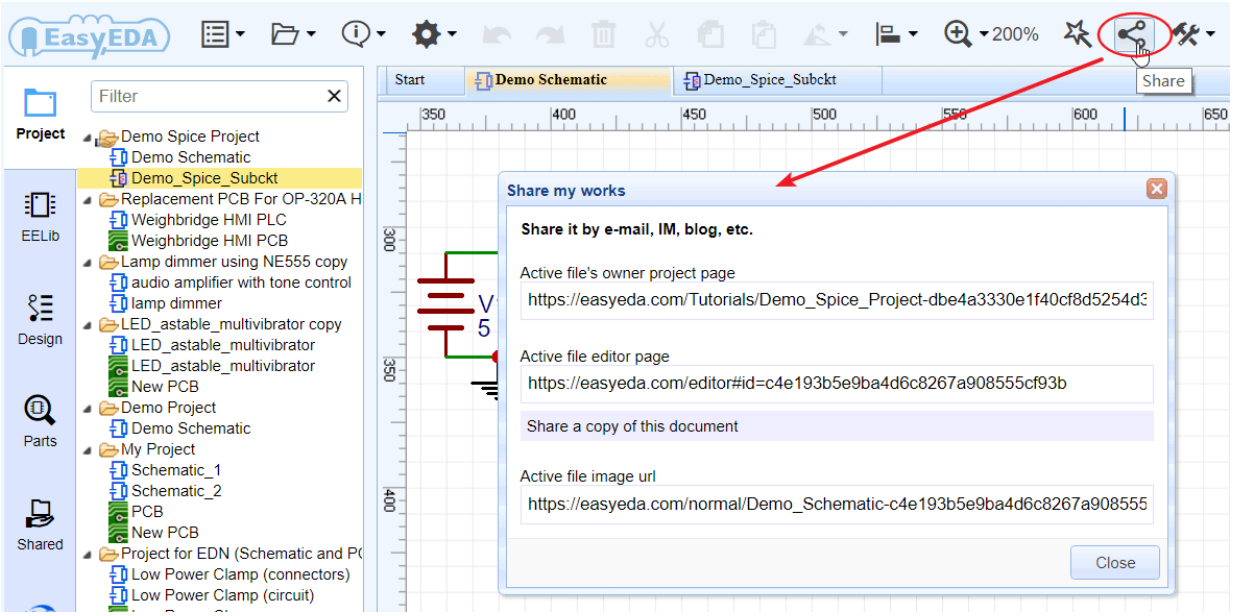
When you click **Edit Project**, it will open a website to allow you to set your project to be public.



After setting the project as public, you will see that the Project folder Icon is now shown as a hand holding the folder.



If you then open one of the documents in this share folder, you can then click the Share icon on the toolbar to open the Share my works dialog.



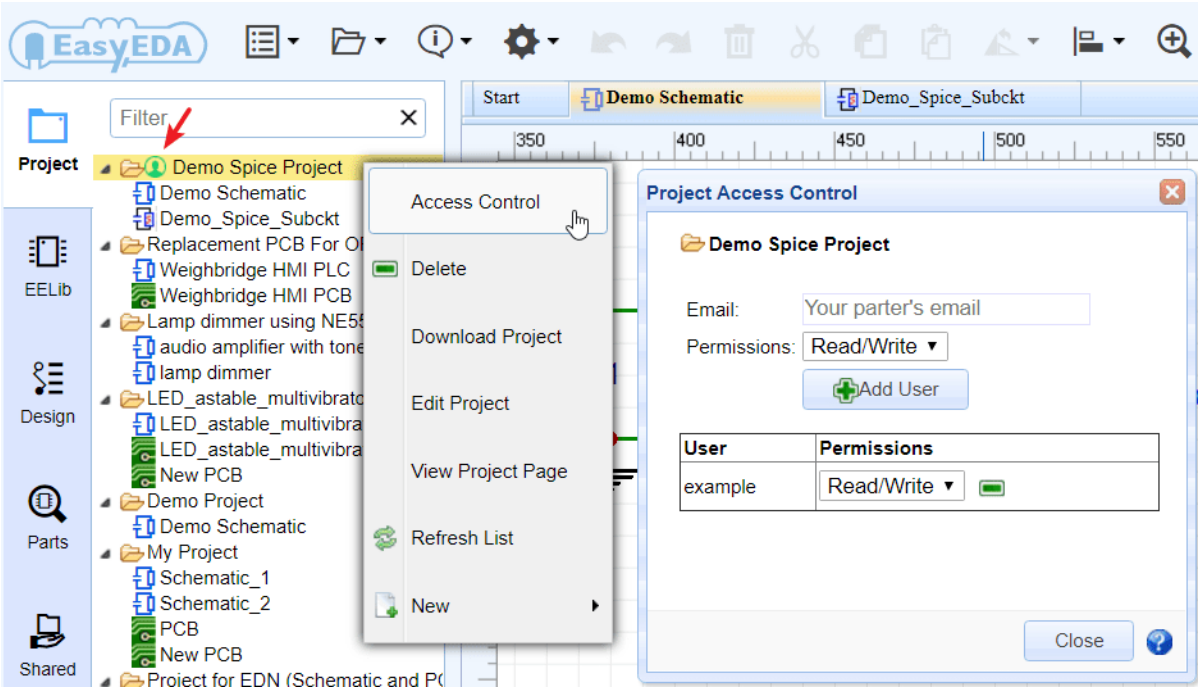
Access Control

How about sharing with selected people?

Can you share a private project with your partner? Can your partner modify your designs?

Yes, you can use **Access control** to do this.

Right click the project and you will see the Access Control on the context menu; clicking on it will open the Access Control dialog. After adding a user, a user icon will show up beside the project folder icon as below.

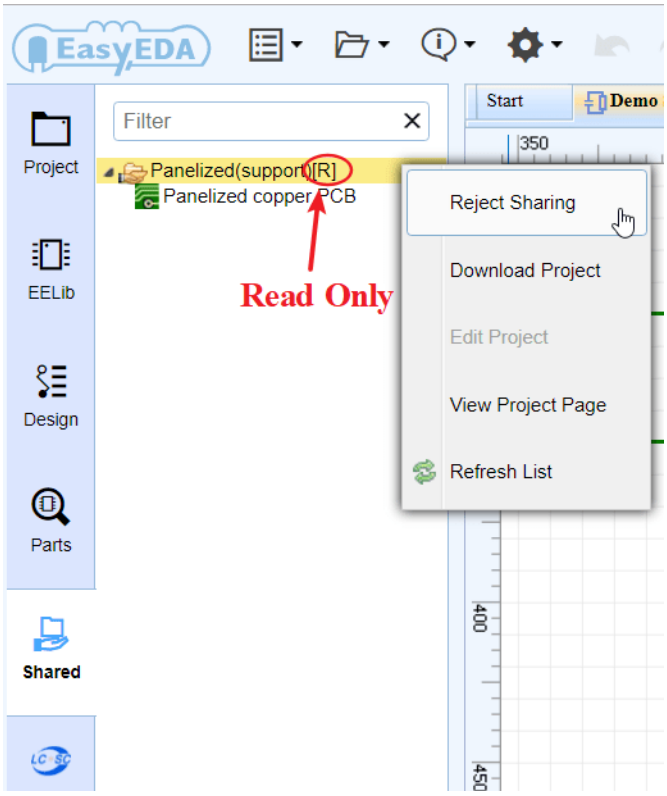


So if you want to share a project with someone,

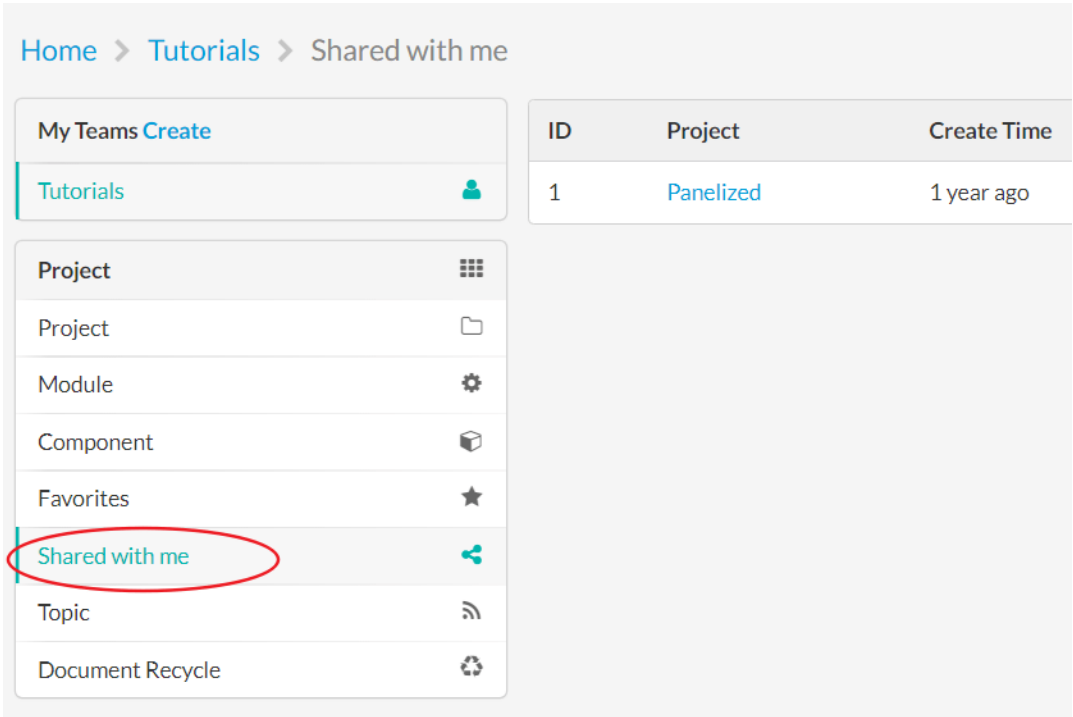
1. You just need to know their E-mail address which they have used to create an account with EasyEDA
2. You can share your project as **read only** or **read/write**.

After setting up **Access Control** and Permissions, your partner will find your project in the **Shared** section from the left **Navigation Panel** when they login.

If you partner doesn't wish to accept the shared project, they can reject it by right clicking on the project in the Shared with Me section and then clicking on Reject Sharing;



And you also can check projects that your partner has shared with you in the account dashboard:



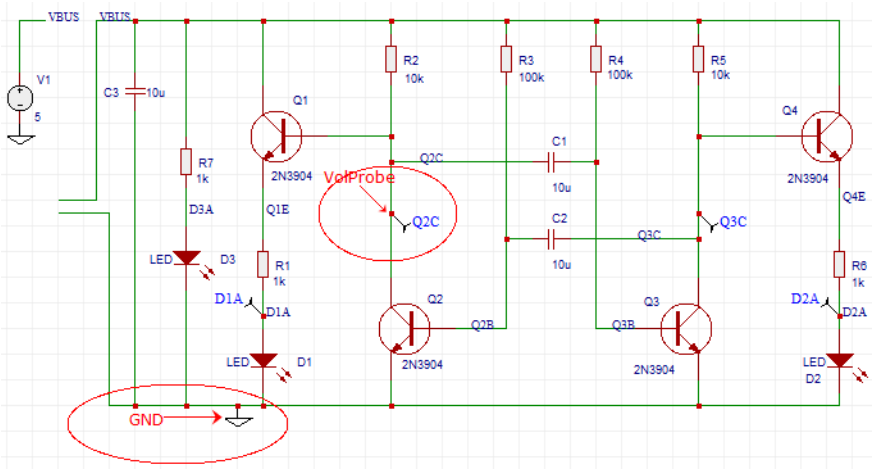
Simulation

Spice Simulation

Build the circuit

To simulate your circuit, at least until you become more familiar with the idea and concepts of simulation, first you should build a circuit as described in the chapter on **Creating The Schematic.**

The circuit below is the **simulation schematic** for the **Astable Multivibrator LED project**:



Now, to make your circuit simulatable, you should note that:

1. You do not have to draw the whole schematic again from scratch. You can `CTRL+C` copy the schematic you have already drawn for the PCB layout and `CTRL+SHIFT+V` paste it into a new schematic canvas and then save it into a new Project folder (with maybe the same name but with "simulation copy" or something added to the end to avoid the Design Manager flagging up duplicates or you accidentally editing the original);
2. You should then remove anything from the schematic that you do not want to include in the simulation. Connectors and mechanical items such as heatsinks and often manually operated switches can be removed. (Although there are situations where these items may need to be included in the simulation; that is getting off the topic and into advanced simulation territory so we will leave it there for the moment...)
3. You may want to replace a simple battery or - to simplify the simulation, reduce the size of the simulation output file and reduce the simulation time, a complicated power supply - with a simple spice V voltage source. You will almost certainly need to add some sort of voltage or current signal input source such as a simple SIN or PULSE source or maybe something more complex such as the EasyEDA electret microphone model, a guitar pickup model, a photodiode or an optoisolator. If you are simulating a power supply or a power amplifier of some sort, you may also need to add a representative load of some sort. Unless you are specifically simulating the effects of wiring impedances for Power Integrity, you can usually omit any power supply decoupling capacitors hung directly off the supply rails: they have no effect, clutter up the circuit, generate lots of useless output data and add to the simulation time;
4. Your circuits **must** have a **GND** net. You can use a NetLabel or NetFlag to add one. You can call this net GND or 0 (the numeral zero);
5. You should use the Schematic Design Manager to help verify that your circuits are wired correctly. It can be hard to debug wiring errors from the Simulation Results... dialog error messages;
6. As you draw a schematic, EasyEDA assigns default net names to all the wires. Any section of wires that are joined will be assigned the same net name. This is how EasyEDA "knows" that those wires are joined together.

These default net names are usually of the form N001, N002, etc.

Adding NetLabels to name nets (wires or nodes) which you would like to observe (probe) makes it much easier to identify traces when the simulator shows them in WaveForm. Remember that in any circuit, you may want to probe the voltages on nets other than the obvious Input or Output nodes that may be the nets you first think you will want to probe.

Probing voltages and currents

To probe voltages, you can add some Voltage Probes which can be found in the Wiring Tools palette. These will appear on your schematic auto-numbered as volProbe1, volprobe2, etc.

When you place a voltage probe onto a net, the name you give the voltage probe will overwrite any name that is already assigned to that net. So if you place a voltage probe called foo on a net call bar, that net is renamed to foo.

Therefore it is strongly recommended that you change the name of the voltage probe to be identical to the name of the net onto which you then place that probe (except for the letter case which is ignored).

If this net name is used somewhere else in the simulation - for example in an expression for an arbitrary source - then it is possible that the voltage probe name overwriting the net name will break the expression and so the simulation will give unexpected results or throw errors. Giving voltage probes identical names to their target nets avoids this problem.

It is also recommended that you name all nets because if you have used an EasyEDA-assigned default net name in an expression then, if you edit the schematic, say to insert an extra resistor or a current probe, then EasyEDA will reassign the default net names to different nets. This breaks the expression and so the simulation will give unexpected results or throw errors.

To probe the current in a wire you can place an instance of the Ammeter component, from the EasyEDA Libs, in series with the wire.

For an alternative method of probing voltages on nets and currents through the EasyEDA Ammeter component, see Probe.

Checking models and subckts

You then need to check that all the devices in the simulation schematic have the necessary and the correct spice models and/or subckts.

Missing spice models and subckts will be indicated in the Simulation Results... dialog after attempting to run a simulation but it is much easier to do this before you try to run a simulation.

Simple components such as resistors, capacitors and inductors do not pull models into the netlist because their models are built-in to Ngspice at a very low level but almost all other components will pull in either a .model statement or a set of line enclosed in the .subckt ... .ends spice

keywords.

By looking at the spice netlist that is generated as a simulation schematic is being created,

Super menu > Miscellaneous > Netlist for Document > Spice...

or

Super menu > Miscellaneous > Netlist for Project > Spice...

it is easier to check that each component in the schematic has pulled into the netlist an associated .model statement or .subckt ... .ends block of lines.

In the astable example spice netlist below, Q1 - Q4 are 2N3904 devices which all pull in - and share - the .model 2N3904 statement.

Similarly, D1 - D3 are the same LED device and pull in the shared .model LED statement.

Astable Multivibrator simulation copy

```
.param pi = 3.141593
V1 VBUS GND  5
R7 D3A VBUS  1k
R6 D2A Q4E  1k
R5 Q3C VBUS  10k
R4 Q3B VBUS  100k
R3 Q2B VBUS  100k
R2 Q2C VBUS  10k
R1 D1A Q1E  1k
Q4 VBUS Q3C Q4E 2N3904
Q3 Q3C Q3B GND 2N3904
Q2 Q2C Q2B GND 2N3904
Q1 VBUS Q2C Q1E 2N3904
D3 D3A GND LED
D2 D2A GND LED
D1 D1A GND LED
C3 GND VBUS  10u
C2 Q2B Q3C  10u
C1 Q2C Q3B  10u


.MODEL 2N3904 npn
+IS=1.26532e-10 BF=206.302 NF=1.5 VAF=1000
+IKF=0.0272221 ISE=2.30771e-09 NE=3.31052 BR=20.6302
+NR=2.89609 VAR=9.39809 IKR=0.272221 ISC=2.30771e-09
+NC=1.9876 RB=5.8376 IRB=50.3624 RBM=0.634251
+RE=0.0001 RC=2.65711 XTB=0.1 XTI=1
+EG=1.05 CJE=4.64214e-12 VJE=0.4 MJE=0.256227
+TF=4.19578e-10 XTF=0.906167 VTF=8.75418 ITF=0.0105823
+CJC=3.76961e-12 VJC=0.4 MJC=0.238109 XCJC=0.8
+FC=0.512134 CJS=0 VJS=0.75 MJS=0.5
+TR=6.82023e-08 PTF=0 KF=0 AF=1
.MODEL LED D
+ IS=661.43E-24
+ N=1.6455
+ RS=4.8592
.control
probe V(D1A) V(D2A) V(Q2C) V(Q3C)
quit
.endc
.END
```

In fact the astable example circuit has no elements defined by subcircuits but the principle is the same as for .model statements.

The example below of a simple 555 timer based monostable, includes a .model statement for a type of 2N7002 MOSFET and a subcircuit for the 555 timer which in turn, calls up .model statements for the bipolar transistors, QN and QP and the diode DA that are used within the subcircuit.

It is quite possible to call one subcircuit from within another subcircuit but let's not get too carried away just yet ... 555 monostable

```
.param pi = 3.141593
XU1 GND XU1_2 OUT VCC XU1_5 XU1_6 XU1_6 VCC 555
VGATE GATE GND  PULSE(0 9 0 10u 10u 10m 300m) AC 0
VBATT VCC GND  9
R4 XU1_2 VCC  2k
R1 XU1_6 VCC  100k
M1 XU1_2 GATE GND GND DI_2N7002K
C4 VCC GND  1u
C2 XU1_5 GND  10n
C1 XU1_6 GND  1u
**************************************
* Bipolar 555 timer model
**
* Rfix added to stop V(out) exceeding V(vcc)
* with no external load on OUTPUT pin.
**
* Last edited 140111
**
*           GND
*           |  TRIGGER
```

```
*           |  |  OUTPUT
*           |  |  |  RESET
*           |  |  |  |  CONTROL
*           |  |  |  |  | THRESHOLD
*           |  |  |  |  |  |  DISCHARGE
*           |  |  |  |  |  |  | VCC
*           |  |  |  |  |  |  | |
.SUBCKT 555 34 32 30 19 23 33 1 21
**
Q4 25 2 3 QP
Q5 34 6 3 QP
Q6 6 6 8 QP
R1 9 21 4.7K
R2 3 21 830
R3 8 21 4.7K
Q7 2 33 5 QN
Q8 2 5 17 QN
Q9 6 4 17 QN
Q10 6 23 4 QN
Q11 12 20 10 QP
R4 10 21 1K
Q12 22 11 12 QP
Q13 14 13 12 QP
Q14 34 32 11 QP
Q15 14 18 13 QP
R5 14 34 100K
R6 22 34 100K
R7 17 34 10K
Q16 1 15 34 QN
Q17 15 19 31 QP
R8 18 23 5K
R9 18 34 5K
R10 21 23 5K
Q18 27 20 21 QP
Q19 20 20 21 QP
R11 20 31 5K
D1 31 24 DA
Q20 24 25 34 QN
Q21 25 22 34 QN
Q22 27 24 34 QN
R12 25 27 4.7K
R13 21 29 6.8K
Q23 21 29 28 QN
Q24 29 27 16 QN
Q25 30 26 34 QN
Q26 21 28 30 QN
D2 30 29 DA
R14 16 15 100
R15 16 26 220
R16 16 34 4.7K
R17 28 30 3.9K
Rfix 30 0 1G
Q3 2 2 9 QP
.MODEL DA D (RS=40 IS=1.0E-14 CJO=1PF)
.MODEL QP PNP (level=1 BF=20 BR=0.02 RC=4 RB=25 IS=1.0E-14 VA=50 NE=2)
+ CJE=12.4P VJE=1.1 MJE=.5 CJC=4.02P VJC=.3 MJC=.3 TF=229P TR=159N)
.MODEL QN NPN (level=1 IS=5.07F NF=1 BF=100 VAF=161 IKF=30M ISE=3.9P NE=2
+ BR=4 NR=1 VAR=16 IKR=45M RE=1.03 RB=4.12 RC=.412 XTB=1.5
+ CJE=12.4P VJE=1.1 MJE=.5 CJC=4.02P VJC=.3 MJC=.3 TF=229P TR=959P)
.ENDS
*SRC=2N7002K;DI_2N7002K;MOSFETs N;Enh;60.0V 0.300A 2.00ohms  Diodes Inc. MOSFET
.MODEL DI_2N7002K  NMOS( LEVEL=1 VTO=2.50 KP=32.0m  GAMMA=3.10
+ PHI=.75  LAMBDA=104u RD=0.280 RS=0.280
+ IS=150f  PB=0.800 MJ=0.460 CBD=98.8p
+ CBS=119p  CGSO=60.0n CGDO=50.0n CGBO=390n  )
*   -- Assumes default L=100U W=100U --
.control
tran 500u 500m
probe V(GATE) V(OUT)
quit
.endc
.END
```

Whoa! I thought this was supposed to be easy?

At this stage you might be forgiven for feeling a sense of panic at the sudden complexity of what should be a simple job of checking that all the symbols in your simulation schematic have the necessary and correct models associated with them.

Well, to quote the Hitchhikers Guide to the Galaxy:

Don't Panic!

All you have to do is check that every different type of device - not every instance - in your simulation schematic has a corresponding .model or .subckt statement associated with it.

If it hasn't then the first thing to check is that you have got all the device names right.

If you still haven't pulled in a .model or a .subckt then it probably means that a simulation model for that device is not available in the EasyEDA libraries. This may be because we haven't been able to find a copyright unrestricted model, we haven't had time to build our own or

we just haven't caught up with entering all the thousands of possible models yet ...

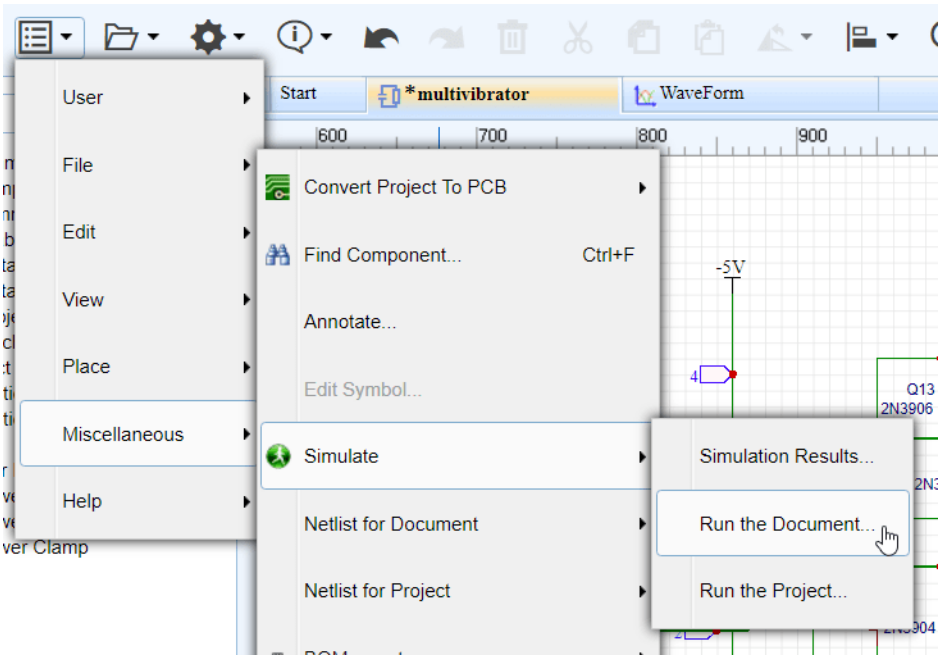
If you're desperate then EasyEDA gives you several ways that you can include third party models in your simulation but more of that later.

If you're really desperate then if you ask us nicely we just might find or even build one for you. Please see the section on How to get help?

Once you are satisfied that you have done everything to pull in the right models then you can save and then run the simulation, but don't worry, EasyEDA will still tell you if you have made any mistakes in the Simulation Results.. dialog. It's just that until you are familiar with using simulation it really is easier if you do the checking before your run a simulation because the error reporting from Ngspice may include warnings and error messages about other things besides just missing models and that can make it very confusing for beginners.

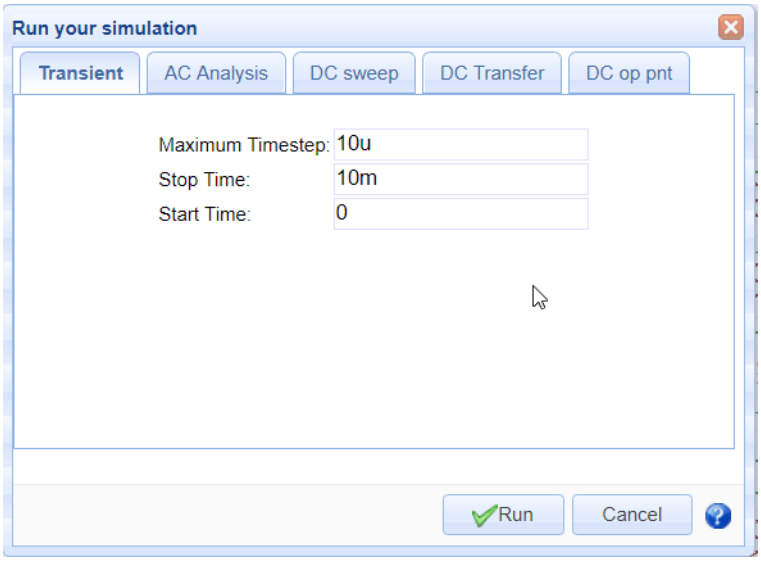
Run Simulation

Your schematic is ready, so now you can run it. **SuperMenu > Miscellaneous > Simulation > Run The...**



Run the Document: Just for the active schematic, you can also open this dialog using the `CTRL+R` hotkeys.

Run the Project: EasyEDA will merge all the schematics in the project to one, and simulate them.



EasyEDA provides the following simulation analyses:

- Transient: the time domain response of the circuit;
- AC Analysis: the frequency domain response of the circuit (including an experimental FFT);
- DC sweep: the DC response of the circuit as a voltage or current source or a component or parameter is swept between user specified limits;
- DC Transfer: computes the DC small-signal value of a transfer function (ratio of output variable to input source), input resistance, and output resistance of the circuit;
- DC op simulation: computes the dc operating point of the circuit with inductors shorted and capacitors opened.

For more information about these analyses, please refer to:

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.1.2.1

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.1.2.2

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.1.2.3

Please note that although using Ngspice for it's simulation engine, at present (140218) EasyEDA does not support all the possible analysis modes available in Ngspice.

Note that for transient simulations, at present (140218):

the maximum value of (Stop Time-Start Time)/(Maximum Timestep) = 1000

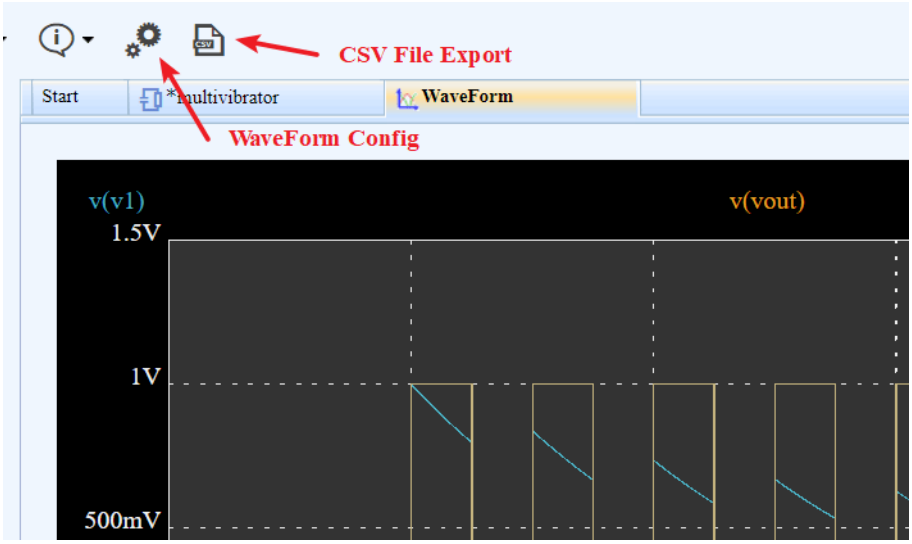
See also CTRL+R to Run Simulation Immediately

WaveForm

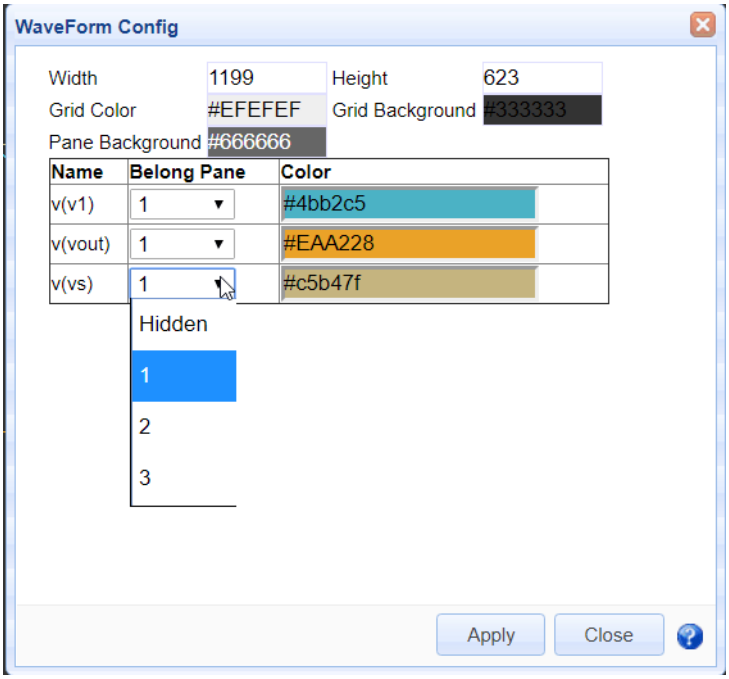
EasyEDA's WaveForm display is super easy but to make sure you don't miss any of the features it supports, we've put some detailed information into this tutorial.

Transient, AC Analysis and DC Sweep simulation results are shown in the WaveForm trace viewer.

After you run a spice simulation which should plot some traces, EasyEDA will automatically open a WaveForm tab like the image below.



The Waveform window width and height, trace, grid and background colours and the placement of traces in up to three panes can all be configured using the WaveForm Config dialog. To open the WaveForm Config dialog, click the Config button on the toolbar above the Waveform window.



WaveForm allows the display of traces in any selection of up to three vertically stacked plot panes. The Y axes automatically scale to fit the units and the range of the traces being displayed. Traces can be hidden but at least one trace must be visible.

X and Y trace data can be seen on-screen just by moving the mouse cursor around the plot area of a pane with the readout adapting to the Y axes in each pane.

Delta X and delta Y trace data can be seen on-screen using a Left-Click and Drag select box, with the readout adapting to the Y axes in each pane. Returning the cursor to within a small radius of the starting point of the select box -without releasing the Left-Click - returns the readout to X and Y trace data.

Left-Click, Drag and then releasing the Left-Click zooms all plots, synchronised across all panes, horizontally. Double clicking anywhere in the WaveForm window resets the zoom.

Vertical plot zoom is not supported but traces are dynamically autoscaled to fit the available pane height as the horizontal zoom is changed.

The window can moved around within the EasyEDA window using the horizontal and vertical scroll bars or using Right-Click and Drag.

WaveForm plot data can be exported in CSV format for further analysis and manipulation in external programs such as LibreOffice Calc, Scilab or Excel, however a particular feature of EasyEDA is that the WaveForm window can not only be saved in an EasyEDA Project but that the plots in a saved WaveForm window can be viewed and manipulated in exactly the same way as when they first appear as a result of a simulation. This makes it easy to compare the results from several simulations.

	A	B	C	D
1	time	v(v1)	v(vout)	v(vs)
2	0.000000000000	0	0	0
3	0.000000000010	0	0	0
4	0.000000000020	0	0	0
5	0.000000000040	0	0	0
6	0.000000000080	0	0	0
7	0.000000000160	0	0	0
8	0.000000000320	0	0	0
9	0.000000000640	0	0	0
10	0.000000001000	0	0	0
11	0.000000001064	0.064	0.06382	0.064
12	0.000000001192	0.192	0.19121	0.192
13	0.000000001448	0.448	0.44463	0.448
14	0.000000001768	0.76816	0.75904	0.76816
15	0.000000002000	1	0.985	1
16	0.000000002064	1	0.98325	1
17	0.000000002192	1	0.97978	1
18	0.000000002448	1	0.97293	1
19	0.000000002960	1	0.95953	1
20	0.000000003985	0.99999	0.93393	1
21	0.000000006034	0.99999	0.88714	1

Once saved in a Project, a WaveForm window can be exported as a .pdf, .png or .svg file into your browser window. This can then be saved to your device so it is easy to create professional quality documentation.

Build Your Own Simulation Component

There are several reasons why you may want to build your own simulation component.

- You may have downloaded a spice model in text form for a device for which EasyEDA has no symbol;
- Perhaps you have designed a simulation schematic of a circuit for which there is no readily available spice model and you need to create your own symbol for it;
- You have a subckt for a device and EasyEDA already has a symbol for it but you want to use your subckt in place of the one already attached to the EasyEDA symbol.

EasyEDA gives you three ways to build your own components so that you can simulate them:

1.From a model in text form

1. If you already have a spice subcircuit in text form, for example one that you have downloaded from a component manufacturer's website but you haven't got a spice symbol for it, then you can create a spice symbol and attach a .subckt definition to it.

2. First make a note of the exact name given in the .subckt line. Spice names are case insensitive but can only be made up from alphabetical, numeric and underscore characters.

   For example: LM741EE_demo would be a valid name and would be seen as identical to lm741ee_Demo but **LM741EE-demo** and **LM741EE~demo** are **invalid** names because they contain invalid characters.

   In this example we shall assume that you have a .subckt with the name: *Demo_Spice_Symbol*

3. Next, create your symbol.

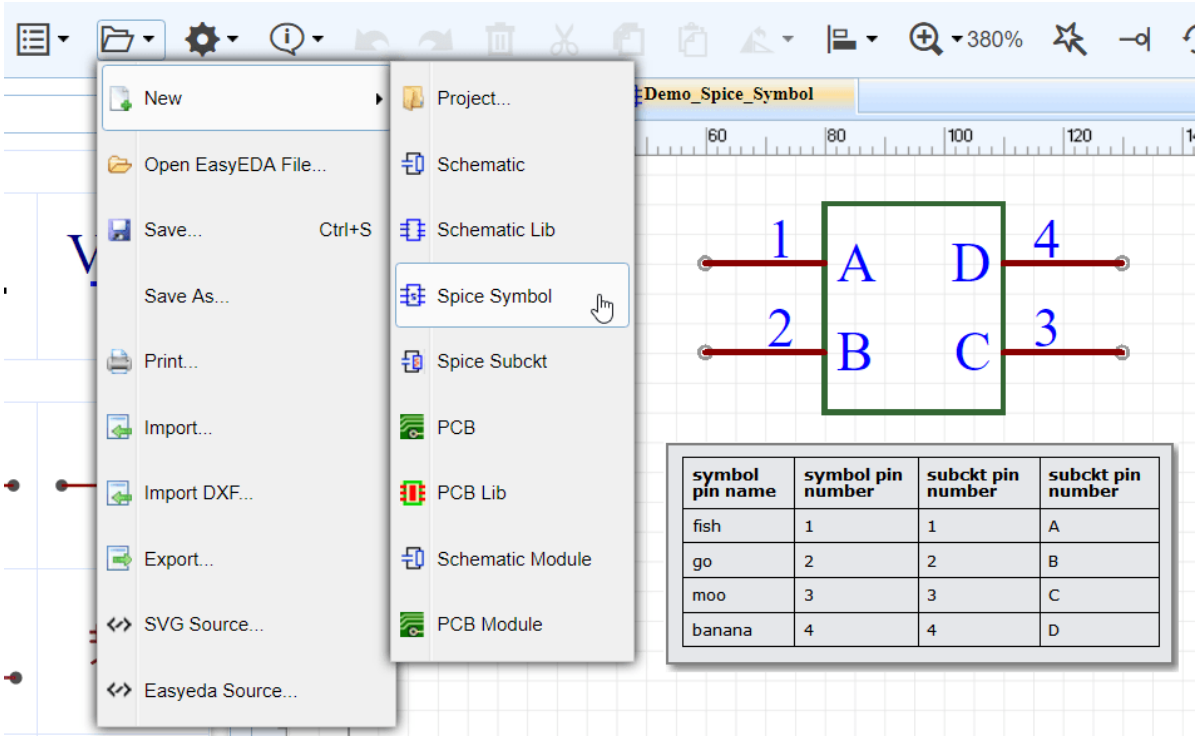
   You need to do this using:

   Document > New > Spice Symbol... instead of: **Document > New > Schematic Lib...**

   because that option does not support attaching a spice model to a schematic symbol.

   Using **Document > New > Spice Symbol...** also automatically sets the Spice Prefix of the symbol to X which is essential for a .subckt definition to attach to your symbol.

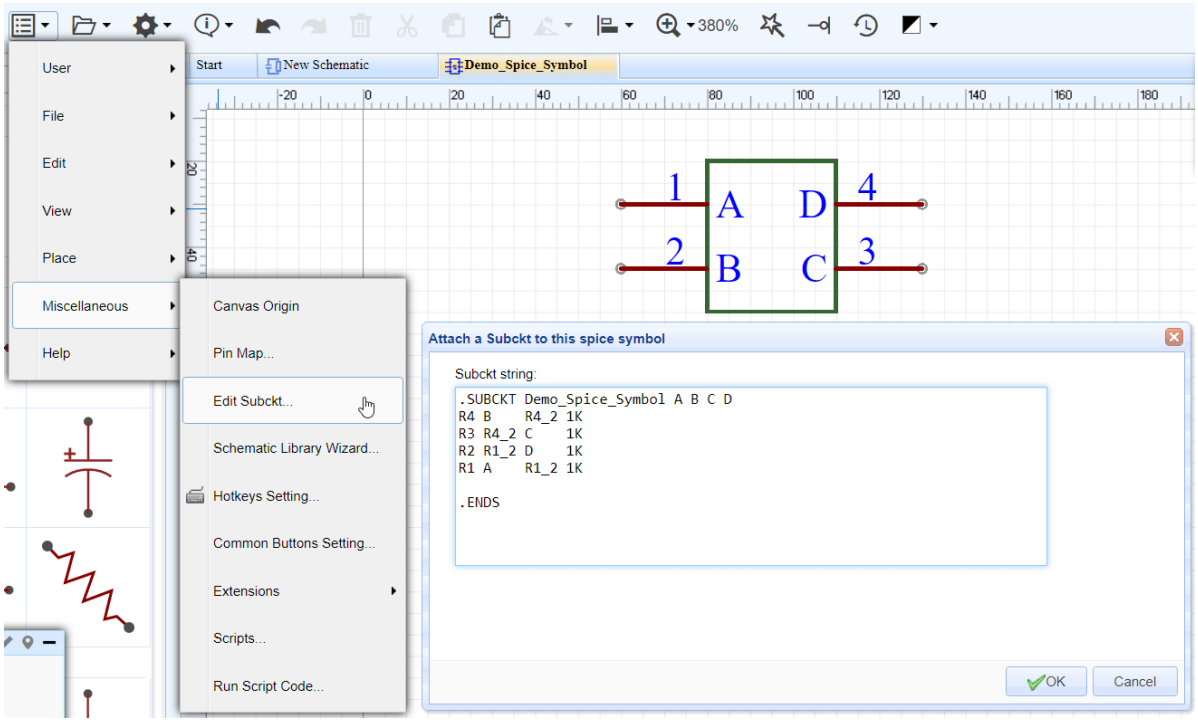
   Remember that the Spice Pin names assigned to the symbol **must** be numbered in the same order that they appear in the .subckt. So, if there are four pins named A, B, C and D in the order 1, 2, 3 and 4 in the subckt, then the corresponding pins on the symbol must be in the same number order. They don't have to have the same names: you could have symbol pins named; fish, go, moo and banana but if they correspond, in the same order, to the .subckt names A, B, C and D then they must be numbered as:



4. You are now ready to attach your subcircuit to the symbol by opening the attached this spice symbol with subckt dialog using:

   Super menu > Miscellaneous > Edit Subckt...

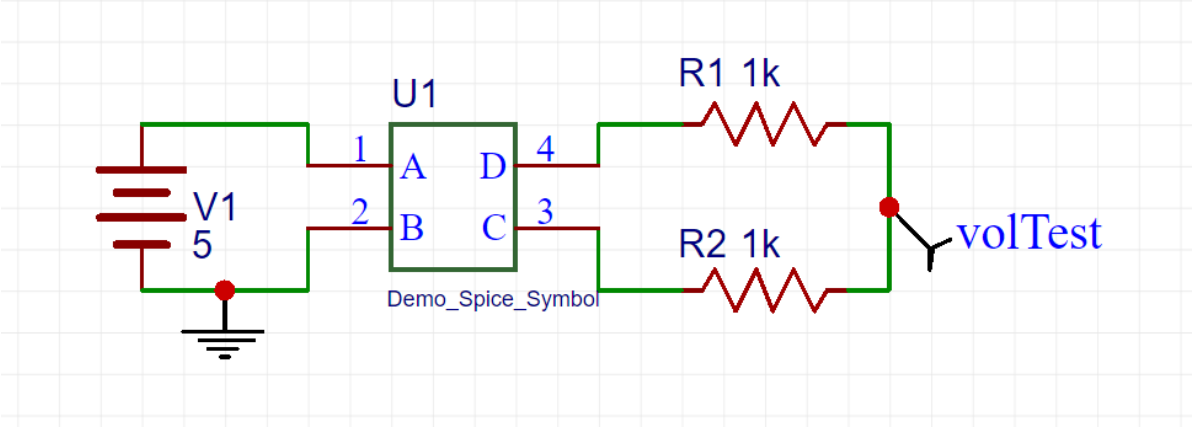
   and then pasting the .subckt definition that you wish to use into the Subckt string: text area.



5. Click OK and save the symbol but remember: the symbol name must be identical to the name of the subckt:

   .SUBCKT Demo_Spice_Symbol A B C D

6. Lastly, add your new spice symbol to a schematic and run a simulation.

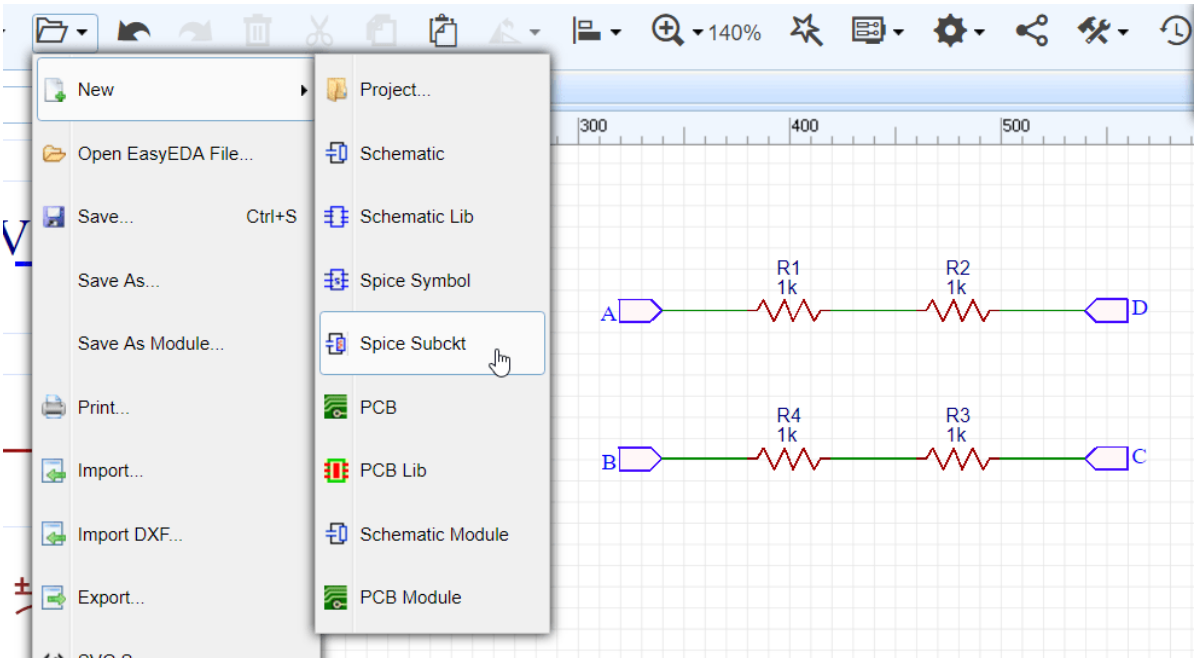


7. If you run a DC op simulation on this example, the result, shown in the Simulation Results... window, should be 2.5V

2.From a subcircuit in schematic form

1. Create a spice symbol and subckt circuit.
2. The same as (1) above, create a spice symbol.
3. Next create a spice subckt as a schematic:

   Document > New > Spice Subckt...



   Draw the schematic that you want EasyEDA to turn into a subckt and attach to your symbol.

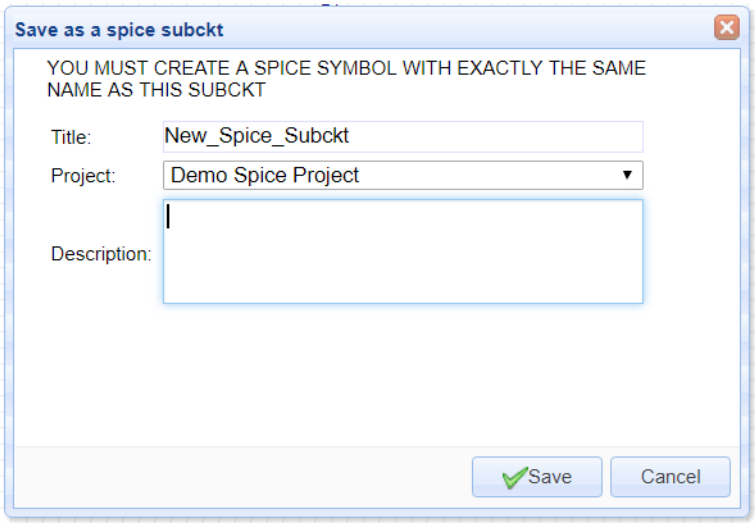
   To connect your simulation schematic to your symbol there must be a net in the circuit that is to be attached to each pin of the symbol. Each of these connecting nets in your circuit must have the same name as that of the symbol pin to which it connects. For example if your symbol has four pins called A, B, C and D then your simulation schematic must have exactly four connecting nets; one called A, one called B, one called C and one called D.

   To attach these nets in the schematic to the pins in the symbol you must name them using NetPort from the Wiring Tools palette.

   Do not use NetLabel or NetFlag.

   NetPort is used to distinguish those subckt nets that are to connect to symbol pins from all other nets named using EasyEDA default net names and those added using NetLabel or NetFlag.

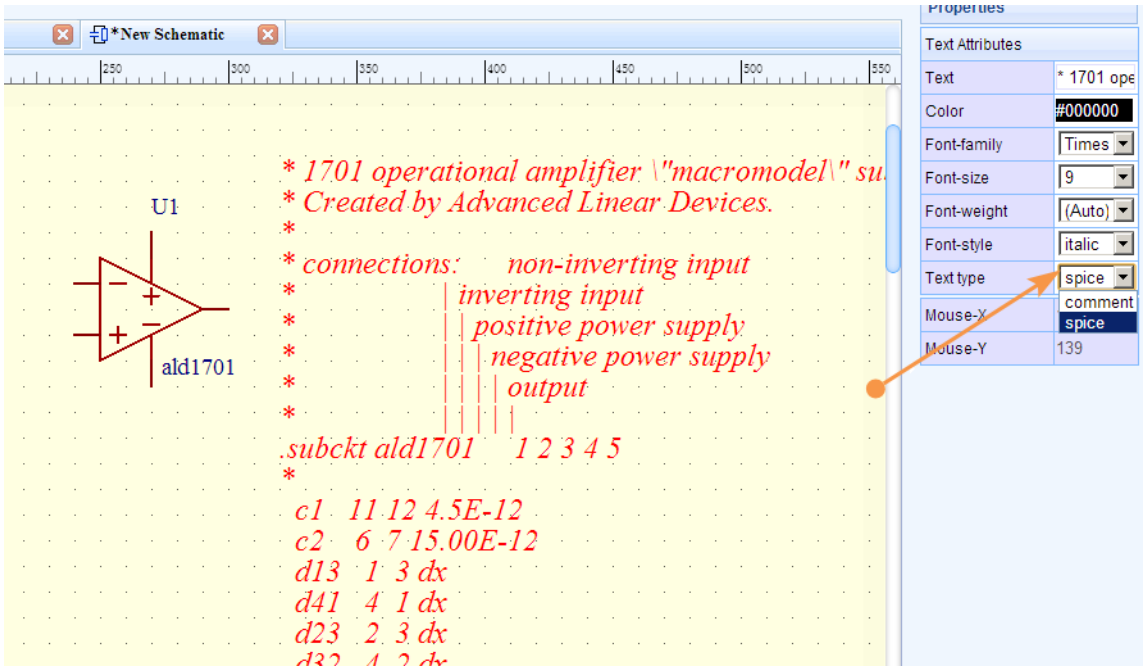
4. Save your spice subckt with exactly the same name as your spice symbol.



5. Lastly, add your new spice symbol to a schematic the same as in (1) above and run a simulation.
6. If you run a DC op simulation on this example, the result, shown in the Simulation Results... window, should be 2.5V.

3.From a spice directive in a schematic

1. When you already have a spice symbol which has a subckt attached to it - for example, an opamp symbol from the EasyEDA Libs - but you want to use a subckt for a different device which is not already in the EasyEDA Libs, then you can use this method to easily attach a subckt to a symbol directly in your schematic.
2. Paste your .subckt text into the schematic.
3. Next, select the pasted text and, in the right hand Properties panel, change the Text type from comment to spice. This will create a spice directive which the simulator will then incorporate into the spice netlist.
4. Next, select the symbol and, either directly in the symbol or in the right hand Properties panel, edit the Model text to exactly the same name as in your pasted subckt.
5. Check that the Spice pin order of the symbol matches that of the pasted .subckt and edit it if necessary (not all subckts for a given type of device use the same Spice Pin order!).
6. Save your schematic and then you can run your simulation.



4.Using .models instead of .subckts

1. All three techniques can be used to attach .model statements to symbols in exactly the same way as .subckts but after placing the symbol in your schematic, you must use:

   Super menu > **Miscellaneous** > **Edit Symbol...** to set the symbol's Spice Prefix to the appropriate letter for the device model you are using.

2. You also have to know the spice pin order for the type of .model statement you are using because, unlike .subckts, the .model statement does not show this explicitly.

   The Spice Prefixes and Spice Pin names and orders for the most commonly used devices for which you may want to use different models are listed below:

Spice Prefix	Device description	Spice pin order
D	Diode	Anode = 1 Cathode = 2
J	Junction field effect transistor (JFET)	D = 1 G = 2 S = 3
M	Metal oxide field effect transistor (MOSFET)	D = 1 G = 2 S = 3
Q	Bipolar junction transistor (BJT)	C = 1 B = 2 E = 3
X	Subcircuit	Depends on subckt
Z	Metal semiconductor field effect transistor (MESFET)	D = 1 G = 2 S = 3

Actually there is a way to save the symbol with the required Spice Prefix so that you don't have to edit it every time you place a new instance of the symbol into a schematic ... but that needs a bit more insight into editing the EasyEDA Source for the symbol so will be left for the moment.

We did say that EasyEDA Source enables some powerful ways to manipulate schematic and spice files and symbols! :)

Advance Tips

EasyEDA uses Ngspice as the simulation engine, so once you get more familiar with it you can use many other commands and features of Ngspice that are not directly available via the EasyEDA UI.

The lists below show which Ngspice commands are currently supported by EasyEDA and which are not.

Ngspice Commands Whitelist

EasyEDA allows these commands:

let define option options unlet op tf tran pss ac dc pz sens disto noise fft fourier meas alter run while repeat dowhile foreach if else end break continue label goto linearize print probe echo

Ngspice Commands Blacklist

EasyEDA does not currently allow these Ngspice commands:

reshape snsave snload circbyline alias deftype display destroy setplot setcirc setscale transpose xgraph gnuplot wrdata wrs2p hardcopy asciiplot write compose print eprint codemodel load cross undefine listing edit dump psd spec show showmod sysinfo altermod resume state stop trace save iplot altermod status delete step remcirc reset aspice jobs rspice bug where newhelp tutorial help oldhelp removecirc quit source shift unset unalias history shell rusage cd version diff rehash cdump mdump mrdump settype strcmp devhelp inventory source

Probe

An alternative to using the volProbe element to probe voltages in a circuit - which avoids the possibility described in Probing voltages and currents of overwriting net names and consequently corrupting any expressions that use them - is to use the Probe command.

For example, to probe the voltages on two nets named in and out all you have to do is enter this text into the schematic:

Probe V(out) V(in)

and then, in the Properties panel, set the Text type to spice to set it to be included in the spice netlist as a spice directive.

You can also use the Probe command to probe a current in your circuit.

To measure the current in a wire you insert an Ammeter, from the EasyEDA Libs, in series with the wire you wish to probe. EasyEDA then inserts a small subckt comprising a 0V, zero resistance, voltage source in series with the wire and then probes the current in that voltage source. Hence although an ammeter in an EasyEDA schematic is shown with an A prefix, it is spice netlisted with an X prefix (for a subckt call) followed by V (for the voltage source).

For example, to add the current in an Ammeter, named A*load*current1, to the command probing the two voltage probes above, you would change the Probe command in your schematic to:

```
Probe V(out) V(in) I(XVA_load_current1)
```

It is also possible to use expressions in a Probe command. In the example above, if we assume that V(out) is connected directly to a grounded load then, to plot the power dissipation of the load, you can add this expression:

```
V(out)*I(XVA_load_current1)
```

the Probe command list:

```
Probe V(out) V(in) I(XVA_load_current1) V(out)*I(XVA_load_current1)
```

Note that your probe list can be as long as you like but all entries in a Probe command list must be entered as a single line of text with no returns.

A useful feature allowing you to easily switch between different sets of probe points is that any number of Probe commands, each with their own list of probe points, can be included in a schematic by setting the Text type of only one at a time to spice and setting all others to comment.

But this is just the tip of the iceberg ...

Using CTRL+R to Run Simulation Immediately

As described in Run Simulation, using:

`CTRL+R`

will open the

Run the Document

or

Run the Project

simulation control dialog.

That approach is a great way for you to quickly and easily set up and Run any of the most commonly used simulation analyses types but EasyEDA gives you a way to harness the real power of **Ngspice**.

Simply by entering your simulation control commands as text, directly into the schematic and setting the Text type to spice, you can set up powerful spice analyses. You can run these straight away, without needing the Run your simulation dialog just using the `CTRL+R` hotkeys.

Using this method it's quick and easy to create and run more advanced simulation analyses and to make automated measurements on your circuit.

Here's a quick insight into how it works but you can skip this if you like and just get into how to make this amazing feature work for you!

EasyEDA automatically embeds the simulation commands set up in the Run your simulation dialogs within a control section. You can see this in the spice netlist for any circuit that has been through a simulation run at least once via:

Simulate > Simulation Results... > Download netlist

The control section starts with the .control command and ends with the .endc command. All commands between these delimiters are run in an Ngspice interactive simulation control mode.

Now, you don't need to worry about these two commands because EasyEDA automatically inserts them in the netlist in the right place to enclose your commands so all you need to do is to enter a list of commands as text, anywhere in the schematic canvas and then, in the Properties panel, set the Text type to spice for it to be included in the spice netlist as a **spice directive**.

The following examples show some of the things you can do using **spice directives**.

Run a **transient simulation** with the following parameters:

Maximum Timestep: 10u **Stop time**: 11ms **Start Time**: 1ms

just add this text anywhere on the schematic canvas:

```
tran 1u 11m 1m
```

set Text type to **spice**

then type:

`CTRL+R`

Run an **AC Analysis** with the following parameters:

Type of Sweep: Decade

Number of points: 100 (per decade)

Start Frequency: 1k

Stop Frequency: 1Meg

just add this text anywhere on the schematic canvas:

```
ac dec 100 1k 1Meg
```

set Text type to **spice**

then type:

`CTRL+R`

Run a **DC Sweep** with the following parameters:

(And, yes, you can sweep component values, not just sources!)

Source to Sweep: R1

Start Value: 1k

Stop Value: 2k

Increment: 100

just add this text anywhere on the schematic canvas:

```
dc R1 1k 2k 100
```

set Text type to **spice**

then type:

`CTRL+R`

A couple of more advanced examples:

Run a **Fourier** analysis:

```
tran 2u 2m 0
fourier 1K V(volOut)
run
probe V(volOut)
```

set Text type to spice

then type:

`CTRL+R`

For more information on Fourier Ngspice, see:

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.17.5.24

Run an **FFT** analysis:

```
tran .1m 2s 0
run
linearize
fft v(out)
probe db(mag(v(out)))
```

set Text type to spice

then type:

`CTRL+R`

For more information on **FFT** in Ngspice, see:

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.17.5.25

Run a DC op pnt analysis and Print the power in the load into the Simulation results window:

```
op
print V(out)*I(XVA_load_current1)
```

set Text type to spice

then type:

`CTRL+R`

Measure the gain and 3dB bandwidth of an amplifier.

This prints the gain and bandwidth values of this x1 and x10 amplifier example:

https://easyeda.com/file*view*Find-gain-and-bandwidth_8GE0KRFDn.htm

in the Simulation Results... window:

```
* This is a control block.
* Note: variables in a control block must start with
* a letter.

* Set up an AC analysis:

ac dec 100 1k 10Meg

* Define a 3dB value:

let neg3dB = 20*log10(sqrt(2)/2)

* Convert the outputs of both amplifiers into dB:

let x1gain_dB = DB(v(x1Avcl))
let x10gain_dB = DB(v(x10Avcl))
```

```
* Find the low frequency gain of each amplifier
* (look at the Bode plots in WaveForm and choose
* a frequency where the gain is level; i.e. well
* below any possible hf gain peaking):

meas ac x1_lfgain_dB find x1gain_dB at=1k
meas ac x10_lfgain_dB find x10gain_dB at=1k

* Subtract 3dB from the lf gains to find
* a value 3dB down from the lf gain:

let x1_3dBdown = x1_lfgain_dB + neg3dB
let x10_3dBdown = x10_lfgain_dB + neg3dB

* Find the frequencies at which the outputs
* are 3dB down from the lf gains:

meas ac x1_f3dB when x1gain_dB = x1_3dBdown
meas ac x10_f3dB when x10gain_dB = x10_3dBdown
```

set Text type to spice

then type:

`CTRL+R`

For more information on the meas statement, see:

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.17.5.37

Simulation eBook

Introduction

EasyEDA is not just a way to draw circuit diagrams and design PCBs. It is also a circuit simulator.

A circuit simulator is basically a specialised mathematical program, optimised to construct and then solve the equations that define the behaviour of the circuit that has been described to it in the circuit diagram.

The circuit simulation program that EasyEDA uses is called ngspice. Ngspice is Free and Open Source Software (FOSS) that is in turn based on a simulator called SPICE that was originally written by Larry Nagel.

What this book is for

This book is an introduction to circuit simulation in EasyEDA using ngspice.

It starts with the basics of how to avoid some of the most common mistakes that cause simulations to fail and then goes on to illustrate how to set up a circuit so that it will simulate successfully and produce meaningful results. It also discusses some aspects of understanding how, what might at first appear to be unexpected or even nonsensical results, can arise.

The book then introduces and illustrates more advanced techniques such as:

- probing signals such as voltages, currents, powers and resistances;
- configuring signal sources;
- setting up different types of analyses;
- making measurements such as rise times, RMS values and bandwidths;
- defining component values using parameters;
- using expressions to calculate component values such as for a resistor to draw a given load current or capacitances for a specified filter cutoff frequency;
- using manufacturer's device models;
- setting up complex simulations including arbitrary voltage and current sources.

What this book is not for

- This is not a book about learning to use EasyEDA to draw schematics. For general information about using EasyEDA please refer to the EasyEDA Tutorial;
- This book does not teach electronics. Whilst there may be examples of circuits and explanatory text that are helpful in understanding electronics, it is assumed that the user already has sufficient knowledge of electronics to understand the content of this book.* Although ngspice is similar to other variants of spice and a lot of the information and techniques in this book may be applicable to some of those variants, this book is written specifically about circuit simulation in EasyEDA using ngspice.
- Except where necessary to help explain some aspect of the behaviour of simulation, this book does *not* go into any detail of how circuit simulation in general and ngspice in particular actually works. For more information about those areas, please see the links below:

More information about Larry Nagel and SPICE is available from here:

http://www.omega-enterprises.net/The%20Origins%20of%20SPICE.html

Larry Nagel's PhD Dissertation:

Laurence W. Nagel., "SPICE2: A Computer Program to Simulate Semiconductor Circuits,"

Memorandum No. ERL-M520, University of California, Berkeley, May 1975.

http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/ERL-520.pdf

is actually very readable and instructive.

For more information about electronic circuit simulation and spice in particular, see:

http://en.wikipedia.org/wiki/Electroniccircuitsimulation

and:

http://en.wikipedia.org/wiki/SPICE

More information about ngpsice is available from here:

http://ngspice.sourceforge.net/presentation.html

Who this book is for

All simulation tools and how they interact with schematic capture tools are different so even for people with experience of using simulation tools, it is worth at least skimming through the sections of this book to get an idea of where to find information if things don't seem to go quite as planned.

However, for people with limited or no previous experience of using simulation tools, this book is *essential* reading.

Why?

Because, as a newcomer to the world of circuit simulation, it is very tempting just to launch straight into trying to build and run lots of interesting and exciting simulations.

Sadly, however, this will almost certainly be a very frustrating and discouraging experience because many of those simulations will either not run, will not function correctly as circuits or will appear to give nonsensical results! This can happen for all sorts of very simple reasons but to newcomers with no experience of simulation, those reasons can seem utterly incomprehensible.

So, to avoid a lot of discouragement and wasted time, it is worth taking a few minutes to understand some of the most common things that can cause a simulation to fail or to run but give unexpected answers.

How the book is structured

This book is not some dry tome with lots of words, diagrams and snippets of code. It is written as a live, interactive document. Rather than having to read lots of text and then having to go into EasyEDA to create circuits to try things out, live simulation examples are embedded directly into the text to illustrate the points being discussed.

Just go to a topic or a section heading; read a couple of paragraphs and then open and run the embedded simulation to see exactly what the text is about.

Every embedded simulation must be saved before it can be run. Users who have joined EasyEDA can save the files to their own project folder. Non-registered users can save the files to Anonymous Files. In either case, examples can be copied and edited so that users can try out different things in whatever way helps their understanding of what the example is trying to explain, building up their own personal library of teaching examples as they progress through the book.

Newcomers are encouraged to work through the book in a linear manner because each section builds on the knowledge and insights gained from all the previous ones. Skipping sections leaves gaps that can catch out the novice and hinder understanding ideas presented in later sections.

- The book begins by introducing some of the concepts and terms that are essential for a basic understanding of what a simulator is and how to use it effectively.
- Then, using simple interactive simulation examples it describes and illustrates how to avoid the most common pitfalls in building, running and interpreting the results from simulations.
- The book goes on to illustrate ways to create and show connectivity using wires and netlabels in schematics. It then discusses techniques for probing voltages and currents and how they relate to and can affect or be affected by the netnames that are assigned to nets either automatically by EasyEDA or manually by the user placing netlabels.
- More advanced techniques for probing voltages and currents using the 'probe' command are described, moving on to using this command to measure power dissipations, resistances and conductances.

This section describes the use of 0V sources to measure currents also introduces and E,F, G and H dependent sources and the very powerful arbitrary or B Dependent Sources.

In this section, the concept of the 'Spice Directive' is also introduced, describing how to turn inactive 'comment' text into active 'spice' text that the simulator then recognises as an instruction to do something.

- The 'probe' concept is then extended by describing how several different 'probe' commands can be typed into a schematic and - simply by selecting which one is switched from being 'comment' text into 'spice' text - they can be used to swap between different selections of values and nodes to be probed.
- The use of the 'let' command to further extend and simplify the probing of signals is introduced with a simple example. This command will feature heavily in later sections about making measurements based on the results of simulations.
- Early interactive simulation examples introduce the basic way of setting up and running a simulation using the green running man 'Simulate...' icon (the Green Man) via the 'Run the Document...' option to open the 'Run your simulation' dialogue and then select the

type of analysis to run.

- Later examples show how to use the CTRL+R Hotkey as a shortcut to the 'Run your simulation' dialogue.

This idea is then extended with the concept of using 'spice directives' to run simulations straight from the CTRL+R Hotkey so that, with a few simple key strokes, it is easy to switch between several different simulations from a single simulation schematic.

- DC operating point, DC Transfer function, DC Sweep, AC (frequency domain) and Transient (time domain) Analyses are described in some detail.

Using the DC Sweep to sweep the value of a resistance in a circuit and to sweep the ambient temperature for circuit is covered in this section.

There are detailed sections on setting up each of the various time

domain signal sources that are available, such as SINE, PULSE and more.

Configuring and using the frequency domain AC source that is built into to every independent signal source is covered.

- The need for and different ways of defining initial conditions such as voltages on nets and across capacitors and currents in inductors are demonstrated.

Using PULSE, EXP, PWL and B Sources to 'kick start' circuits is also illustrated in this section.

- The concept of using parameters (i.e. variables that are used in the simulation) to define things like multiple component values and signal source settings is described.
- The use of parameters in expressions (i.e. formulas or equations) to calculate values such as the capacitance for a given RC time constant with a given resistance is covered and then extended into using them in B Sources.
- The concept of predefined functions and their use in expressions is described in detail extending into the creation of user defined functions using the .func statement.
- The concept and scope of device models in simulation is discussed in detail.
- Ways to add 3rd party models and subcircuits into schematics and to use them with the predefined schematic symbols from the EasyEDA Libs is covered in detail.
- The use of 3rd party models is extended into attaching them to custom schematic symbols.
- The use of expressions and functions in the creation of custom behavioural models is covered with reference to some of the in-house EasyEDA (EE suffix) models.

Introductory concepts of Spice simulation

For every circuit being simulated, EasyEDA converts the schematic into a textual description of the circuit that is then passed to the simulator.

This textual description of the circuit is called a **spice netlist.**

- Note that the spice netlist is not the same as the netlist that is generated from a schematic and which is then passed though to PCB layout.

The netlist is a list of all the components used, how they are connected together and descriptions of them, called **models**, so that the simulator knows the behaviour of each component is to be simulated. The netlist also includes instructions to the simulator, called **spice directives**, to tell it what type of **analysis** is to be run. It may also include lists of values, called **parameters**, that are to be used to directly define component values or as part of more extensive calculations, called **expressions**.

Just like real ones, circuits to simulated require power sources and often signal sources. Therefore the netlist will include any **voltage sources** and **current sources**. These can be **Independent Sources** or **Dependent Sources**. Independent Sources just generate DC voltages or currents or can generate time domain signals such as sinusoids and pulses or they can generate a signal of a given amplitude and phase for frequency domain simulations (in fact they can be configured to generate all three but that will be covered later). Dependent Sources can also be used to generate DC voltages or currents but their main use is to generate outputs that are **functions** of that other signals in the circuit, for example by using an expression to describe the output of a source as a current that is equal to the sum of the squares of two input voltages.

The netlists of more advanced simulations can include instructions, called **measure statements**, to the simulator to perform calculations on the results of the simulation itself, for example to measure the rise time of a pulse, the RMS value of a signal or the 3dB bandwidth of a filter circuit.

About naming conventions

Before going any further it is important to understand the naming conventions used throughout this book and throughout all the simulations

Fields on a line are separated by one or more blanks, a comma, an equal (=) sign, or a left or right parenthesis; extra spaces are ignored. A line may be continued by entering a "+" (plus) in column 1 of the following line; ngspice continues reading beginning with column 2. A name field must begin with a letter (A through Z) and cannot contain any delimiters. A number field may be an integer field (12, -44), a floating point field (3.14159), either an integer or floating point number followed by an integer exponent (1e-14, 2.65e3), or either an integer or a floating point number followed by one of the following scale factors:

Suffix	Name	Factor
T	Tera	1e12
G	Giga	1e9
Meg	Mega	1e6
K	Kilo	1e3
mil	Mil	25.4×1e-

meg	Meg	6
m	milli	1e-3
u	micro	1e-6
n	nano	1e-9
p	pico	1e-12
f	femto	1e-15

Letters immediately following a number that are not scale factors are ignored, and letters immediately following a scale factor are ignored. Hence, 10, 10V, 10Volts, and 10Hz all represent the same number, and M, MA, MSec, and MMhos all represent the same scale factor. Note that 1000, 1000.0, 1000Hz, 1e3, 1.0e3, 1kHz, and 1k all represent the same number. Note that M or m denote 'milli', i.e. 1e?3

- The suffix Meg MUST to be used for 1e6

Node names are case insensitive.

Node names may either be plain numbers or arbitrary character strings, not starting with a number.

The ground node must be named "0" (zero). For compatibility reasons "gnd" is accepted as the ground node and will internally be treated as a global node and be converted to "0".

Note the difference in ngspice where the nodes are treated as character strings and not evaluated as numbers, thus "0" and "00" are distinct nodes in ngspice (whereas they are not in SPICE2).

Ngspice and therefore, EasyEDA, requires that the following topological constraints are satisfied:

- Each circuit has to have a ground node (gnd or 0)!
- The circuit cannot contain a loop of voltage sources and/or inductors and cannot contain a cut-set (series connected set) of current sources and/or capacitors.
- Each node in the circuit must have a dc path to ground.
- Every node must have at least two connections except for transmission line nodes (to permit unterminated transmission lines) and MOSFET substrate nodes (which have two internal connections anyway).

These constraints will be covered in more detail later,

Introduction to using a simulator

Using a simulator is not quite the same as building a real circuit. There are many things that can catch out the newcomer to simulation because neither real world nor most simulator components are ideal. The departures from ideal are often different between the two and if these differences between real and simulated components in a circuit are not clearly understood, they can lead to confusion when the results of a even a simple simulation are different from those expected.

Part of the problem is that a user's expectations are informed by experience of real world measurements. Therefore, it is important to understand exactly what measurements are being made and how the simulation circuit will affect the results. In cases where there are differences between real and simulated results this may also require a deeper understanding ofwhat is going on in those circuits and what assumptions are being made - often unconsciously - about them.

Learning to use a simulator means thinking more about what the real world really looks like, how it differs from the theoretical world of textbook problems and simple diagrammatic circuit representations and therefore what the results of measurements are likely to be.

Avoiding common mistakes

The following section describes and illustrates some of the most common mistakes, misunderstandings and causes for confusion.

Prefix conflict error

This error message will appear:

If:

There are two copies of the same schematic in the same project folder: CctA and Copy_of_CctA.

In this case, the component prefixes (reference designators) will be the same in both CctA and Copy_of_CctA.

Or:

There are two different schematics in the same project folder, CctA and CctB but there are some components in CctB that have the same prefixes as in CctA.

Then:

If a simulation run is attempted using:

Green Man > Run the Project

then EasyEDA will try to merge all the schematics in the Project into a single schematic and then run a simulation on that big schematic.

This will fail because EasyEDA will find repeated instances of components with the same prefix.

Else:

if you try to run a simulation using:

Green Man > Run the Document

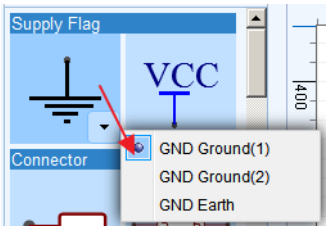
or:

CTRL+R

then EasyEDA will run a simulation only on the schematic in the current active tab in the editor window (if you have only one schematic open then it will run that one but if you have more than one schematic open then it will run only the one in the selected tab).

If you have given a unique prefix to every component in that selected sheet then this simulation will run because EasyEDA will find only single instances of every component in the schematic.

All simulation circuits MUST have a ground node



A feature of the way simulators work is that they MUST have a ground node (also referred to as the 0 net) somewhere in the circuit. All voltages probed in the circuit, unless *explicitly* probed as voltage differences are then measured with respect to that ground node. The ground node can be placed anywhere in the circuit that is convenient for the purposes of the simulation but it must exist somewhere in that circuit.

SIMULATION SCHEMATICS THAT DO NOT HAVE A GROUND NODE WILL NOT RUN.

This is illustrated in the following two examples:

With no ground symbol, the simulation will not run:

All simulation schematics MUST have a ground 01

Once any of the available ground symbols is added, the simulation will run:

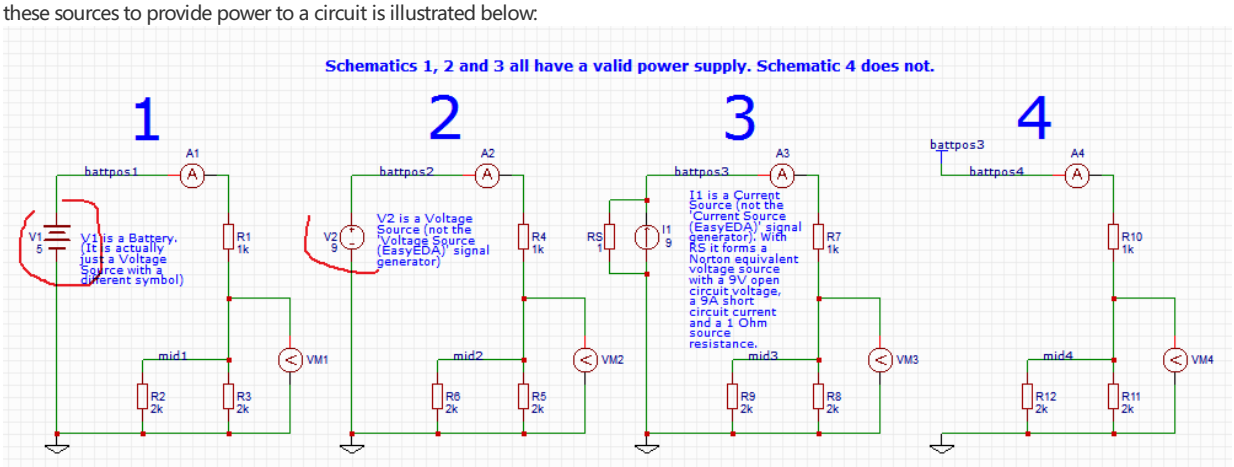
All simulation schematics MUST have a ground 02

All simulation circuits MUST have a power and/or signal source

In exactly the same way that a real circuit must have some sort of power supply - even if that power supply is actually the signal source itself (for example, a crystal set or a volume control potentiometer) or comes from a capacitor pre-charged to some voltage or an inductor pre-charged to some current prior switching the circuit on - a simulation schematic must have a power and/or a signal source or an initial condition such as a capacitor pre-charged to some voltage or an inductor pre-charged to some current prior to the start of the simulation.

Power supplies can be built from ideal zero series resistance voltage sources. Simple but more realstic models of battery and voltage regulator supplies can be built using voltage sources with some series resistance or current sources with a parallel resistance (i.e. Thevenin or Norton equivalent sources).

EasyEDA provides a wide range of signal sources. These will be covered in detail in their own section on Sources but the basic use of some of these sources to provide power to a circuit is illustrated below:



All simulation schematics MUST have a power supply too

Every point in a simulation schematic MUST have a DC path to ground

Unlike real circuits, every point in a simulation schematic MUST have a DC path to ground (or 0 net). Attempting to run a simulation with a node that has no DC path to ground will fail with errors.

Before discussing the significance of the DC path to ground, however, it is essential to look at some of the basic components and sources that are used in almost all simulations and to understand how they affect these DC paths.

Values and DC paths of RLC components

This section is about the values of - and DC paths through - resistors, inductors and capacitors.

- Inductors in ngspice have zero series resistance (i.e. ESR = 0).
    - Therefore, inductors have a DC path through them.
    - Inductors in ngspice have no parasitic parallel capacitance or resistance.

- Capacitors in ngspice have no parasitic parallel conductance (i.e. they have an infinite DC resistance).
    - Therefore capacitors in ngspice do not have a DC path.
    - Capacitors in ngspice have zero series resistance (i.e. ESR = 0).
    - Capacitors in ngspice have no parasitic series inductance (i.e. ESL = 0).
- Inductors and capacitors in ngspice can be set to positive and to negative values and to zero. Beware that setting a component to a negative value may cause the circuit connected to them to exhibit instability or oscillation. This in turn may cause the simulation to fail.
- Resistors obviously have a DC path through them.
- Resistors in ngspice can be set to positive and to negative values but cannot be set to exactly zero: they MUST have a non-zero value. Setting a resistance to zero will cause the simulation run to fail with an error. This is basically because any voltage difference across a zero resistance (such as may occur normally in a circuit or even just as a consequence of numerical "noise" in the simulation calculations) will generate an infinite current, which will obviously crash out of the top end of the simulators calculation dynamic range. This applies to all resistors in ngspice, including the Ron and Roff values in switches and resistances given in device models. Beware that setting a resistor to a negative value may cause the circuit connected to it to exhibit instability or oscillation. This in turn may cause the simulation to fail.
- A common cause of confusion in simulation is the apparently unexpected behaviour of circuits using open circuit switches. Just like switches in the real world, switches in EasyEDA have non-zero ON resistances (Ron). They also have finite OFF resistances (Roff). If an EasyEDA switch is connected in series with a voltage source into an open circuit load (such as presented by a VolProbe) then the voltage at the output of the switch will be equal to the open circuit voltage of the voltage source *whether the switch is open or closed*. This is different from the real world experience because in a real circuit, the voltage would be measured with a voltmeter or an oscilloscope probe having a much lower resistance than the open circuit resistance of the switch.
  To illustrate this, let’ s take a simple example of a switch connected in series with an ideal voltage source set to 1V and then measure the V(Output) of the switch when open and closed.
  With the switch closed the expected output is V(Output) = 1V.
  And that’ s what the simulation shows.
  OK. With the switch open the expected output is V(Output) = 0V.
  What the simulation actually shows is V(Output) = 1V.
  What’ s that all about? Surely the switch must not be simulating properly and is stuck closed?
  Well, no actually; the switch and voltage are doing exactly what they should. It’ s just that these results are unexpected because there's a misunderstandingabout how the simulated circuit differs from a real circuit.
  No real switch has an infinite OFF state resistance but similarly, there is no such thing as a real open circuit that presents an infinite load impedance.
  The Voltage and Current Controlled Switches and the Static Switches in EasyEDA are not ideal: they all have a finite OFF resistance. For the voltage and current controlled switches, the OFF resistances are specified by the user editable R(off)(Ω) parameter in the right hand Properties panel with a default value of 1GΩ. The Static Switches have a fixed 10GΩ OFF resistance. In either case these resistances are large but certainly not infinite.
  However, the effective resistance to ground at the Output node using a Volprobe or plot expression, really is infinite. Therefore when operating into an infinite resistance load there would be no difference between the ON and OFF state voltages at the Output node.
  Note that if a load resistance of 1GΩ were to be connected from the Output node to ground, then there would be a clear ON/OFF state difference.
  This is illustrated in the following simulation:
  EasyEDA switches are not ideal
  Some further examples of the effects of finite switch resistances are illustrated in this next simulation:
  Effects of finite switch resistances
  Note that, in ngspice, the Ron and Roff values used in switches can be set such that Ron > Roff. This is a useful way to invert the logic sense of a switch.
  At this point it is worth noting that if it is important that the simulation results are as close as possible to those expected to be observed when probing a real circuit using real test equipment, it is sometimes useful to place a realistic load on wires (nets) in the simulation schematic where voltage measurements are to be made in the real circuit. Similarly for current measurements, realistic ammeter insertion impedances should be connected in series with the wire. To avoid unnecessary loading of the simulated circuit however, only place such loads in the locations where external measurement devices are to be connected to the real circuit and only in the same numbers as there are measurement instruments being used at the same time. For example, if there are two oscilloscope probes connected to a circuit but one of them is moved around, only connect loads representing simulated oscilloscope probes to two places in the simulated circuit. If different places need to be probed then move the simulated probes and rerun the simulation.

DC paths through Voltage and Current Sources

This section is about the DC paths through Voltage and Current Sources.

- Voltage sources in ngspice (including independent V and dependent B and E voltage sources) have zero source resistance and have no current limit. The output voltage will be constant for any load current. This is true when sourcing or sinking current. sinking. This is not the same behaviour as regulated and current limited bench power supplies set to give a constant voltage output. Except for specialised supplies such as Source Measure Units, bench supplies can usually only source currents for positive outputs or sink currents for negative output voltages, up to some current limit set by the user.
    - Voltage Sources in ngspice therefore have a DC path through them.
      As a consequence of their zero source resistance, in the same way that damagingly high currents will flow if real batteries are connected in parallel without some resistance between them, voltage sources in simulation schematics cannot be connected in parallel without some resistance in series between them. This is true even if they are set to exactly the same voltage. Attempting to do so will cause the simulation run to fail with an error.
      Similarly, in the same way that damagingly high currents will flow if an inductor is connected directly across a real power supply without some resistance between them, inductors in simulation schematics cannot be connected directly in parallel with voltage sources without some resistance in series between them. Attempting to do so will cause the simulation run to fail with an error.
- Current sources in ngspice (including independent I and dependent B and F sources) have an infinite source resistance and have no voltage limit. The output current will be constant for any load impedance and voltage across the current source. This is not the same behaviour as regulated and current limited bench power supplies when they are set to give a constant current output. Except for specialised supplies such as Source Measure Units, bench supplies can usually only source currents up to some maximum positive voltage

at the output or sink currents down to some minimum negative voltage at the output, as set by the user.

- Current Sources in ngspice therefore do not have a DC path through them.

As a consequence of their generating a constant current through an infinite source resistance, in the same way that damagingly high voltages can be generated if two real current sources are connected in series without some finite resistance across each source, current sources in simulation cannot be connected in series without some finite resistance connected in parallel with each source. Attempting to do so will cause the simulation run to fail with an error.

- In simulation, connecting a capacitor in parallel with a current source generating a current, I, without also connecting a resistor, R, in parallel, will cause the voltage on the capacitor to ramp towards infinity. Even with the resistor in parallel, the voltage will ramp to I*R which may still be a large voltage. If such a circuit exists in a simulation schematic then, unless it is shorted out by a switch with some suitably low value resistance or set to some initial voltage, the voltage across the capacitor will already have ramped towards infinity at time t=0, i.e. as the simulation starts. This can lead to unexpectedly high voltages right from the start of a simulation.

The effects of adding DC paths

This section is about creating DC paths and some of the effects they can have.

- A favourite of elementary electrical engineering classes and an example of a very tricky problem to solve using a simulator is a circuit that has two capacitors in series. This must have a DC path to ground from the common point between the two caps but the path to ground does not have to be direct. It can be via other elements in a circuit. For example if one end of one of the capacitors is connected to ground with the other end of the other capacitor connected directly to a grounded voltage source or to a Thevenin Source (a voltage source in series with a resistance) or to a Norton Source (a current source in parallel with a resistance) then placing a single high value resistor in parallel with either of the capacitors or simply from the junction of the two capacitors to ground would be sufficient.

The following simple example will not run because there is no DC path to ground from the common node B, between the two capacitors:

Capacitors in series 01

- Resistors placed across capacitors to provide a DC path can be scaled so that the R*C time constant that they will create with the capacitors is very large compared to the time interval over which a Transient simulation is to be run. Another way to look at this for an AC Analysis is that the 1/(2*pi*RC) frequency that they form is far outside the frequency range of interest.

The value of these resistances can be anywhere between milliΩ (1e-3 Ω) up to 1G Ω (1e9 Ω). In many cases the values can be smaller or larger than this range but in some circuits that can lead to simulations failing with errors. This is usually because the ratio of the largest to the smallest voltages or currents in the circuit is too high for the simulator to handle. As a general rule, keeping the ratio between the smallest and the largest component value for any given type of passive component to no more than 1e12 should help avoid this sort of simulation failure.

- It must be noted however that whilst a purely theoretical analysis of the voltages across capacitors in series due to a current flowing through them based on Q=I*t=CV*, may yield sensible values, setting up both real and simulated circuits to demonstrate this can be quite challenging. Voltage measurements in real circuits can be difficult whilst the need for a DC path in simulations can present different but no less tricky problems.

- Similarly, a transformer coupled circuit where the primary side is driven from mains live and neutral and one side of the secondary circuit is connected to earth (which is probably where the ground symbol would be placed in the circuit diagram), must have a DC path back to this ground from one or both ends of the transformer primary (or from a centre tap if one is available).

Similarly to placing resistors across capacitors, DC path resistors used with inductors can be scaled so that the R/L time constant that they will create is very large compared to the time interval over which a Transient simulation is to be run. Another way to look at this for an AC Analysis is that the 1/(2*pi*R*C) frequency that they form is far outside the frequency range of interest. capacitors

Another example of where very high value resistors would be added is in transformer coupled datacommunications networks such a 100BaseTx or 1000BaseT Ethernet. Here, the connections between the two transceivers are floating and so have to have DC paths to ground added to keep the simulator happy. Using resistances that are high in comparison to the network cable characteristic impedance and hence termination resistance is important in order not to introduce an impedance mismatch and so disrupt the signal integrity. In practice it is simplest therefore to add two resistors: one from each of the two signal wires to ground. This preserves the symmetry of the signalling on the wires and doubles the effective resistance between them.

In many transformer coupled circuits, the simplest solution is to ground one side of both the primary and the secondary sides of the circuit. This may look strange because the circuits are no longer galvanically isolated by the transformer but - as long as the reason for connecting the circuits this way is clearly indicated in the schematic (and removed or otherwise modified before passing into PCB layout!) - then it is a simple and very useful solution. Not only does it remove any problems with how large or small the DC path resistor has to be, it also refers the voltages on both the primary and the secondary sides to ground and so simplifies probing of many of the voltages on both sides of the transformer which might otherwise have to be probed differentially.

Some examples of this use of the ground return path resistor can be seen in the following collection of examples of transformers and coupled inductors, including a simpleexample of an open loop flyback converter:

Transformers and coupled inductors

Transformers and coupled inductors 1

Transformers and coupled inductors 2

Transformers and coupled inductors 3

Open loop flyback converter

- To avoid confusion between passive schematics intended to be passed to PCB layout and simulation schematics that may have had DC path resistors added, it is a good idea to clearly identify any resistors added to a schematic soley to provide a DC path to ground. This can be done using resistor names such as *RDCPATH1* or by labelling them for example as *For simulation only*.

- It must be remembered that all voltages probed in a schematic are with respect to ground. This is particularly important to remember when probing signals that are floating, such as the transformer coupled examples discussed above. This is when B or E Sources can be

used to probe two floating voltages and subtract them to simply generate the difference between them.

Common problems with DC paths

A DC path to ground is often provided by the rest of the circuit but here are some cases that are often overlooked:

- All current sources (independent I and dependent B and F sources) are ideal: they have an infinite DC resistance (and AC impedance) and so do not have a DC path through them. Connecting a current source across a capacitor with one side grounded will throw an error even if the current source is set to zero (the capacitor voltage would ramp to infinity if a non-zero current were set and that would throw a different error!). So a resistor must be connected across the current source, to provide the DC path to ground to the other side of the current source / capacitor. A similar problem arises if two current sources are connected in series even if the two currents are identical (if they're not then the common node flies off to infinity again);
- Switches have their own internal DC path between the switch terminals because they have finite OFF resistances. However, **voltage controlled switches** do not have a DC path between their control inputs.
- **Current controlled switches**, **Current Controlled Current Sources (CCCS)**and **Current Controlled Voltage Sources (CCVS)**all have a zero resistance short circuit between their control inputs.Atfirst sight it might appear that connecting the output of a current source to the input of a current controlled switch would not cause a problem because the switch control input places a short circuit across the current source output. There is, however, no DC path to ground from the these connections so a resistor must be placed from one of the current controlled switch input pins to ground. If the current source output/switch control input part of the circuit is not connected to anything else then the resistor can be replaced by a connection to ground.

This is illustrated by the placement of the RDC*PATH*TO_GROUND 1kresistor connected to output side of the CCCS, F1, and input side of thecurrent controlled switch W1 in the example below:

Controlling EasyEDA switches

- Capacitors (unless using the more advanced options in ngspice) are ideal: they have no parasitic (leakage) DC resistance in parallel with them. This is why both ends of a capacitor must have a DC path to ground either through the external circuit or explicitly by adding a resistor;
- Although the primary and the secondary or secondaries of transformers have DC paths, there is of course no DC path from primary to secondary. This must be added either by connecting one side of the primary and of the secondary to ground directly or through a resistor.
- Voltage sources (including independent V and dependent B and E voltage sources) have the opposite problem. They are ideal so they have zero resistance. The same is true for simple inductors. If you connect voltage sources directly in parallel then ngspice will throw an error even if the voltage sources are set to the same value (if they're not then an infinite current would flow and that would then throw a different error). The same problem arises if you connect a voltage source directly in parallel with a simple, ideal, inductor because the voltage source tries to drive an infinite current through the inductor.

This problem often occurs when driving a transformer from a voltage source. Adding a small series resistance fixes this little gotcha (in practice there will always be a finite winding resistance anyway!).

Components are connected by netnames

It is important to understand that although components in a schematic can be shown joined by wires or by netlabels, in a spice netlist they are joined by purely by the net names given to them either automatically by EasyEDA or manually by the user placing netlabels. This includes nets joined by any of the three NetFlag GND ground symbols. It also includes nets joined by the Net Port, NetFlag VCC and NetFlag +5 symbols as illustrated below:

Nets can be joined by netnames 01

It is equally important to understand that when a wire between two components in a schematic is broken then - unless the wire on both sides of the break is explicitly given the same netname by manually placing netlabels with the same name - then *the wires on each side of the break will have different netnames* because even though one side may already have a manually assigned netname, the other side will automatically have a new and arbitrary netname assigned to it by EasyEDA.

The significance of manually assigning net names will become apparent a little later when looking at how to probe voltages and using the voltage on nets in expressions for B sources.

Nets can be joined by netnames 02

As just illustrated, breaking a wire creates two segments of wire that are no longer connected to each other. This is because EasyEDA automatically assigns different netlabels to each end of a broken wire.

To rejoin the connection, add netlabels with the same name to each segment of the broken wire. NetPorts and NetFlags can be placed anywhere on a net but take care that the little grey connection dot is placed onto the wire.

To avoid accidentally connecting nets together that are not intended to be connected when manually assigning netlabels, take care to ensure the netnames assigned are unique to each net. For instance, in the following example, using the name 'mid' instead of 'ammeterneg' would connect the negative side of A1 to the 'mid' net and would short out R1.

Nets can be joined by netnames 03

Probing signals

After spending ages building a schematic the time has finally arrived: the first simulation is run. The **Simulation Results...** window pop opens and there it is ...

Nothing.

No warnings.

No error messages.

No dire messages about *Timestep too short* or *Trouble with node X*

What's gone wrong?

The most likely cause is one of the most overlooked beginners' mistakes: there are no signal probes in the schematic. The simulation has run fine. It just didn't produce any results simply because it had not been asked to!

In order to view any useful output from a simulation, the circuit must have at least one voltage probe or one ammeter in the circuit. In basic simulations these will be the VolProbe and Ammeter symbols from the EasyEDA Libs. In more advanced simulations, these can be implemented using a **probe** command. In either case, at least one type of probe must exist in the simulation schematic. If no probe is present, the simulation will run but the **Simulation Results...** window will be empty and no WaveForm window will open.

Probing voltages

All voltage measurements in real circuits are actually measurements of voltage differences. In many cases such as when probing a voltage using an oscilloscope probe, it is easy to forget that the voltage being measured is, in reality, the difference between the voltage at the probe tip and wherever the probe ground lead is connected. In the same way it is easy to forget that probing a single ended voltage in a simulation schematic is with respect to wherever the ground node has been placed.

In a real circuit, probing a voltage between any two points places a resistive load between them. With a good quality voltmeter that resistance may be very high, in the order of hundreds of MegΩ. With an x10 oscilloscope probe it will be 10MegΩ. There will be some stray capacitance across that resistance. There will also be stray lead inductances. If the voltage being measured is an AC signal then impedances due to these stray and parasitic components will also load the circuit.

Note that in simulations, voltage probes present an infinite resistance and have no stray capacitance or inductance. In effect, voltage probes have an infinite bandwidth.

The following example illustrates some of the probing techniques decribed above:

Probing voltages 01

The following example shows a number of ways to measure voltages with respect to ground or differentially using;

- The EasyEDA Voltmeter;
- An `E` source (a.k.a. Voltage Controlled Voltage Source or VCVS) with a Voltage probe to probe the output;
- A `B`, source (a.k.a. behavioural or dependent source) configured as a VCVS using a Voltage probe to probe the output.

The schematic also demonstrates the importance of:

1. Giving voltage probes names that are identical to the nets to which they are attached;
2. Naming all nets in a schematic;

Probing voltages 02

Probing currents

In a real circuit, probing the current in a wire places a resistive load between them. This will cause some voltage drop across the ammeter. With a good quality ammeter that voltage drop may be very low, in the order of milivolts. There will be some stray capacitance across the insertion resistance and from the ammeter connections to ground. There will also be stray lead inductances. If the current being measured is an AC signal then impedances due to these stray and parasitic components will also load the circuit.

Note that in simulations, current probes present zero insertion resistance and have no stray capacitance or inductance. In effect, current probes have an infinite bandwidth.

The following example shows a number of ways to measure currents with respect to ground or differentially using;

- As a current using the Ammeter symbol;
- As a linearly scaled voltage using an H Current Controlled Voltage Source (CCVS) (or an F Current Controlled Current Source (CCCS) with a resistor);
- As a linearly scaled current using an F Current Controlled Current Source (CCCS) driving an Ammeter;
- As a voltage that can be an arbitrary function of the current flowing through a 0V Voltage Source using a BV source (or a BI source with a resistor);
- As a current that can be an arbitrary function of the current flowing through a 0V Voltage Source using a BI source driving an Ammeter.

Note however, that although a 0V source can be used to *monitor* a current, it cannot be used to *measure* a current so that it can be directly displayed in the Simulation Results... window or plotted in Waveform.

Probing currents 01

Configuring Voltage and Current Sources

EasyEDA Libs provides a range of what are referred to in ngspice as **Dependent Sources**.

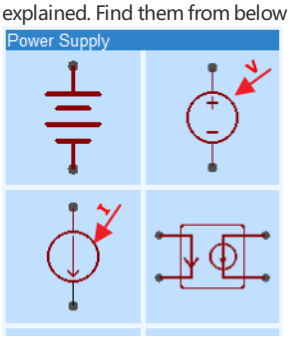
These are Voltage and Current Sources whose outputs are defined by a list of values or parameters. The outputs do not depend on anything else.

These sources have already been discussed in terms of their DC source resistances and DC paths.

In most of the examples so far, they have been used to provide DC supply voltages either as ideal voltage sources or as Thevenin or Norton Sources.

Their use to provide time domain (time varying) signals has been introduced in the examples about transformers but has not so far been

explained. Find them from below



This section describes in detail how any dependent source can be set up to provide the following types of signal sources:

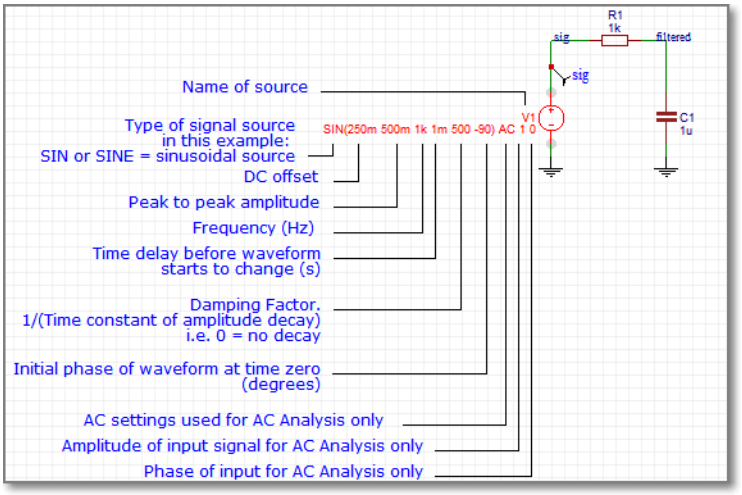
- SINE or SIN: a sinusoidal signal;
- PULSE: a general pulse waveform;
- EXP: a single pulse with exponential rising and falling edges;
- SFFM: (Single Frequency Frequency Modulated) a single sinusoidal carrier, frequency modulated by single sinusoidal frequency;
- AM: (Amplitude Modulated) a single sinusoidal carrier, amplitude modulated by a single frequency;
- PWL: (PieceWise Linear sources} an arbitrary waveform source with signals created as a list of times and levels with the signal linearly interpolated between each time point.

Although the examples in this section only illustrate how to configure Dependent Voltage Sources, Dependent Current Sources are configured in exactly the same way.

Configuring the SIN or SINE option

Configuring the SINE option to create an unmodulated, single frequency sinusoidal signal source.

Spice Sinusoidal Source example



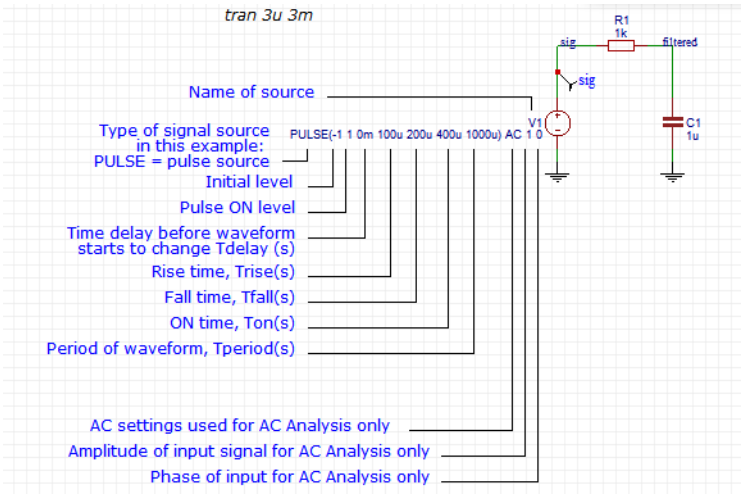
More ways to use the SIN (or SINE) option

Spice Sinusoidal Source: more examples

Configuring the PULSE option

Configuring the PULSE option to create a pulse signal source.

Spice PULSE Source



More ways to use the PULSE option

Spice PULSE Source: more examples

Configuring the EXP option

Configuring the EXP option to create a single pulse source with exponential rising and falling edges.

Spice EXP Source

Configuring the SFFM option

Configuring the SFFM option to create a simple, single frequency, frequency modulated sinusoidal signal source.

Spice SFFM Source

Configuring the AM option

Configuring the AM option to create a simple, single frequency, amplitude modulated sinusoidal signal source.

Spice AM Source

Configuring the PWL option

Configuring the PWL option to create an arbitrary piecewise linear waveform signal source.

Spice PWL Source

Configuring the AC source option

It can be quite hard to visualise what the Amplitude and the Phase settings in the AC Source options of the Independent Voltage and Current Sources really mean when the signals in an AC Analysis cannot be viewed in the time domain in a Transient Analysis. To try to help visualise these settings and what they represent, the following couple of examples demonstrate the settings in ways that can be related back to their equivalents in the time domain.

The first example shows how more than one AC Source can be configured in a circuit to represent different signal sources at the same frequency but with different phases. The example also shows how the phase settings relate to the same signals in the time domain.

In this example both AC Sources are set to the same amplitude of 1. They could be set to different amplitudes: try it and compare the results with the same amplitude changes in the time domain part of the signal sources.

Configuring AC Sources 01

Note, however, that the AC Analysis assumes that the circuit is perfectly linear so even if an AC Source amplitude 100 were to be specified, the output would still look as if it came from a perfectly linear circuit. Compare that with what happens if the time domain parts of the sources are set to 100!

The DC offset of the inphase AC Source in this example is important because it biases Q1 into a range where both the emitter and collector swings are operating in the linear region.

This can clearly be seen by probing V(Q1E) and V(Q1C) in a Transient Analysis. If the DC Offset is increased, eventually the V(Q1C) as V(Q1E) rises will until they meet as Q1 saturates and V(Q1E) starts to pull V(Q1C) back up again. At the point where this happens the small signal gain of the collector output passes though zero and then becomes a noninverting gain of somewhat less than unity.

If the DC offset is reduced to near ground or even below it, Q1 is cut off so both the collector and the emitter output small signal gains fall effectively to zero. Again this can clearly be seen in a Transient Analysis.

What is not so obvious is that although these effects still occur in the AC Analysis, because they are represented in the frequency domain, they can sometimes be hard to interpret.

So, the information to take away from this is that if an AC Analysis seems to be showing a lower than expected gain then it is worth checking that the DC operating point of the circuit is not forcing some part of it into saturation or cutoff. One example of this is incorrectly biasing an opamp input so that the output has hit one or the other of the supply rails. Forgetting to connect up a power supply rail is another common mistake.

It must also be understood that although any number of AC Sources can be placed in a circuit, each with their own amplitude and phase, all the sources will operate at exactly the same frequency as this is determined by the AC Analysis settings and not by the sources themselves.

In a circuit with several Independent Sources in it, AC sources can simply be added to, removed from or moved around it simply by adding the AC amplitude and phase values to the required source. So in the example above, the response of the inphase and outofphase side of the all pass network can be observed simply by setting one AC Source or the other to have zero amplitude or just by deleting the AC parts of the Source configuration.

Another example of this might be that the frequency response of an amplifier from signal input to output can be plotted using an AC Source at the input source whilst the frequency response of the amplifer from power supply ripple to output can be plotted by swapping the AC Source settings to the voltage source being used for the power supply.

The second simulation also shows how more than one AC Source can be configured in a circuit to represent different signal sources at the same frequency but this time with different amplitudes and phases. This example also shows how the amplitude and phase settings relate to the same signals in the time domain.

Configuring AC Sources 02

Setting up Analyses

What are Analyses?

An analysis is simply the instruction to the simulator telling itwhat type of simulation to carry out on the spice netlist.

There are several different types of analysis that can be carriedout when a simulation is run.

EasyEDA supports a subset of the spice analyses that are availablein ngspice. The supported analyses supported by the Simulate...dialogue box (accessed via the green running man > Run the Documentor Run the Project options or directly via CTRL+R) are described below.

SPICE Analyses available from the Green Man / Simulate... button

Clicking on the Green Man (Simulate...) icon opens the Simulation options menu.

- Run the Document...

  This is to run a simulation on a single schematic sheet.

- Run the Project...

  This is to run a simulation on all the schematic sheets in a project.

Note that the component prefixes (reference designators) must be unique across all schematic sheets: i.e. there must be no duplicate reference designators.

Attempting to run a simulation on any sheet or collection of sheets containing duplicate reference designators will generate the 'Prefix conflict error'.

By doing:

Simulate... > Run the Document... / Runthe Project...

the following SPICE analyses can be run from a simple dialogue box.For more information about what they do, please scroll down to therelevant sections.

1) DC op pnt

2) DC Transfer

3) DC Sweep

4) AC Analysis

5) Transient

SPICE Analyses and Control Statement Syntax

The following SPICE analyses can also be entered directly into atext box in a schematic.

Several analysis statements can be entered in a single schematic butone and only one can be made active for any one simulation run. To makean analysis statement active, do:

Properties > Text Attributes > Texttype > spice

To make an analysis statement inactive, do:

Properties > Text Attributes > Texttype > comment

They can then be run simply by doing:

CTRL+R

Several analysis statements can be entered in a single schematic butone and only one can be made active for any one simulation run.

1) OP: Perform an Operating Point Analysis

General form:

op

Example:

op

Causes SPICE to perform an operating-point analysis to determine thethe quiescent state of the circuit with inductors shorted andcapacitors opened. The results of this analysis are used to calculatevalues for the the linearised, small-signal models of nonlinear devices.

2) TF: Perform a DC Transfer Function Analysis

The dc transfer function analysis portion of SPICE computes thefollowing small signal characteristics:

the ratio of output variable to inputvariable (gain or tranfer gain)

the resistance with respect to the inputsource

the resistance with respect to the outputterminals

The TF statement can be used to find the Thevenin small signalequivalent resistance. (The Thevenin voltage is given by the nodevoltage at the open circuit terminal, as a result of the OP statement).

General form:

```
tf OUTvar inSRC
```

Examples:

```
tf V(5, 3) VIN

tf I(VLOAD) VIN
```

The TF command defines the small-signal output and input for the DCsmall-signal analysis. OUTvar is the small-signal output variable andinSRC is the small-signal input source. If this line is included, SPICEcomputes the DC small-signal value of the transfer function(output/input), input resistance and the output resistance.

3) DC: Perform a DC-Sweep Analysis

During a DC-sweep analysis SPICE steps the value of a specifiedindependent voltage or current source over the user-specified range andperforms an operating point analysis at each value. This permits theevaluation of the DC transfer function, and also provides a mechanismfor plotting the characteristic curves of devices and models.

General form:

dc Source-Name Vstart Vstop Vincr [Source2 Vstart2 Vstop2 Vincr2]

Examples:

```
dc vin 0.25 5.0 0.25

dc vin 0 10 .5 vgs 0 5 1

dc vce 0 10 .25 ib 0 10u 1u

dc R1 0 1k 100

dc TEMP 0 100 1
```

The parameters define the dc transfer-curve source and sweep limits.Source-Name is the name of an independent voltage or current source, aresistor or the circuit temperature. Vstart, Vstop, and Vincr are thestarting, final, and incrementing values respectively. The firstexample causes the value of the voltage source vin to be swept from0.25 volts to 5.0 volts in increments of 0.25 volts. A second source(Source2) may optionally be specified with associated sweep parameters.In this case, the first source is swept over it's range for each valueof the second source.

It is worth highlighting that the DC Sweep Spice Analysis allows notjust voltage and current sources to be swept but also temperature andresistances because Source-Name can also refer to a resistor in thecircuit or to the keyword TEMP meaning temperature in degrees Celcius.

The following simulations illustrate sweeping:

- a voltage source;

Plot and compare diode forward currents vs. voltage

- a resistor value;

Sweep a resistor value

- the ambient temperature applied to every component in the simulation:

Sweep the ambient temperature

4) AC: Perform a Small-Signal AC (frequency domain) Analysis

The ac small-signal portion of SPICE computes the ac outputvariables as a function of frequency. The program first computes the dcoperating point of the circuit and determines linearized, small-signalmodels for all of the nonlinear devices in the circuit. The resultantlinear circuit is then analyzed over a user-specified range offrequencies. The desired output of an ac small-signal analysis isusually a transfer function (voltage gain, transimpedance, etc). If thecircuit has only one ac input, it is convenient to set that input tounity and zero phase, so that output variables have the same value asthe transfer function of the output variable with respect to the input.

General form:

```
ac ( DEC | OCT | LIN ) N Fstart Fstop
```

Examples:

```
ac dec 10 1 10K

ac dec 10 1k 100Meg

ac lin 100 1 100HZ
```

Use:

'**dec**' for decade variation, in which caseN is the number of points per decade;

'**oct**' for octave variation, in which caseN is the number of points per octave;

'**lin**' for linear variation, when N is thetotal number of points.

Fstart is the starting frequency, and Fstop is the final frequency.

5) TRAN: Perform a Transient (time domain) Analysis

The transient analysis portion of SPICE computes the transientoutput variables as a function of time over a user-specified timeinterval. The

initial conditions are automatically determined by a dcanalysis. All sources which are not time dependent (for example, powersupplies) are set to their dc value.

General form:

```
tran Tstep Tstop [ Tstart [ Tmax ] ] [uic ]
```

Examples:

```
tran 1ns 100ns 0ns 2ns

tran 1ns 1000ns 500ns 10ns

tran 10ns 1us 0us 20ns uic

Tstep is the suggested computing increment.

Tstop is the final time.

Tstart is the initial time.
```

If Tstart is omitted, it is assumed to be zero. The transientanalysis always begins at time zero.

In the interval<zero, tstart="">, the circuit is analyzed (toreach a steady state), but no outputs are stored. In the interval<tstart, tstop="">, the circuit is analyzed and outputs are stored. Tmax is themaximum step-size that SPICE uses; try Tmax=(Tstop-Tstart)/50.0 tostart with.</tstart,></zero,>

The optional keyword 'uic' (use initial conditions) indicates thatthe user does not want ngspice to solve for the quiescent operatingpoint before beginning the transient analysis. If this keyword isspecified, ngspice uses the values specified using IC=... on thevarious elements as the initial transient condition and proceeds withthe analysis. If the .ic control line has been specified, then thenode voltages on the .ic line are used to compute the initialconditions for the devices. IC=... will take precedence over the valuesgiven in the .ic control line. If neither IC=... nor the .ic controlline is given for a specific node, node voltage zero is assumed.

Please see the description below of the .ic control line for itsinterpretation when uic is not specified.

IC: Set Initial Conditions

General form:

```
.IC V(n1)=VAL &amp;lt;V(n2)=VAL&amp;gt;&amp;lt;...&amp;gt;
```

Example:

```
.IC V(11)=5 V(4)=-5 V(2)=2.2
```

The IC line is for setting initial transient conditions. It has twodifferent interpretations depending on whether the UIC parameter isspecified on the .TRAN control line. One should not confuse this linewith the .NODESET line. The .NODESET line is only to help DCconvergence, and does not affect final bias solution (except formulti-stable circuits). The two interpretationsof this line are as follows:

- When the uic parameter is specified on the .tran line, thenthe node voltages specified on the .iccontrol line are used to compute the capacitor, diode, BJT, JFET, andMOSFET initial conditions. This is equivalent to specifying the ic=...parameter on each device line, but is much more convenient. Theic=... parameter can still be specified and takes precedence over the.ic values. Since no dc bias (initial transient) solution is computedbefore the transient analysis, one should take care to specify all dcsource voltages on the .ic control line if they are to be used tocompute device initial conditions.
- When the uic parameter is not specified on the .tran controlline, the dc bias (initial transient) solution is computed before thetransient analysis. In this case, the node voltages specified on the.ic control line is forced to the desired initial values during thebias solution. During transient analysis, the constraint on these nodevoltages is removed. This is the preferred method since it allowsngspice to compute a consistent dc solution.
- Note that the 'uic' option must be used with caution.

Normally, a DC operating point analysis is performed before starting the transient analysis. Theresults of this DC operating point analysis provide the initial conditions for the circuit at time t=0.

The 'uic' spice directive suppresses this initialization.

The initial conditions of some circuit elements can be specified on an instance-per-instance basis. Forexample: transistors can be specified to be in an OFF initial state; switches can be specified to be inan ON or an OFF initial state; the .IC spice directive allows the voltages on nets at t=0 to bespecified.

If the 'uic' option is added to a tran spice directive then all specified initial conditions are used.

It is important to realise however that if the 'uic' directive is used without explicitly stating theinitial conditions then, because the DC operating point analysis is omitted, default values are assumed.This can cause problems in some simulations because the default values can lead to nonphysical initialconditions around a circuit. For example, consider an ideal voltage source connected in parallel to anideal capacitor. Unless it is specified otherwise, the default initial value of the voltage source istaken as zero. Therefore, the voltage across the capacitor is also zero at t=0. Then, in the first timestep, the voltage source is set to the operating output voltage so an infinite current is drawn from itto charge the capacitor up to this operating voltage. The simulator cannot find a short enough timestep to make this current finite, and a "time step too small convergence fail" message is issued.

Note that if the 'uic' option is not used then any .IC directives included in the simulation are used anyway.

Initial conditions and starting up circuits

Some background and basic start-up techniques

All time domain simulations start at time t=0.

Even though it is possible to start plotting the results of a simulation at some time after t=0, the simulation itself always starts at t=0.

The initial DC conditions of a circuit are completely defined by the initial levels or DC offsets of any sources present in the circuit. So for example a DC power rail that is set to 9V is treated by the simulator as having been at 9V for all time prior to t=0.

Similalrly a SINE source with a -1V DC offset or a PULSE source with an initial level of 1V are treated by the simulator as having been at -1V DC for all time prior to t=0.

Therefore the voltages in a circuit, such as bjt base bias potential dividers and so on, will have been set to DC steady state values for all t<0. Consequently, the voltages across all capacitors and the currents through all inductors in the circuit will have reached their DC steady state values prior to the simulation starting at t=0.

Even if the circuit is an oscillator, prior to t=0 it will have been assumed to be in a stable non-oscillating steady state. At t=0 the circuit will then start from those initial DC conditions. It will then either continue in that steady non-osillating state or will slowly drift away from the steady DC state and oscillations will build up.

The initial state of oscillators based on a tuned circuit such as phase shift, Wien Bridge and Crystal Oscillators will be defined by their DC bias conditions. If there are no noise sources in the circuit (the default state for all components unless otherwise specified such as resistors defined to have noise contributions) then there is nothing to nudge the circuit away from equilibrium and so it may never start oscillating.

Although in most cases such oscillators will eventually start up due to the 'hidden' noise source which is simply due to the mathematical noise generated by the finite resoluton and rounding errors of the calculations carried out in running a simulaton, this can take a very long time compared to the time taken to run the oscillator in a stable oscillatory state for a few cycles. Crystal oscillators in particular can take many hunderds of thousands of times the oscillator period to start up and reach a stable state.

To minimise the simulation time spent waiting for an oscillator to start, it is useful to introduce some initial start-up condition to 'kick-start' the circuit into oscillation.

The simplest way to kick-start a circuit into oscillation is to replace a simple DC supply source with a PULSE source set to an initial level of the desired power supply voltage but configured to generate a short pulse of the supply voltage plus or minus some small voltage. So for example a circuit with a 9V supply that is to run for 1ms with a time step of 1us could have a PULSE source set to an initial level of 9V pulsed down to 8.5V for 1us with 100ns rise and fall times. Or an initial level of 8.5V with a delay of 1us before a 100ns risetime step up to a pulse level of 9V.

The same trick can be carried out using a voltage source inserted into the circuit almost anywhere in the circuit simply to inject a small step or pulse into a bias voltage but it must be remembered that if the voltage step or pulse does not return to 0 after the kick then it will represent an offset voltage in that part of the circuit.

An example of this is in forcing crystal oscillators to start.

Crystal oscillators take a very long time to start up because of the extremely high Q of the crystal.

The same thing is true of their simulations so, to avoid simulations taking too long to run and generating massive data files, they may need to be run in stages with increasing start and stop times but a small (Tstop - Tstart) value.

As discussed earlier, injecting a small step into the supply voltage or an internal node of the circuit can help to start the oscillations a little earlier but the idea of injecting a small step or pulse into a circuit to kick it into life is taken to an extreme in the EasyEDA crystal model.

The technique used in the XTALfast subckt is to include a PULSE source internal to the crystal model to introduce a very high amplitude impulse at t=0 inside the model. This starts the oscillator almost instantly and because an impulse with a very wideband spectrum is used rather than a step (or a sinusoidal burst at the crystals natural resonant frequency), the oscillations start at the actual resonant frequency of the crystal in the application circuit.

The start up time of the example below using the XTALfast subckt can be compared with the same crystal model but with an unassisted start simply by editing the name of the crystal model from XTALfast to XTALnofast.

Crystal oscillator using the EasyEDA quick starting crystal model

The EXP and PWL sources can also be used as kick-starter supply sources.

Most relaxation oscillators such as the classic two transistor astable multivibrator or the 555 timer have two stable states. Oscillation is normally maintained by the circuit switching between these two states however, under DC bias prior to t=0, these circuits often settle into one or the other of these stable states and so are stuck there at t=0. This means that they never start oscillating.

This is an example of a simple RC relaxation oscillator that does not start up by itself:

Relaxation oscillator startup 01

This type of circuit may need a rather more vigorous kick to get it started. This can be done using a PULSE source but instead of introducing a small step, the supply is ramped up to the desired supply voltage from 0. So, for example, by setting an initial level of 0 and a pulse level of 9 with zero delay time, with a risetime of 200us the circuit starts up cleanly up from all internal nodes being at zero:

Relaxation oscillator startup 03

Another possibility - which will be explained in more detail later - is to use an expression that is a function of time in a B source.

For example, this expression in a B source:

```
V=9_(1-exp(-1_time/100u))
```

generates a voltage that starts at zero and rises exponentially to a final value of 9V with a time constant of 100us:

Relaxation oscillator startup 04

With symmetrical circuits such as the two transistor astable multivibrator, even this may not be enough to disturb the equilibrium enough to get them oscillating. It may be necessary to introduce some delibrate asymmetry or imbalance into the circuit, for example by making a one base pullup resistor or a timing capacitor a fraction different from the other. Even changes of less than the expected real component tolerance can be enough to tip the circuit into self sustained oscillation with a zero to rated voltage ramped supply start-up.

Sometimes complex circuits - or simple circuits with complex models - may fail to simulate because the simulator cannot find the DC operating point prior to t=0. Such circuits will very often simulate fine if the supply (or supplies) are ramped up from 0. In the same way as slightly imbalancing symmetrical components can help start-up, slightly imbalancing the voltages, delay or rise times of the supply voltage ramps may help start-up more 'difficult' simulations.

There is, however, a down side of using the start-up from zero supply ramp. As already stated, at the start of a transient simulation, the voltages across all capacitors and the currents through all inductors in the circuit will have reached their DC steady state values prior to the simulation starting at t=0. If the simulation starts with the supply voltages set to zero at t=0 then obviously all the internal voltages and currents must also be zero (except for some small offsets due to the initial levels of any signal sources and sometimes, badly designed sources internal to models that do not collapse to zero with zero supply voltages).

If all the internal voltages and currents are zero at t=0 then it may take much longer than the desired simulation stop time for all the internal nodes to reach their DC steady states.

A simple solution to this is to run the simulation for long enough for everything to have settled but to only plot the results for just long enough before the stop time to show the signals of interest. So, for example a circuit that is driven by a 1kHz source but which takes 95ms to settle could be observed from say, t=98ms to t=100ms by setting the transient anaylsis to have a maximum timestep of 1us, a stop time of 100ms and a Start time of 98ms, like this:

```
tran 1u 100m 98m
```

This solution works very well but, in this example, over 95% of the simulation time is used just to get the circuit to a steady state before any useful results are generated. This is very wasteful of simulation time and for complex simulations such as switch mode power supply (SMPS) simulations - where exactly this situation is likely to arise - can take many minutes of real time.

This is where another technique for setting initial conditions in a circuit can sometimes be useful.

Setting initial voltages on nets and currents through components

There are occasions where it is required to start a simulation in some predetermined state. For example, a capacitor may be required to start a transient simulation at time t=0, precharged to some given voltage. Similarly the current in an inductor may need to be specified at time t=0. In a larger simulation it may be helpful to precharge the output smoothing capacitor of a power supply to approximately the right voltage to save the time taken for it to be charged up from zero. If the capacitor is at the output of an SMPS then it may be useful to charge the inductor(s) in the SMPS to their average operating current(s) too.

Using the ic spice directive

Using the `.ic` spice directive to set an initial voltage condition on a net

For more information about the .ic spice directive see:

About-spice-analyses-in-EasyEDA

Use of the '.ic' spice directive is illustrated in these two examples:

Setting initial circuit conditions 01

Relaxation oscillator startup 02

These next examples show two ways to use append the 'uic' option to a Transient Analysis but for the reasons already given in the description of the 'uic' option in the section on 'IC: Set Initial Conditions' in 'Setting up Analyses', care should be taken in using this option.

Relaxation oscillator startup 05

Relaxation oscillator startup 06

Using a current source to set an initial current through an inductor

The `.ic` spice directive can only be used to set initial voltage conditions on one or more nets. It cannot be used to set an initial current through a component. This example illustrates a simple way to use a current source to set an initial current through a component.

Setting initial circuit conditions 02

Setting a capacitor voltage using an XSPICE capacitor model

An alternative to using the '.ic': setting the initial voltage across a capacitor using an XSPICE capacitor model in place of the default EasyEDA capacitor.

Setting initial circuit conditions 03

Setting an inductor current using an XSPICE inductor model

An alternative to using a current source in parallel with an inductor: setting the initial current through an inductor using an XSPICE inductor model in place of the default EasyEDA inductor.

Setting initial circuit conditions 04

Some circuits may start up on their own but simply changing a component model may cause it to fail to start-up. Don't be afraid to try these techniques before spending ages trying to find some other obscure cause.

Using a 1V source to help start-up

This is an advanced technique which has limited application but which in the right circumstances can be very useful.

By placing a 1V voltage source in a schematic, naming it's output as, say, 'unity' and then multiplying using expressions in B Sources by V(unity), it is possible to force all the B sources in the simulation to start from zero during the early DC operating point parts of a transient simulation.

This can sometimes help produce a clean startup from zero internal initial states.

It must be borne in mind however, that as with other techniques that bring a simulation up from zero, any attemtp to run an OP, TF or an AC simulation will usually return zero results because all the internal states and in many cases the gains of B sources will have been forced to a zero initial value.

Replacing ideal and Thevenin voltage sources with band-limited Norton Sources to help start-up

This is technique is very widely applicable since it helps improve the overall convergence of simulations and not just their startup behaviour.

Ideal voltage sources in simulations are capable of producing infinite currents so a load which looks capacitive at any point during the simulation has the potential to cause an instantaneous infinite current flow. This can cause a simulation to go into overflow or to fail to find a valid next step from which to proceed. In either case, the simulation will fail to converge. To prevent this it is good practice to always include some resistance in series with every voltage source, whether that is an independent V, dependent E or H source or a dependent B source configured as a voltage source.

This turns all ideal voltage sources into Thevenin Sources, i.e. voltage sources with a finite source resistance.

Internally, however, the simulator is actually more efficient at calculating the currents and voltages around a Norton Source than a Thevenin source. So, the next step of the process is to convert all the Thevenin sources in a simulation into Norton Sources.

A Norton Source is the exact equivalent circuit of a Thevenin Source. To convert a Thevenin source to the Norton Equivalent source, the voltage source of the Thevenin source is replaced with a current source equal to the short circuit current of the Thevenin source and the series resistance of the Thevenin source is reconnected in parallel with this current source. Both sources than have identical source resistances, open circuit voltages and short circuit currents.

So far these steps of converting ideal voltage sources to Thevein sources and then to Norton equivalent sources generally increases the simulation speed of most simulations but the final step of limiting the bandwidth of the Norton sources can significantly improve the start-up and convergence of many simulations.

Limiting the bandwidth (or band-limiting) a Norton Source is easy: all that is required is to connect a capacitor in parallel with the source resistance that has already been connected in parallel with the current source. This creates a lowpass filter across the output of the Norton source which reduces the output at high frequencies. This means that even if the source is passing or generating highly non-linear signals that can produce exactly the kind of very fast and even discontinuous signals that spice struggles to deal with in its internal calculations, as the frequency content of these signals and transitions exceeds the lowpass filter cutoff frequency, they are reduced to relatively slower edged (i.e. reduced high-frequency content) signals. Such signals then have finite rise and fall times and so spice no longer struggles to deal with them. They become much more spice friendly continuous signals.

Even better is that if there are several such band-limited sources sprinkled around the circuit (such as the input signal sources) then very often the derivatives of these signals also become continuous. This is also a big help to spice because much of the internal calculation in spice is in estimating not just where a signal is but also making estimates about where it is going and how fast it is getting there.

Some words of warning are needed here though. Replacing voltage and Thevein sources with band-limited Norton sources is a great technique but it requires a high degree of understanding of what is really going on in a simulation circuit.

The statement above about:

"...all that is required is to connect a capacitor in parallel with the source resistance that has already been connected in parallel with the current source. This creates a lowpass filter across the output of the Norton source which reduces the output at high frequencies."

is all very well but it skirts around the question of what should the value of this capacitance actually be? There is no signal answer that question because the cutoff frequency required to avoid convergence or start-up problems will vary from one circuit to another and even from one location to another in a circuit. The single biggest factor affecting the value of simply that the cutoff frequency must be much higher than the working frequencies of not the circuit itslef but of the devices in it. So for example, an audio amplifier circuit designed for a bandwidth of 20kHz may use an opamp that is operated with a closed loop bandwidth that is 200kHz but which internally may have a gain bandwidth of 2MHz or even 20MHz.

In practice, setting the cutoff frequency with an RC time constant of 1ps (1e-12s) i.e. approximately 160GHz should be safe for most 'ordinary' simulations of amplifiers, linear and switch mode power supply circuits but for high frequency and RF circuits the band-limiting must be increased accordingly.

The value of the resistor in this parallel RC circuit can be also be very tricky to decide. For many lightly loaded voltage sources, a value of 1 Ω is OK and simplifies the calculation of the parallel capacitance to 1pF. In some sources, however, it may be necessary to use a higher (or lower value) of parallel resistance. This in turn will require the use of a proportionately smaller (or larger) capacitance for the same cutoff frequency.

This band-limiting technique should generally not be applied around any existing current sources in a simulation circuit. Placing an RC circuit in parallel with a pure current source may do more damage to the operation of the circuit by reducing the in-band source impedance of the current source than the band-limiting avoids!

Although not an example of start-up or initialisation problem, the Ideal and Thevein to Norton Source conversion is demonstrated in steps (i), (ii) and (iii) of this simulation:

Parameters, expressions, functions and B Sources

Using the 'OFF' option to help start-up

Some components, such as switches, bjts, jfets, MOSFETs and MESFETs have an 'OFF' option to specify the device to be in an intial OFF state.

Switches also have an 'ON' option to specify the device to be in an intial ON state.

This option can be very useful to ensure that for example, one side of a two transistor bistable or monostable circuit or an astable mutlivibrator is off, so avoiding the situation described above where both transistors are on in the intitial state. Whilst this does not in itself guarantee that the circuit will start up from t=0 but it may simplify any other measures that have to be taken to ensure it does.

These states are simply invoked by appending the keyword OFF (or, for switches only, ON) after the device name. This can be done either by directly editing the name in place in the schematic or via the right hand panel Properties dialogue. For example, to set a switch with the name MYSWITCH to be initially ON the name should be edited to:

MYSWITCH ON

Similarly, to set a bjt with the name 2N2222 to be initially OFF the name should be edited to:

2N2222 OFF

Expressions

Expressions can be used to define component values and to help configure Voltage and Current Sources.

Operators

In expressions, parentheses are evaluated before the other operators. The operators are evaluated following a list of precedence as shwn in the table below. For equal precedence binary ops, evaluation goes left to right. Functions operate on real values only!

Operator	Alias	Precedence	Description
-		1	unary negate (see Note 1 below)
!		1	unary not
**	^	2	power (but see also the pow(x,a), pwr(x,a) and pwrs(x,a) functions)
*		3	multiply
/		3	divide
%		3	modulo (does not work in B Source expressions)
\		3	integer divide (does not work in B Source expressions)
+		4	add
-		4	subtract
==		5	equality
!=	<>	5	non-equal
<=		5	less or equal
>=		5	greater or equal
<		5	less than
>		5	greater than
&&		6	boolean and
||		7	boolean or
c?x:y		8	ternary operator (See also the if(x,y,z)

			and ifx(x,y,z) functions)

Note 1

At the time of writing (141021), the ngspice implementation of the unary negate or subtract symbol works as expected when used in this type of expression:

```
.param myparameter={A-B}
```

but may produce unexpected results when used in an expression for a B Source (or in a .func statement) like this:

```
V=5*(1-exp(-time/1m))
```

This expression must be written instead as:

```
V=5_(1-exp(-1_time/1m))
```

This is a feature of ngspice. It is hoped that this anomaly will be corrected at some point in the future and that this will feed into later revisions of EasyEDA.

The number zero is used to represent boolean False. Any other number represents boolean True. The result of logical operators is 1 or 0.

Some examples of logical operators used to defined value of voltage sources:

```
V1or 1 0 {1 | | 0}

V2and 2 0 {1 &amp;&amp; 0}

V3not 3 0 { ! 1}

V4mod 4 0 {5 % 3}

V5div 5 0 {5 \ 3}

V6not 6 0 { ! 0}
```

Note that when used directly in component and source value fields, expressions MUST be on a single line. When used like this, expressions cannot be wrapped over more than one line.

Using Expressions to define component values

The -3dB frequency, fc, of a first order RC lowpass filter is given by:

```
fc = 1/(2_pi_R*C)
```

Exactly the same expression applies to a first order RC highpass filter.

If fc is specified as 10kHz and R is chosen as 1k then:

```
C = 1/(2_pi_1k*10k)
```

Suppose the high pass filter output is required to be attenuated by a factor, A.

The total value of R for the highpass filter is still 1k but it must be split into a lower resistor with a value given by:

```
Rlower = (Rupper+Rlower)*1/A
```

whilst the upper value is given by:

```
Rupper = (Rupper+Rlower)*(1-1/A)
```

If we choose A=3 then for the chosen value of R=1k

```
Rlower = 1k*1/3
```

and:

```
Rupper = 1k*2/3
```

Simply by entering the right hand side of these expressions into the component values fields, enclosed in curly brackets like this:

```
{expression}
```

the value of those components will be defined directly by those expressions, as illustrated in Rupper and Rlower in this example.

Using Expressions to configure voltage and current sources

In this example, PULSE source V1 is configured to generate a signal with a 20us rise and fall time, a frequency of 5kHz and exactly equal high and low times: in other words, a slow edged squarewave of 200us period and a 50% duty cycle.

Because a PULSE source is defined in terms of Trise and Ton, it can be helpful to think of the time interval from the start of the rising (leading)

edge to the start of the falling (trailing) edge as the 'pulse width', Twidth:

```
Twidth = Trise+Ton
```

It is then a simple matter to define the PULSE source in terms of Trise and Twidth without having to manually calculate a value for Ton because:

```
Ton = Twidth-Trise
```

From this we can also see that if the 'duty cycle' is defined as:

```
D = Twidth/Tperiod
```

then for a given D:

```
Ton = D*Tperiod - Trise
```

Lastly, it is sometimes convenient to define the period of a PULSE source in terms of a frequency:

```
Frequency = 1/Tperiod
```

To use an expression in a source, simply enter it in place of the value you wish to calculate and enclose it in curly brackets like this:

```
{expression}
```

The use of expressions is illustrated in the following example:

Using expressions 01

Parameters

Usually, the values of components are specified directly in the component's value field. There are, however, occasions where it is desirable to be able to set or change the value of several components at once without having to edit each individual component value.

The simple resistor attenuator circuit used to illustrate several of the early examples has one resistor of 1k and two of 2k. Instead of entering the value 1k into one resistor resistor 2k into each of the others, it is possible to set up two variables to represent these values.

To create two variables, R1val=1k and R2val=2k, a `.param` statement is placed into the schematic and turned into a spice directive (by doing: **Properties > Text type = spice)**:

.param R1val=1k R2val=2k

The parameters are then used to define the values of the components in their value fields.

By placing more than one .param statement in a schematic and doing:

Properties > Text type = comment

and

Properties > Text type = spice

It is possible to switch sets of values without having to edit individual components evey time.

Using .param statements therefore makes it possible to:

- change the value of several components a single edit;
- define parameters in terms of other parameters;
- define parameters in terms of functions of other parameters.

Note that it is also possible to have several .param statements active in a simulation at the same time but to avoid the conflict caused by duplicate definitions, the parameter identifier names in each set must be unique.

The syntax of .param statements is:

```
.param
 <param_name1>=<value1> <param_name2>=<value2> ... <param_nameN>=<valueN>
```

'.param' statements can wrap over more than one line by using the '+' continuation character: .param + = + = + ... + + ... + + ... + =

Parameters can be numbers, other defined parameters or expressions made up from any combination of numbers and defined parameters.

Parameter identifier names must begin with an alphabetic character. The other characters must be either alphabetic, a number, or ! # $ % [ ] _ as special characters.

Note that when used in .param statements, expressions can wrap over more than one line by using the '+' continuation character:

```
.param
+ <param_name>=<value1>
+ <param_name2>=<value2>
+ <param_name3>=<{expression1}>
+ <param_name4>=<{part of expression2
+ continuation of expression2}>
+ ...
+ <param_nameN>=<valueN>
```

Care must be taken in choosing the line break points to clearly distinguish the use of the '+' character as a continuation character from any mathematical use of the '+' character as an addition operator in an expression.

For example:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2+
+ y^2)}</xmp>
```

and:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2 +
+y^2)}
```

are valid wrappings of expression in a .param statement which will give the expected results, whereas:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2
+ y^2)}
```

or:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2+
y^2)}
```

or:

```
.param x=3 y=4
+ hypotenuse={sqrt(x^2
+y^2)}
```

may give unexpected results or may fail with errors.

- **Note that the variables TIME, TEMPER or TEMP and HERTZ are NOT valid identifier names.**

Note that to use a parameter in a component value field, it must be enclosed in curly brackets:

{...}

A parameter should also be enclosed in curly brackets if it is being used to define the value of another parameter, as in the example here of:

```
R3val={R2val}
```

Although not shown here, the use of curly brackets to enclose expressions containing parameters which are then used to define other parameters is mandatory so, even though the use of curly brackets in the example shown above is not mandatory, it is good practice to always enclose in curly brackets any parameter or expression used in a parameter definition.

Note, however, that parameters in expressions used in B Sources must should not be enclosed in curly brackets.

The basic use of parameters is illustrated below:

Using parameters 01

Using parameters in expressions

Expressions and parameters can be combined to simplify and automate the calculation of component and source configuration values, as illustrated in the following example:

Using parameters in expressions 01

Functions

The sections on expressions and parameters shows how component and source values can be defined using arithmetic equations. The examples used to illustrate this used only simple linear expressions. This section introduces the concept of functions.

Functions hugely expand the power of parameters and expressions by allowing the creation of expressions including non-linear functions of other parameters

Predefined functions

EasyEDA has a number of pre-defined functions. Many of them are immediately available to be used in expressions because they are built-in to ngspice or they are automatically added in to the spice netlist by EasyEDA at the time of first saving a schematic.

Some are not yet automatically added in to the netlist so their definitions must be pasted in manually before they can be used. These function definitions are clearly indicated in the examples illustrating each function.

These functions will be appended to the list of those automatically added to the netlist soon.

All the currently available pre-defined functions are listed, together with illustrative examples, in the table below.

Note that all the functions in this list can be used in any context in EasyEDA: in the value fields and in expressions for component values, Independent Sources and for B Sources.

Table of functions

Function	Description	ngspice native or EasyEDA special	Where useable
abs(x)	Absolute value of x	ngspice	E, I, B
acos(x)	Arc cosine of x. Fails to converge for x < -1 and x > +1. Use invcos(x) instead.	ngspice	E, I, B
acosh(x)	Real part of the arc hyperbolic cosine of x, e.g., acosh(.5) returns 0, not 1.0472i	EasyEDA	E, I, B
arctan(x)	Alternate syntax to atan(x)	ngspice	E, I, B
asin(x)	Arc sine of x. Fails to converge for x < -1 and x > +1. Use invsin(x) instead.	ngspice	E, I, B
asinh(x)	Arc hyperbolic sine	EasyEDA	E, I, B
atan(x)	Arc tangent of x	ngspice	E, I, B
atan2(y,x)	4 quadrant arc tangent of x/y (tan^-1(x/y))	EasyEDA	E, I, B
atanh(x)	Arc hyperbolic tangent. (Limited output swing to avoid numerical under/overflow failure.)	EasyEDA	E, I, B
buf(x)	Returns 1 if x > 0.5, else 0	EasyEDA	E, I, B
ceil(x)	Integer equal or greater than x	ngspice	E, I, B
cos(x)	Cosine of x	ngspice	E, I, B
cosh(x)	Hyperbolic cosine of x	ngspice	E, I, B
exp(x)	e to the x	ngspice	E, I, B
floor(x)	Integer equal to or less than x	ngspice	E, I, B
if(x,y,z)	IF x > 0.5, THEN y ELSE z	EasyEDA	E, I, B
ifx(x,y,z)	IF x, THEN y ELSE z	EasyEDA	E, I, B
int(x)	Convert x to integer	EasyEDA	E, I, B
inv(x)	Returns 0 if x > 0.5, else 1	EasyEDA	E, I, B
invcos(x)	Real part of the arc cosine of x, e.g., acos(-5) returns 3.14159, not 3.14159+2.29243i	EasyEDA	E, I, B
invsin(x)	Real part of the arc sine of x, asin(-5) returns -	EasyEDA	E, I, B

	1.57080, not -1.57080+2.29243i		
invtan(x)	Alternate syntax to atan(x)	EasyEDA	E, I, B
limit(x, L, U)	Value of x, bounded by L and U	EasyEDA	E, I, B
ln(x)	Natural logarithm of x. Fails with errors for negative x. Use log(x) instead.	ngspice	E, I, B
log(x)	Natural logarithm of x. Generates a real valued output for all x, limited to a minimum of approximately -230.5 for x <= 1e-100.	EasyEDA	E, I, B
log(x)	Base 10 logarithm. Generates a real valued output for all x, limited to a minimum of -100 for x <= 1e-100.	EasyEDA	E, I, B
max(x,y)	The greater of x or y	ngspice	E, I, B
min(x,y)	The smaller of x or y	ngspice	E, I, B
pow(x,a)	Real part of x raised to the power of a. Zero for negative x and fractional a	EasyEDA	E, I, B
pwr(x,a)	The absolute value of x raised to the power of a	EasyEDA	E, I, B
pwrs(x,a)	pwr(x) multiplied by the sign of x	EasyEDA	E, I, B
sgn(x)	Sign of x. Returns -1 for x < 0, 0 for x == 0 (where == means 'exactly equal to') and 1 for x > 0	ngspice	E, I, B
sin(x)	Sine of x	ngspice	E, I, B
sinh(x)	Hyperbolic sine of x	ngspice	E, I, B
softlim(ip, lo, hi, sharp)	Value of ip, bounded by lo and hi with the sharpness of the transition between linear and limited regions defined by 'sharp'.	EasyEDA	E, I, B
sqr(x)	Square of x	EasyEDA	E, I, B
sqrt(x)	Real part of the square root of x. Zero for negative	EasyEDA	E, I, B

	x		
stp(x)	Alternate syntax for u(x)	EasyEDA	E, I, B
tan(x)	Tangent of x	ngspice	E, I, B
tanh(x)	Hyperbolic tangent of x	ngspice	E, I, B
u(x)	Unit step, i.e., 1 if x > 0., else 0	EasyEDA	E, I, B
u2(x)	Returns 1 for x >= to 1, x for x between 0 and 1, 0 for x <= 0.	EasyEDA	E, I, B
uramp(x)	x if x > 0, else 0	EasyEDA	E, I, B

User defined functions

There may be occasions where a function is required that maybe has to be used in several places in a schematic or it is useful in several different schematics. To save having to copy and paste a complicated expression as a block of text each time it is needed, the **.func** statement makes it is possible to create a user defined function.

The syntax of the .func statement is very simple:

```
.func myfunctionname(a,b,c, ...n) {expression of functions of a, b, c ... n}
```

For example:

```
.func hypotenuse(x,y) {sqrt(x^2+y^2)}
```

defines a function that calculates the length of the hypotenuse of a right angle triangle with sides length x and y.

Once a function has been defined in a schematic in a project in this way it can be used anywhere in any schematic in that project, i.e. it does not have to be defined separately in every sheet in which it is used in a project. It does, however, have to be defined in a sheet in every project in which it is to be used but, if it's a really useful function, let us know and maybe we'll add it to the growing list of pre-defined functions!

To use the function all that is then required is to paste **hypotenuse(x,y)** into wherever it is needed and to substitute the 'x' and 'y' with the required variables. So, for example to use the function in a parametric expression:

```
.param a=3 b=4 hypot=hypotenuse(a,b)
```

or in a current output B Source driven by two voltages, V(oneside) and V(otherside):

```
I=hypotenuse(V(oneside), V(otherside))
```

There are many examples of functions defined by the .func statement in the simulations linked to in the table above and in all EasyEDA spice netlists for the automatically appended predefined functions.

Note that when used in .func statements, expressions can wrap over more than one line by using the '+' continuation character.

There are several examples of this in the simulations linked to in the table above and in all EasyEDA spice netlists for the automatically appended predefined functions. For example:

```
.func POW(x,a)
+ {((a-(int(a)))==0)||(sgn(x)>=0))_( max(exp(ln(uramp(x))_a),0) +
+ (2_(0.5-ABS((int(a))-2_int(a/2))))_max(exp(ln(uramp(-1_x))*a),0) )}
```

can be found appended to every EasyEDA spice netlist.

B sources spice simulation

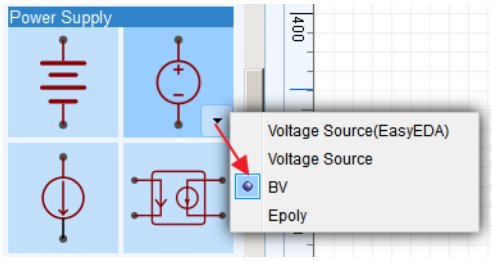
B sources are one of the most powerful components in EasyEDA. They are available as a BV voltage source and as BI current source (although in fact at the spice netlist level they are the same device just defined to have a voltage or a current output).

The function of every B source is defined by an equation.

The left hand side of the equation defines whether the output of the source is a voltage or a current.

The right hand side is an expression made up from numbers, the basic arithmetic operators and functions not only of parameters but, crucially, of dynamic voltages and currents from within the circuit being simulated. In other words, B sources can perform a range of functions in simulations that is limited only by the imagination of the simulation designer.

**How to add BV source in EasyEDA **



The syntax of the equations to define a BV source in EasyEDA is very simple:

V=expression

For example:

```
V=3*V(a,b)
```

defines a BV source that generates an output voltage equal to 3 times the difference between the voltage on the 'a' net (V(a)) and the voltage on the 'b' net (V(b)).

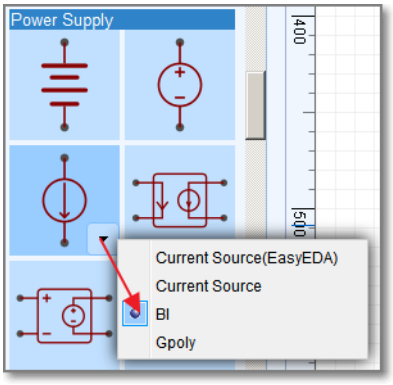
```
V=scale*uramp(V(a,b))/ABS(I(Vimon))
```

defines a BV source that generates an output voltage equal to the parameter, scale, multiplied by the positive difference between the voltage on the 'a' net (V(a)) and the voltage on the 'b' net (V(b)), divided by the absolute value of the current through the 0V source Vimon (I(Vimon)).

```
V=Vswing_tanh(V(a,b)_Avol)
```

defines a differential gain block with an small signal gain of Avol and an output voltage swing which is limited with a tanh function to +/- Vswing.

The syntax of the equations to define a BI source in EasyEDA is equally simple: **How to add BV source in EasyEDA **



I=expression

```
I=V(a)*I(Vimon)
```

defines a BI source that generates an output current equal to the voltage on the 'a' net (V(a)) multiplied by the current through the 0V source Vimon (I(Vimon)).

```
I=LIMIT(V(a), 3, minval^2)
```

defines a BI source that generates an output current equal to the voltage on the 'a' net (V(a)) but clamped to the value of 3 and the square of the value of the minval parameter for all values of V(a) outside the range defined by 3 and minval^2.

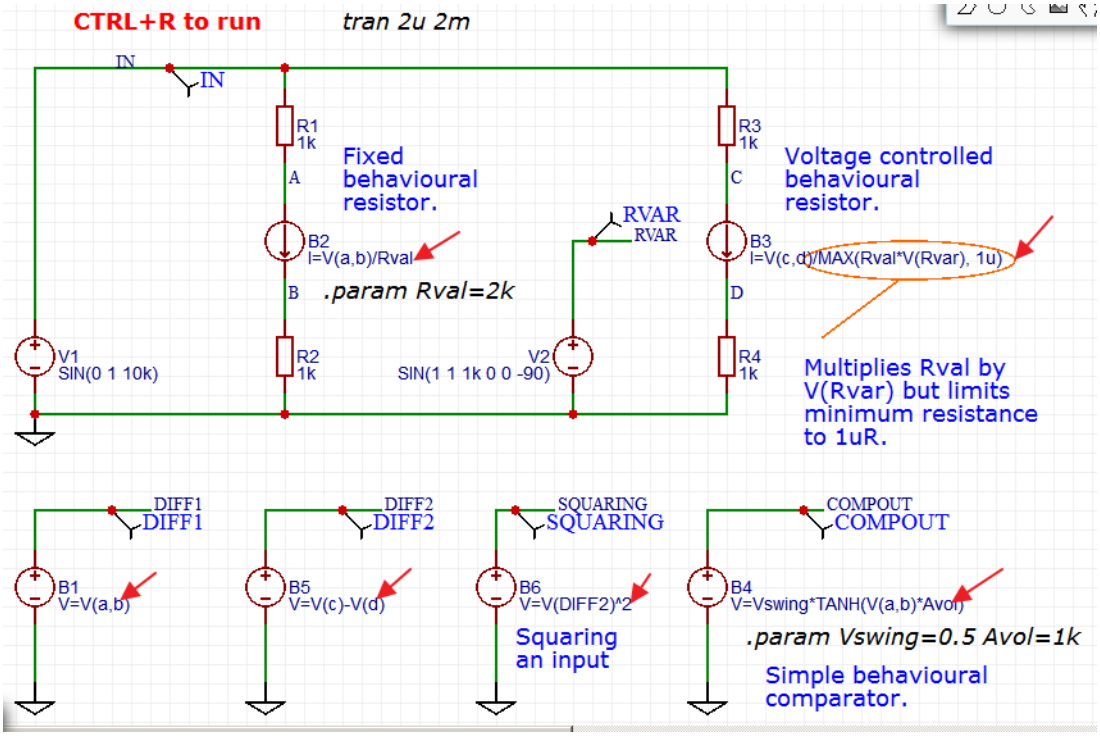
```
I=V(a,b)/Rval
```

when the '-' and '+' terminals of the B source are named 'a' and 'b' respectively then this expression defines a resistor of value Rval.

Note that curly brackets are not used in expressions for B Sources.

There are several examples of the uses of B Sources in the following simulations.

1. B Sources 01 example->



1. limit-x-L-U
2. Parameters, expressions, functions and B Sources

Note that when entered directly in a B Source value field in a schematic, expressions MUST be on a single line. When used like this, they cannot be wrapped over more than one line.

Expressions entered into a netlist, however, such as inside a .subckt model definition, can wrap over more than one line by using the '+' continuation character. Several examples of this can be found by inspecting the netlists of circuits using some of the EasyEDA .subckt models. For example in the netlist of the opamp5pEE Parameterised 5 pin opamp model there are these B sources:

```
Bipbias1 inp isum I=(ibias+ios)*V(supply_ok)
+ ( uramp(V(inn)-(V(vp)+inmax)) - uramp(-V(inn)+(V(vn)-inmin)) )/Rser
```

and

```
Bstg1 0 stage1 I=Islew_tanh(V(indiff)_Kg)
+ - ( uramp(V(stage1)-(V(vp)-outhi)) - uramp(-V(stage1)+(V(vn)+outlo)) )/Rser
+ - sel*( uramp(V(stage1)-(V(out)+oooclmphi)) - uramp(-V(stage1)+(V(out)-oooclmplo)) )/Rser
```

and in the opamp_ANF01 .subckt example found elsewhere in this document, there is another example:

```
B1 out 0
+ V=(TANH((V(inp)-V(inn))_{Avol}_2/(V(vcc)-V(vee)))*(V(vcc)-V(vee))
+ +(V(vcc)+V(vee)))/2
```

Device models

In order to simulate the behaviour of the individual components, they have to be described mathematically. The underlying equations that describe the behaviour of a component are written into the simulator program (sometimes they can be added by the user).

The equations that describe basic components such as resistors (I = V/R), capacitors (I = C*dV/dt) and inductors (V = L*di/dt) may be reasonably straightforward. The equations that describe diodes, bipolar (bjt) and a variety of field effect (jfet) and MOS transistors become increasingly complex, sometimes with several equations describing the behaviour of different aspects of device performance in different regions of operation.

Because these sets of equations are very much based on the semiconductor physics of devices and the manufacturing processes used to fabricate them, for some families of devices, such as MOSFETs, different sets of equations may be used to describe devices in the same family. The different sets of equations may be used because the manufacturer wishes to describe the operation of their devices to a greater or lesser degree of accuracy.

Although the equations themselves are hidden deep in the source code for a simulator, in general the coefficients of the sets of equations are collected together in the form of a list. Individual devices of any particular device family can then be described by a list of coefficients.

This list of coefficients is called a **model** .

The individual coefficients in a model are called the **model parameters**.

A device model written in this way is called a **.model** statement.

Some devices such as Thyristors, opamps, linear regulators and switch mode supply chips are made up from a number of other devices connected together to form subcircuits.

A spice netlist of a device defined by a subcircuit is also referred to as a **model**.

A device model written in this way is called a **.subckt**.

Subcircuit models may themselves contain .model statements.

Subcircuits can also contain parameters and can also have parameters passed to them to change their characteristics for example to tailor them to a particular device variant.

Why are there different models for the same device?

Because each family of devices (resistors, diodes, bjts, jfets, MOSFETs etc.) is described by one or more sets of equations, each family has one or more models available for it.

One reason there are different models available for devices in the same family is because manufacturers give device models away for free. Therefore they do not want to spend any more time on developing device models than they need to. Basically, the more complex a model is, the more time the manufacturer has to spend on making measurements in order to derive the model parameters. Therefore if they feel that a device can be adequately described by a simple model then they will use that rather than a more accurate but more complex and so more expensive one.

Another reason there may be differences between models of the same device is that there may be slight differences in the semiconductor fabrication processes of different manufacturers.

In the same way that there may be more than one .model available for a device. there may be different .subckt defined models available.

There may be differences between .subckt models because there are implementation differences in the device models and/or the physical devices from different manufacturers. For example there are slight differences in internal timings and even a subtle difference in the internal circuitry of the oscillator section of the UC384x family of SMPS controllers between the various different manufacturers.

Sometimes, there are differences in the models just to get around the copyright protection. Some differences are to optimise the model for a particular simulator and some differences are simply down to the preferences of the model writer.

.model statements

In the spice netlist of a circuit, the user can see the models listed in .model statements. When a schematic is saved, these .model statements are pulled in to the netlist by EasyEDA recognising the symbols and their associated device names given in the schematic. Each model may either be pulled in from a library or - for devices that are not in the EasyEDA libraries - by downloading a model from a manufacturer's website and then manually pasting it directly into the schematic (the process of doing this will be described later).

Ngspice model types

To help identify model types and in particular if they are for N or P type devices, the following table of model types may be helpful.

Code	Model Type
R	Semiconductor resistor model
C	Semiconductor capacitor model
L	Inductor model
SW	Voltage controlled switch
CSW	Current controlled switch
URC	Uniform distributed RC model
LTRA	Lossy transmission line model
D	Diode model
NPN	NPN BJT model
PNP	PNP BJT model
NJF	N-channel JFET model
PJF	P-channel JFET model
NMOS	N-channel MOSFET model
PMOS	P-channel MOSFET model
NMF	N-channel MESFET model
PMF	P-channel MESFET model

Although it is beyond the scope of this document to go into detail there are some other points about models that are worth mentioning.

- Models for the basic resistors, capacitors and inductors used in a schematic are usually not shown in the netlist;
- Some device models have a full list of parameters, some may only have a partially completed list. Missing parameters in models are simply replaced by default values.
- Different simulators support different sets of models so in some cases the simulator may warn the user that some parameters are unrecognised and so are ignored. This generally has little effect on the simulation results but if the user is particularly concerned about their effects than the only option is to change to using a simulator that supports all the relevant parameters.

.subckt definitions

Not all devices are described by .model statements.

Models of more complex devices such as Thyristors (SCRs, Triacs and also Diacs), Insulated Gate Bipolar Transistors (IGBTs), operational amplifiers (opamps) and even many MOSFETs are often made up by connecting lower level devices to make a circuit that behaves like the desired device. This is called a **subcircuit**. The spice netlist of this subcircuit is then used to create a type of device model defined by what is called a **.subckt**. The low level components in subcircuits are described by the same sort of models (those lists of parameters or coefficients) as for the basic diodes etc., already referred to so a .subckt will often contain a list of .model statements describing the devices that are used to build the .subckt itself. Complex .subckts may even call other .subckts.

Behavioural models

Using Behavioural Voltage and Current Sources and expressions it is possible to create what are called **behavioural models** of components. These are models that behave like a device but which have little of the actual underlying realistic circuit defined and are mostly - or perhaps completely - described by explicitly defined expressions (equations). The models for most devices internally comprising more than one active component, i.e. ICs, are largely behavioural. This is a way of hiding the detailed information about the manufacturer's process technology that low level spice modelling reveals.

The use of expressions and behavioural sources in EasyEDA is explained later in the book.

What if there is no model available for a device?

Not all devices have spice models that can be run in ngspice. There are a number of possible reasons for this.

1. Some models are encrypted and can only be run in certain proprietary simulation tools;
2. Some proprietary simulators support models that are not available in ngspice;
3. Some devices have models that only run in specific non-spice based simulation tools and which, for whatever reason, cannot be translated into spice models;
4. Some devices do not have publically available models;
5. Many devices predating the creation of the original spice program do not have models;
6. Models for some devices simply do not exist because the manufacturers have never created them;
7. Some models may be unavailable in EasyEDA because they are restricted by copyright or end user licenses so they can only be run in certain proprietary simulation tools or cannot be shared publically.

In cases (1) to (3), there is no way they can be run in ngspice. They must be run in the simulation tools for which they were written.

In cases (4) to (7), it is sometimes possible to find an equivalent, alternative or similar device for which a spice model is available. The user must exercise caution and use their judgement in deciding if such an approach offers satisfactory simulation results.

It must be noted that spice was not originally written with support for thermionic devices (valves or tubes) so models for such devices exist only in .subckt form. They are usually created by enthusiasts rather than manufacturers and so they (a) can be hard to find and (b) should be used with caution. EasyEDA does have a library of valve models gathered from sources that we believe have written reasonably accurate models.

Note that models obtained from manufacturers are often subject to copyright restrictions. Please respect any copyright notices contained either in end user license agreements that may have to be accepted prior to the granting of access to a downloadable copy of a model or in the models themselves.

Similarly, models contained in the libraries of commercial simulation tools are subject to copyright restrictions.

It is often possible to find device models offered in forums, discussion groups and various online collections of models. Again, the user must exercise caution and use their judgement in deciding if such models really are suitable. Often it is not possible to establish where they originate from so their validity is very hard to verify.It is also possible that such models have been copied in breach of the originators copyright.

The relationship between spice models and device datasheets

Although some of the device models in EasyEDA have been specially written so that the user can easily tailor them to simulate a range of devices by editing parameters that can be found directly in - or inferred from - device datasheets (see: About the relationship between spice models and real world behaviour below), most of them are off-the-shelf models from the device manufacturers.

It is important to understand that, for many of these off-the-shelf models, the underlying equations and therefore the .model parameters and .subckt definitions bear little relationship to the sort of information that is given in typical component datasheets. Therefore it is usually not possible to take a device datasheet and simply write down a device models from the information given in it.

Whilst it is possible to extract spice parameters for a variety of devices from device datasheets and from actual device measurements, it is beyond the scope of this document to describe how this can be done.

More information about what the model parameters mean in diodes, bipolar transistors and MOSFETs, is available from:

http://www3.imperial.ac.uk/pls/portallive/docs/1/56133736.PDF

with individual slide sets:

http://www3.imperial.ac.uk/pls/portallive/docs/1/7292571.PDF

http://www3.imperial.ac.uk/pls/portallive/docs/1/7292572.PDF

http://www3.imperial.ac.uk/pls/portallive/docs/1/7292573.PDF

For more detailed information about bjt's in particular, this book:

Modelling the Bipolar Transistor by Ian Getreu

is available from:

http://www.lulu.com/spotlight/iangetreu

and

http://www.amazon.com/Modeling-Bipolar-Transistor-Ian-Getreu/dp/B000EYPQLU

Another excellent (and free) book about transistor modelling, is available by going to:

http://www.aeng.com/spice_modeling.htm

and registering to get a copy of:

Definitive Handbook of Transistor Modeling

More information about ngpsice is available from here:

http://ngspice.sourceforge.net/presentation.html

More information about Larry Nagel and SPICE is available from here:

http://www.omega-enterprises.net/The%20Origins%20of%20SPICE.html

Larry's PhD dissertation Dissertation:

Laurence W. Nagel., "SPICE2: A Computer Program to Simulate Semiconductor Circuits,"

*Memorandum No. ERL-M520, University of California, Berkeley, May 1975.

http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/ERL-520.pdf

is actually very readable and instructive.

For more information about electronic circuit simulation and spice in particular, see:

http://en.wikipedia.org/wiki/Electroniccircuitsimulation

and:

http://en.wikipedia.org/wiki/SPICE

The relationship between spice models and real world behaviour

Not all spice models are created equal. Here are just some of the things to be aware of.

Models of the same device from different manufacturers may offer differing degrees of accuracy. Sometimes models are kept simple in the interests of speeding up the simulations at the expense of accuracy. Sometimes they are complex because accuracy is considered to be more important than simulation speed. Models may contain some text at the beginnings of them to describe some of their limitations or their special features. It is often useful to read this information as it can help improve the convergence of simulations using them.

Not all diode models simulate reverse breakdown voltage.

Zener diode models can be of varying accuracy and are best put into test jigs to run a curve trace on them to compare them with the datasheet. Zener diodes are sometimes used as white noise sources. Zener models do not accurately generate the levels and spectrum of noise seen in real devices.

None of the bjt models simulate the reverse bias base-emitter breakdown voltage. Very few model collector-emitter or collector-base junction breakdown voltages.

Some models, particularly of high speed and high frequency devices may include package parasitics such as lead inductances and pin capacitances. Such models are almost always .subckt definitions of devices defined by .model statements but which have the parasitics connected to form a subckt. If the high frequency behaviour is not important, simulation speed can be improved by using only the .model statement without the parasitics. This .model can be cut and pasted out of the .subckt definition but often the .model statement will be for a transistor that is defined as a .model in it's own right somewhere else on the manufacturers site or as an equivalent from another vendor.

Thyristor and Triac models can be of varying accuracy or simulate only a limited selection of all the device parameters.

EasyEDA has an in-house behavioural Thyristor macromodel and a behavioural Triac macromodel.

As far as possible, the EasyEDA in-house Thyristor and Triac models model almost all the datasheet parameters of the target device with the exception of di/dt behaviour with inductive loads. These devices can be tailored to model almost any device simply using the values taken from the datasheet for the target device.

Metal Oxide Varistors (MOVs) are a nightmare to model and are best avoided! Even the commercially available models sometimes do not run reliably in all conditions.

Some opamp models are hugely detailed and can be very accurate but care must be taken to check that they are written using a syntax that is compatible with ngspice. Devices tailored for some of the commercial simulators will not run in ngspice without some syntax changes. Some may require special **.option**s to be invoked for the simulator.

Beware that even some quite complex opamp models do not simulate supply current drains even as simple DC quiescent currents let alone the dynamic behaviour with load currents added in. This can be an advantage since it reduces the signal currents that have to be simulated. It also means that there is absolutely no point in including any supply rail decoupling for those device that are known to not model supply current drains since they do not draw any current: they only use the supply voltage to define things like common mode range or output swing.

Here is an example of a third party opamp model that does not model supply or output currents:

LM108 test jig

Some opamp models may make no attempt to accurately simulate the output stage behaviour versus load current. Similarly, many device models do not simulate the behaviour of inputs and outputs when they are taken above or below the supply rails.

Few device models simulate the excessive supply current drain of a supply reversed misconnection or a correctly connected device that is subject to a supply voltage above the stated absolute maximum supply differential.

There are a several device models in EasyEDA that have been specially written to reproduce the real world behaviour of the devices that they model.

For example, the EasyEDA in-house opamp behavioural macromodel can be set up to give an output voltage swing anywhere from a rail-to-rail to the more restricted swings of non-rail-to-rail output opamps. The output swing can be asymmetric.

Input resistance, bias and offset current and input offset voltage are modelled.

The input differential and common mode voltage ranges are modelled.

The current drain behaviour of the device if input or output pins are taken above or below the supply rails or if the supply polarities are reversed are modelled. Output polarity reversal due to inputs exceeding the common mode range is modelled for devices that exhibit such behaviour.

Frequency dependent common mode and power supply rejection are modelled.

Noise and temperature dependent effects are not modelled at present.

EasyEDA has an in-house behavioural macromodel which can be tailored to model a wide range of 3 terminal fixed and adjustable positive and negative linear voltage regulators which feature similar real-world behaviour to the opamp models.

For all of the in-house EasyEDA models, more information about them can be found in the .subckt definition itself simply by viewing the spice netlist of any saved circuit thay have been put into.

How to change the model attached to a symbol

Please note that before attempting to edit device models, it is *essential* that the user is familiar with and understands the relationship between spice pin names and numbering, described in the section on 'Schematic symbols: prefixes and pin numbers'.

We are working to provide a search function for the device simulation models available in EasyEDA.

Right now there are a couple of ways to change the model for a device.

1) Place a device from the EasyEDA Libs and then edit the device model name either in place in the schematic or in the right hand properties panel.

For instance, when an NPN bjt is placed in a schematic, it comes in with a default name of editing the model name of 2DC2412R. This name pulls the associated default 2DC2412R model into the spice netlist. Editing the device name from 2DC2412R to 2N2222 will pull the 2N2222 model from EasyEDA's spice model library into the netlist.

The problem here is that until a model search function is up and running this approach is obviously too hit and miss for an arbitrary choice because there no way to see which models are available to choose from.

2) The second option is a bit more fiddly but it allows almost any unencrypted device model to be run in a simulation. The process is similar for both .model and .subckt defined models.

These processes are described in detail in 'Associating spice models with schematic symbols' as part of the following section on 'Schematic symbols: prefixes and pin numbers' but it is strongly recommended that the whole of that section is read in order, to make sense of the detailed descriptions.

Schematic symbols: prefixes and pin numbers

Please note that before attempting to edit device models, it is *essential* that the user is familiar with and understands the relationship between spice pin names and numbering, described in this section.

Device and subcircuit (or hierarchical block) symbols created for use in schematics that are intended to be run as spice simulations, in addition to having a PCB Prefix that is used for the reference designator in the schematic, also have a Spice Prefix. They also have two sets of pin numbers: PCB pins and Spice pins.

PCB and Spice Prefix

The rules on the assignment of the PCB Prefix or reference designator of a schematic symbol are somewhat dependent on the EDA tool and on the user's preferences. Depending on how a device is graphically represented by it's schematic symbol it may have a different PCB Prefix or reference designator. For example, a single discrete MOSFET device may have a PCB Prefix of Q, M or perhaps TR, whereas if it is part of a monolithic multiple transistor array it may have a PCB Prefix of U or IC.

The rules on the assignment of the Spice Prefix of a schematic symbol are strict. This is because the Spice Prefix is used to tell the simulator which circuit element the symbol represents and therefore which simulation model it is to use.

Simulation models for most of the spice circuit elements are in the form of a single-line .model statement however some of them may be in the form of a multi-line .subckt subcircuit definition. For example, some MOSFETs may be described by a .model statement in which case their Spice Prefix is M but many MOSFETs are described by a .subckt and so their Spice Prefix is X.

Therefore, irrespective of the PCB Prefix chosen for a schematic symbol, the Spice Prefix for a schematic symbol representing a given circuit element must match the type of model required to simulate that instance of that circuit element in your schematic.

For example, if there are two different n-channel MOSFETs in a schematic; Q1, a BSS123 which is modelled by a .model statement: ~~~~ *SRC=BSS123;DI*BSS123;MOSFETs N;Enh;100V 0.170A 1.00ohms Diodes Inc. MOSFET .MODEL DI*BSS123 NMOS(LEVEL=1 VTO=1.00 KP=6.37m GAMMA=1.24 + PHI=.75 LAMBDA=625u RD=0.140 RS=0.140 + IS=85.0f PB=0.800 MJ=0.460 CBD=19.8p + CBS=23.7p CGSO=36.0n CGDO=30.0n CGBO=124n * -- Assumes default L=100U W=100U -- ~~~~ and Q2, a BSS127S which is modelled by a .subckt: ~~~~ *----------

BSS127S Spice Model ---------- .SUBCKT BSS127S 10 20 30 * TERMINALS: D G S M1 1 2 3 3 NMOS L = 1E-006 W = 1E-006 RD 10 1 84.22 RS 30 3 0.001 RG 20 2 29 CGS 2 3 1.958E-011 EGD 12 0 2 1 1 VFB 14 0 0 FFB 2 1 VFB 1 CGD 13 14 2E-011 R1 13 0 1 D1 12 13 DLIM DDG 15 14 DCGD R2 12 15 1 D2 15 0 DLIM DSD 3 10 DSUB .MODEL NMOS NMOS LEVEL = 3 VMAX = 8E+005 ETA = 1E-012 VTO = 3.419 + TOX = 6E-008 NSUB = 1E+016 KP = 0.127 U0 = 400 KAPPA = 1.044E-015 .MODEL DCGD D CJO = 1.135E-011 VJ = 0.9232 M = 0.9816 .MODEL DSUB D IS = 2.294E-010 N = 1.601 RS = 0.1079 BV = 65 + CJO = 1.956E-011 VJ = 1.514 M = 0.8171 .MODEL DLIM D IS = 0.0001 .ENDS *Diodes BSS127S Spice Model v1.0 Last Revised 2012/6/6 ~~~~ then even though both have the same PCB Prefix of Q: Q1 must have a Spice Prefix of M and Q2 must have a Spice Prefix of X.

A list of Spice Prefixes and their associated circuit elements is given in the table below.

Element description	Spice Prefix	Comment
A	XSPICE code model	analogue, digital, mixed signal
B	Behavioural (arbitrary) source	
C	Capacitor	
D	Diode	
E	Voltage-controlled voltage source (VCVS)	linear, non-linear
F	Current-controlled current source (CCCS)	linear
G	Voltage-controlled current source (VCCS)	linear, non-linear
H	Current-controlled voltage source (CCVS)	linear
I	Current source	
J	Junction field effect transistor (JFET)	spice pin order: D G S
K	Coupled (Mutual) Inductors	
L	Inductor	
M	Metal oxide field effect transistor (MOSFET)	spice pin order: D G S
N	Numerical device for GSS	
O	Lossy transmission line	
P	Coupled multiconductor line (CPL)	
Q	Bipolar junction transistor (BJT)	spice pin order: C B E
R	Resistor	
S	Switch (voltage-controlled)	
T	Lossless transmission line	
U	Uniformly distributed RC line	
V	Voltage source	
W	Switch (current-controlled)	
X	Subcircuit	spice pin order: depends on subckt
Y	Single lossy transmission line (TXL)	
Z	Metal semiconductor field effect transistor (MESFET)	spice pin order: D G S

For more information on circuit elements in Ngspice, please refer to:

http://ngspice.sourceforge.net/docs/ngspice-manual.pdf#subsection.2.1.2

PCB and Spice pin numbers

The two sets of pin numbers are:

1. PCB pin number: these are the numbers for the real, physical device pins in its package. They are required so that the pins of a device symbol in a schematic can be mapped onto the physical pins of a PCB footprint. In other words, so that the connections shown in the schematic, end up connected properly by copper on the PCB.
2. Spice pin number or pin order: these are the numbers that map the pins on the symbol to their respective functions in the spice model or

subcircuit.

Actually the spice pin ordering has a slightly deeper meaning.

Spice has no concept of component symbols: they are a construct of the schematic editor.

When a spice netlist is generated, the symbol in the schematic editor is either - in the case of model defined devices such as resistors, capacitors, inductors, diodes, transistors and sources - mapped directly to the relevant models (defined by the device prefix such as R, C, L, D, Q and so on), or in the case of a subcircuit, converted into a subcircuit call statement.

The spice pin ordering for the majority of built-in models such as resistors, capacitors, inductors, diodes, transistors and sources are defined and generally taken care of by the schematic editor, more care has to be taken with the spice pin ordering of subcircuits.

This can be illustrated by a simple opamp with 5 pins: inverting and non-inverting inputs; output and positive and negative supply pins but the principle applies to all spice subcircuits.

The subcircuit call for this opamp might look like this in the spice netlist:

~~~~ X1 input feedback vpos vneg output opamp_ANF01 ~~~~

where:

`X1` is the name of the subcircuit in the top level (i.e. the calling) circuit;

`input feedback vpos vneg output` are the netnames in the circuit calling (i.e. containing) the subcircuit and

`opamp_ANF01` is the name of the subcircuit being called.

- Pay special attention to the order of the netnames in the subcircuit call.

The spice pin ordering for the majority of opamp subcircuits is like that shown in the example below: ~~~~

- opamp_ANF01 *
- Simplified behavioural opamp
- Node assignments
- noninverting input
- | inverting input
- | | positive supply
- | | | negative supply
- | | | | output
- | | | | |
- spice pin order: 1 2 3 4 5
- | | | | | .subckt opamp_ANF01 inp inn vcc vee out ; these are the netnames
- used internally to the subcircuit. * B1 out 0 + V=(TANH((V(inp)-V(inn))*{Avol}*2/(V(vcc)-V(vee)))*(V(vcc)-V(vee)) + +(V(vcc)+V(vee)))/2 .ends opamp_ANF01

~~~~

Note that the spice pin order of the subcircuit call is in exactly the same order as that of the subcircuit.

Although the physical pin numbering of any device is critical for successfully mapping the pins of a schematic symbol onto a physical package footprint when laying out the PCB, because spice only knows about single devices and does not care about how they are physically packaged, each instance of any device in a spice schematic has to be mapped onto the same pins of the spice model or subcircuit, irrespective of where it is in any physical package.

Therefore, for the physical, package pin numbering of the four opamps in a quad opamp in say, a SOIC14 or a DIP14 package as shown below, to work with the example subcircuit above, the spice pin ordering would be:

Opamp A	**PCB Pin Number**	**Spice Pin Order**
OUT	1	5
IN-	2	2
IN+	3	1
V+	4	3
V-	11	4
Opamp B	**PCB Pin Number**	**Spice Pin Order**
OUT	7	5
IN-	6	2
IN+	5	1
V+	4	3
V-	11	4
Opamp C	**PCB Pin Number**	**Spice Pin Order**
OUT	8	5

IN-	9	2
IN+	10	1
V+	4	3
V-	11	4
Opamp D	**PCB Pin Number**	**Spice Pin Order**
OUT	14	5
IN-	13	2
IN+	12	1
V+	4	3
V-	11	4

The physical package pin numbering reflects that of each opamp in the package.

The spice pin ordering is the same for each instance of the individual opamps.

Of course there is only one physical instance of each supply pin on the schematic symbol for the quad opamp in this example but each spice subcircuit must have the supply pins explicitly defined.

Exactly how this is handled at the schematic symbol level depends on how the schematic capture package deals with symbols for multiple devices with shared supply pins but the generation of a spice netlist from the schematic will always generate the complete set of pins required in the subcircuit calls.

In cases where the subcircuit is built by the user as opposed to where it is supplied by a vendor for a particular device, exactly the same rules apply except that it is up to the user to specify the subcircuit pin order and to construct the symbol appropriately.

Although as described earlier, built-in spice models usually have defined spice pin orders, not all subcircuits have the same spice pin numbering. Therefore if your spice circuit throws errors - especially if there are warnings about pin numbers or pin names - it is worth remembering to check that the pin order of the symbol that is netlisted to form the calling statement matches that of the subcircuit that is being called!

Associating spice models to schematic symbols

For .MODEL defined models

1. Find a spice .model for your target device;
2. Copy and paste it into a text placeholder (the `T` hotkey) in your schematic (but please respect the EULA and copyright of commercial files);
3. In the right hand properties panel, change the text type from comment to spice;

   Properties > Text type > spice

4. Place a symbol for the device from the EasyEDA Libs palette onto the schematic;

5. Edit the model name to the exact name of the model in the pasted file;
6. Done!

There's an example of this here:

Playing with model parameters

This is another example showing using a generic depletion mode MOSFET.

It also shows a way to hack a MOSFET defined by a LEVEL 3 .model statement but which has a problem with some of the parameters not being recognised as being part of the model by ngspice, so that it can still be used directly with the MOSFET symbol.

In this example the L and W parameters of the original model are recognised as part of the .model statement. Note also that some of the other parameters are also simply not recognised by ngspice.

Here's how:

1. Find a spice .model for your target device;
2. Copy and paste the .model statement into the schematic canvas;
3. Turn it into a spice directive:

   Properties > Text Attributes > Text type > spice

4. Place an N channel depletion mode MOSFET symbol onto the schematic;

5. Edit the 'model' attribute for M1 to include the unrecognised or modified L and W parameters so they look like this:

   `IXTT20N50D L=2E-6 W=5.5`

   This can be done either in place or via:

   Part Attributes > Model > `IXTT20N50D L=2E-6 W=5.5`

Note that adding an asterisk at the start of the two lines in the .model statement that define the L and W parameters (and any other unrecognised parameters as deemed necessary) will comment them out. This stops these parameters being reported as model issues in the simulation report but it is not required to do so.

This process is illustrated in the following example:

N channel depletion mode MOSFET using a .model statement

For .SUBCKT defined models

The process described above works fine for simple .model defined models but for .subckt defined models it is a little more complicated because you need to tell EasyEDA that the model is a .subckt and not a simple .model.

Note that even some humble diode models are in fact .subckt defined to include things like package parasitics. For example, compare the 1N4148 and the 1N4148W-V models in the netlist.

There are three stages in attaching a .subckt to a symbol that already has a spice prefix of 'X' and so is expecting to call a .subckt statement.

- Place the .subckt text into the schematic and activate it;
- Place the symbol in the schematic;
- Change the name of the symbol to exactly the same as the name of the .subckt;

The detailed steps to associate a new .subckt model to the symbol are:

1. Find a spice .SUBCKT for your target device;
2. Copy and paste it into a text box (the `T` hotkey) in your schematic (but please respect the EULA and copyright of commercial files);
3. In the right hand properties panel, change the text type from comment to spice;

   Properties > Text type > spice

4. Place a symbol for the device from the EasyEDA Libs palette onto the schematic;

5. Edit the model name to the exact name of the model in the pasted file;
6. Press the `I` Hotkey or:

   Click the blue **Edit Symbol...** button in the Properties panel:

   Properties > Edit Symbol...

   or do:

   Super Menu > Miscellaneous > Edit Symbol

7. In the **Modify your symbol information** dialogue box, check that the **Spice Prefix** is `X`;

8. Check that the NUMBER of pins in 'Edit Pin Map information' is exactly the same as in the .SUBCKT pasted into the schematic: if it is not then the wrong symbol has been placed for the chosen .SUBCKT (or vice versa) so a different symbol (or .SUBCKT) must be chosen.

   Note that 'number of pins' here means how many pins, not the pin numbers or names used to describe the nets they connect to in the .subckt netlist;

9. Check that the ORDER of the pins in **Edit Pin Map information** is exactly the same as in the .SUBCKT pasted into the schematic. This can be very confusing because the pin NAMES may be different between the symbol and the .SUBCKT so it is first necessary to reconcile the two sets of names before attempting to confirm their order.
10. Click **OK** in the **Modify your symbol information** dialogue box;
11. Done!

    This process is illustrated in the following example:

    Attaching a .subckt to a symbol 01

Some of the EasyEDA symbols from the EasyEDA Libs have a Spice Prefix that is expecting to call a .model statement (i.e. any symbol with a Spice Prefix other than 'X'!). For example, the bjts symbols have a Spice Prefix of 'Q' whilst the MOSFET symbols have a Spice Prefix of 'M'. There are some diodes, for example some zener diodes, which some are defined by .models and others by .subckts. Even the humble 1N4148 may have a .model from one manufacturer and a .subckt from another. Some bjt models are defined by .subckts such as most Darlington transistors and some Avalanche transistors. Apart from some low power MOSFETs and those used in IC design, most MOSFET models are defined by .subckts. Therefore it is quite common to have to associate a .subckt to a symbol with a Spice Prefix that is expecting to call a .model statement.

This can be done in four stages as illustrated in the following example, the NMOS_E symbol placed into the schematic from the EasyEDA Libs palette must be edited to change the 'Spice Prefix' of the symbol from 'M' (for a .model defined part) to 'X' (for a .subckt defined part).

- Place the .subckt text into the schematic and activate it;
- Place the symbol in the schematic;
- Change the name of the symbol to exactly the same as the name of the .subckt;
- Change the 'Spice Prefix' of the symbol from 'M' (for a .model defined part) to 'X' (for a .subckt defined part).

The detailed steps to associate a new model to a symbol and to tell EasyEDA that a device model is a .subckt and not a simple .model are:

1. Find a spice .SUBCKT for your target device;
2. Copy and paste it into a text box (the `T` hotkey) in your schematic (but please respect the EULA and copyright of commercial files);
3. In the right hand properties panel, change the text type from comment to spice;

   Properties > Text type > spice

4. Place a symbol for the device from the EasyEDA Libs palette onto the schematic;

5. Edit the model name to the exact name of the model in the pasted file;
6. Press the `I` Hotkey or:

Click the blue **Edit Symbol...** button in the Properties panel:

Properties > Edit Symbol...

or do:

Super Menu > Miscellaneous > Edit Symbol

7. In the 'Modify your symbol information' dialogue box, change the **Spice Prefix** from `M` (for a .model defined part) to `X` (for a .subckt defined part);
8. Check that the NUMBER of pins in **Edit Pin Map information** is exactly the same as in the .SUBCKT pasted into the schematic: if it is not then the wrong symbol has been placed for the chosen .SUBCKT (or vice versa) so a different symbol (or .SUBCKT) must be chosen.

   Note that 'number of pins' here means how many pins, not the pin numbers or names used to describe the nets they connect to in the .subckt netlist;
9. Check that the ORDER of the pins in **Edit Pin Map information** is exactly the same as in the .SUBCKT pasted into the schematic. This can be very confusing because the pin NAMES may be different between the symbol and the .SUBCKT so it is first necessary to reconcile the two sets of names before attempting to confirm their order.
10. Click OK in the **Modify your symbol information** dialogue box;
11. Done!

This process is illustrated in the following example:

Attaching a .subckt to a symbol 02

Another example of the process described above to change the Spice Prefix of a symbol is illustrated with the same EasyEDA N channel depletion mode MOSFET symbol from the EasyEDA Libs that was used earlier with the IXTT20N50D .model statement. In this example the MOSFET symbol is attached to a .subckt that has been created from the orignal IXTT20N50D .model statement in order to wrap up the L=2E-6 W=5.5 parameters and so make using the original model easier.

An N-channel depletion mode MOSFET using an EasyEDA .subckt

Attaching models to custom symbols

This is basically the same as attaching a model to any of the predefined symbols from the EasyEDA Libs except that the symbol is one that has been created from scratch or by editing an existing symbol. The rules for assigning and checking that the spice prefix matches the type of model to be attached (i.e. 'X' for .subckt or any Spice Prefix other than 'X' for .model) and checking that the spice pin numbering matches that of the type of device defined by the .model statement or by the pin sequence of a .subckt defined model.

Custom modelling

This is an advanced topic and this section will be filled out when time permits.

However, browsing through some of the EasyEDA in-house models can be informative because, although they do not come with a 'How it works' written into them, there is some documentation in their .subckt definitions that may give some insights into the wild and wonderful world of custom modelling.

Some examples to look through the netlists of are:

LDR test

Electret microphone model

Electret microphone model .subckt

Electret microphone model test jig

LM56EE demo jig

How to include a 5 Pin Comparator in a schematic

and this project:

An EasyEDA logic family

Making measurements of simulation results

In the same way that digital storage oscilloscopes (DSO) allow measurements to be made of signals displayed on the screen using cursors and directly reading values from their positions and from mathematical analysis of the waveform data stored in the DSO memory, EasyEDA allows measurements to be made of simulation results directly using cursors and by analysis of the data used to create the WaveForm trace display using the **meas** command.

Using the WaveForm display cursors

The WaveForm X and Y cursor functions are a simple and quick way to make measurements of points in waveforms and to make measurements of differences between points.

WaveForm allows the display of traces in any selection of up to three vertically stacked plot panes. The Y axes automatically scale to fit the units and the range of the traces being displayed. Traces can be hidden but at least one trace must be visible. X and Y trace data can be seen on-screen just by moving the mouse cursor around the plot area of a pane with the readout adapting to the Y axes in each pane.

Delta X and delta Y trace data can be seen on-screen using a Left-Click and Drag select box, with the readout adapting to the Y axes in each pane. Returning the cursor to within a small radius of the starting point of the select box - without releasing the Left-Click - returns the readout to X and Y trace data.

The cursor placement and results produced are volatile, meaning that they cannot be copied and pasted. They are not saved as part of a saved WaveForm file. However, using a screenshot utility, it is possible to save an image of the WaveForm display showing the cursor positions and their associated readout.

Note that the screenshot utility must have a user definable time delay to allow cursor placement to be carried out between initiating the screenshot and the screenshot actually being taken.

For more information on displaying simulation results in Waveform, please refer to WaveForm in the EasyEDA Tutorial.

Using the meas command

The meas command is used to analyse the output data of a dc, ac, tran or fft (or spec) simulation. The command is executed immediately after the simulation has finished.

The meaning of terms in MEAS commands

The meas type **{DC|AC|TRAN|SP}** depends on the data which are to be evaluated, either originating from a DC, AC analysis, TRANsient or SPectrum analysis (using the fft (or spec) analysis) simulation.

result will be a vector containing the result of the measurement.

trig*variable*, **targvariable**, and **out_variable** are vectors stemming from the simulation, e.g. a voltage vector v(out).

VAL=val expects a real number val. val may also be a real parameter or an expression enclosed by in â€ or {} that expands to a real number.

TD=td and **AT=time** expect a time value if the meas type is **tran**. For **ac** and **sp**, **AT** will be a frequency value and **TD** is ignored.

For **dc** analysis **AT** is a voltage (or current), and **TD** is ignored.

CROSS=# requires an integer number #. **CROSS=LAST** is possible as well. The same is expected by **RISE** and **FALL**.

Frequency and time values may start at 0 and extend to positive real numbers. Voltage (or current) inputs for the independent (scale) axis in a dc analysis may start or end at arbitrary real valued numbers.

Examples of the forms and syntaxes of MEAS commands

Trig Targ

A meas command using the Trig Targ syntax of **General Form 1** measures the difference in dc voltage, frequency or time between two points selected from one or two output vectors. The following examples all are using transient simulation. Measurements for tran analysis start after a delay time td. If other examples are run with ac simulation or spectrum analysis, time may be replaced by frequency whilst after a dc simulation, the independent variable may become a voltage or current.

General Form 1:

MEAS {DC | AC | TRAN | SP} result TRIG trig_variable VAL=val

<TD=td> <CROSS=# | cross="LAST">

<RISE=# | rise="LAST"> <FALL=# | fall="LAST">

<TRIG at="time"> TARG targ_variable VAL=val <TD=td>

<CROSS=# | cross="LAST">

<RISE=# | rise="LAST"> <FALL=# | fall="LAST">

<TARG at="time">

Measure statement example:

```
meas tran tdiff TRIG v(1) VAL=0.5 RISE=1 TARG v(1) VAL=0.5 RISE=2
```

measures the time difference between v(1) reaching 0.5 V for the first time on its first rising slope (TRIG) versus reaching 0.5 V again on its second rising slope (TARG); i.e. it measures the signal period.

Output:

```
tdiff = 1.000000e-003 targ= 1.083343e-003 trig= 8.334295e-005
```

Measure statement example:

```
meas tran tdiff TRIG v(1) VAL=0.5 RISE=1 TARG v(1) VAL=0.5 RISE=3
```

measures the time difference between v(1) reaching 0.5 V for the first time on its rising slope versus reaching 0.5 V on its rising slope for the third time (i.e. two periods).

Measure statement example:

```
meas tran tdiff TRIG v(1) VAL=0.5 RISE=1 TARG v(1) VAL=0.5 FALL=1
```

measures the time difference between v(1) reaching 0.5V for the first time on its rising slope versus reaching 0.5 V on its first falling slope.

Measure statement example:

```
meas tran tdiff TRIG v(1) VAL=0 FALL=3 TARG v(2) VAL=0 FALL=3
```

measures the time difference between v(1) reaching 0V its third falling slope versus v(2) reach- ing 0 V on its third falling slope.

Measure statement example:

```
meas tran tdiff TRIG v(1) VAL=-0.6 CROSS=1 TARG v(2) VAL=-0.8 CROSS=1
```

measures the time difference between v(1) crossing -0.6 V for the first time (any slope) versus v(2) crossing -0.8 V for the first time (any slope).

```
Measure statement example:
```

meas tran tdiff TRIG AT=1m TARG v(2) VAL=-0.8 CROSS=3

measures the time difference between the time point 1ms versus the time when v(2) crosses -0.8 V for the third time (any slope).

Find ... When

The FIND and WHEN functions allow to measure any dependent or independent time, frequency, or dc parameter, when two signals cross each other or a signal crosses a given value.

Measurements start after a delay TD and may be restricted to a range between FROM and TO.

General Form 2:

MEAS {DC | AC | TRAN | SP } result WHEN out_variable=val

<TD= td> <FROM=val> <TO= val>

<CROSS=# | cross="LAST">

<RISE| rise="LAST"> <FALL=# | fall="LAST">

Measure statement example:

```
meas tran teval WHEN v(2)=0.7 CROSS=LAST

measures the time point when v(2) crosses 0.7 V for the last time (any slope).
```

General Form 3:

MEAS {DC | AC | TRAN | SP } result WHEN out*variable=out*variable2

<TD= td> <FROM= val> <TO= val>

<CROSS=# cross="LAST">

<RISE | rise="LAST"> <FALL=# | fall="LAST">

Measure statement example:

```
meas tran teval WHEN v(2)=v(1) RISE=LAST
```

measures the time point when v(2) and v(1) are equal, v(2) rising for the last time.

General Form 4:

MEAS {DC | AC | TRAN | SP } result FIND out*variable WHEN out*variable2=val

<TD= td> <FROM= val> <TO= val>

<CROSS=# | cross="LAST"> <RISE | rise="LAST">

<FALL=# | fall="LAST">

Measure statement example:

```
meas tran yeval FIND v(2) WHEN v(1)=-0.4 FALL=LAST
```

returns the dependent (y) variable drawn from v(2) at the time point when v(1) equals a value of -0.4, v(1) falling for the last time.

General Form 5:

MEAS {DC | AC | TRAN | SP } result FIND out*variable WHEN out*variable2=val=out_variable3

<TD= td> <CROSS=# | cross="LAST"> <RISE | rise="LAST"> <FALL | fall="LAST">

Measure statement example:

```
meas tran yeval FIND v(2) WHEN v(1)=v(3) FALL=2
```

returns the dependent (y) variable drawn from v(2) at the time point when v(1) crosses v(3), v(1) falling for the second time.

General Form 6:

MEAS {DC | AC | TRAN | SP } result FIND out_variable AT=val

Measure statement example:

```
meas tran yeval FIND v(2) AT=2m
```

returns the dependent (y) variable drawn from v(2) at the time point 2 ms (given by AT=time).

AVG | MIN | MAX | PP | RMS | MIN*AT* / *MAX*AT

General Form 7:

MEAS {DC | AC | TRAN | SP } result {AVG | MIN |MAX| PP | RMS | MIN*AT* / *MAX*AT} out_variable

<TD= td> <FROM=val> <TO=val>

Measure statement example:

```
meas tran ymax MAX v(2) from=2m to=3m
```

returns the maximum value of v(2) inside the time interval between 2 ms and 3 ms.

Measure statement example:

```
meas tran tymax MAX_AT v(2) from=2m to=3m
```

returns the time point of the maximum value of v(2) inside the time interval between 2 ms and 3 ms.

Measure statement example:

```
meas tran ypp PP v(1) from=2m to=4m
```

returns the peak to peak value of v(1) inside the time interval between 2 ms and 4 ms.

Measure statement example:

```
meas tran yrms RMS v(1) from=2m to=4m
```

returns the root mean square value of v(1) inside the time interval between 2 ms and 4 ms.

Measure statement example:

```
meas tran yavg AVG v(1) from=2m to=4m
```

returns the average value of v(1) inside the time interval between 2 ms and 4 ms.

INTEG

A meas statement with the INTEG form returns the area under out_variable inside the time interval between FROM val and TO val.

General Form 8:

MEAS {DC | AC | TRAN | SP} result INTEG<RAL> out_variable

<TD=td> <FROM=val> <TO=val>

Measure statement example:

```
meas tran yint INTEG v(2) from=2m to=3m
```

returns the area under v(2) inside the time interval between 2 ms and 3 ms.

DERIV

Please note that meas {DC|AC|TRAN|SP} result DERIV<ATIVE> ... is not yet available in ngspice

More measure statements

```
meas tran inv_delay 2 trig v(in) val='vp/2' td=1n fall=1 targ v(out) val='vp/2' rise=1

meas tran test_data1 trig AT=1n targ v(out) val='vp/2' rise=3

meas tran out_slew trig v(out) val=' 0.2_vp' rise=2 targ v(out) val=' 0.8_vp' rise=2

meas tran skew when v(out)=0.6

meas tran skew2 when v(out)=skew_meas

meas tran skew3 when v(out)=skew_meas fall=2

meas tran skew4 when v(out)=skew_meas fall=LAST

meas tran skew5 FIND v(out) AT=2n

meas tran v0_min min i(v0) from='dfall' to='dfall+period'

meas tran v0_avg avg i(v0) from='dfall' to='dfall+period'

meas tran v0_integ integ i(v0) from='dfall' to='dfall+period'

meas tran v0_rms rms i(v0) from='dfall' to='dfall+period'
```

```
meas dc is_at FIND i(vs) AT=1

meas dc is_max max i(vs) from=0 to=3.5

meas dc vds_at when i(vs)=0.01

meas ac vout_at FIND v(out) AT=1MEG

meas ac vout_atd FIND vdb (out) AT=1MEG

meas ac vout_max max v(out) from=1 k to=10MEG

meas ac freq_at when v(out)=0.1

meas ac vout_diff trig v(out) val=01 rise=1 targ v(out) val=01 fall=1

meas ac fixed_diff trig AT=10k targ v(out) val=0.1 rise=1

meas ac vout_avg avg v(out) from=10k to=1MEG

meas ac vout_integ integ v(out) from=20k to=500k

meas ac freq_at2 when v(out)=01 fall=LAST

meas ac vout_rms rms v(out) from=10 to=1G
```

The following examples illustrate some of the measurements that can be made this way.

Measuring WaveForm parameters 01

Measuring WaveForm parameters 02

Measuring settling time

Find gain and bandwidth

Essential Check

Introduction

By following - and constantly checking against - a set of procedures, it is possible to avoid just about all of the common mistakes and omissions that can significantly delay or even stop a schematic being successfully converted to a PCB and then that PCB being successfully updated from the schematic as a design progresses.

It can also significantly reduce the likelihood of a PCB being made that subsequently is found to not work correctly due to mistakes made during the creation of the original schematic (Schematic Capture).

After spending hours on Schematic Capture, it is very frustrating to be presented with error messages about prefix conflicts, missing or invalid packages when first attempting to pass a schematic through to the PCB Editor by clicking on the **Convert Project to PCB…** button or, after making changes to a schematic, similar error messages or having components that disappear from the PCB when attempting to update an existing PCB using the **Update PCB…** button in the Schematic Editor or the **Import Changes…** button in the PCB Editor.

These issues can be avoided by running through a series of checks for the first time each new Part (i.e. the first instance) is placed into the schematic.

There are several other issues that arise from mistakes in and omissions from the schematic that people encounter only after they are part way through a PCB design or - worse still - only when they receive their PCBs in the post.

Almost all of these other issues can be avoided by running through a further series of checks (i) during Schematic Capture, (ii) once Schematic Capture is complete but before first attempting to convert the schematic into a PCB and (iii) when updating the PCB as work progresses.

This document pulls together all the essential procedures to follow and things to check in the schematic before clicking on the **Convert Project to PCB…**, **Update PCB…** or **Import Changes…** buttons.

Things to understand before using this document.

Before using the document it is important that the following points are clearly understood:

What constitutes a Part in a schematic and a PCB?

A Part is any element of the circuit that is to be mounted on the PCB plus any element which is ultimately intended to be mounted on or form an integral part of the PCB such as heat sinks, PCB mounting holes, mounting holes for PCB mounted potentiometers and switches (for example where a PCB is used as a front panel or as a self-contained test jig), test points, wire links or jumpers, fuse holders and even image based elements such as high voltage warnings and logos.

Note that fuses that are fitted into PCB mounted fuse holders are best dealt with in a schematic by showing the fuse using a fuse symbol in the schematic but assigning to that fuse symbol the BoM information - including the package - that is for the required fuse holder. The required fuse ratings, type and supplier information can then be included in the BoM using Add new parameter function.

Other socketed devices can be treated in the same way.

What is the relationship between Parts, Schematic Symbols and PCB Packages?

Any Part must have a Schematic Symbol to represent it in the schematic (a.k.a. Schematic Lib) and that Schematic Symbol must have a PCB Package (a.k.a. PCB Lib) assigned to it either when the symbol is created or after placing the first instance of it into the schematic.

The associated PCB Package must exist in the library.

- Ensuring that every Part has a Schematic Symbol with a PCB Package associated with it **and** that that PCB Package actually exists in the Parts (SHIFT+F) library will avoid a **Missing package** error being issued on Conversion or Update to PCB.

It is possible that a component may comprise more than one device in a package, for example logic gates. Some symbols represent both devices in a single symbol but quite often a separate symbol is used to represent each of the devices. This may mean that some of the pin numbers and/or names on the symbols representing each of the two devices may be different although both may have the same power and ground pin numbers and/or names.

High pin count devices such as processors and FPGAs may be split into several symbols representing different sections or ports. It is important to ensure that pin numbers and names are unique across all the symbols.

It is possible that a component may be available in different packages. For example the LM358-N dual operational amplifier is available in several different packages. The pin numbering and/or naming of the symbol may be different depending on which package the component is supplied in.

It is easy in EasyEDA to change the pin numbering and/or naming for a Schematic Symbol (using the `I` Hotkey) or a PCB Package so it may be tempting to think of an LM358-N as the same part in a different package and just put down a symbol, edit the package assigned to it and then hack the pin numbers and names about until they match the PCB package.

However, an LM358-N in a SOIC-8 package has a different pinout, a different part number and has to be physically ordered as a different part from an LM358-N in a DSBGA-8 package.

- They are therefore two *different* Parts.

When thought of like this it should be clear that there should be one Schematic Symbol (or pair if each device has a separate symbol) and a matching PCB Package for an LM358-N in a SOIC-8 package and another Schematic Symbol (or pair if each device has a separate symbol) and a matching PCB Package for an LM358-N in a DSBGA-8 package.

- Ensuring that the pin numbers/names of the Schematic Symbol for a Part are correct, unique and match those of the PCB package associated *with that particular part* will avoid the generation of the **Invalid package** error being issued on Conversion or Update to PCB.

Why do Parts in a PCB disappear when the PCB is updated from the schematic?

It is important to understand that any Part that is supposed to form part of or be mounted on the PCB must have a corresponding Schematic Symbol in the schematic.

If it *does* then as soon as the PCB is created, the PCB Package for that Part , even such a seemingly abstract item as a mounting hole, warning sign or a logo, will be pulled into the PCB layout without having to be added to the PCB later by hand.

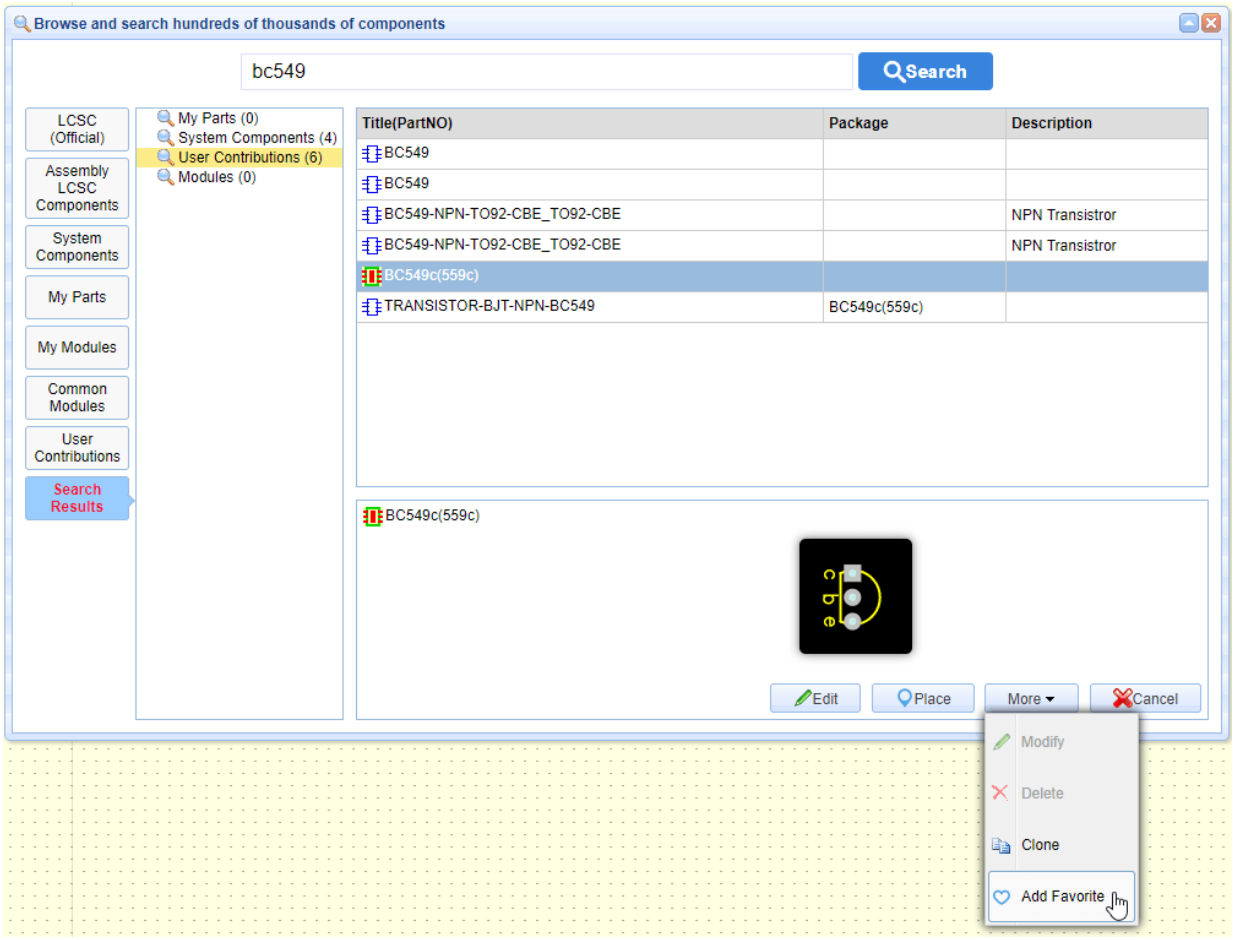
If it *does not* then not only will the PCB package for that Part not be pulled into the PCB layout as it is created but when it is added to the PCB later by hand and the PCB is then updated to bring in changes made to the original schematic, *that PCB package will be deleted*.

Such elements can be added to the schematic later and then imported into the PCB but if they do not exist in the schematic at the time the PCB is updated from that schematic then they will always be deleted and will therefore have to be added back to the PCB by hand.

What is the relationship between User Contributions and the other Parts categories (LCSC (Official), Assembly LCSC Components, System Components, My Parts, My Modules and Common Modules)?

Any Schematic Symbol or PCB Package chosen from the **User Contributions** category MUST be added to your local library by doing:

Parts (or SHIFT+F) > Search for and select the part then > More > Add Favorite



- Failure to do this will result in the package not being found and a **Missing package** error being issued on Conversion or Update to PCB.

Packages in the other library sections will be found automatically.

Note however that although Schematic Symbols and PCB packages created by a user within a Team will automatically appear in that user's My Parts library, once that user swaps to another Team, those parts will no longer appear in their My Parts library but will only be available via the **Add Favorite** option from the User Contributions library.

Procedures and Checklist

- Verify that the Schematic has been drawn to show clearly how the components in the circuit are connected together;

The schematic must help the reader understand signal and power flow in the circuit with inputs on the left, outputs on the right, positive supplies at the top, negative supplies at the bottom and netlabels used to clarify connections and reduce congestion.

Components may be grouped by function and boxes may be drawn around them.

Decoupling components may be drawn adjacent to the devices they are associated with or symbols with dedicated sub-parts for power pins can be used to reduce congestion;

- Use the Design Manager (left-click the **Design** button in the left hand panel) to check that all components are present in the schematic and that all nets have been assigned reasonable mnemonic names and have at least two connection.

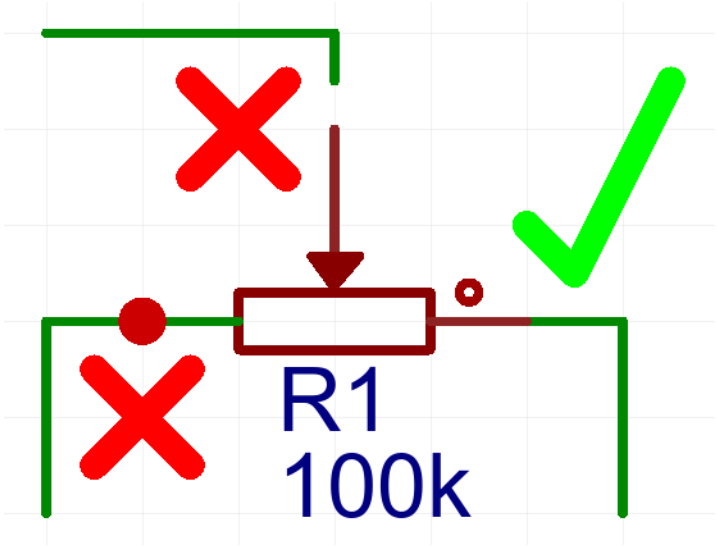
Naming nets instead of relying on the EasyEDA auto-generated alphanumeric names makes signal tracing and debugging the final PCB much easier but care must be taken to ensure that names are correct and that there are no unintended duplicate names or accidental increments in numbered nets;

- Verify that there are no duplicate prefixes in the schematic:

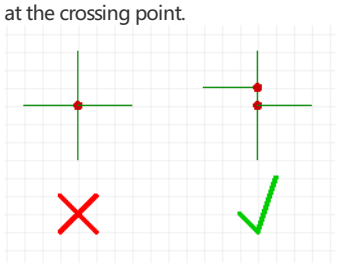
https://easyeda.com/forum/topic/How_to_resolve_quotPrefix_Conflictquot_error_-gpbca8642

Remember to check across all sheets of a multi-sheet schematic;

- Check that the schematic is drawn correctly.
  In particular, check that no nets have been accidentally cross connected, that wires have join dots where they are intended to be joined, that they are properly connected to component pins and that nets joined by netlabels are correctly named and that there are no unintended duplicate net names.



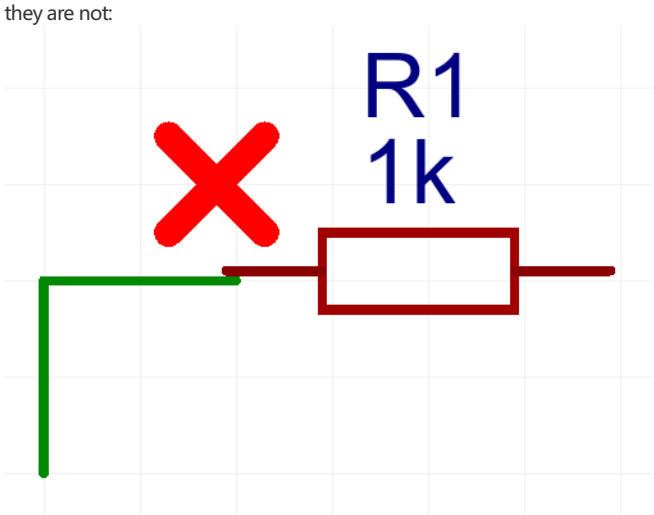
- Verify that junctions of 4 or more wires are drawn to show staggered junctions to avoid confusion with wires that cross but are not joined at the crossing point.



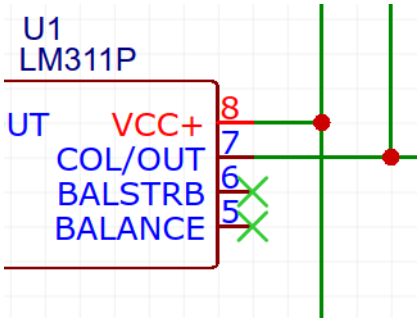
- Check that all parts and nets have been placed with the Canvas Attribute **Snap** = **Yes**:

Selected Objects	0
◢ Canvas Attributes	
Background	#FFFFFF
Visible Grid	Yes
Grid Color	#CCCCCC
Grid Style	line
Grid Size	10
Snap	Yes
Snap Size	10
ALT Snap	5

and that no parts have been placed off grid so that although they may appear to be connected on close inspection it can be seen that they are not:



- Check that pins have been terminated (pulled up, down, left open etc.) as specified in manufacturers' datasheets.
- Check that all unconnected pins have **No Connect** symbols attached directly to them. Unconnected pins without No Connect symbols attached directly to them will show up as alphanumeric net names in the Design Manager but will not highlight when clicked on in the Design Manager.
  No Connect symbols must be attached directly to component pins. There should be no wire between the No Connect symbol and the pin.



Note that the No Connect symbol changed in V4.8.5 of EasyEDA from a red cross to a green one to make the highlighted state of a selected symbol clear;

- Check that the device ratings are suitable for the circuit in which they are to be used. For example, capacitor, diode, transistor, connector and switch voltage ratings, transistor, resistor and zener diode power dissipations, inductor, diode (including LED), transistor, connector and switch current ratings.

Although these parameters should have been checked at the time of specifying the components as an essential part of the circuit design stage prior to or during Schematic Capture, there is plenty of scope for them to have gone astray during the part selection, placement and editing steps of Schematic Capture.

An undetected mistake now can result in the wrong size part being chosen. For example a larger diameter or even a taller electrolytic capacitor may be needed. Whilst this is easy to correct in the PCB design stage, at best this may waste time in having to redesign part of the PCB. At worst the mistake may not be discovered before the PCB design is completed and sent for manufacturing.

- Check that diode (including LED) and bipolar transistor base-emitter junction reverse breakdown voltage ratings and that input differential and common mode voltage ratings of operational amplifiers and comparators are not exceeded during any state of operation of the circuit including power up and power down.

Consider adding diode or MOSFET reverse supply protection especially for battery powered circuits.

An example of MOSFET reverse protection is described in:

https://easyeda.com/example/Uberclamp*Schematic*PCB*and*BoM-r4YgysK2k

Pay special attention to this in operational amplifier or comparator devices that exhibit `output phase reversal` under some input conditions. For more information about this see:

http://www.analog.com/media/en/training-seminars/tutorials/MT-036.pdf

For example, the TL081 exhibits this behaviour but it is not documented in more recent versions of the datasheet. See Applications Hints on page 5 of this earlier version:

http://www.physics.ucc.ie/fpetersweb/FrankWeb/courses/PY2108/spec%20sheets/TL081%20OpAmp.pdf

Consider adding diode or MOSFET reverse supply protection especially for battery powered circuits.

An example of MOSFET reverse protection is described in:

https://easyeda.com/example/Uberclamp*Schematic*PCB*and*BoM-r4YgysK2k

- Check that LED currents are supplied through series current limiting resistors or from constant current sources.

For background on this please see:

https://easyeda.com/andyfierman/LEDs*must*have*series*resistors-OoGYgCK2k

- Check that signal connectors have sufficient ground pins to maintain signal integrity by minimising signal return path impedances (i.e. ground loop area). This is especially important in designs with high speed signals through the connectors but can also be important for lower speed signalling with long wire interconnects and/or fast edge speeds.
- Check that power connectors have sufficient ground and power pins pins to maintain power integrity by minimising power and ground return path impedances.
- Verify that device power supply decoupling complies with manufacturers' recommendations.

Where possible, check datasheets, applications notes and schematics and PCBs for Reference Designs or Evaluation Boards.

For some background on the importance of adequate decoupling please see:

https://easyeda.com/andyfierman/Power_supply_decoupling_and_why_it_matters_-451e18a0d36b4f208394b2a2ff7642c9

- Verify that a Schematic Symbol and an associated PCB Package has been created for every Part needed to construct the complete PCB.

Remember to include Schematic Symbols and an associated PCB Packages for things like heat sinks, PCB mounting holes, mounting holes for PCB mounted potentiometers and switches (for example where a PCB is used as a front panel or as a self-contained test jig), test points, wire links or jumpers, fuse holders and even image based elements such as high voltage warnings and logos;

Look in:

Parts (or SHIFT+F) > MyParts > Schematic Lib > Favorite Schematic Lib

and:

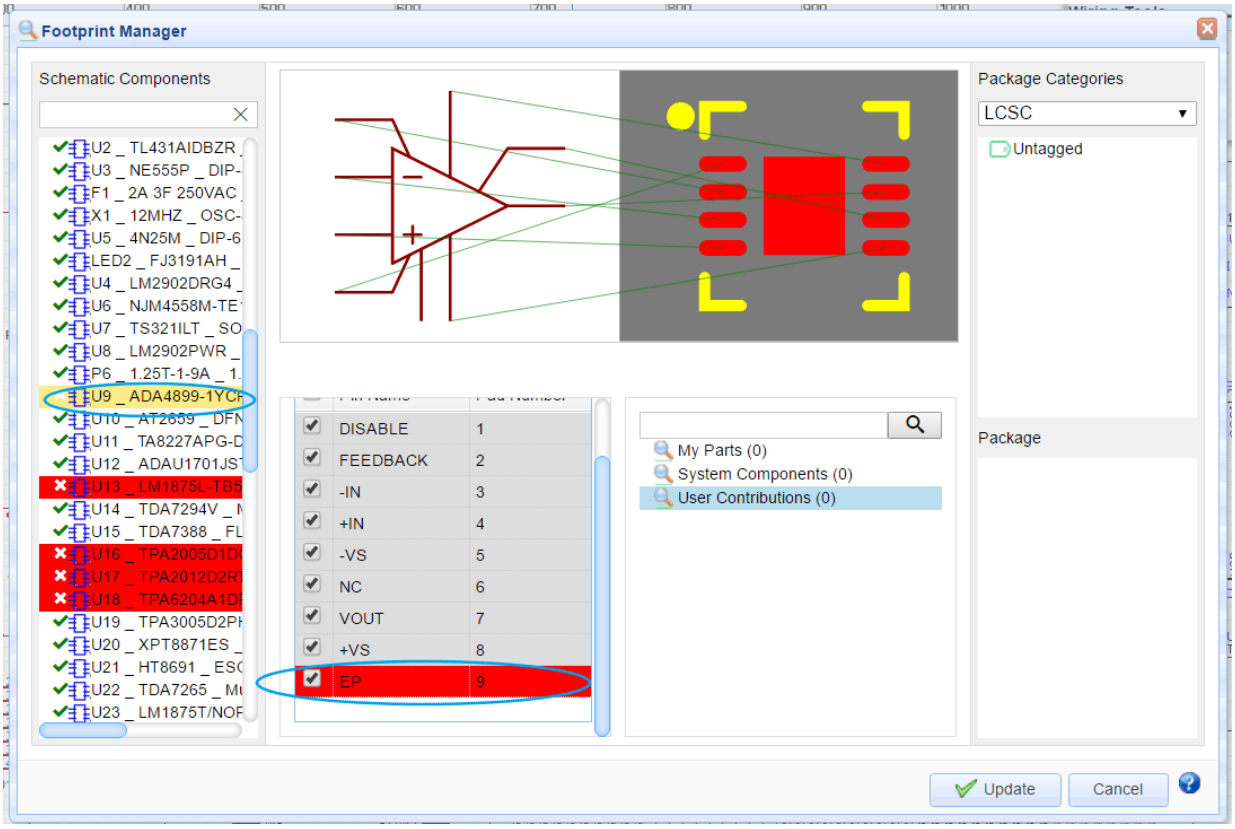
Parts (or SHIFT+F) > MyParts > Package > Favorite Package

and verify that every Schematic Symbol and associated PCB Package chosen from the **User Contributions** category has been added to your local library;

- Verify that the pin numbers/names of the Schematic Symbol(s) for every Part are correct, unique and match those of the PCB package associated *with that particular part*;
- Verify that the pin order (pin mapping) of the PCB Package associated with every part is correct.

This task is simplified using the EasyEDA Footprint Manager:

https://easyeda.com/Doc/Tutorial/Schematic.htm#Footprint-Manager



Remember that in EasyEDA, the PCB Footprint is viewed looking down onto the component side of the board. This view is assumed to be with all components mounted on the Top Layer. Packages can subsequently be placed on the top or bottom layers as required.

- Verify that all necessary information about the specific components (and any suitable alternatives) that are to be used in the circuit and which are ultimately intended to be mounted on or form an integral part of the PCB, from which a Bill of Materials (BoM) can be generated has been added to the Schematic Symbols.

For more information about this please see:

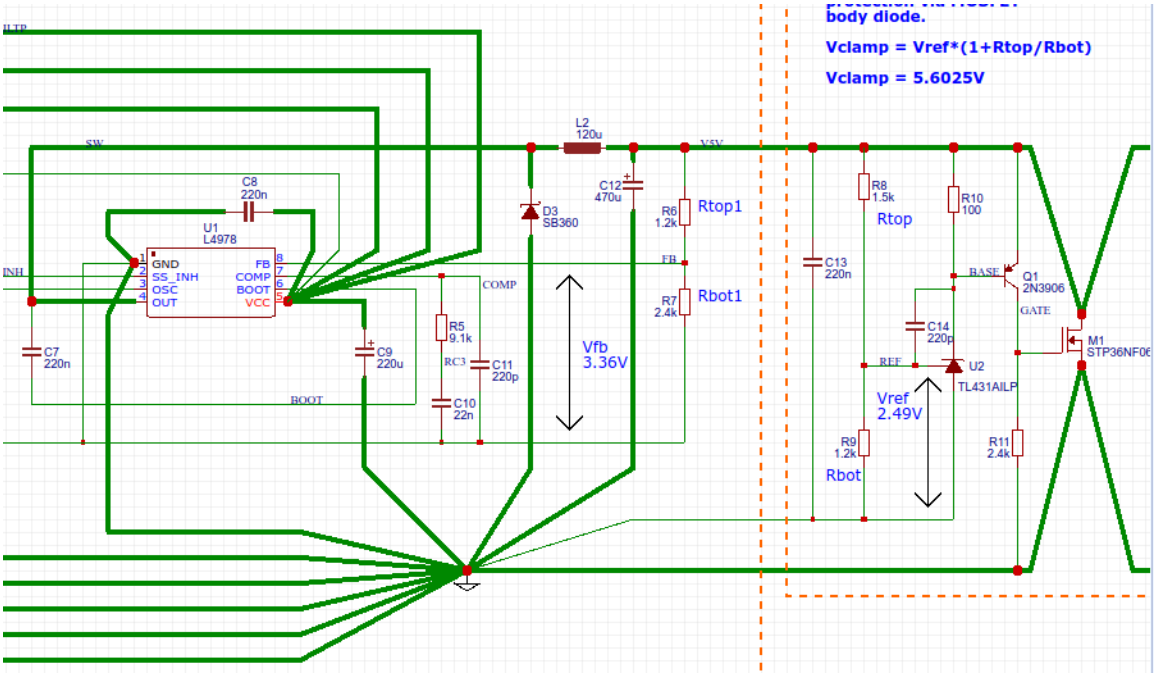
https://easyeda.com/forum/topic/How_to_add_extra_information_to_the_Bill_of_Materials_BOM-Hp9rJCUcu

- Verify that any necessary information relating to the physical placement of components and layout of copper traces and areas has been annotated in the schematic.

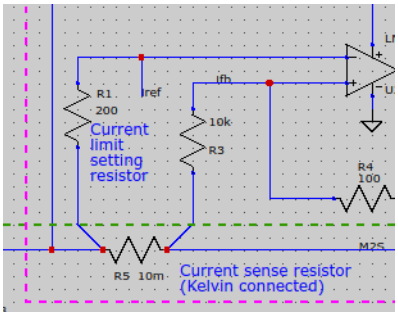
Including text (or even diagrammatic information in the schematic about component positioning and orientation, clearances around heatsinks for airflow or copper areas for heat sinking, current and voltage ratings of traces, trace length matching, controlled impedance transmission lines and differential pairing can all help in the following stages of PCB design.

Nets that are carrying high currents may be drawn using thicker wires (Stroke width).

Nets can be drawn converging at star points to help illustrate where this type of PCB layout is required on the PCB:



Kelvin connections to current sense resistors can be drawn in a similar way:



Nets can be colour coded but beware using red because it can be very hard to see when such nets are highlighted.

- Generate - and check - the Bill of Materials (BoM) information and check the availability of components.

Whilst it is easy to change parts in the PCB design stage, at best this may waste time in having to redesign part of the PCB. At worst the unavailability of a part may not be discovered before the PCB design is completed and sent for manufacturing.

- Use the Design Manager (left-click the **Design** button in the left hand panel) to check everything again!

FAQ

Please spend a few minutes reading this FAQ, it will save you lots of time getting started with EasyEDA.

Hot Question

How to find the list of hotkeys.

https://easyeda.com/Doc/Tutorial/Introduction.htm#Hotkeys

Where are my files?

Your files are stored on EasyEDA servers, so you can access them anywhere and share them with your partners.

Why does EasyEDA focus on Cloud based EDA?

EasyEDA is built for people who like to work anywhere, who like to build projects together with other team members, who like to share their projects, who like something that operates like a github for hardware design. The only way to meet these needs is to build a Cloud version EDA.

How can I work if there is no internet?

Although most of the time there are ways to access the internet easily and cheaply there may be times when, for whatever the reason, internet access is simply not possible. For times like this, EasyEDA is working to provide a desktop client soon.

Does EasyEDA have a desktop version?

At present, no but EasyEDA is developing and testing a desktop version to be introduced soon.

A Windows version will be available at the end of this year. Mac and Linux versions will be available early next year.

Which Browser is best for EasyEDA?

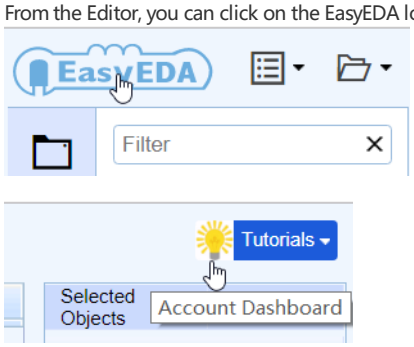
Chrome. Firefox and Safari are OK too. If you are restricted to using other browsers, it would be better to download the EasyEDA desktop client when it becomes available (see above).

How to go to your dashboard.

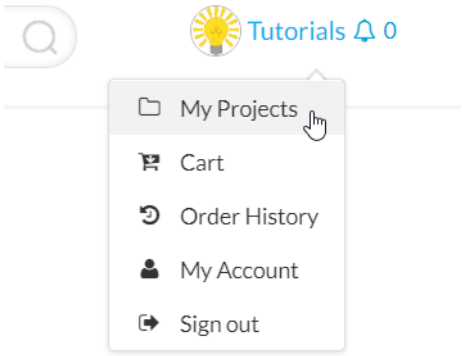
In the Dashboard, you can check all your Projects, Modules, Components and Favorites, projects others have shared with you, forum posts and orders.

There are two ways to arrive there.

1. From the Editor, you can click on the EasyEDA logo or user logo:



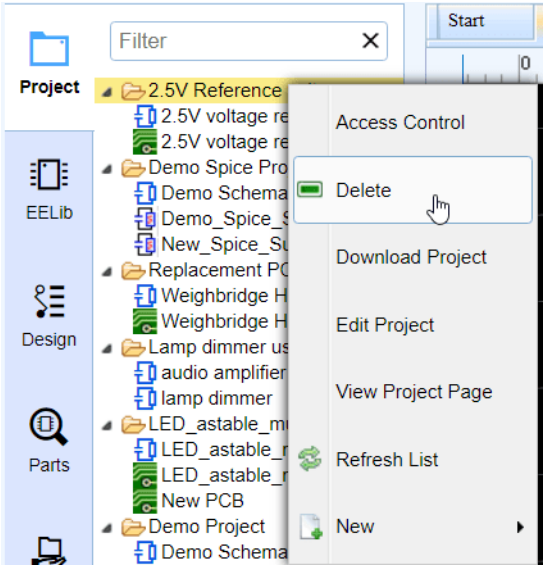
2. From the homepage, you can click My Projects:



Projects and Files

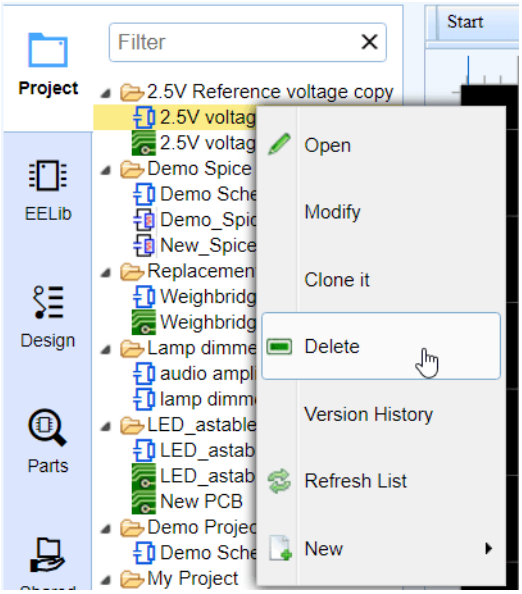
How to delete a project.

Select it and right click to open a context menu, like the image below.



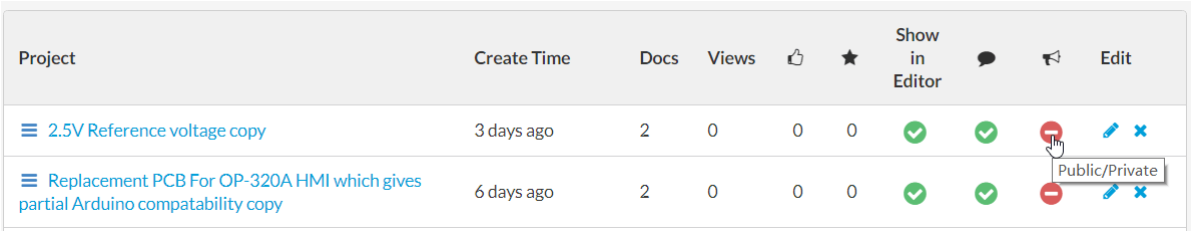
How to delete a schematic or PCB.

Select it and right click to open a context menu, like the image below.



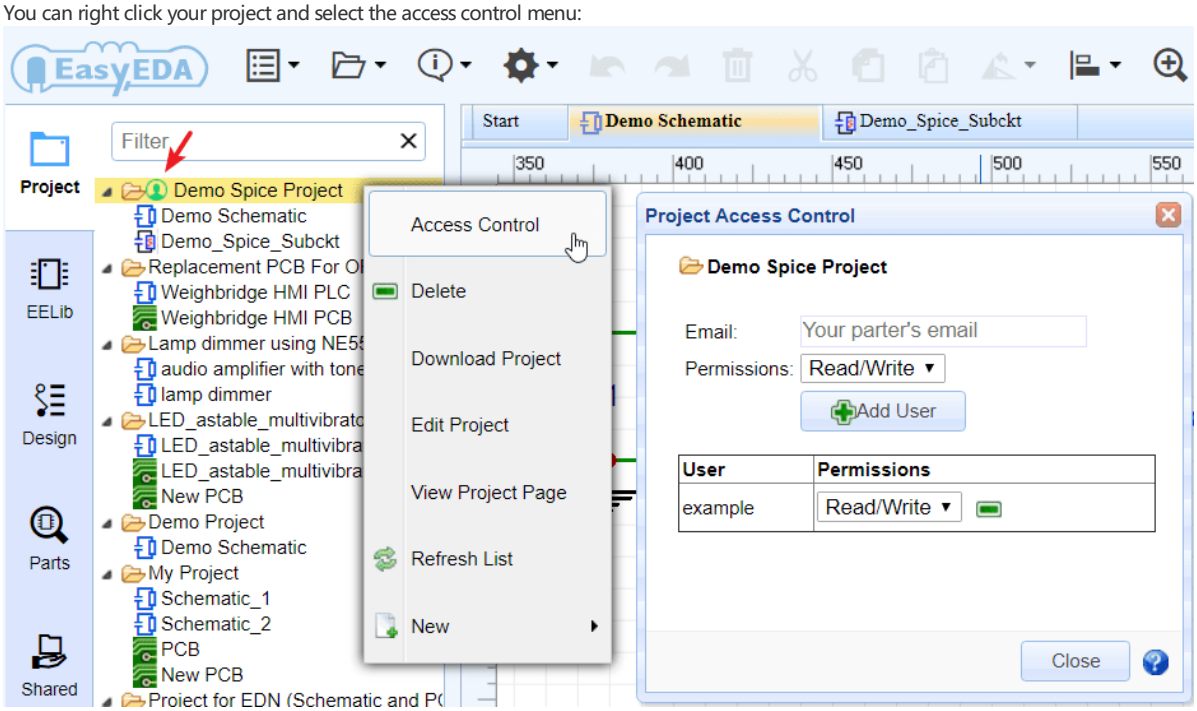
How to share a project with others.

1. Make your project public.
   Open https://easyeda.com/projects/mylists, then click the red `No entry' icon where indicated by the arrows. This icon will change to a green 'Tick' icon to show that the project is now public.



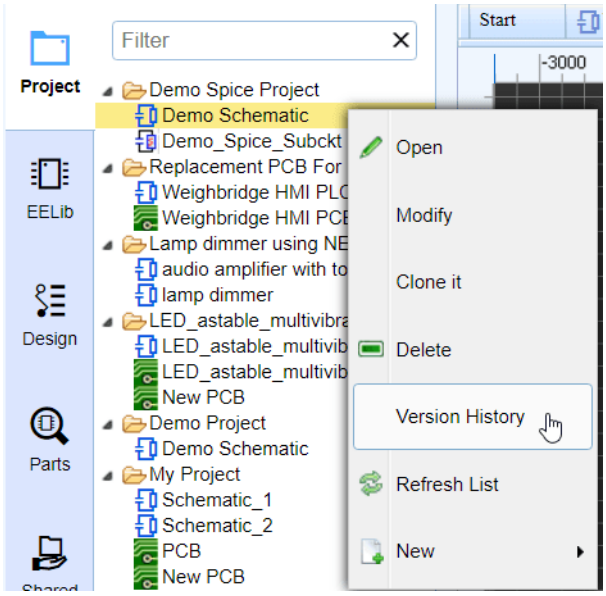
Project	Create Time	Docs	Views			Show in Editor			Edit
2.5V Reference voltage copy	3 days ago	2	0	0	0				
Replacement PCB For OP-320A HMI which gives partial Arduino compatability copy	6 days ago	2	0	0	0				

2. To share a project privately with only selected collaborators via:
   Access Control
   You can right click your project and select the access control menu:



How to find the version history of schematics and PCBs.

The version history of your EasyEDA schematics and PCBs can be accessed by right-clicking on the file you wish to query to open the context menu as shown in the image below:



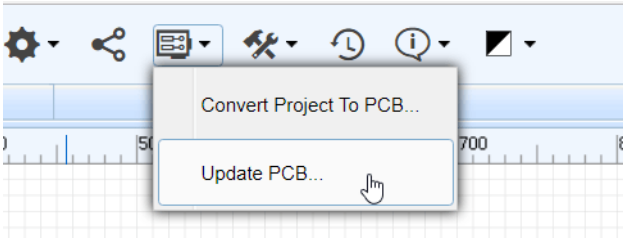
Then click on the version number that you wish to view.

Note: *saving a previous version will restore that version to being the current version of the file.*

Schematic

If I update the schematic, how do I then update the PCB?

The initial conversion of a schematic to PCB is done from within the Schematic Editor using the `Convert Project to PCB...` button as illustrated in the toolbar below but a new `Update PCB` button has been added so that modifications to the schematic can immediately be passed forward to update a selected PCB without having the PCB editor window already open.

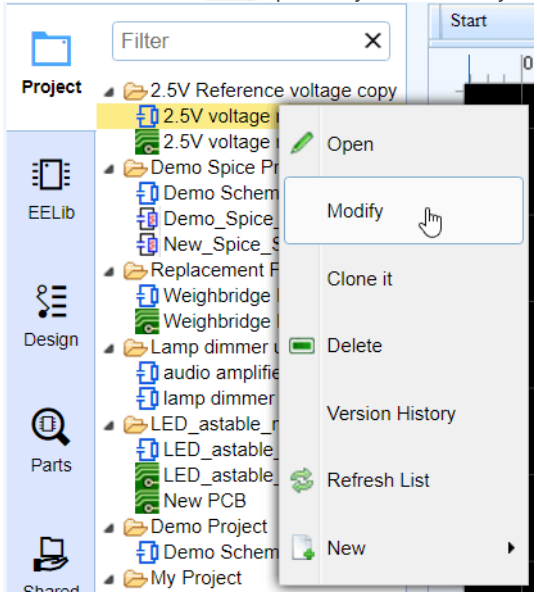


Alternatively, you can import changes from the schematic from within the PCB Editor:

https://easyeda.com/Doc/Tutorial/PCB.htm#Import-Changes

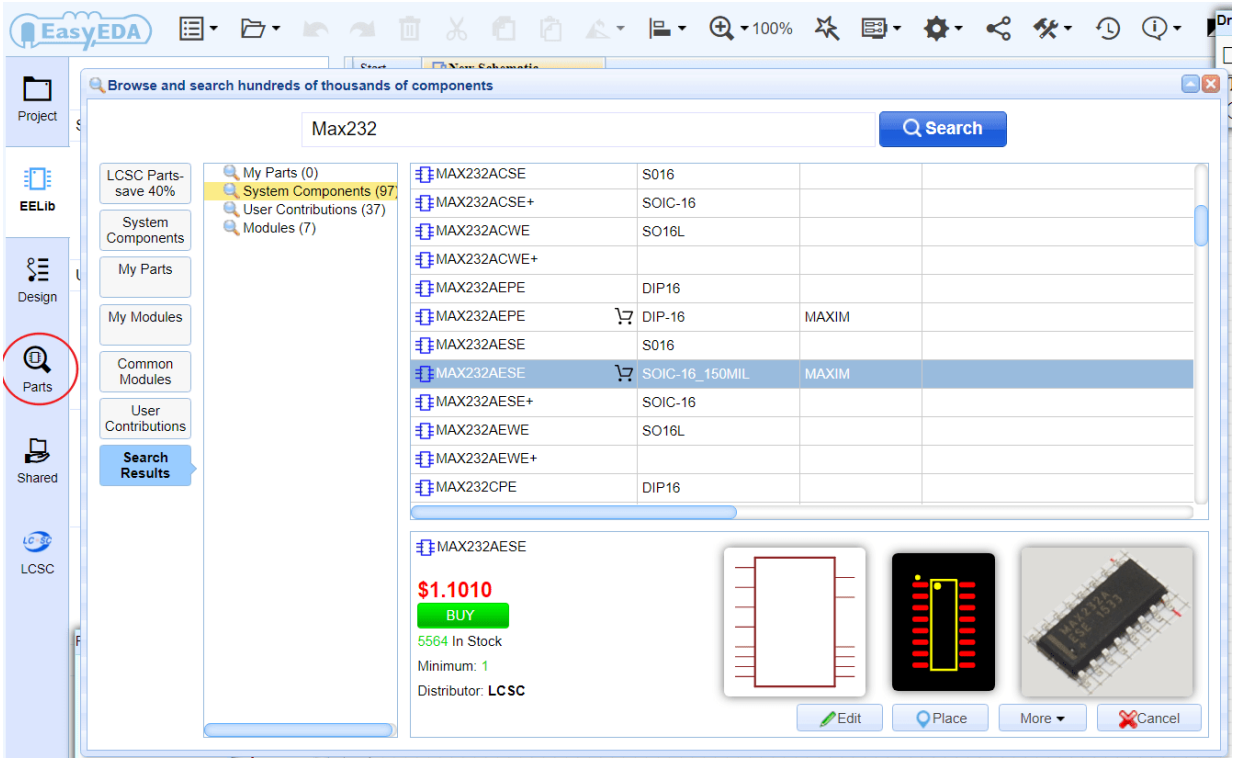
How to rename a Sheet/Document or modify description.

In this menu, there is a `Modify` option, so you can rename your files.



How to find components

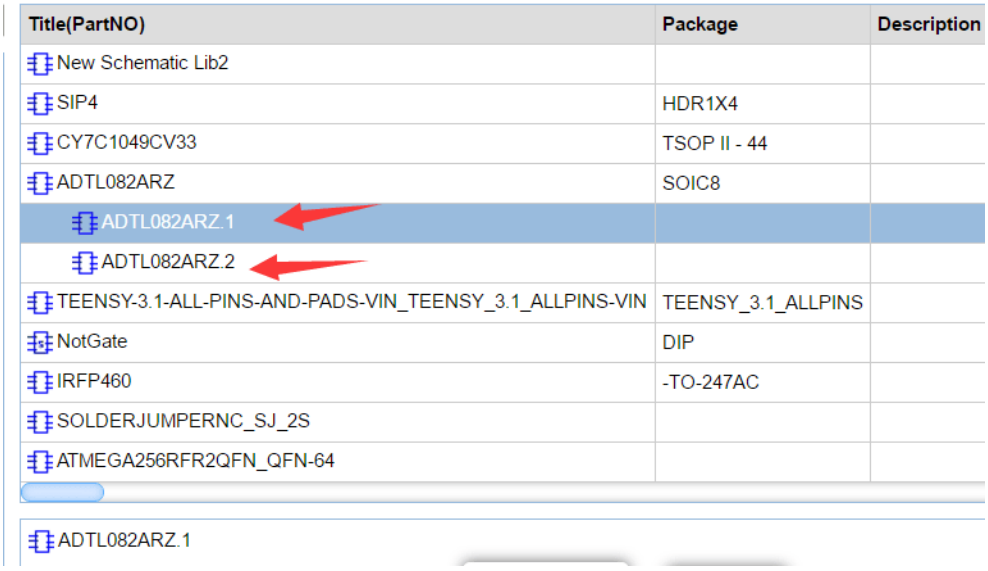
The component search function has been significantly improved to make finding part symbols and footprints quicker and easier. Press `SHIFT+F` or click on the `Parts` icon on the left navigation panel:



In the new components dialog, it is easy to select the right components via tags and you can set tags for your own components.

How to add sub parts to a schematic.

You can add sub parts to a schematic one by one but please note that the sub parts prefix must be in the form of U1.1 U1.2 etc, and not U1.A U1.B.



What is the unit of the schematic sheet?

The basic unit of the schematic sheet is the pixel. 1 pixel is about 10mil (0.001 inch) but please note that this use of the pixels as a unit in a schematic is just for reference.

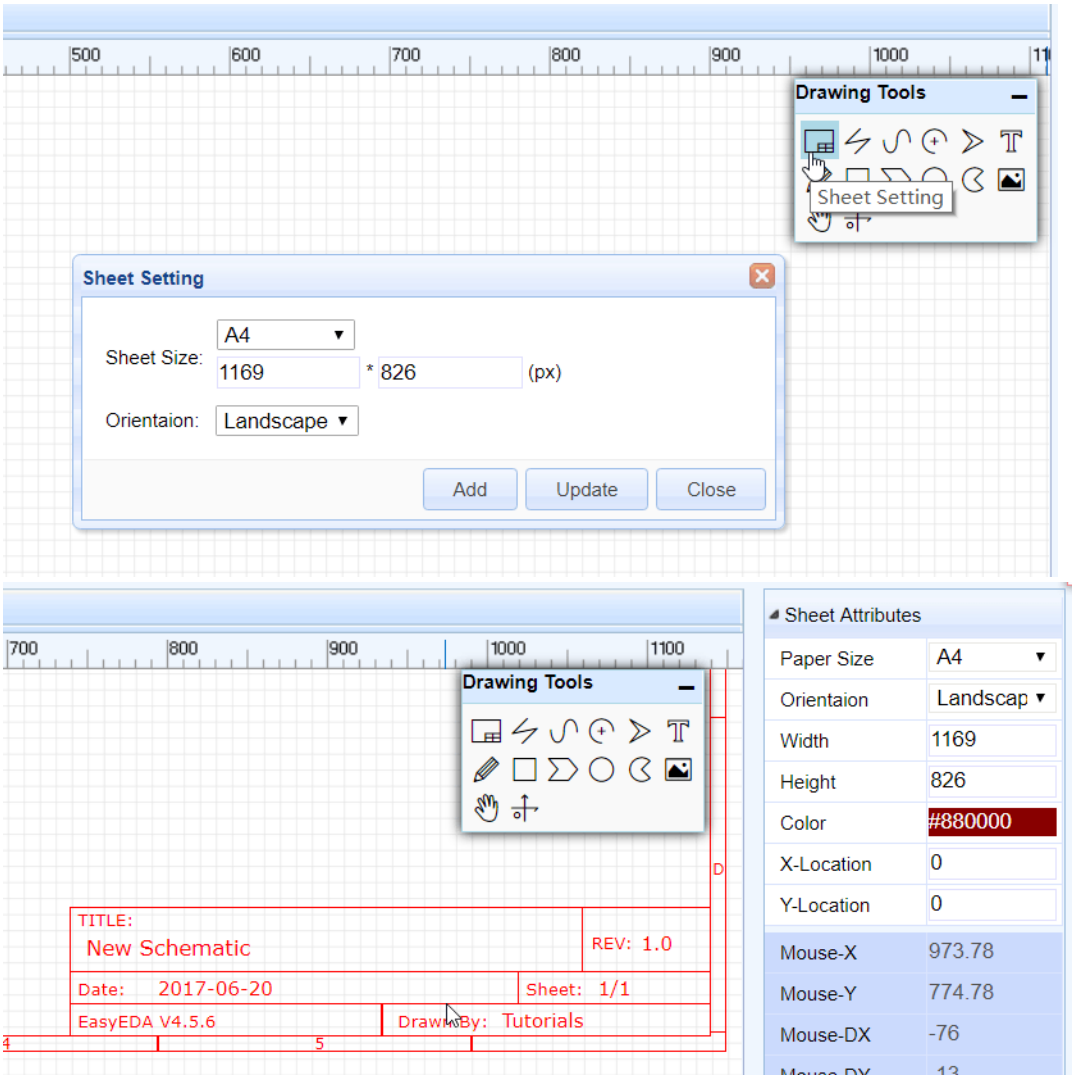
For a complex project, I want to split the schematic over several sheets. Does EasyEDA support hierarchy?

Please check out this link https://easyeda.com/Doc/Tutorial/Schematic.htm#Hierarchy

How to change the sheet size and modify the design information.

To change the sheet size, move the mouse anywhere over the lower right area of the drawing border or frame until the whole border highlights red and then right-click on it. Paper size and orientation can then be changed in `Sheet Attributes` in the right hand panel.

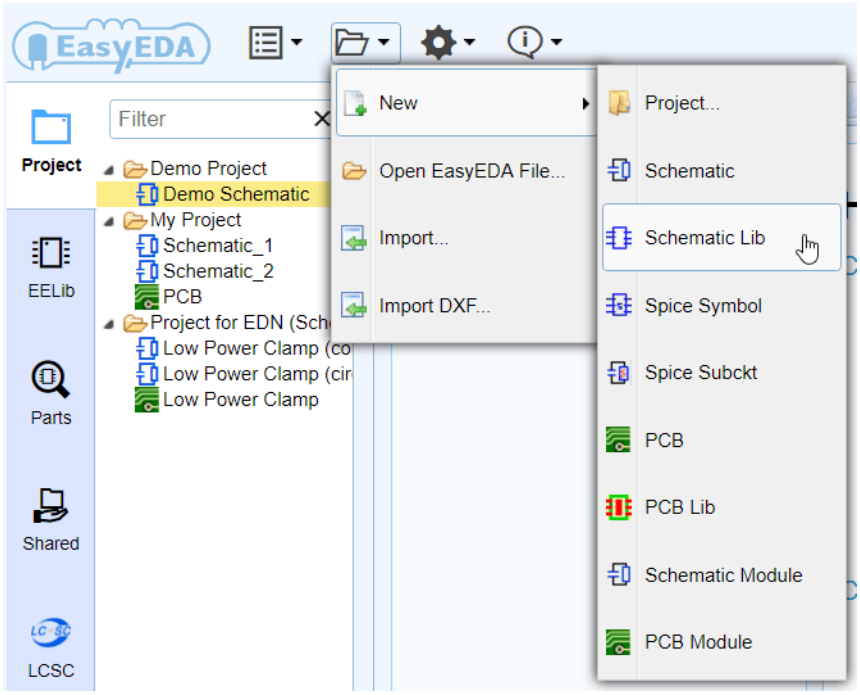
To modify the design information, left-click on the relevant blue text in the lower right area of the drawing border or frame to change it in `Text Attributes` in the right hand panel. Double left-clicking the blue text will allow you to type new information directly into the field.



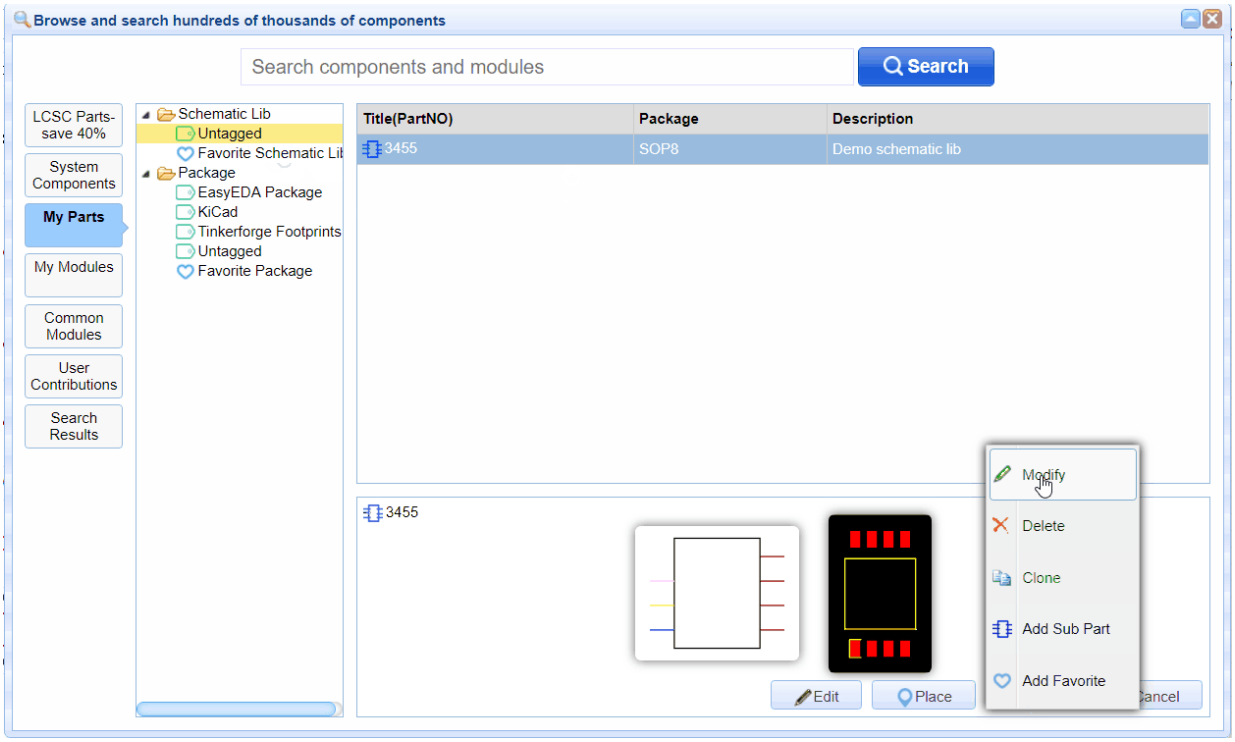
Schematic library symbol

How to create a schematic library symbol.

Document > New > Schematic Lib

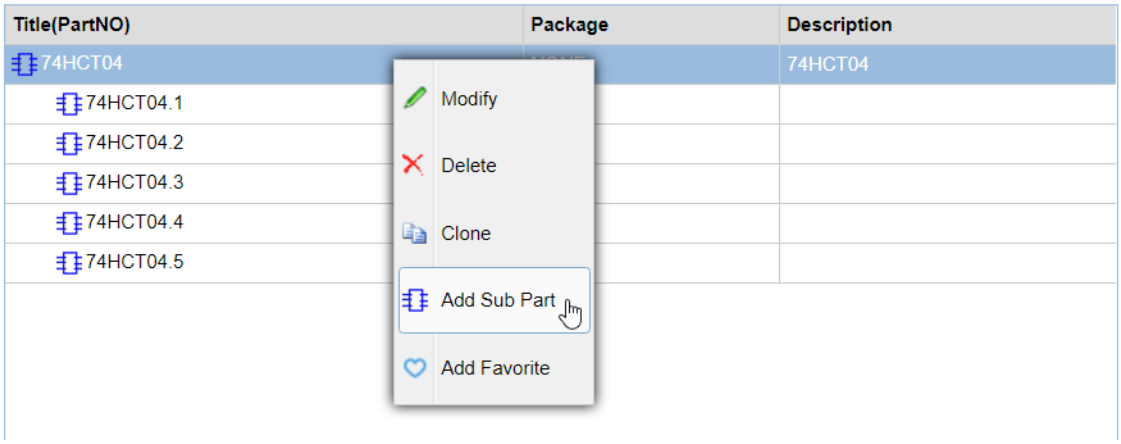


How to tag my schematic library symbol.



How to create sub parts for multi-part components.

In My Parts, Right click the part then select **Add Sub Part** from the menu that opens:



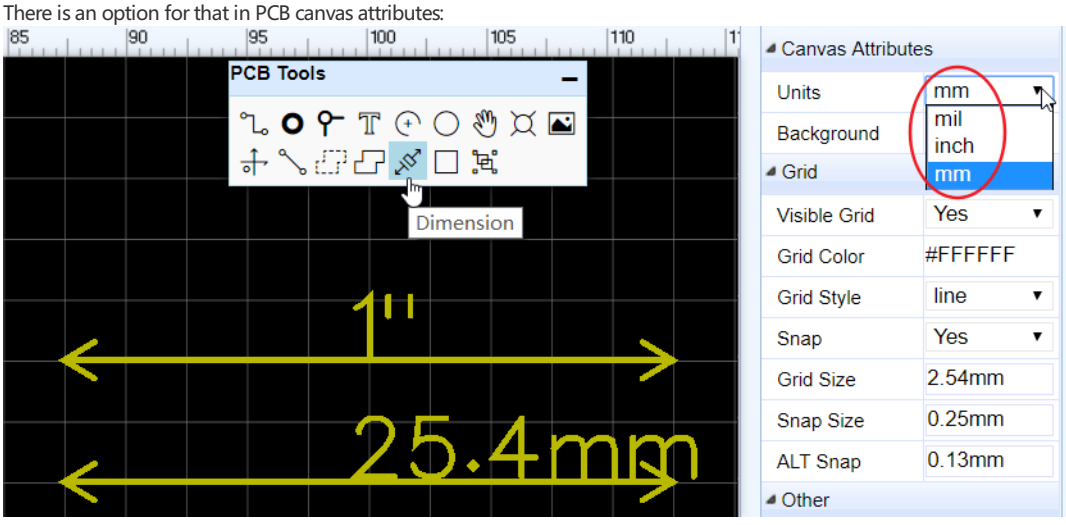
How to change the Package for a component.

https://easyeda.com/Doc/Tutorial/Schematic.htm#Update-Package

PCB

How to change the Units of PCB from mil to mm or inch.

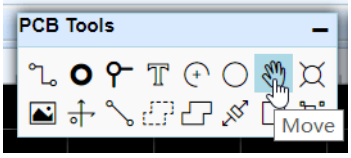
There is an option for that in PCB canvas attributes:



How to pick and move the components on the PCB canvas quickly.

Before routing the PCB, the components need to be positioned in suitable places on the PCB. In the PCB Editor, it can sometimes be quite

difficult to select components by clicking on the silkscreen outline or the pads. To select and move them more easily, please use drag mode (Hot Key `D`) or click the `Move` icon in the PCB Tools toolbar:



Can I create a PCB without creating schematic?

Yes but for any but the simplest PCBs, please see:

https://easyeda.com/forum/topic/The_best_way_to_design_a_PCB_in_EasyEDA-ThR3pwqlC

How to add more fonts for PCB.

You can refer to Text of PCB section.

How to insert an Image/Logo to PCB.

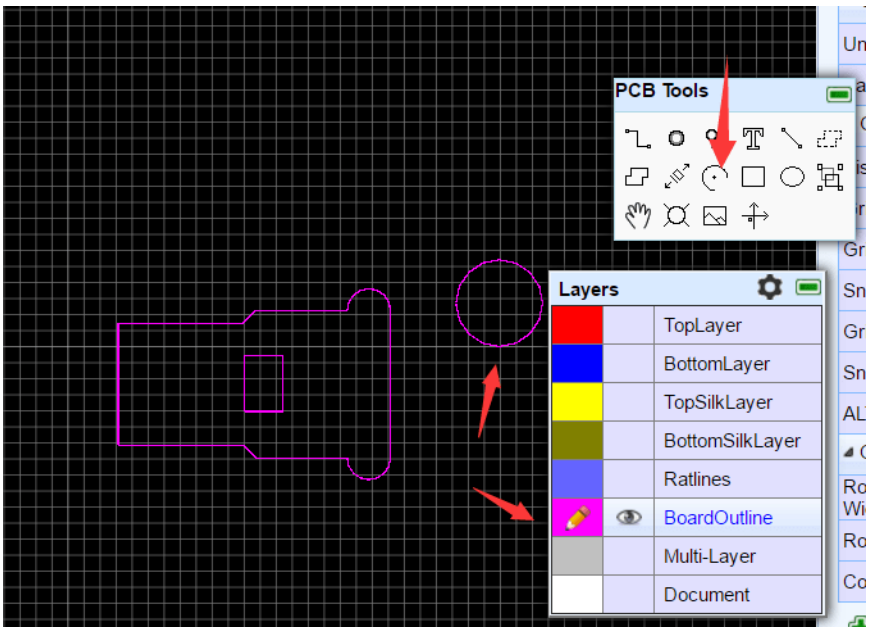
You can refer to Image of PCB section.

How to insert a DXF board outline.

You can refer to Import DXF File of Import section.

How to create non rectangular pcb outline such as round?

You can import a DXF file for the board outline. For a round board outline, you can use an arc to do that, you just need to change to the board outline layer, then draw 1 arc like in the image below (need to adjust a bit later), you can use lines and arcs to create complex board outlines.



How to add a slot and cut out.

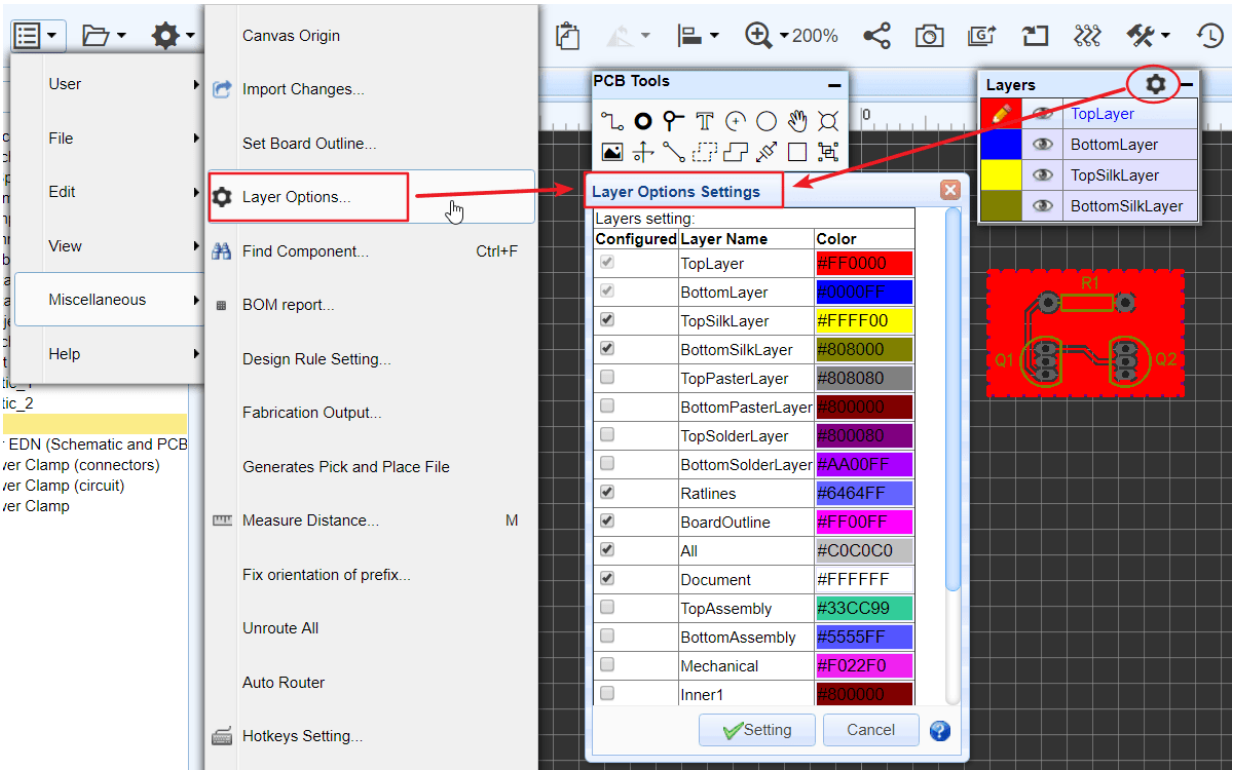
https://easyeda.com/Doc/Tutorial/PCB.htm#Pad and https://easyeda.com/Doc/Tutorial/PCB.htm#Solid-Region

How to measure dimensions on a PCB.

https://easyeda.com/Doc/Tutorial/PCB.htm#Measure-Dimension

How to add more layers.

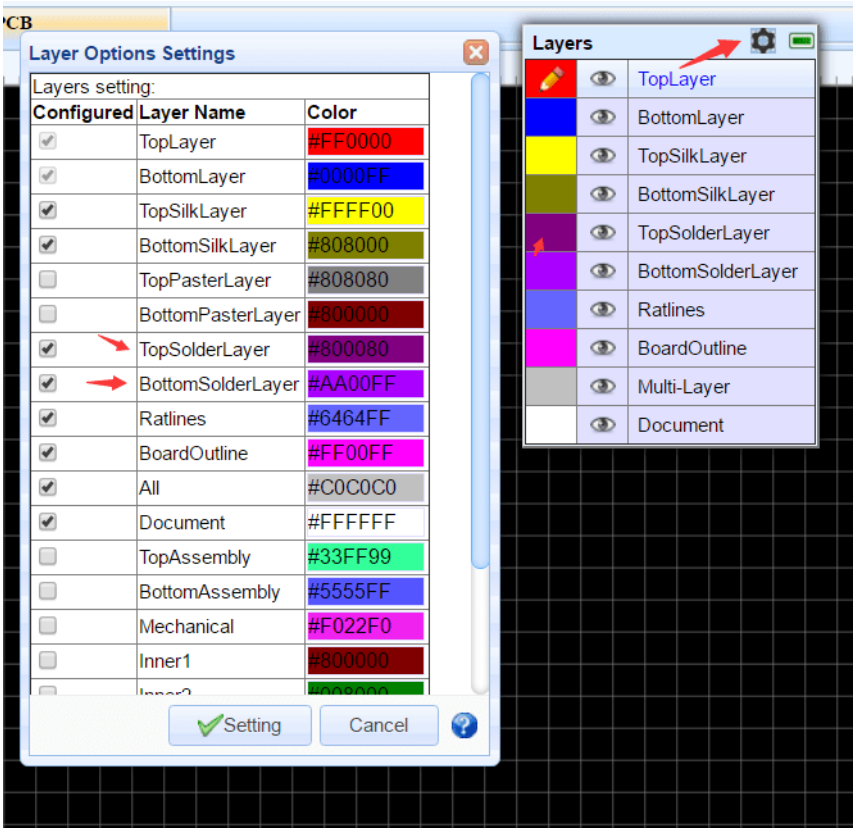
Click the layer options button, then tick the extra layers in the dialog that opens.



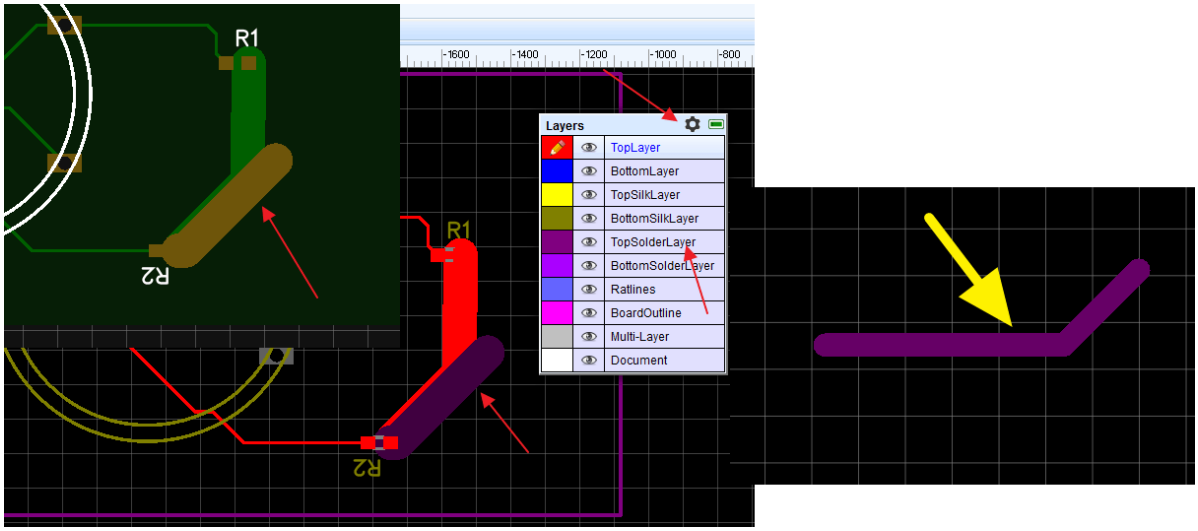
How to add solder mask.

It is possible to get boards with the copper exposed so that you can apply a layer of solder over those tracks to further increase their current carrying capacity. In this case, you need to add solder mask over a copper (copper area, track, solid region). EasyEDA will add solder mask for pads automatically. Sometimes however, you may need to add an aperture in the solder mask to expose and area of copper.

1. First, add a top or bottom solder mask layer, as required.



2. Next, draw a region in the solder mask layer over a copper item as illustrated in the image below. This in effect draws an aperture in the solder mask so that the copper item inside the region, in this case the track, will be not be covered by the green film of solder mask.



A common mistake is to just draw a solder mask, without a copper area, like the track pointed to by the yellow arrow. That is incorrect and does not produce the desired result.

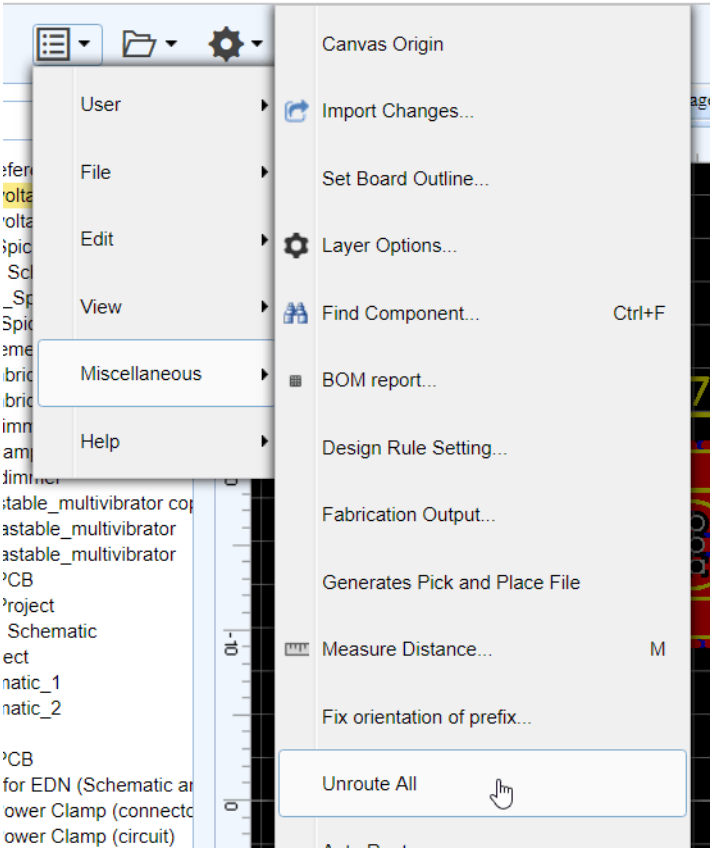
How do I set the dimensions of my PCB in the layout?

PCB's dimension/size depends on the board outline, you can create your board outline, please refer to Board Outline of the PCB section.

My PCB is complex, how can I be sure that I have routed all of the tracks?

Please refer to Design Manager of PCB section.

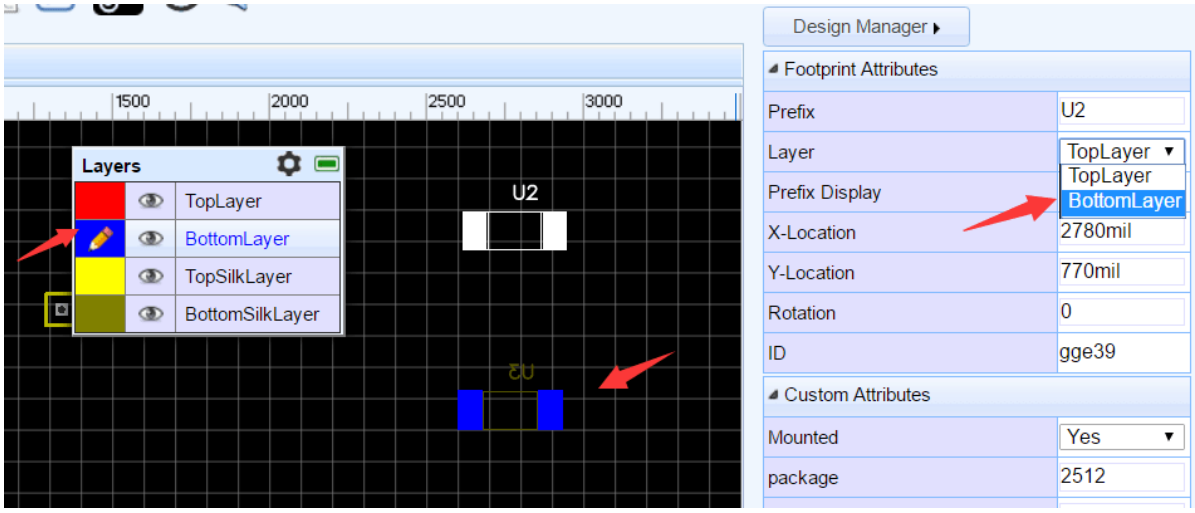
I need to start my layout again, how can I remove all of the tracks?



How to put a component on the bottom layer?

There are two ways to do this.

1. If your active layer is the bottom layer, then every component you place will be placed on the bottom layer automatically.
2. You can place a component then select it and change its layer attribute to `Bottom layer` in the right hand panel.

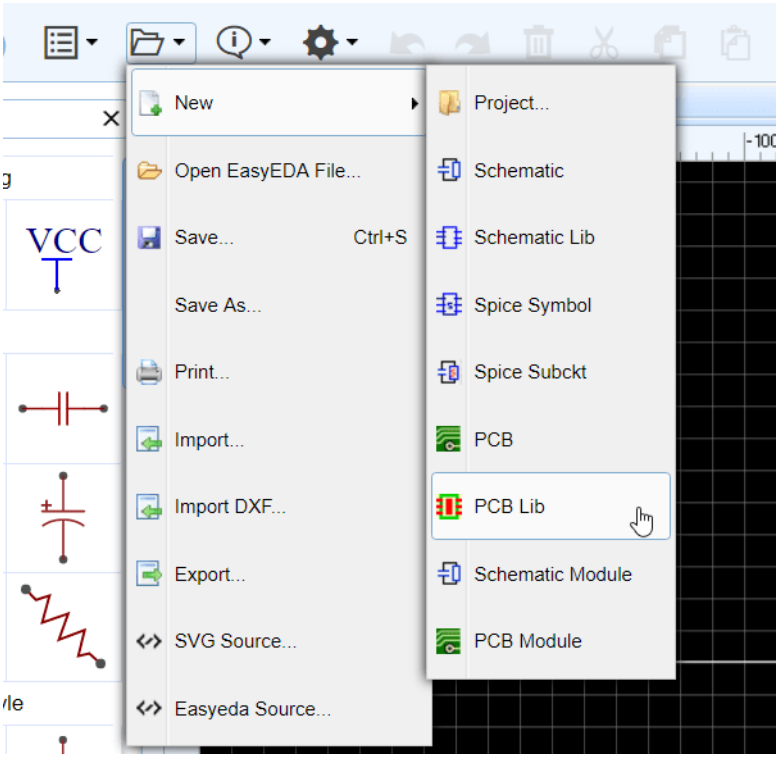


I can't convert schematic to PCB. Why is this?

1. You have not set the right packages for your components.
2. https://easyeda.com/Doc/Tutorial/Schematic.htm#Prefix-Conflict-Error
3. https://easyeda.com/Doc/Tutorial/PCB.htm#Invalid-Packages

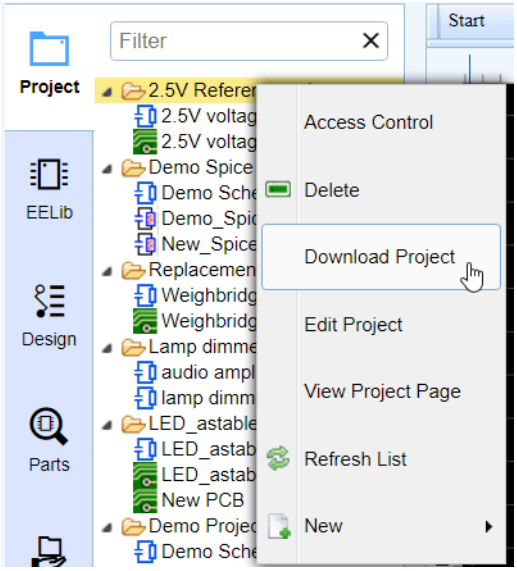
PCB package.

How to create a PCB package/library.



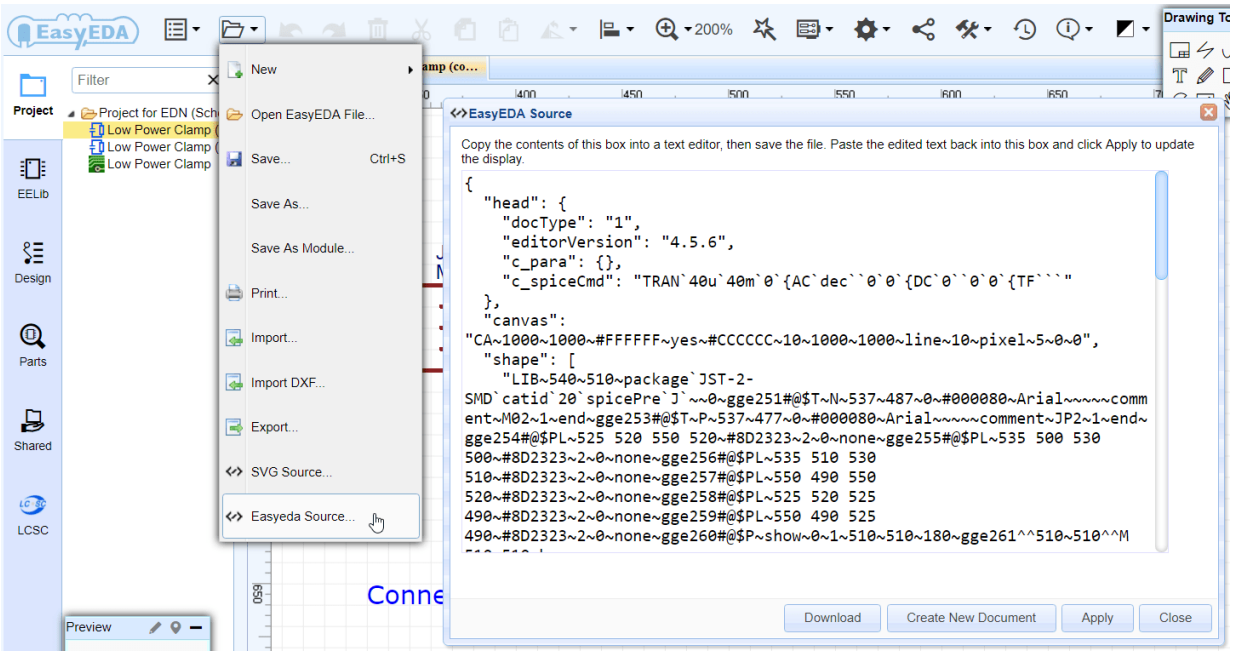
Keep in Mind

1. There is no need to back up your schematics and PCBs manually. After the first save of any file, EasyEDA will back up all saved files automatically under the Version History. If you want to back up your files locally, you can download a copy of the whole project or of individual files in a project in EasyEDA Source (JSON) format:



and;

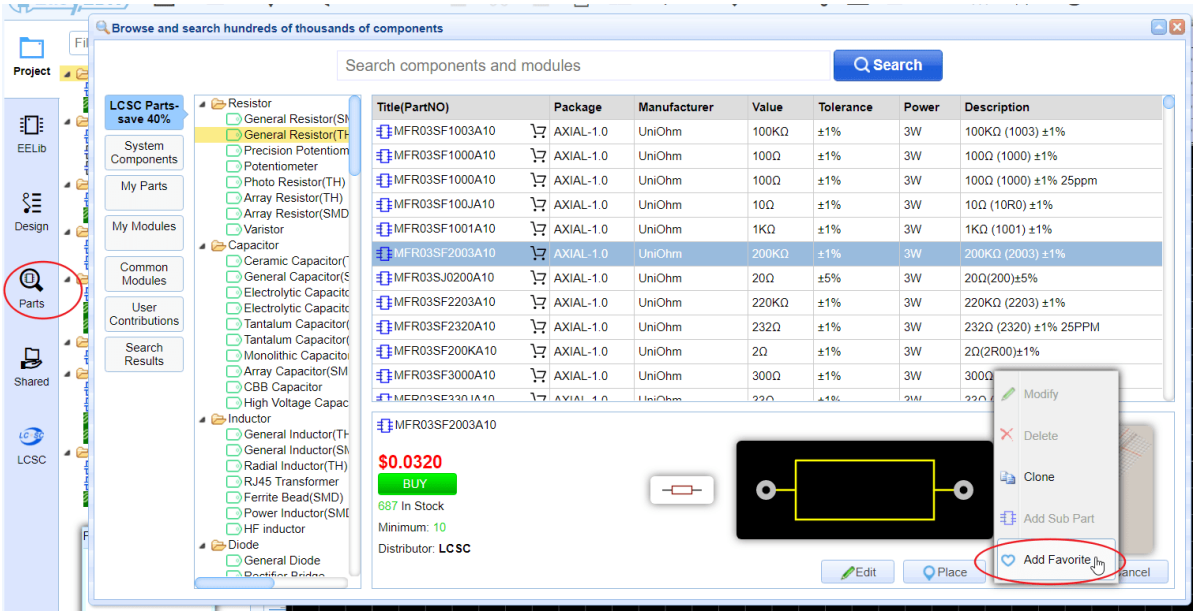
Document > EasyEDA Source > Download



2. If you need help, you can contact us or ask via our Support Forum; we will respond ASAP.

Most Common Errors on EasyEDA.

1. Manually creating backup schematics into the same project. When a project is converted to PCB, EasyEDA will merge all of the schematics under the same project into a single PCB. If there are multiple copies of the same schematic in a project then this will create errors such as duplicate part prefixes. Especially if you are new to EasyEDA, just keep one copy of each unique schematic in any one project.
2. Saving schematic and PCB into different projects. Unless you are absolutely sure that you will not need to update (Synchronise) your PCB from changes made to your schematic then please keep the schematics and PCB under the same project.
3. Bad packages. Schematic symbols must have the appropriate footprints assigned to them, these footprints must exist in the library and - for any footprint that you have not created yourself - you must have clicked on the **Favorite** option in the component search window to add it to your **Favorite Parts** list in the left hand Navigation panel.



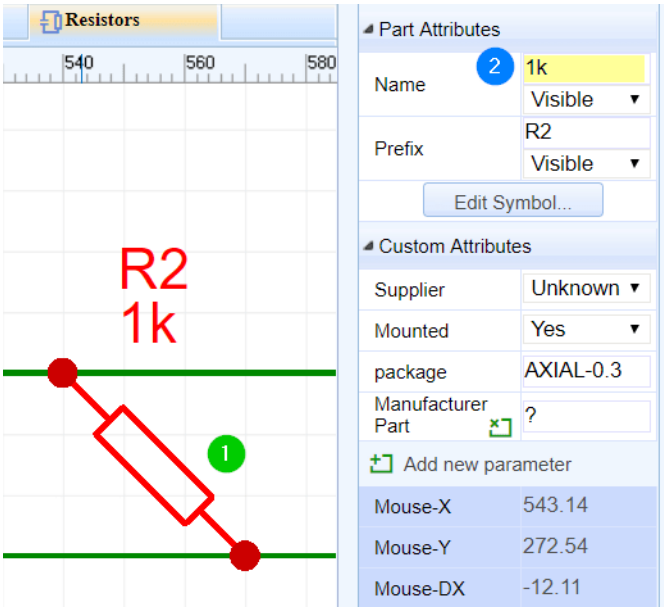
4. Using the polyline from the Drawing Tools Palette to connect symbol pins. To connect components together, you must use Wires from the Wiring Tools Palette.

Spice Simulation FAQ

EasyEDA's main target is schematic and PCB, not simulation. EasyEDA only support simple schematics simulation.

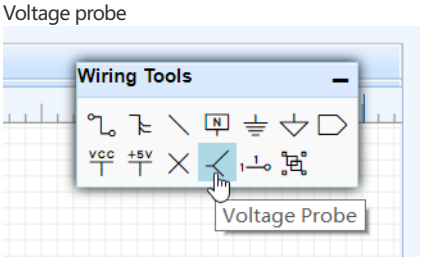
How to set the resistance of a resistor

You can use the name attribute. Just set the name or double click the value text.



Where Can I find the Probe?

Voltage probe



Why I can't simulate my schematic

EasyEDA only has very few simulation models, EasyEDA is powered by http://ngspice.sourceforge.net/ please check Ngspice to know what can be simulated.

Others.

Does EasyEDA canvas use the Cartesian coordinate system?

Yes and no.

It uses X and Y coordinates where the horizontal X coordinate is positive to the right of the origin and negative to the left but the vertical Y coordinate is positive **below** the origin and negative above it.

Actually, we think our coordinate system is not very good but it is hard to change.

PCB Order

After laying out your PCB, you probably want to order some **PCBs**. We have made it easy for you to save time and money by using our **awesome service** to order **low cost**, **high quality** PCBs *directly* from EasyEDA. More importantly, if you are not satisfied with the quality of our PCBs, EasyEDA will refund your money in full.

Although EasyEDA makes it easy to order PCBs for your projects and offers an exceptionally low PCB Manufacturing fee, you are free to download the Gerber files and order your PCBs from any other vendor. However, if you like EasyEDA, please give us a chance to fab. the PCB for you. We think you won't be disappointed.

PCB Quality

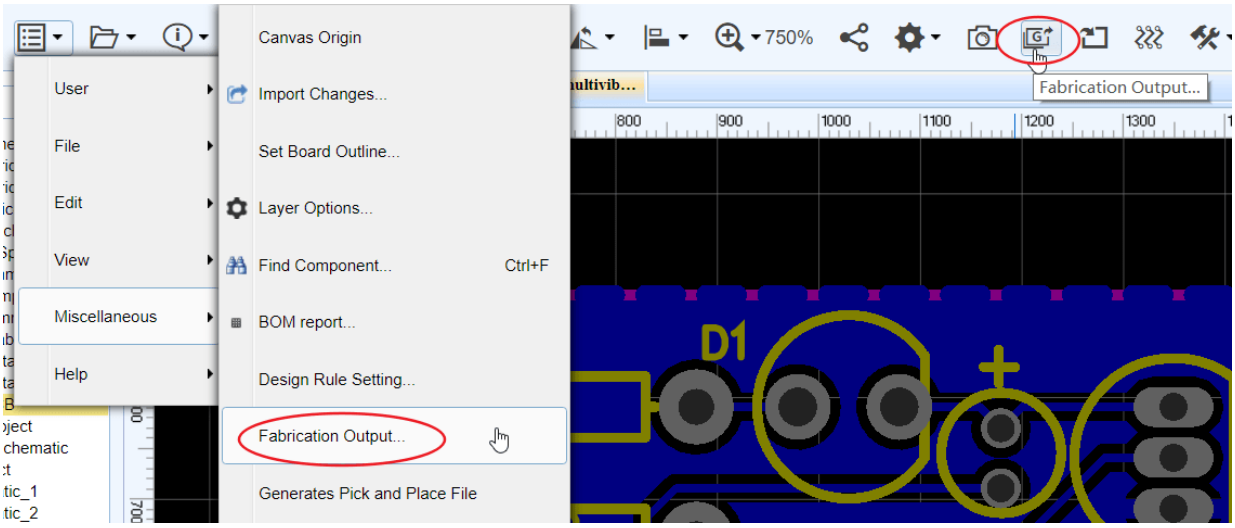
As engineers, we have spent more than 6 years building EasyEDA. As artisans, we believe that if you like using our EDA tools, then you will like our PCBs.

EasyEDA's PCBs are in a group buy model and all PCBs are given 100% E-test. This allows us to provide you with good quality, tested PCBs at a great price. We have shipped thousands to our users, all of whom like our PCBs.

No matter how good we tell you our service is, maybe you still have doubts. The easy way to remove those doubts is to try it out because - as we have said above - if you are not satisfied with the quality of our PCBs, we will refund your money in full. For details of this offer, please check the Warranty and Return.

Order Button

To order PCBs from us, just click the **Fabrication Output...** button in the PCB Editor window, as shown in the image below, and you will be redirected to an order page. In that page you can place an order quickly and easily. At the same time, at the click of a button, you can check the Gerber and drill files in our Online Gerber Viewer and then download your files. Obviously, we hope that you will support EasyEDA by ordering your PCBs from us but you are welcome to download the Gerber and drill files and send them to your favorite PCB house.



Essential Check Before Placing a PCB Order

A simple mistake can make a batch of PCBs useless so before submitting an order for PCBs, there are a lot of things to check.

The list below is a good starting point for the essentials but it by no means exhaustive!

1) Check that all nets and netnames are as intended. For example nets can be accidentally connected together while drawing and editing a

schematic and also by inadvertently assigning duplicated netlabels;

2) Check that the appropriate package has been assigned to each and every schematic symbol. Don't forget that different transistors, capacitors and even resistors may be in different packages in different locations in a schematic. Also check that components have been rated correctly as this may affect their package size (don't forget their height too!);

3) Check that the pin designations of the schematic symbol and the PCB package are the same. The pins on the schematic symbol for a bipolar transistor may be labelled B, C, and E but if the corresponding pins on the PCB footprint are labelled as 1, 2, and 3 then EasyEDA will flag this as an error when an attempt is made to convert the schematic into a PCB. It is simple to correct or - better - avoid this: change the labelling on the corresponding pins of the PCB footprint to B, C and E **or** change the labelling on the corresponding pins of the schematic symbol to 1, 2 and 3.

To change the pin labelling in the schematic to match the PCB footprint: select the part, press the `i` key then edit the **Names** in the **Edit Pin Map Information** section of the **Modify symbol information** dialogue that opens and click `OK` when finished.

To change the pin labelling in the PCB to match the schematic symbol: select each pin of the relevant part, then edit the pin **Names** in the right hand **Properties** panel.

Note: do not confuse the schematic symbol pin numbering with the spice pins numbering. For more about this see:

Schematic symbols: prefixes and pin numbers in:

https://docs.google.com/document/u/1/d/1OWZVVFRAe_2NW3WratpkA_SGuHa5AcRow5ZRfvcoVTU/pub#h.pkwqa1

4) Check that the pin designations of the PCB footprint chosen for each and every device actually matches the pinout of the device that will be soldered to it. It is very easy to assign a SOT23 package to a BC846 bipolar transistor where the pin order is:

Pin 1 = Base

Pin 2 = Emitter

Pin 3 = Collector

and then to forget that the pin order for a MMBF5485 junction FET going round the same SOT23 package in the same order is:

Pin 1 = Drain

Pin 2 = Source

Pin 3 = Gate

To change the pin labelling in the schematic to match the PCB footprint: select the part, press the `i` key then edit the order of the **PCB Pin** information in the **Edit Pin Map Information** section of the **Modify symbol information** dialogue that opens and click `OK` when finished.

To change the pin labelling in the PCB to match the schematic symbol: select each pin of the relevant part, then edit the pin **Numbers** in the right hand **Properties** panel.

5) Check that all necessary Bill of Materials (BoM) information is present and correct. Correct it and add more if required, making sure that fields such as `Description` are consistently labelled so that they form a coherent column structure in the BoM;

6) Check that all PCB footprints are correct for the intended devices (yes: whether they have come from the library or you have created them yourself: check them thoroughly);

7) Check that silkscreen markings such as the polarity markings for electrolytic capacitors and diodes are the right way round. Even if the pin names, numbering and sequence around tha package are correct, it can all go wrong if the footprint markings show the device in the wrong orientation.

9) Check that devices have been placed on the correct side of the board;

10) Refresh and check all the Components and Nets in the schematic Design Manager tab in the right hand panel;

11) Refresh and check all the Components, Nets and DRC Errors in the PCB Design Manager tab in the right hand panel;

12) Check connector and on-board pot and switch orientations;

13) Check the dimensions and locations of mounting holes and any components that have to line up with respect to these mounting holes or to apertures in an enclosure;

14) Check that the order of the top and bottom (and any inner layers) is correct;

15) Check that a Board Outline exists, is closed and that it is shaped and dimensioned correctly and is on the correct (Board Outline) layer;

16) Check that silkscreen markings do not overlap pads;

17) Check that all required silkscreen markings are present, in the correct locations on the correct layers, are within the recommended dimensions, are legible and and spelled correctly;

18) Check that any additional information such as notes about the PCB stackup etc., are present and on the correct (Documentation) layer;

19) Check that no board outline, silkscreen or documentation layer information has accidentally been placed on any copper layers;

20) Having completed the layout, check that the assembled component heights do not foul any enclosure (At the time of writing (160922) this is not something that can be acheived directly in EasyEDA as there is no 3D viewer yet available so this must be checked by other means);

21) If copper areas are used with heat shunt spokes enabled, check that any tracks that join pins that are also joined by copper areas run within the area of a spoke. If they do not - for example a 45 degree diagonal track coming out of a pad with 90 degree heat shunt spokes - the track forms an extra spoke which increases the heat shunting to a pad and so may make soldering more difficult;

22) Check that all copper areas assigned to a net are joined by reasonable widths of copper, i.e. they are not just joined by thin slivers of copper;

23) Check that any tracks that require special routing considerations such as Kelvin connections to low value current sense resistors or increased clearances for high voltage traces have been correctly implemented;

24) Check clearances of copper on all layers and components on both sides to the edges of the board;

25) Check that no traces have been set to hidden in the Design Manager Nets list;

26) Use the Design Manager (the **Design** button in the left hand panel) to check that all components are present in both the schematic and the PCB and that all nets have at least two connection;

27) Use the Design Manager (the **Design** button in the left hand panel) to check, investigate and correct all DRC errors. A completed PCB design should have no DRC errors;

28) Check that the Gerbers to be generated are from the correct version of the PCB layout. It may have changed as a result of the above checks so always regenerate the Gerbers from the latest version unless there is a specific reason to use them from an earlier version. Putting a version number on the PCB helps but is not perfect as this has to be updated manually anyway;

29) Download and check items (6) to (24) in the Gerbers in either the EasyEDA Gerber Viewer:

https://gerber-viewer.easyeda.com/

or using a 3rd party Gerber viewer such as the free and open source gerbv:

http://gerbv.sf.net/
http://flatcam.org/
http://kicad-pcb.org/
http://www.gerber-viewer.com/

30) Check the order options such as number of boards, copper finish, silkscreen colour, solder mask colour, panellisation, any solder paste mask requirement and so on;

31) Lastly, check that the order is being placed with the correct delivery option. The default delivery method is by express courier. This is the most expensive option but avoids the mistake of ordering boards that are needed urgently with a slow delivery method.

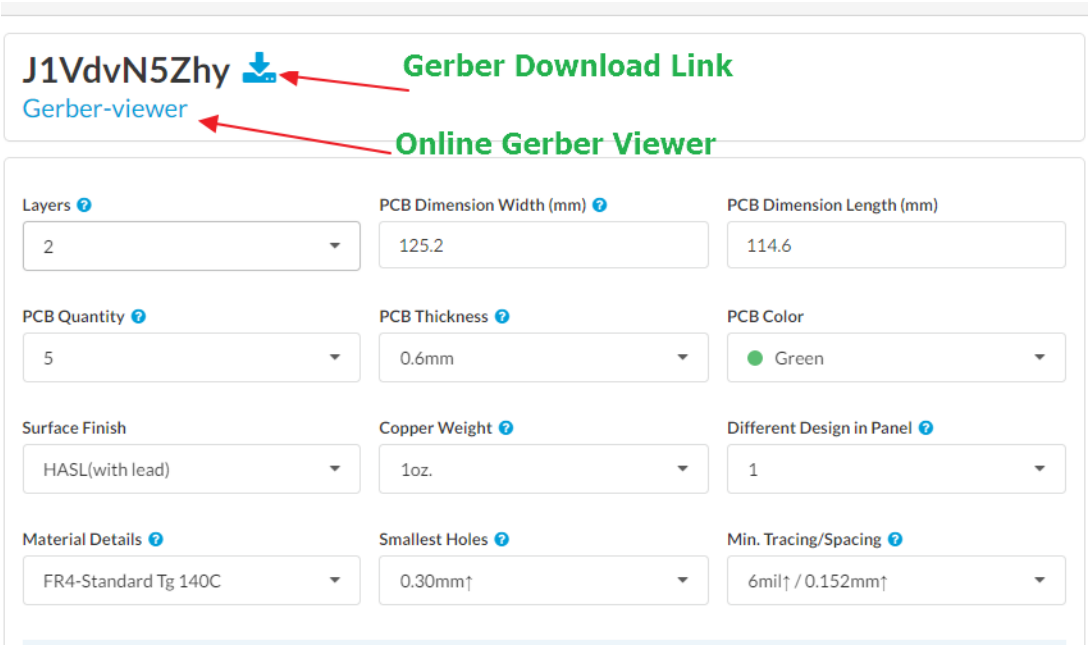
PCB Order from EasyEDA Editor

When you click the `Fabrication Output...` button your order is coming from within the EasyEDA editor environment so you don't need to input information about **Layers**, **Width** and **Height**; EasyEDA fills this information in for you.

On the order form page you will find a real time price. Most of the time this price is the final cost however if, for example, you change the Layers to 4 or 6, you will find the price field changes to **Quote**. If that happens, don't worry: just click the `Save to Cart` button and we will email a quote for the final price to you ASAP.

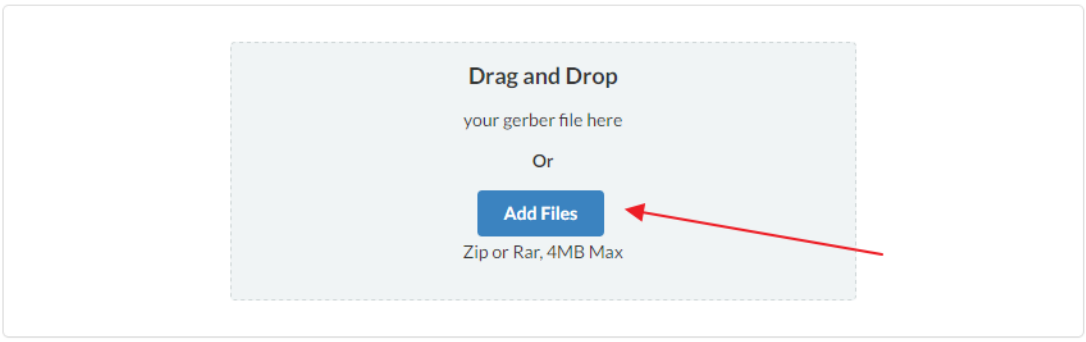
Note:

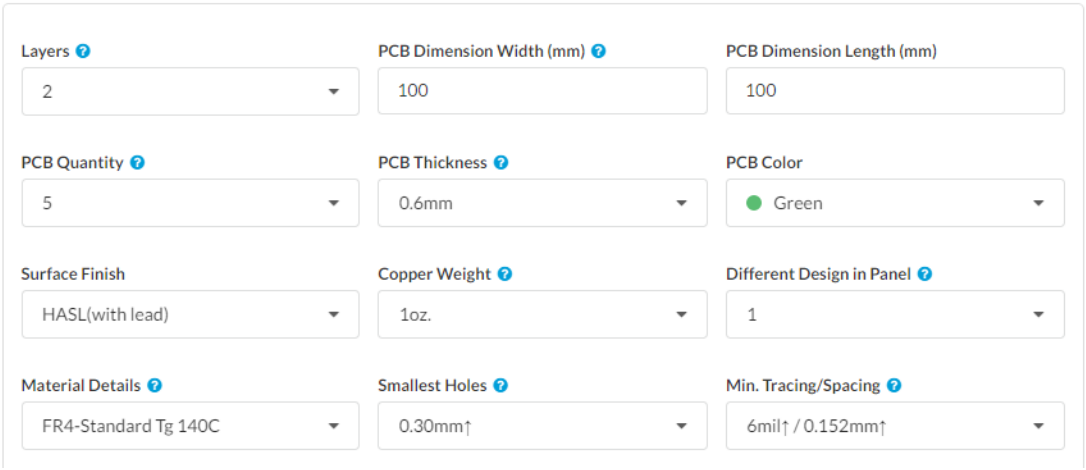
- *When you add your PCB to the cart, EasyEDA saves a copy of Gerber files at that time.*
- *If you then change your PCB back in the PCB editor, **EasyEDA does not synchronize your Gerber files to the updated PCB design**.*
- *The only way is to delete the earlier version of the PCB from your Cart and then add the updated design as a new item.*



PCB Order from Order Link

If you would like to upload your own Gerber files from a third party PCB tool such as Eagle, Pads, or Altium Design, just click on this link https://easyeda.com/order to order. This page will let you upload your own Gerber Files.





PCB Capabilities

Number of Copper Layers*	1-16
PCB Material	FR-4, FR4-Tg, FR4-High Tg
Available Solder Mask Colors	Green, Red, Yellow, Blue, White, Black
Silk Screen Colors	White, Black (For White Solder Mask only)
Minimum Quantity	5
Minimum dimensions*	0.4cm x 0.4cm
Maximum dimensions*	100cm x 100cm

- If your PCB requires more than the default maximum of 6 layers (up to a maximum of 16) or larger dimensions, then please contact us before placing your order
- If your PCB dimensions are bigger than 45cm * 45cm, it may add some additional cost

Manufacturing Specifications:

Item	Specs	
	Unit: mm	Unit: mil
Available Board Thickness	0.4, 0.6 (except 4 layer), 0.8, 1.0, 1.2, 1.6, 2.0	15.7, 23.6 (except 4 layer), 31.5, 39.4, 47.2, 63.0, 78.7
Thickness Tolerance	(t >= 1.0) ± 10%	(t >= 39.4) ± 10%
Thickness Tolerance	(t < 1.0) ± 0.1%	(t < 39.4) ± 0.1%
Insulation Layer Thickness	0.075 - 5.0	2.95 - 196.85
Minimum trace width	0.089	3.5
Minimum inner trace width	0.127	5
Minimum trace/vias/pads space	0.102	4
Minimum inner trace/vias/pads space	0.102	4
Minimum silkscreen width	0.1524	6
Minimum silkscreen text size	0.8128	32
Outer Layer Copper Thickness	> 0.03	> 1.18

Drilled Hole Diameter (Mechanical)	0.3 - 6.35	11.81 - 250.00
Drilled Hole Diameter (Laser)	0.2 - 0.3	7.87 - 11.81
Diameter Tolerance (Mechanical)	±0.08	± 3.148
Solder Mask Bridges	0.1	3.94
Circuit to edge	≥0.3	≥11.8
Slot	≥0.6	≥23.6
Slot Tolerance(Mechanical)	±0.15	±6
Aspect Ratio	8:1	
Solder Mask Type	Photosensitive ink	

If you have any special PCB requirements, please contact us before placing your order.

Price

All Prices stated are FOB Shenzhen. This does not include transportation costs which shall be borne by the customer.

Manufacturing Price

Price is dependent on many factors, such as the quantity of PCBs you order, PCB Color, Surface Finish, PCB Thickness, PCB Dimensions, Hole size etc.

EasyEDA uses a group buy business model and we are sure it will be hard to find a better PCB supplier than EasyEDA offering the same price and quality.

EasyEDA needs 2~4 days to manufacture the PCBs after you submit payment.

Shipping Costs

Method	note	Price	Service
Air Mail	Delivery Time: 8-35 days. Most of our users receive their PCBs in two weeks.	From $6	Usually http://www.singpost.com/
Express	Delivery Time: 3-7 days	From $24	Usually delivered by http://www.DHL.com/

Note: The shipping cost is estimated. EasyEDA will always try to find the best shipping option. If you are in some Remote Areas, we will ask you to pay for more or change to some other express service such as Fedex, UPS. Sometimes, we will use Hongkongpost for delivery by Airmail.

File Name

If your Gerber file names are good, this will save us a lot of time in checking your design. There are many different PCB design software packages so there are many variations of Gerber file names and filename extensions.

Gerber Type

File Type	
.TXT, .DRL .dri , .drd	Drill holes
.GML, .GKO, .GML, outline EDGE_CUTS.GBR	Outline
.GTP F.PASTE.GBR	Solder Paste Top
.GBP B_PASTE.GBR	Solder Paste Bottom
.GTS .stc, .smt, F.MASK.GBR	Solder Mask Top
.GBS .sts, .smb B.MASK.GBR	Solder Mask Bottom
.GTO .sst F.MASK.GBR	Silkscreen Top
.GBO .ssb B.SILKS.GBR	Silkscreen Bottom
.GTL .cmp, .top F.CU.GBR	Top Layer
.GBL .sol, .bot, B.CU.GBR	Bottom Layer
Inner layer in 4 layers PCB:	
.GL2	Layer 2
.GL3	Layer 3

If you don't know how to map your files, don't worry about changing the file names and please contact support for help.

We encourage you to use our free online gerber viewer to check your gerber files before placing an order.

E-Test

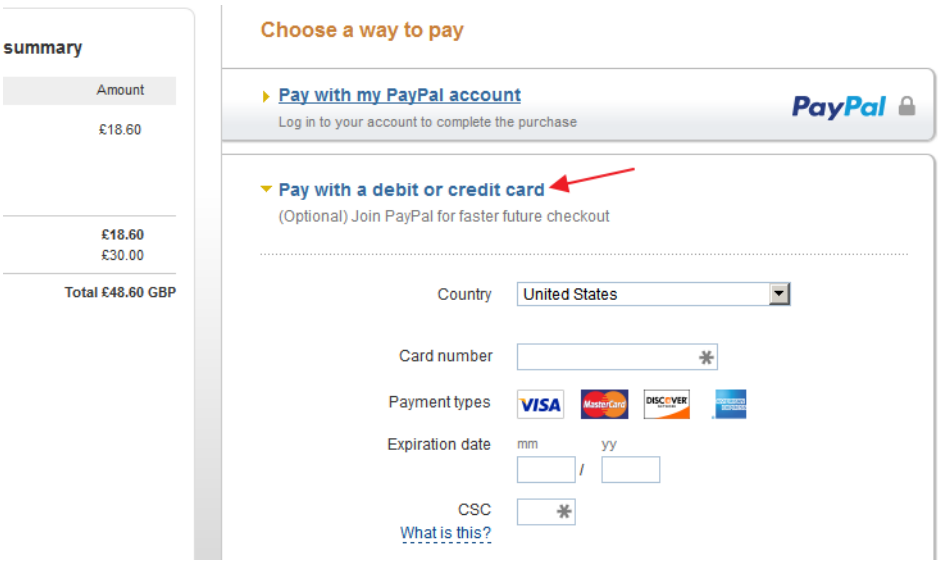
All PCBs undergo a 100% AOI (Automated Optical Inspection) to make sure that all tracks and pads are connected. In addition to this the PCBs can be tested by a flying probe to make sure that all vias are connected, because this is not visible by the AOI. Single layer PCBs do not require this test because there are no vias but boards with 2 layers and above will always be 100% tested with a flying probe.

Payment

We accept the PayPal, Credit Card and Wire Transfers.

PayPal and Credit Card

We use Paypal as our payment; it is safe and easy. If you don't have a Paypal account, you can still use Paypal to pay with a debit or credit card.



Wire Transfers

Wire Transfers can only be used on orders with a grand total (subtotal plus all additions and deductions but excluding shipping fees) of at least $600. For orders > $2000, payment by Wire Transfer is preferred. In this circumstance, 3.5% extra discount will be applied for the grand total (subtotal plus all additions and deductions but excluding shipping fees). Wire Transfer payments usually take 3-5 business days to clear. We will not ship your order until your payment is verified by our bank. Please send a copy of the Wire Transfer receipt to our customer service because although it is not sufficient to release an order, it will help us to push the delivery date.

Customs, Duties and Taxes

You should expect to pay any amount charged by the government in your respective country. This includes but is not limited to: duties, taxes and any extra fees charged by the courier company. We will not be held responsible for any extra charges once the original package has been shipped. If the customer refuses to pay these extra charges, the return shipping and any additional fees will be taken out of the cost of the order, with any remaining funds being refunded to the customer. Customs are quite different in each country. Please include information about particular Customs requirements as necessary, while you are placing your order: we will support you as much as possible.

Warranty and Return

For your first order for a PCB laid out in **EasyEDA**, we have the top return policy on the planet! If you don't like them, just send an email to

support@easyeda.com, no reason needed. We will provide your full money back - including product + shipping costs - in one working day.

For subsequent orders, because you now know the quality level of our PCBs, if you are not satisfied with a product you bought from us for whatever reason, you just need to email us some pictures of the product and explain why you are not happy with it. We will then refund the full money of the product. Shipping fees will only be refunded if the return is a result of a shipping error on our part.

PCBOrderFAQ

PCB Parameter Description

PCB Dimension

EasyEDA supports the design of PCBs up to 50cm * 50cm however we suggest our customers try to limit the design to a PCB size of no larger than 45cm * 45cm. We may have to charge more for PCB sizes greater than this because they are harder to fabricate and need a bigger box with more packaging to protect them in shipping.

- If the size is smaller than 2cm, you need to panelize them to big size, or we can't help you to do the E-test.

PCB Quantity

EasyEDA uses a group buy model, so the price is very low, but the minimum number of PCBs should be 5 pcs. So for example, if you need 2 pcs, you need to order 5 pcs and similarly, if you need 7 pcs you need to order 10 pcs.

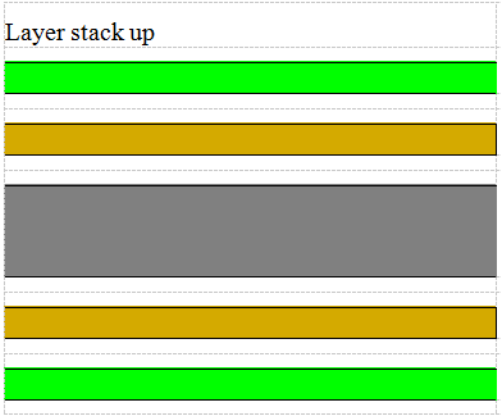
PCB Thickness

EasyEDA PCB provides 0.4mm, 0.6mm, 0.8mm, 1.0mm, 1.2mm, 1.6mm, 2.0mm thickness to choose from. If you need a 2.5mm or 3mm thickness board, please ask us for a quote via email. Please note that not all thicknesses will be available for all numbers of layers; for example no PCB house can fabricate a 32 layer PCB with a 0.4mm thickness!

PCB Stack up

2 Layer PCB stackup 1Oz

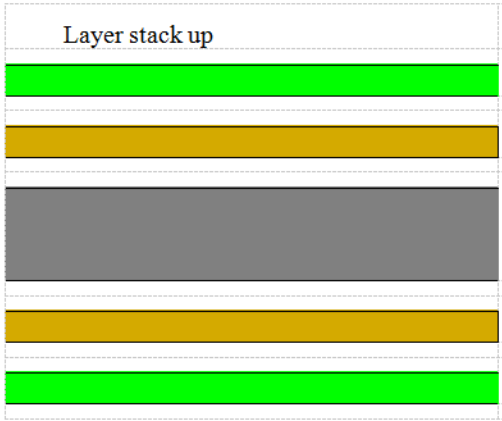
2 Layers **1.6mm (0.062 inch) Green for 1 Oz**



Layer	Thickness (mm)
Top Solder Mask	0.01
Top Layer	0.035
Core	1.5
Bottom Layer	0.035
Bottom Solder Mask	0.01

2 Layer PCB stackup 2Oz

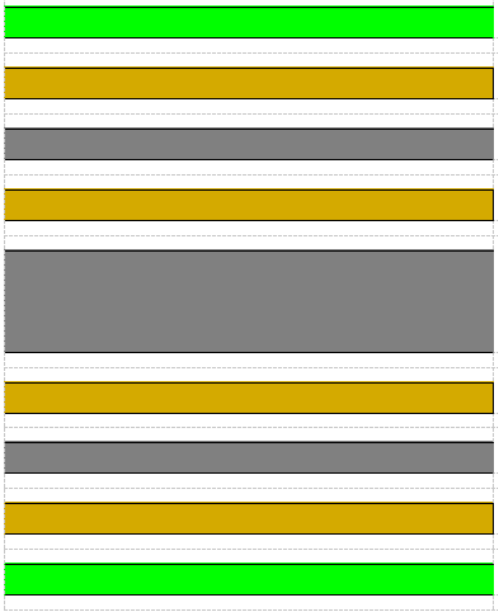
2 Layers **1.6mm (0.062 inch) Green for 2 Oz**



Layer	Thickness (mm)
Top Solder Mask	0.01
Top Layer	0.070
Core	1.5
Bottom Layer	0.070
Bottom Solder Mask	0.01

4 Layer PCB stackup 1Oz

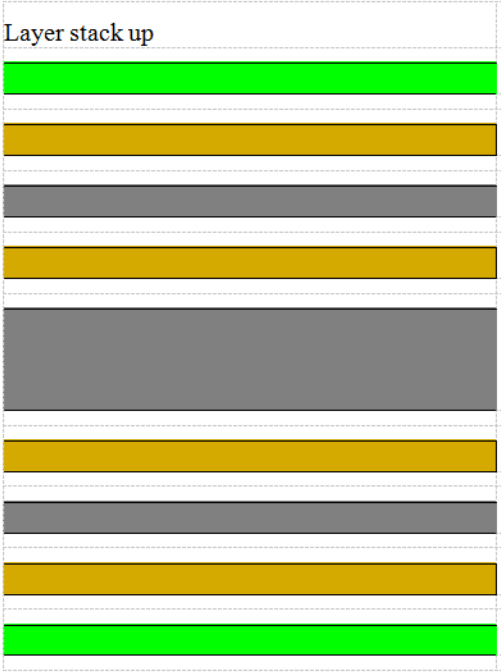
4 Layers **1.6mm (0.062 inch) Green for 1 Oz**



Layer stack up	Layer	Thickness (mm)
	Top Solder Mask	0.01
	Top Layer	0.035
	Prepreg	0.18
	Inner 1	0.017
	Core	1.12
	Inner 2	0.017
	Prepreg	0.18
	Bottom Layer	0.035
	Bottom Solder Mask	0.01

4 Layer PCB stackup 2Oz

4 Layers **1.6mm (0.062 inch) Green for 2 Oz**



Layer stack up	Layer	Thickness (mm)
	Top Solder Mask	0.01
	Top Layer	0.07
	Prepreg	0.18
	Inner 1	0.017
	Core	1.12
	Inner 2	0.017
	Prepreg	0.18
	Bottom Layer	0.07
	Bottom Solder Mask	0.01

6 Layer PCB stackup 1Oz

6 Layers **1.6mm (0.062 inch) Green for 1 Oz**

Layer stack up	Layer	Thickness (mm)
	Top Solder Mask	0.01
	Top Layer	0.035
	Prepreg	0.1
	Inner 1	0.017
	Core	0.57s
	Inner 2	0.017
	Prepreg	0.1
	Inner 3	0.017
	Core	0.57
	Inner 4	0.017
	Prepreg	0.1
	Bottom Layer	0.035
	Bottom Solder Mask	0.01

For other numbers of layers or for a different layer stack up, please email us before placing your order.

Copper Weight

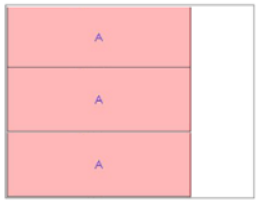
You can select 1oz, 2oz in our order page.

1. For 1Oz, the track width and clearance can be 3mil.
2. For 2Oz, please make sure the clearance is bigger than 8mil.

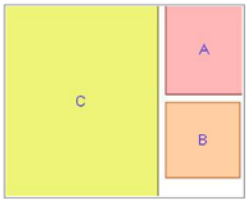
Different Design in Panel

Some customers would like to merge more than 1 PCB in the same Gerber. We know you want to save money but this may make it **hard to cut the board outline** and more importantly this will take a lot more time to pick up and package the PCB. Although by ,doing this, you just have the one order, this complicates the fabrication of the panel and separation of the individual PCBs, so we will usually charge more for this. Similarly, using holes or slots as break off sections between boards are treated the same way as putting more than one design on a panel, each with its' own board outline.

Note: This additional charge only applies if the PCBs on a panel are different. Boards such as in the left hand image below will not incur an additional charge because they are easy to pick up but boards such as in the right hand image would incur an additional charge.

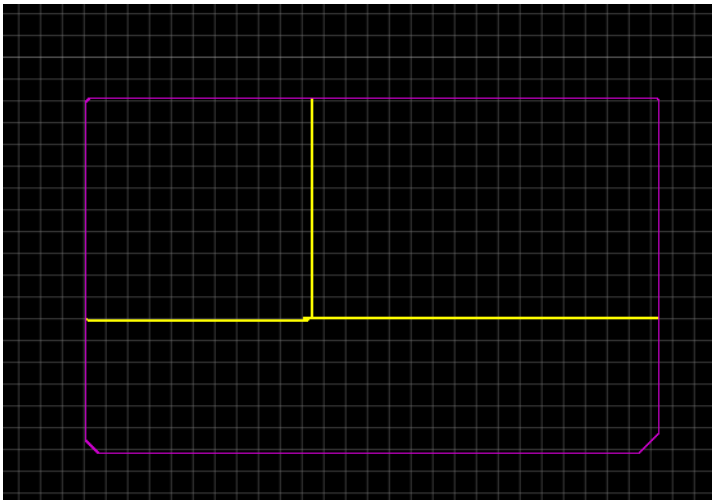


3 Same PCBs



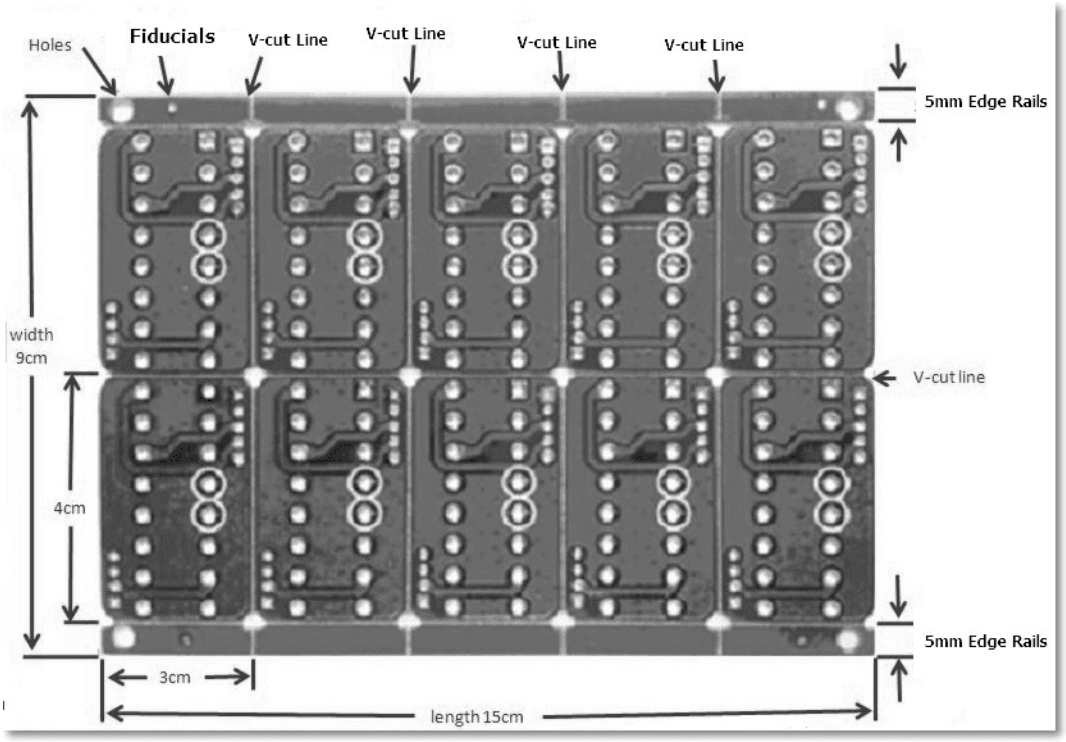
3 Different PCBs

If you are just in the prototype phase and want to save money, then you can use the following trick. Place your different designs all inside one overall board outline and use lines drawn in the silk layer to mark out the separate the PCBs. Then, when you receive the PCB, carefully cut them apart yourself (we recommend you do this before you assemble the PCBs!). Like the PCB shown below, the yellow lines are drawn in the silk layers showing how you can merge 3 different PCBs in one gerber without incurring any extra costs.

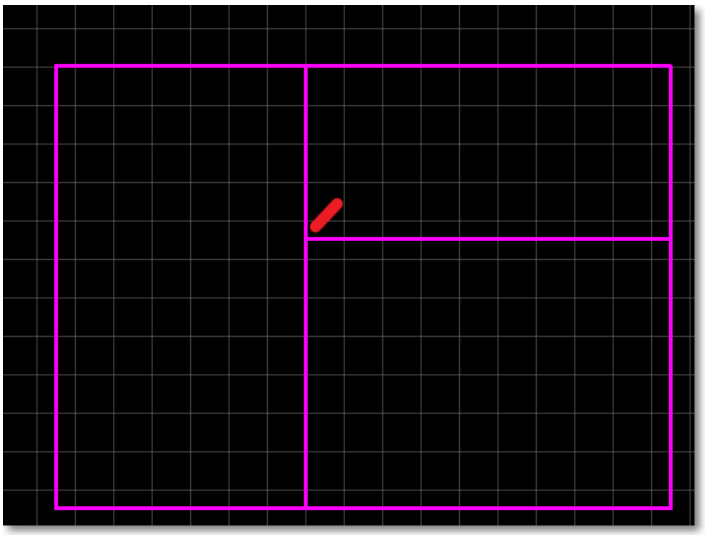


V-cut/V-Groove

This will help you to build a PCB Array to save time by making it faster to solder the PCB in SMT.



1. The V-cut line should be the same as the outline of the Sub-PCB. That is to say, there is zero space between the sub-boards.
2. The PCB panel needs to be larger than 8 * 8cm.
3. The V-cut line must cross the whole panel or else the factory can't add a V-groove on the PCB because the milling cutter may destroy any sub-PCBs on the V-cut line. They also cannot stop part way across the panel. The V-cut lines shown below are not acceptable:



Material Details

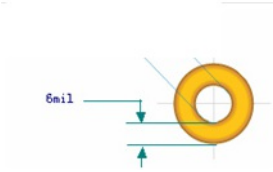
EasyEDA supports FR4-Standard Tg 140C, FR4-Tg 150C, FR4-High Tg 170C. The FR4 TG's lead time may be more than 6 days. 90% of EasyEDA orders use FR4-Standard. For more information about this, please check the FR4 Material pdf

Smallest Holes Diameter

0.3mm for mechanical drill, 0.2mm for the laser drill. To save money, please use a minimum drill diameter of 0.3mm.

Ring

The width of the ring around vias or pads should be wider than 6mil/0.15mm.



Min. Tracing/Spacing

We support down to 4mil but to save money, please use 6mil.

Grid size

Make sure the Grid filled size is bigger than 8mil/8mil (track/space), if less than that, we will change it to 8mil/8mil .

Impedance Control

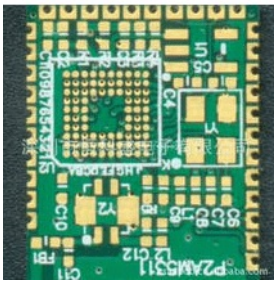
We support 5% and 10% precision. Please add enough information about your impedance control requirements to help us to fabricate your PCB.

Gold Fingers

If you wish to built PCBs that plug directly into edge connectors, such as memory cards, please choose **Gold Fingers** as shown in the image below:

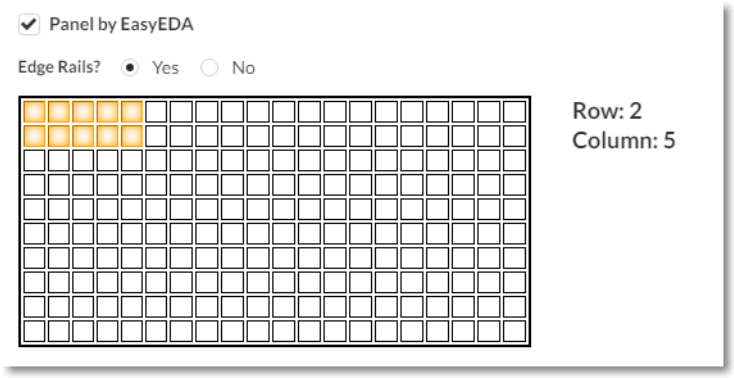
Half-cut/Castellated Holes

If you need to build some PCBs as shown in the image below, please choose **Half-cut** holes.



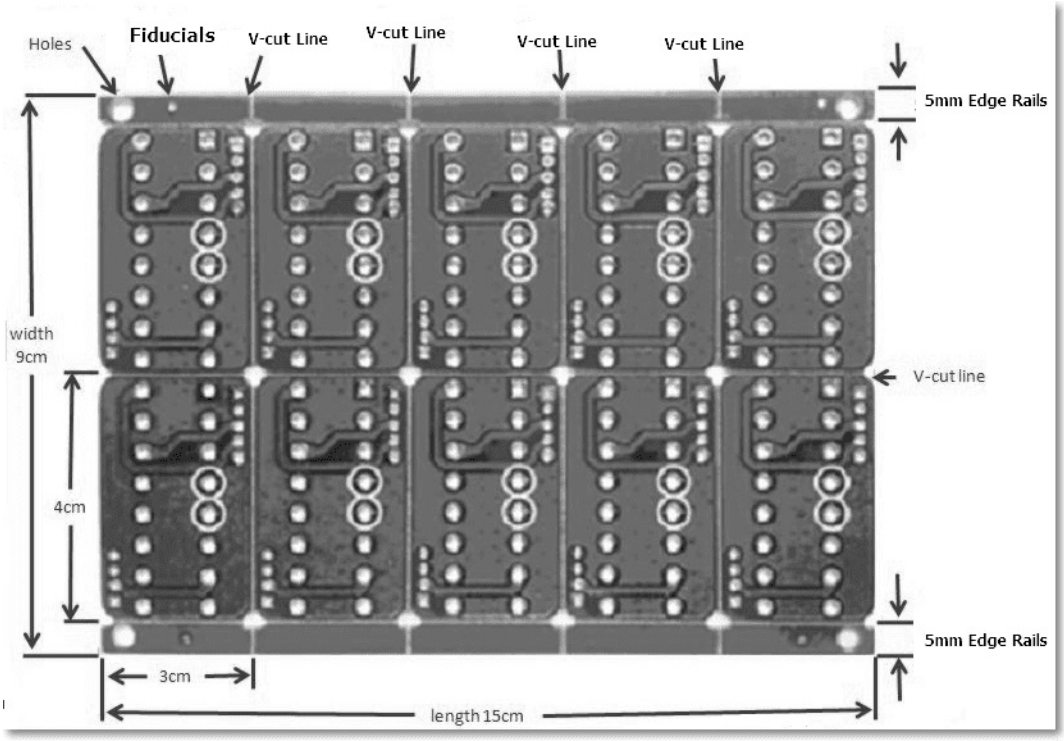
Panel by EasyEDA

When your Gerber is for just one design and you need EasyEDA to help you to duplicate many copies onto one panel, you can use this option. You can drag across the rectangles to select then click on the panel to set how many rows and columns as shown in the image below:



For this image, we will merge 10 small PCBs to 1 big panel. If you order 5 pcs, then we will send 5 big panels to you, each with 10 PCBs on it so you will end up with 50 small PCBs.

If you select the **Edge Rails**, we will add a 5mm board edge as shown in the image below: This is 2 rows and 5 cols panel PCB.

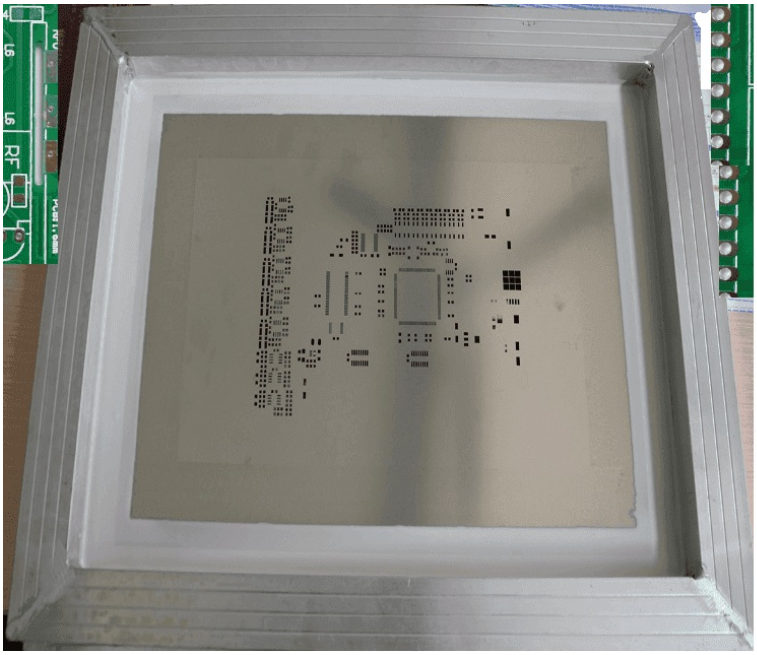


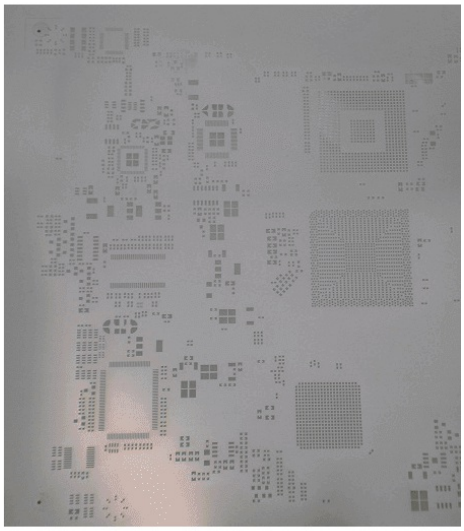
Note:

1. *Make sure the single PCB size is bigger than 2cm x 2cm, or we will charge $20 for the v-cut cost. Small PCB is hard to v-cut.*
2. *Make sure the board outline is simple, for complex board outlines, you need to panelize by yourself. Or you can pay us $15 to do that, we will send the panelized gerber to you to confirm.*

Stencil Parameter Description

A Stencil can help you to solder the PCB quickly. For efficient and reliable SMT assembly a Stencil is a must. EasyEDA can provide the option of NON-FRAMEWORK (or frameless) and FRAMEWORK stencils. The right hand image below shows a frameless stencil. Frameless stencils are cheaper and lower weight(0.2Kg) so they can help to reduce the shipping cost.

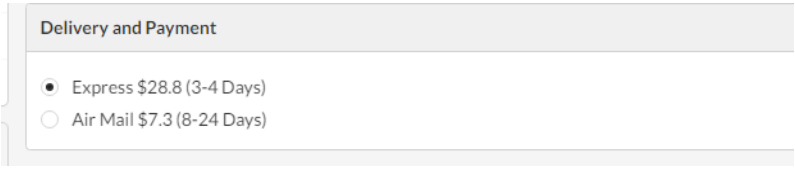




Order FAQ

How to check the Shipping Cost?

We can ship PCBs to any country. The shipping cost depends on the weight of the boards, country, and shipping method. Before you pay, you can see the shipping cost and select the shipping method. So you just need to add it to your cart and then you will see some options like in the image below:



How to order lots of PCB together?

EasyEDA allows you to order many different PCBs together, just add the PCB to cart one by one , at last to pay for together.

How to remove the customer ID on the PCB?

EasyEDA uses a group buy model to save the cost of production, for picking up your PCB easier, we need to add a very small string to your PCB, the string might be under some IC, and if you solder the PCB, the string will be hidden. If you don't like it, there are two ways.

1. Email us, we will fabricate your PCB in another way, but the cost will be higher.
2. When you order, you can use https://easyeda.com/Doc/Tutorial/PCBOrderFAQ#Panel-by-EasyEDA and keep the edge, we will add the string at the dirty edge, so there is no string on your product PCB.

EasyEDA API Plug

Before reading this capture, please check Open EasyEDA File Format first.

Why Need API

After route the PCB, you found out that you need to enlarge all tracks size a bit little, How? After route the PCB, you found out that all Vias' hole size is too small, How to fix this? How to create a board outline using code? EasyEDA API will let you control your designs in an easy way.

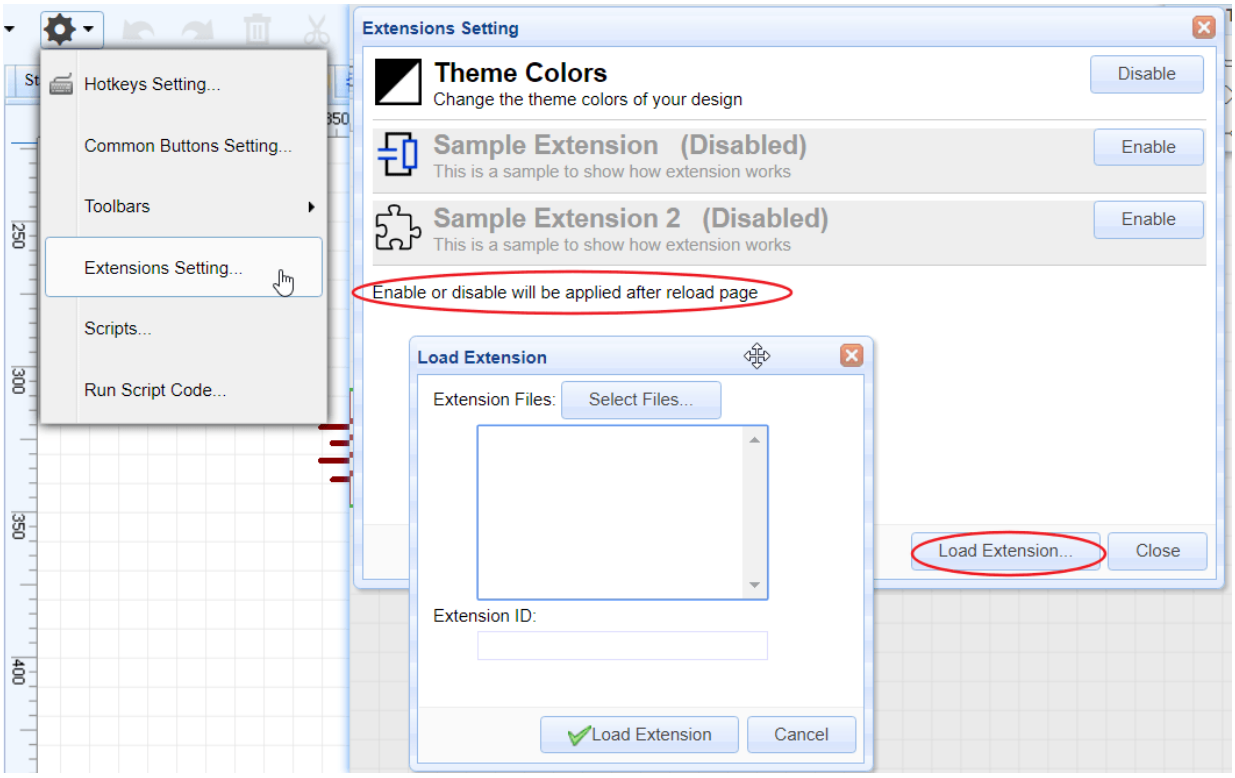
How to use API

How to find the plug entrance

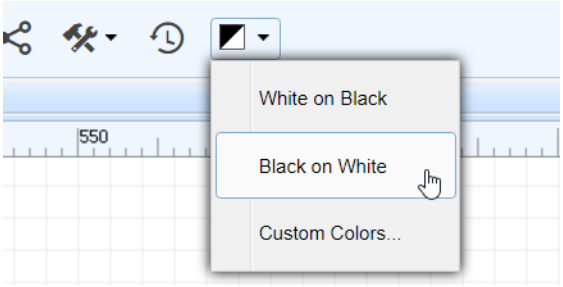
You can click **Config Icon** > **Extensions Setting** on the top toolbar image as below.

Extensions Setting

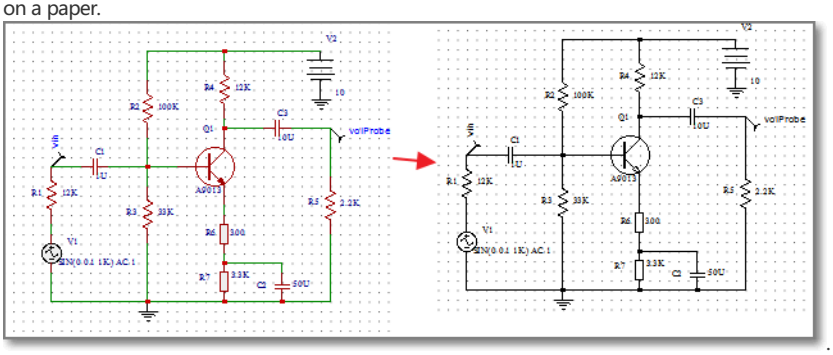
You can enable or disable the default extensions, after enable, please **reload** the EasyEDA editor. We will give you a file about how to create an extensions soon.



If you enable the **Theme Colors** Extension, you will find a button on the tool bar like bellow image:



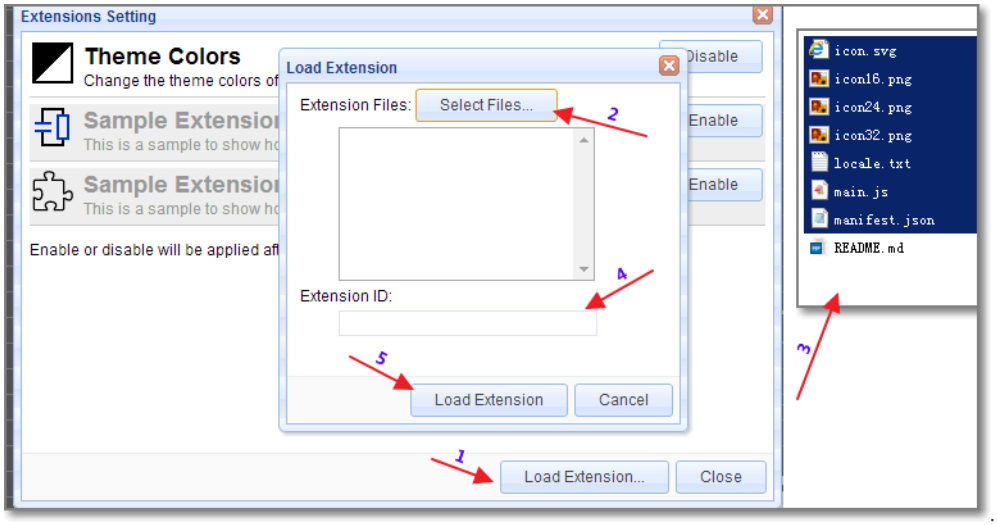
If you click the **Black On White**, you will find your schematic changes like bellow image, this is useful when you would like to print your design on a paper.



You can check our **github** codes of this API via https://github.com/dillonHe/EasyEDA-Documents/tree/master/API/example/theme, check the **manifest.json** and **main.js** out, you will find out how to create an extension.

How to install an extension

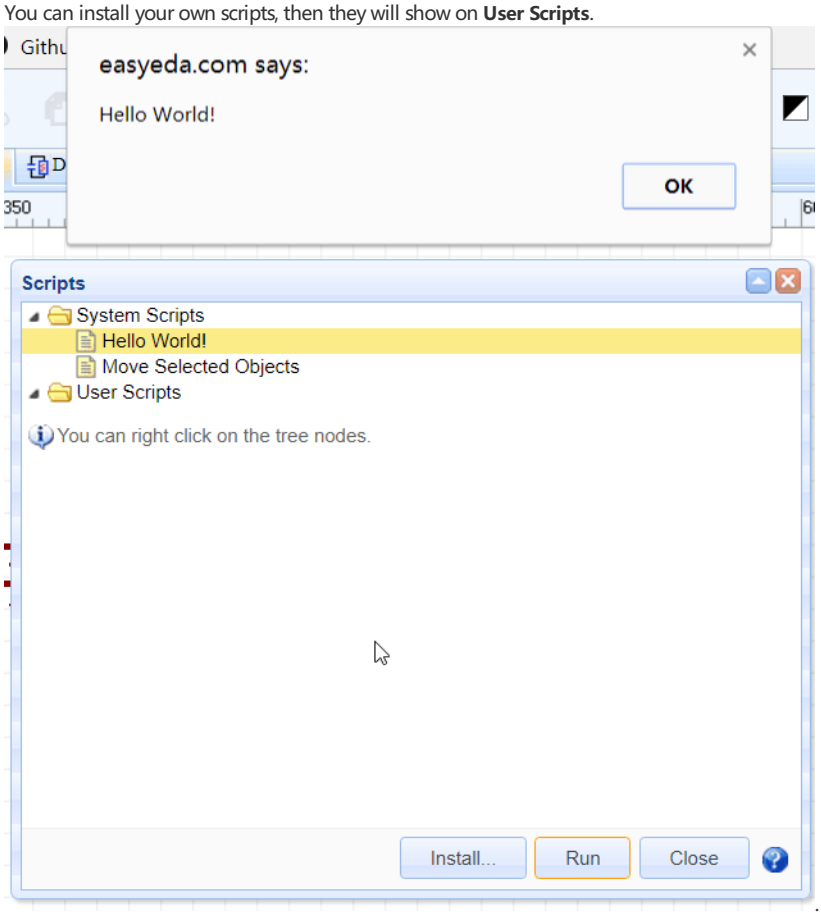
1. Click the Load Extension button
2. Click the select file button
3. Select **All** the files.
4. Type a name
5. Click the load button.
6. **Close** EasyEDA editor and open it again.



Scripts

If you just need some simple functions, you don't need to create an extension. You just need to create a single Javascipt file and keep it in this list.

1. You can select the `Hello World`, then click the `Run` button, the response as below image.
2. You can select some items, then try `Move Selected Objects`.
3. You can install your own scripts, then they will show on **User Scripts**.

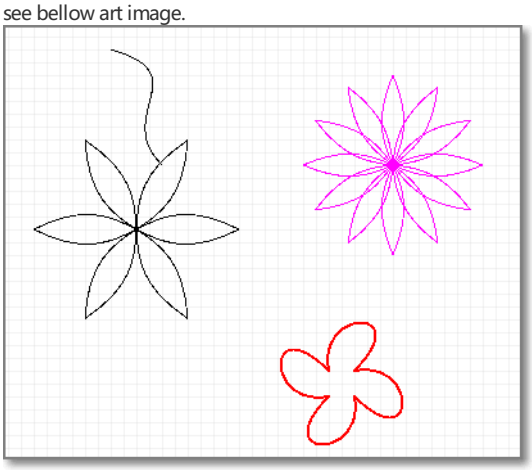


Run Script code

In some case, you just need to run the function one time, such as create a user define board outline in codes, changing the Track width, change the hole size etc. You can use this way.

example 1 Art

You can open an empty schematic and copy this example javascript codes to the text box to run a test. After clicking the `Run` button, you will see bellow art image.

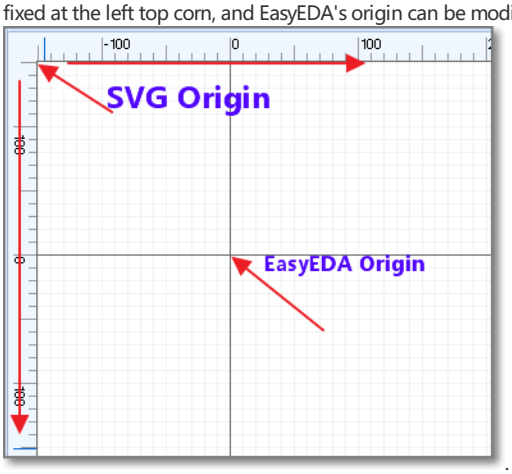


example 2 Change track width and via hole size

You can open a **PCB** and copy this example javascript codes to the text box to run a test. After that, All tracks will be 10mil.

EasyEDA Coordinate System

EasyEDA's editor is based SVG, SVG viewport,(Coordinates increase **left-to-right** and **top-to-bottom**, the same as EasyEDA). But SVG's origin is fixed at the left top corn, and EasyEDA's origin can be modified at the any place.



Be careful this, they are different from **Cartesian coordinate system**

Unit

There are two kinds of unit in our editor, SVG Canvas unit and real world EasyEDA unit. SVG Canvas unit is **Pixel**. The real world EasyEDA unit in schematic is also **Pixel**, but in PCB, there are **mm**, **mil** and **inch**. We use bellow map to convert Canvas to real world. - 1 pixel = 10 mil - 1 pixel = 0.254mm - 1 pixel = 0.01inch There are API for these convert.

```
//mm2pixel: convert 10mm to pixel
var result = api('unitConvert', {type:'mm2pixel',value:10});
```

```
//mil2pixel: convert 10mil to pixel
var result = api('unitConvert', {type:'mil2pixel',value:10});
```

There are other convert method, such as `inch2pixel`, `pixel2mm`, `pixel2mil` and `pixel2inch`.

All EasyEDA's value is based pixel, if you can keep in mind that 1 pixel equal 10mil or 0.254 mm, you don't need to use any convert function. For example, if you want to change a Track to 20mil, so you just need to use 2.

API List

Get EasyEDA Source

1. get EasyEDA JSON objects, type is `json`, you can check PCB Json object out to know more.

   ```
   var result = api('getSource', {type:'json'});
   ```

2. get EasyEDA compress string, EasyEDA save this string to our database, it is a bit little hard to read and understand, but it is small in size. EasyEDA save this string to our database.

   ```
   var result = api('getSource', {type:'compress'});
   ```

3. Get SVG string

   ```
   var result = api('getSource', {type:'svg'});
   ```

Check the Get EasyEDA source example codes.

Apply Source

After you can use your codes to hack EasyEDA's source, then you need to apply the source to EasyEDA's editor. You can

1. Apply as compress string

   ```
   //will open a new editor and convert compressStr to EasyEDA file.
   api('applySource', {source:'compressStr', createNew: true});
   ```

2. Apply as Json object.

   ```
   //will modify the active file and convert json object to EasyEDA file.
   api('applySource', {source: json, createNew: !true});
   ```

Check the Apply Source example codes.

Get Shape

If you want to get an EasyEDA json object by **id**, you can try to use bellow code.

```
      var obj = api('getShape', {id:'gge13'})
```

Delete Shapes

Removing shapes by follow code

Update Shape

If you want to modify an EasyEDA object, you can use this API.

```
//Change the net to GND and the shape to ELLIPSE
 api('updateShape', {
    "shapeType": "PAD",
    "jsonCache": {
    "gId": "gge5",
    "net": "GND"
    "shape": "ELLIPSE"
    });
```

shapeType and gId are must provided.
1. Schematic
`shapeType`, `schlib`, `rect`, `polyline`, `polygon`, `wire`, `bus`, `image`, `circle`, `ellipse`, `line`, `path`, `arc`, `annotation`, `junction`, `netlabel`, `busentry`, `arrowhead`, `noconnectflag`, `pin`, `netflag`
2. PCB
`shapeType`, `FOOTPRINT`, `TRACK`, `COPPERAREA`, `SOLIDREGION`, `RECT`, `CIRCLE`, `TEXT`, `ARC`, `DIMENSION`, `PAD`, `VIA`, `HOLE`

Create Shape

If you want to create EasyEDA shape by codes, you can try. We will provide more information about this API soon, now we just provide examples. You will find out how to do.

```
    /** with shortUrl
     * @example
     * api('createShape', {shapeType:'schlib', shortUrl:'nxlVIGgQO', from:'system', title:'556_DIL14', x:400, y:300});
     * api('createShape', {shapeType:'FOOTPRINT', shortUrl:'RrkewO60i', from:'system', title:'ARDUINO_PRO_MINI', x:400, y:300});
```

```
 */
/** with jsonCache object
 * @example
 * api('createShape', {
 *   "shapeType": "PAD",
 *   "jsonCache": {
 *     "gId": "gge5",
 *     "layerid": "11",
 *     "shape": "ELLIPSE",
 *     "x": 382,
 *     "y": 208,
 *     "net": "",
 *     "width": 6,
 *     "height": 6,
 *     "number": "1",
 *     "holeR": 1.8,
 *     "pointArr": [],
 *     "rotation": "0"
 *   }
 * });
 *
 * @example
 * api('createShape', {
 *     "shapeType": "polygon",
 *     "stroke": "#000000",
 *     "stroke-width": "1",
 *     "stroke-style": "dashed",
 *     "fill": "none",
 *     "points": [
 *         {"x": 390, "y": 580},
 *         {"x": 450, "y": 450},
 *         {"x": 520, "y": 580},
 *         {"x": 610, "y": 490}
 *     ]
 * });
 *
 * @example
 * api('createShape', {
 *     "shapeType": "arrowhead",
 *     "x": 300,
 *     "y": 300,
 *     "color": "#339933",
 *     "size": "3",
 *     "rotation": 0
 * });
 *
 * @example
 * var ts = ["no_connect_flag", "arrowhead", "busentry", "netLabel_GNd", "netLabel_GnD", "netLabel_gnD", "netLabel_Bar", "netLabel_V
 * for(var i=0;i<ts.length;i++){
 *     api('createShape', {
 *         "shapeType": ts[i],
 *         "x": 300 + i%5*50,
 *         "y": 300 + (i/5|0)*50
 *     });
 * }
**/
/** with cached or pre-defined libs
 * @example
 * api('createShape', {"shapeType": "pcblib", from:'GeneralPackages', title:'C0402', x:400, y:300});
 * @example
 * api('createShape', {"shapeType": "schlib", from:'EasyEDALibs', title:'HDR2X2', x:400, y:300});
 *
**/
/**
 * @example 4
 * api('createShape', {
 *     "shapeType": "schlib",
 *     "gId": "gge6",
 *     "head": {},
 *     "itemOrder": [],
 *     "annotation": {
 *         "gge8": api('createShape', 'annotation', {}),
 *         "gge9": api('createShape', 'annotation', {})
 *     },
 *     "pin": {
 *         "gge11": api('createShape', 'pin', {}),
 *         "gge14": api('createShape', 'pin', {})
 *     },
 *     "polyline": {
 *         "gge10": api('createShape', 'polyline', {}),
 *         "gge12": api('createShape', 'polyline', {})
 *     }
 * });
 *
 * @example 5
 * api('createShape', {
 *     "shapeType": "schlib",
 *     "gId": "gge6",
 *     "head": {},
 *     "children": [
```

```
     *         api('createShape', 'polyline', {}),
     *         api('createShape', 'polyline', {}),
     *         api('createShape', 'pin', {}),
     *         api('createShape', 'pin', {}),
     *         api('createShape', 'annotation', {}),
     *         api('createShape', 'annotation', {})
     *     ]
     * });
     *
     * @example 6
     * api('createShape', {
     *     "shapeType": "schlib",
     *     "gId": "gge6",
     *     "head": {},
     *     "children": api('createShape', [
     *         ['polyline', {}],
     *         ['polyline', {}],
     *         ['pin', {}],
     *         ['pin', {}],
     *         ['annotation', {}],
     *         ['annotation', {}]
     *     ])
     * });
    **/
```

UI

If you want to create an extension, not just a run one time script, maybe need toolbar button. You can check the example before you read.

Create Toolbar Button

```
//@example create a button
api('createToolbarButton', {
 icon:'extensions/theme/icon.svg',
 title:'Theme Colors...',
 fordoctype:'sch,schlib',
 cmd:"extension-theme-setting"
});

* @example toolbar with menu
* api('createToolbarButton', {
*  icon:'extensions/theme/icon.svg',
*  title:'Theme Colors...',
*  fordoctype:'sch,schlib',
*  "menu" : [
*      {"text":"White on Black", "cmd":"extension-theme-WhiteOnBlack"},
*      {"text":"Black on White", "cmd":"extension-theme-BlackOnWhite"},
*      {"text":"Custom Colors...", "cmd":"extension-theme-setting"}
*  ]
* });
```

Create Extension Menu

```
/**
 * @example
 * api('createExtensionMenu', [
 *  {
 *      "text":"Theme Colors...",
 *      "fordoctype": "sch,schlib",
 *      "cmd": "extension-theme-white"
 *  }
 * ]);
 */
```

Create Dialog

check the example

Command List

Clone

```
  // clone gge2 gge3 and return their new ids.
  var newIds = api('clone', {ids:["gge2","gge3"]})
```

Delete

```
  api('delete', {ids:["gge2","gge3"]});
```

Rotate

```
// rotate ids to 90 degree
api('rotate', {ids:["gge2","gge3"],degree:90});
```

Rotate Left

```
//anticlockwise
api('rotate_left', {ids:["gge2","gge3"]});
```

Rotate Right

```
//clockwise
api('rotate_right', {ids:["gge2","gge3"]});
```

Fliph

```
api('fliph', {ids:["gge2","gge3"]});
```

Flipv

```
api('flipv', {ids:["gge2","gge3"]});
```

Align Left

```
api('align_left', {ids:["gge2","gge3"]});
```

Align Right

```
api('align_right', {ids:["gge2","gge3"]});
```

Align Top

```
api('align_top', {ids:["gge2","gge3"]});
```

Align Bottom

```
api('align_bottom', {ids:["gge2","gge3"]});
```

Selection

Change or get selection states of EasyEDA objects in editor.

Select

```
  // gge2 and gge3 will be marked as selected.
  api('select', {ids:["gge2","gge3"]});
```

Select None

```
  //no objects will be selected.
  api('selectNone');
```

Get Selected Ids

```
 var ids = api('getSelectedIds');
```

Move

You can use Update Shape to change the shapes position, but the Move method is better in this case.

Move Objects

Move shapes in relative coordinates, like move the shapes in arrow keys.

```
//Move gge2 and gge3 from left to right in 20pixel or 200mil step
//from top to bottom in 20pixel or 200mil step.
api('moveObjs', {objs:[{gId:"gge2"},{gId:"gge3"}], addX: 20, addY: 20});

//Move gge2 and gge3 from right to left in 20pixel or 200mil step
api('moveObjs', {objs:["gge2","gge3"], addX:-20});

//Move selected objects from left to right in 20pixel or 200mil step
api('moveObjs', {addX:20});
```

Move Objects To

How to move a `VIA` or `junction` to position `{x:'10mil', y:'10mil'}` ?, Move shapes to absolute coordinates.

```
 //Move gge2 and gge3 to Canvas postion 20,20, the real coordinates are dedpend the origin.
 api('moveObjsTo', {objs:[{gId:"gge2"},{gId:"gge3"}], x:20, y:20});

//move gge2 and gge3 to 10mm, 10mm coordinates
```

```
 api('moveObjsTo', {objs:["gge2","gge3"], x: api('coordConvert', {type:'real2canvas',x: '10mm'}), y: api('coordConvert', {type:'real2car

//Move selected objects to Canvas postion 20,20, the real coordinates are dedpend the origin.
 api('moveObjsTo', {x:20, y:20});
```

It is very easy to understand to move a PAD, VIA, Junction to absolution coordinates. But what are the effects of moving TRACK, FOOTPRINT, netlabel to some where. Just try to play the codes, you will find out the regular pattern.

SetOriginXY

EasyEDA's canvas origin is 0,0, you can't change it. But the real coordinates can be mapped to any where.

```
//set the real origin point to canvas x = 400, y = 300. X,Y is pixel all the time.
var result = api('setOriginXY', {x:400,y:300});
```

Coordinate Convert

You can use mm or mil or inch as units, but when you apply the Parameters to SVG graph, you must use coordinate convert.

```
//convert the canvas x 400 to real postion, the value is depent your units and origin point.
var result = api('coordConvert', {type:'canvas2real',x:400})

//the default units is your canvas units, but you can add a units like 300mm.
//if your PCB's units is mil, then you will get the canvas coordinate 400mil,300mm.
 var result = api('coordConvert', {type:'real2canvas',x:400,y:'300mm'});
```

If you set the origin to **0,0**. It is very easy to map the coordinate in your mind, you don't need to use API to convert. the canvas coordinate **100,100** equal the real coordinate **1000mil, 1000mil** or **1inch, 1inch** or **393.7mm, 393.7mm**

Value Convert

How to set the pad's hole size to 20mm? How to set the Track width to 20mil?

```
//the default units is your canvas units, but you can add a units like mm, mil, inch, even pixel.
 var result = api('valConvert', {type:'real2canvas',val:400});
 result = api('valConvert', {type:'real2canvas',val:'400mm'})

 //convert the 400 pixel to real value, the value is depent your units , if the unit is mil, the result should be 4000
 //result = api('valConvert', {type:'canvas2real',val:400})
```

If you can keep in mind 1pixel in canvas equal 10mil, so you don't need this API, you can do it in raw way. For example, If you want to update the track size to 20mil, you can do.

```
api('updateShape', {
   "shapeType": "TRACK",
   "jsonCache": {
       "gId": "gge5",
       "strokeWidth": 2
   }
   });
```

Or

```
api('updateShape', {
   "shapeType": "TRACK",
   "jsonCache": {
       "gId": "gge5",
       "strokeWidth":  api('valConvert', {type:'real2canvas',val:'20mil'})
   }
});
```

Get SVG Arc Path

SVG Arc path Parameter is very complex, We provide a API to convert human read ARC parameter to SVG path.

```
var result = api('getSvgArcPathByCRA', {cx:0, cy:0, rx:90, ry:90, startAngle:0.1, endAngle:0.7, sweepFlag:1});
```

result should be `M89.55037487502231 8.985007498214534A90 90 0 0 1 68.83579685560396 57.97959185139219`

Examples

Check Github example

Enjoy it, if you have any questions, do let us know.

Open File Format

Common Information

EasyEDA is a free, zero-install, Web and cloud-based EDA tool suite, integrating powerful schematic capture, mixed-mode circuit simulation and PCB layout.

EasyEDA team tries to make our users happy. We provide an open ASCII file format. With this file format, you can create a schematic or PCB using some codes, even with Notepad. When you try to add hundreds of LEDs to a schematic or PCB batch, you will find out that you can use codes to create an EasyEDA file, then import it to EasyEDA. It is fun and quick.

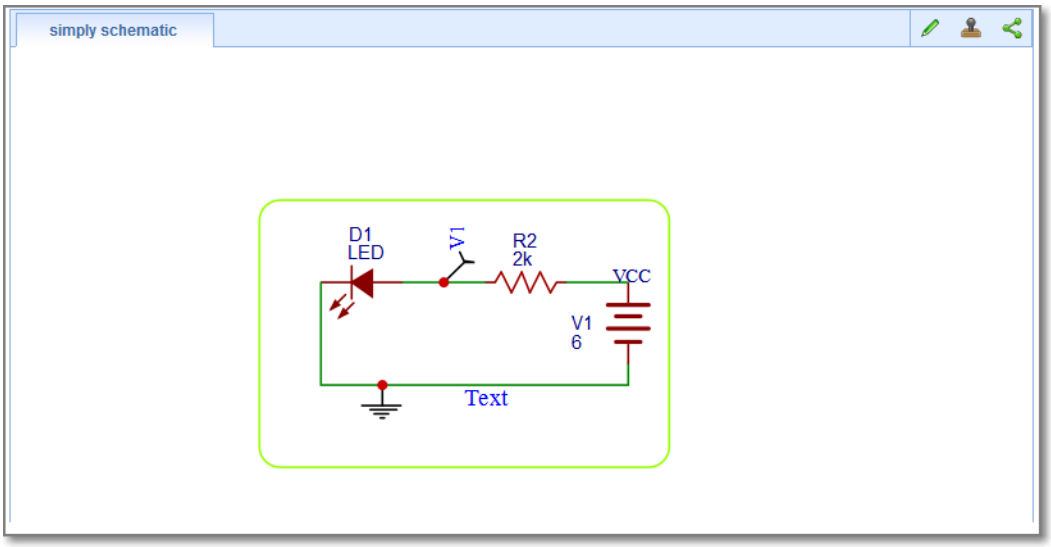
EasyEDA's file is a JSON file, but we compress all of the shape's attributes to a simple string, which will make the file size smaller and saving to server faster. More importantly, with this solution you can create some very big designs. Most browsers will crash when trying to decode the big JSON files. But EasyEDA will provide an **API** to let you to access the EasyEDA friendly JSON object, so you can hack the designs in the EasyEDA editor.

Ok, let's explain them with examples.

Example

Schematic Example

Schematic Example

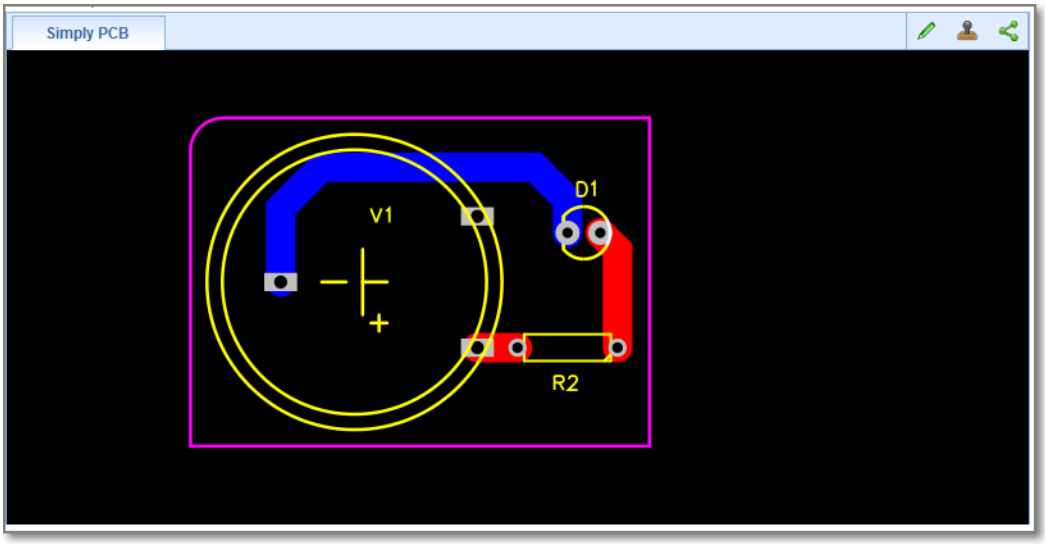


Schematic JSON File Source

check it via github gist Schematic json

PCB Example

PCB Example



PCB JSON File Source

check it via github gist PCB json

General File Struct

Delimiter Mark

From the above JSON source, you can find there are lots of `, ~, ^^ and #@$
characters. These characters are used as **delimiter mark**. These characters are not used frequently in design.

Note: Although these characters were not picked properly at the very beginning, we can't change these, EasyEDA already has lots of existing designs

~ (Tilde)

~ is used to separate the attributes of the shapes. Taking rectangle as an example. `R~170~100~10~10~200~130~#99FF00~1~0~none~gge36~`, when use pure JSON file, it should be look like below, check it via github gist rect json.

So EasyEDA's source is small in file size and will transfer from the internet faster.

`(Back Quote)

` is used to separate the custom attributes.

package`LED3MM stands package:LED3MM

^^(Double Circumflex)

^^ is used to join segments, just used in *netFlag* , *Pin* and *pAD*.

#@$(Octothorpe Ampersat Dollar)

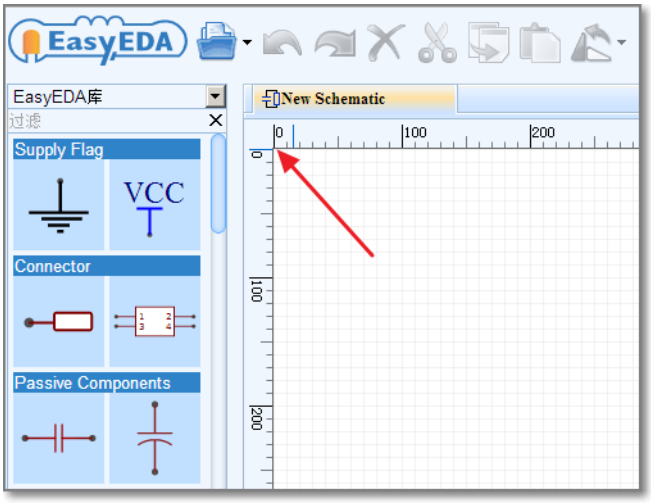
Union the characters **#@$** as a supper mark, it will be used to implode the shapes to a string, it's only used in *Schlib* and *PCBLIB*.

Document Type

check it via github gist document type

Canvas Coordinates

The canvas is a two-dimensional grid.
The upper-left corner of the canvas has the coordinate (0,0)



SVG

EasyEDA uses Scalable Vector Graphics (SVG) which is an XML-based vector image format for two-dimensional graphics to realize the shapes.

Q&A

1. How to check the json file format

Check the EasyEDA source dialog out, copy the text to text area, then click the `Apply` button. That is all.

EasyEDA Schematic File Format

Note: Schematic, Schematic Library, Spice Symbol, Subpart and Subckt are used the same file format. Please check Schematic JSON File Source out before keeping read.

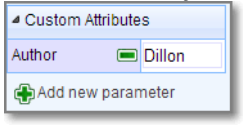
Head

Head information for schematic and subckt.

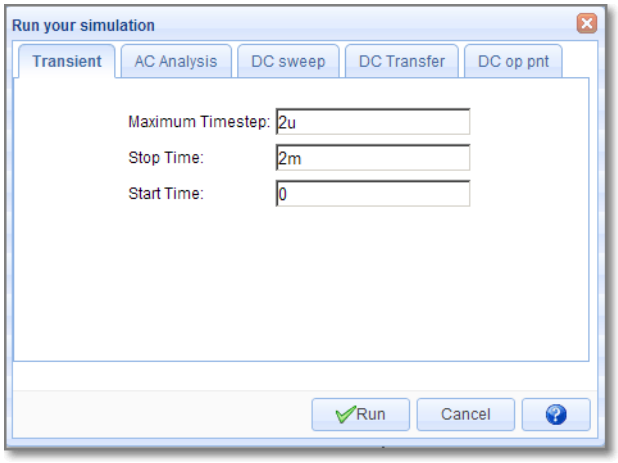
```
"head":"1~1.7.5~Author`Dillon`~TRAN`2u`2m`0`{AC`dec``0`0`{DC`0``0`0`{TF```"
```

Format:

1. document type :`1`
2. document version: `1.7.5`h
3. custom attributes: **key: value** pairs, separate with `, added via **Add new parameter**



4. spice simulation configure store, Now can set four types `tran`, `AC`, `DC`, `TF`, every type split with `{`. When opening the simulation dialog, these information will be listed in like below image.



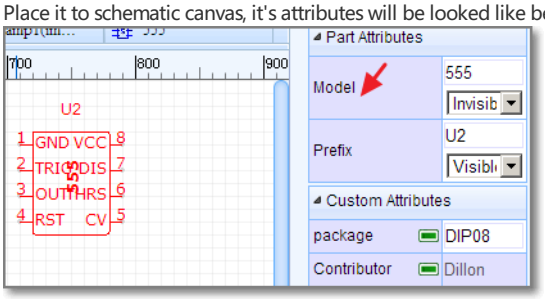
Head information for Schematic Library, Spice Symbol and Subpart

```
"head":"7~1.7.5~400~300~package`DIP08`nameDisplay`0`nameAlias`Model`Model`555`name`555`pre`U?`spicePre``Contributor`Dillon"
```

Format:

1. document type :`7`
2. document version: `1.7.5`
3. origin x position. **Reserved field, can't be changeded**
4. origin y position. **Reserved field, can't be changeded**
5. custom attributes: **key: value** pairs, separate with `, added via **Add new parameter**.
   package: DIP08
   nameDispaly: 0 (hide it is name when placed to schematic)
   nameAlias: Model
   name:555
   pre:U? , when place to schematic, will be marked as U1, U2. subpart will be set as *U?.1*, *U?.2* etc.
   spicePre:X, *X* stands for a subckt.
   sourceId:xxxxxxxxx (just for schematic Lib and spice symbol)

Place it to schematic canvas, it's attributes will be looked like below image. The name field is alias as Model and it is invisible.



Canvas

```
"canvas":"CA~1200~1200~#FFFFFF~yes~#CCCCCC~10~1200~1200~line~10~pixel~5~400~300"
```

Format:
1. command: CA 2. view box width: 1200, View Box Width / Canvas width = scaleX 3. view box height: 1200,View Box Height / Canvas Height = scaleY 4. back ground: #FFFFFF 5. grid visible: yes/none 6. grid color: #CCCCCC 7. grid size: 10 pixel 8. canvas width: 1200 pixel 9. canvas height: 1200 pixel 10. grid style: line/dot 11. snap size: 10 pixel 12. unit: pixel(Always pixel) 13. ALT snap size:5 (Snap Size when pressing the `ALT` Key) 14. origin x position 15. origin y position

Canvas setting image

Shapes

The shape is an array. EasyEDA store various shape in this field, they are different with a command which locate at the begin of the string.

```
"shape":[
    "PL~210 100 260 100~#000000~2~0~none~gge58",
    "R~210~110~~~50~30~#000000~1~0~none~gge61",
    "I~90~90~271~105~0~https://easyeda.com/assets/static/images/logo-140x39.png~gge62",
    "PG~310 100 350 130 300 150 290 150 270 120~#000000~2~0~none~gge64",
    "PT~M230 170 C270 200 270 170 240 150 240 150 240 150 240 150~#000000~2~0~none~gge65"
]
```

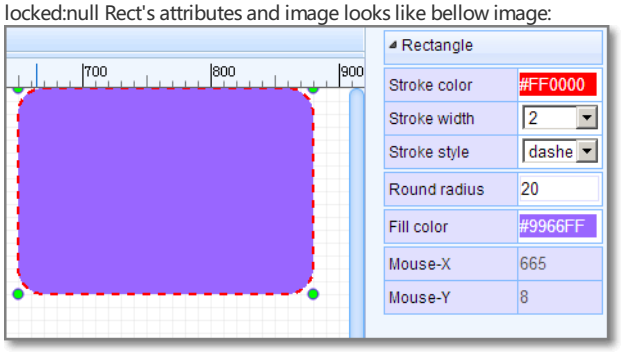
Rectangle

```
"R~650~0~20~20~230~160~#FF0000~2~1~#9966FF~gge5"
```

Format:

Check Rect element of SVG out.

1. command: R
2. x: 650
3. y: 0
4. rx: 20
5. ry: 20
6. width: 230
7. height: 160
8. strokeColor: #FF0000
9. strokeWidth: 2 //pixel
10. strokeStyle: 1
11. fillColor: #9966FF
12. id: gge36
13. locked:null Rect's attributes and image looks like bellow image:

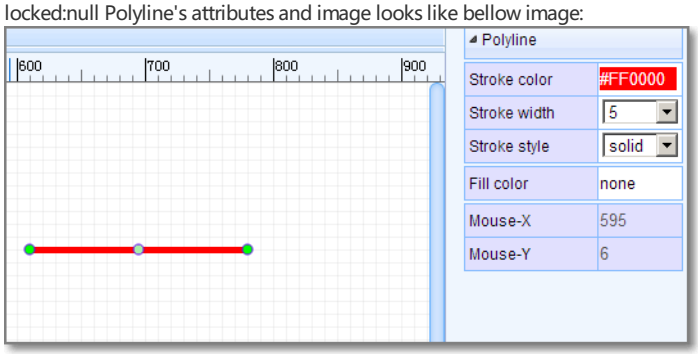


Polyline

```
"PL~610 130 780 130~#FF0000~5~0~none~gge6"
```

Format:
Check Polyline element of SVG out.

1. command: PL
2. points: 610 130 780 130
3. strokeColor: #FF0000
4. strokeWidth: 5 //pixel
5. strokeStyle: 0
6. fillColor: none
7. id: gge6
8. locked:null Polyline's attributes and image looks like bellow image:



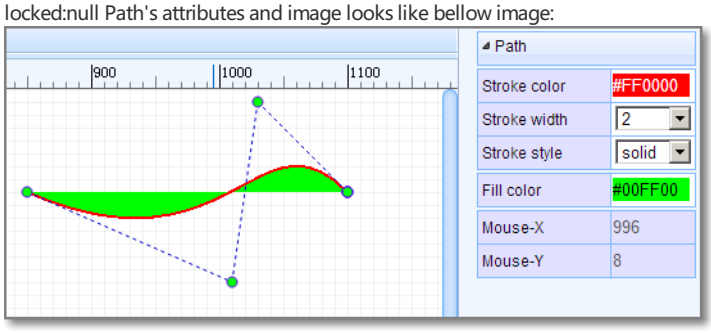
Path

```
"PT~M670 300 C830 370 850 230 920 300 920 300 920 300 920 300~#000000~1~0~none~gge17"
```

Format:
Check Path element of SVG out.

1. command: PT
2. pathString:M670 300 C830 370 850 230 920 300 920 300 920 300 920 300
3. strokeColor: #FF0000

4. strokeWidth: 5 //pixel
5. strokeStyle: 0
6. fillColor: none
7. id: gge6
8. locked:null Path's attributes and image looks like bellow image:



bezier is a **path** too.

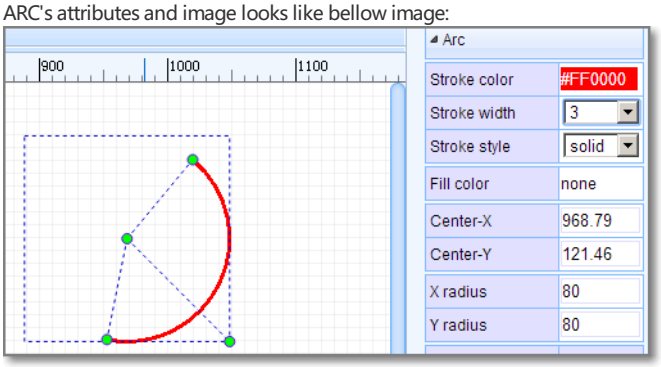
Arc

```
"A~M 1020 60 A 80 80 0 0 1 953.096 199.904~968.78,121.45,1048.785,201.457,1018.785,61.457,948.785,221.45~#FF0000~3~0~none~gge19"
```

Format:
Arc is a **Path** element, Check Path element of SVG out.

1. command: A
2. pathString:M670 300 C830 370 850 230 920 300 920 300 920 300 920 300
3. helperDots: the four green dots
4. strokeColor: #FF0000
5. strokeWidth: 3 //pixel
6. strokeStyle: 0
7. fillColor: none
8. id: gge19
9. locked:null

ARC's attributes and image looks like bellow image:



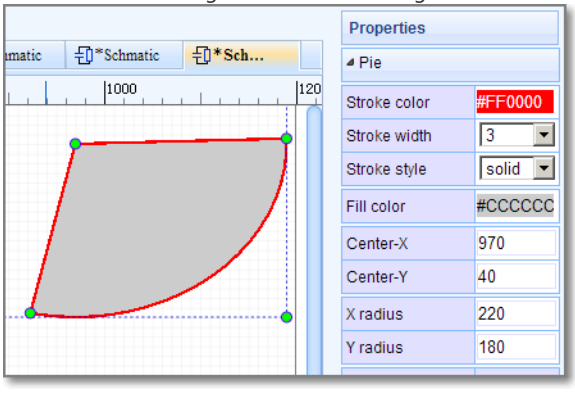
Pie

```
"PI~M 970 40 L 1189.9 34.4509 A 220 180 0 0 1 923.103 215.863 Z~970,40,1190,220,1327.7106323242188,30.973068237304688,923.1032104492188,
```

Pie is a **Path** element, Check Path element of SVG out. Pie is similar with Arc, the pathString has a `Z`

1. command: PI
2. pathString:M 970 40 L 1189.9 34.4509 A 220 180 0 0 1 923.103 215.863 Z
3. helperDots: the four green dots
4. strokeColor: #FF0000
5. strokeWidth: 3 //pixel
6. strokeStyle: 0
7. fillColor: none
8. id: gge19
9. locked:null

Pie's attributes and image looks like bellow image:



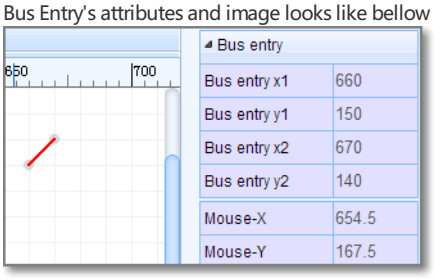
Bus Entry

```
"BE~0~660~150~670~140~gge15"
```

Format:

1. command: BE
2. rotation:0
3. start x1: 660
4. start y1: 150
5. end x1: 670
6. end y1: 140
7. id: gge15
8. locked:null

Bus Entry's attributes and image looks like bellow image:

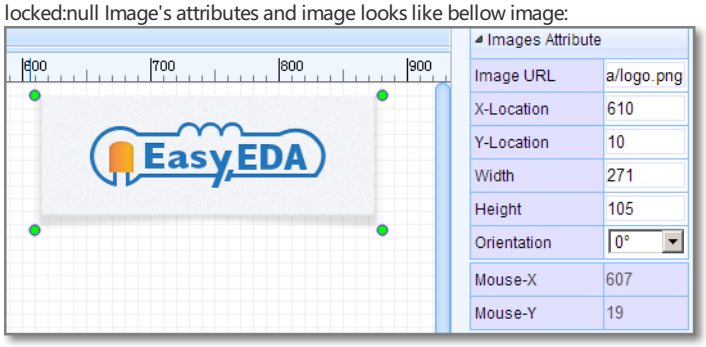


Image

```
"I~610~10~271~105~0~https://easyeda.com/assets/static/images/logo-140x39.png~gge12"
```

Format:
Check Image element of SVG out.

1. command: I
2. x: 610
3. y: 10
4. width: 271
5. height: 105
6. rotation: 0
7. href:https://easyeda.com/assets/static/images/logo-140x39.png
8. id: gge12
9. locked:null Image's attributes and image looks like bellow image:

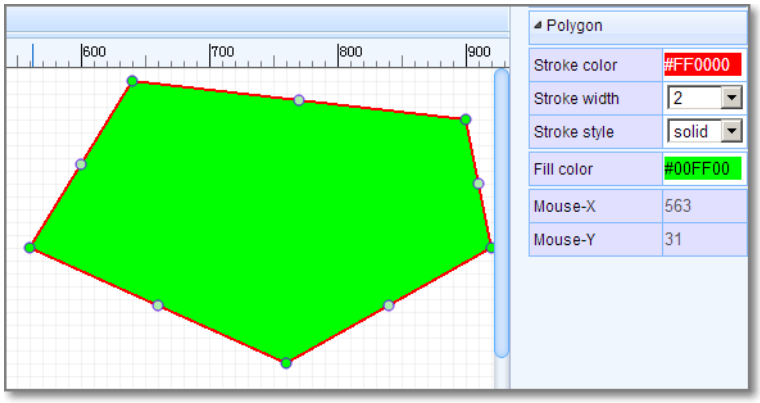


Polygon

```
"PG~640 10 900 40 920 140 760 230 560 140~#FF0000~2~0~#00FF00~gge10"
```

Format:
Check Polygon element of SVG out.

1. command: PG
2. points: 640 10 900 40 920 140 760 230 560 140
3. strokeColor: #FF0000
4. strokeWidth: 2 //pixel
5. strokeStyle: 0
6. fillColor: #00FF00
7. id: gge10
8. locked:null Polygon's attributes and image looks like bellow image:



Line

```
"L~360~160~510~160~#FF0000~2~0~none~gge11"
```

Format:
Check Line element of SVG out.

1. command: L
2. x1:360
3. y1:160
4. x2:510
5. y2:160
6. strokeColor: #FF0000
7. strokeWidth: 2 //pixel
8. strokeStyle: 0
9. fillColor: #00FF00
10. id: gge11
11. locked:null

Circle

```
"C~710~170~105~#FF0000~2~0~#0000FF~gge12"
```

Format:
Check Circle element of SVG out.

1. command: C
2. cx:720
3. cy:90
4. r:105
5. strokeColor: #FF0000
6. strokeWidth: 2 //pixel
7. strokeStyle: 0
8. fillColor: #0000FF
9. id: gge12
10. locked:null

Bus

```
"B~570 130 680 130 680 210~#008800~2~0~none~gge19"
```

Bus is similar with Polyline, Bus is start with `B`, polyline start with `PL`.

Pin

```
"P~show~0~1~670~30~~gge23^^670~30^^M 670 30 h -20~#880000^^1~648~33~0~1~end~~11pt^^1~655~29~0~1~start~~11pt^^0~653~30^^0~M 650 27 L 647
```

A Pin has seven segments, join these segments with ^^ (Double Circumflex) as a string like above.

1. **Pin configure** `P~show~0~1~670~30~~gge23`
    1. command: P
    2. display: show/'' (*bad design, should use yes/none*)
    3. electric: 0, can be ['Undefined', 'Input','Output','I/O','Power']
    4. spice pin number: 1
    5. position x: 670
    6. position y: 30
    7. rotation: null, can be ['null' or 0, '90', '180', '270']
    8. id: gge23
    9. locked: null
2. **pin dot** `670~30`
   The gray dot at the end of the Pin, it is important.

    1. pin dot x: 670

2. pin dot y: 30
3. **pin path** `M 670 30 h -20~#880000`
    1. path: M 670 30 h -20, a 20 pixel horizontal line start from **pin dot**
    2. pin color: #880000
4. **name** `1~648~33~0~1~end~~11pt`
    1. visible : 1/0 stand show or hide
    2. position x: 648
    3. position y: 33
    4. rotation: 0
    5. text: **1**
    6. text anchor: end
    7. font family: null, default is **Verdana**
    8. font size: 11pt, default is 7pt
5. **number** `1~655~29~0~1~start~~11pt`

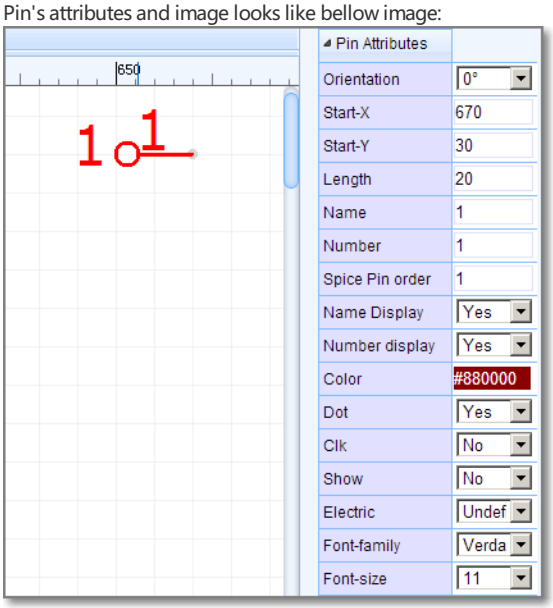
    the same as **name** above

6. **dot** `0~653~30`

    stands for not. a circle with radius in 3 pixel

    1. visible : 0/1 hide / show
    2. circle x: 653
    3. circle y: 30
7. **clock** `0~M 650 27 L 647 30 L 650 33`
    1. visible: 0/1 hide / show
    2. clock path: M 650 27 L 647 30 L 650 33

Pin's attributes and image looks like bellow image:



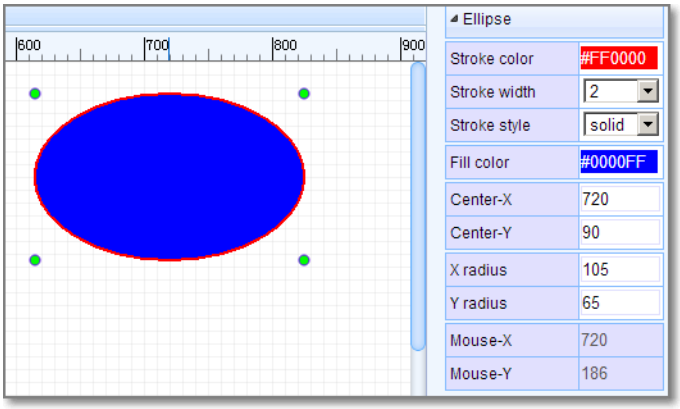
Ellipse

```
"E~720~90~105~65~#FF0000~2~0~#0000FF~gge12"
```

Format:
Check Ellipse element of SVG out.

1. command: E
2. cx:720
3. cy:90
4. rx:105
5. ry:65
6. strokeColor: #FF0000
7. strokeWidth: 2 //pixel
8. strokeStyle: 0
9. fillColor: #0000FF
10. id: gge12
11. locked:null

Ellipse's attributes and image looks like bellow image:



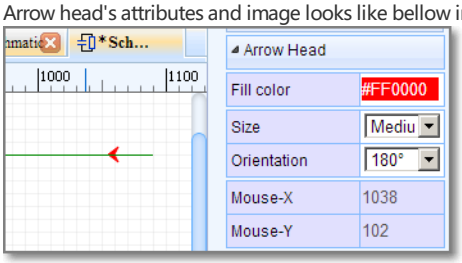
Arrowhead

```
"AR~part_arrowhead~1060~120~gge23~180~M 1060 120 L 1063 126 L 1055 120 L 1063 114 Z~#FF0000"
```

Format:

1. command: AR
2. part Type:part_arrowhead, not used
3. x:1060
4. y:120
5. id:gge23
6. rotation: 180
7. path String: M 1060 120 L 1063 126 L 1055 120 L 1063 114 Z
8. fillColor: #FF0000
9. locked:null

Arrow head's attributes and image looks like bellow image:



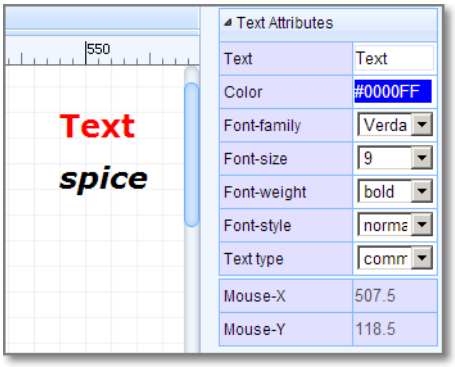
Annotations

```
"T~L~540~60~0~#0000FF~~9pt~bold~normal~~comment~Text~1~start~gge26"
```

Check Text element of SVG out. **Format:**

1. command: T
2. mark: L // `L` = label, `N` = Name, `P` = prefix `N,P` are for Schlib
3. position x:540
4. position y:60
5. rotation:0
6. fill color: #0000FF
7. font family: null, default is **Verdana**
8. font size: 9pt
9. font-weight: bold
10. font style: normal
11. dominant baseline: null
12. text type: comment // **comment** or **spice** command
13. string: Text
14. visible: 1/0 show/hide (use for mark `N` or `P`)
15. text anchor: start (start middle end)
16. id:gge26
17. locked:null

Text's attributes and image looks like bellow image:



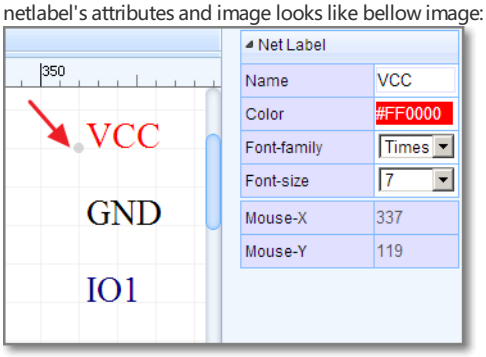
Netlabels

```
    "N~360~100~0~#FF0000~VCC~gge32~start~362~100~Times New Roman~",
```

Format:

1. command: N
2. pin dot x: 360
3. pin dot y: 100
4. rotation: 0
5. fill color: #FF0000
6. name: VCC
7. id: gge32
8. text anchor: start (start middle end)
9. postion x: 362
10. postion y: 100
11. font family: Times New Roman
12. font size:null default is 7pt
13. locked:null

netlabel's attributes and image looks like bellow image:



Netflags

Netflag is very similar with netlabel

```
"F~part_netLabel_gnD~330~110~~gge41^^330~110^^GND~#000080~319~97~0~start~0~Times New Roman~9pt^^PL~330 120 330 110~#000000~1~0~none~gge4
```

A Netflag has several segments, join these segments with ^^(Double Circumflex) as a string like above.

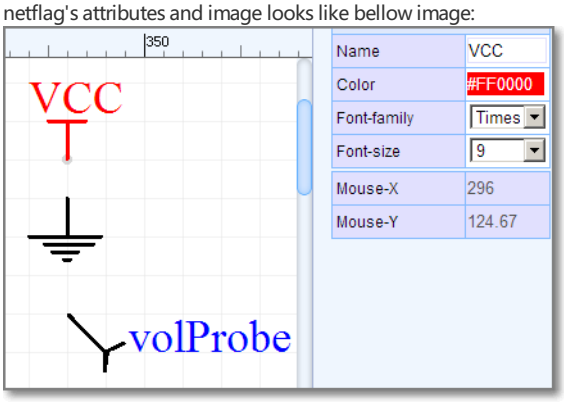
1. **configure** `P~show~0~1~670~30~~gge23`
    1. command: F
    2. part id: part_netLabel_gnD
    3. position x: 330
    4. position y: 110
    5. rotation: null [0, 90, 180, 270]
    6. id: gge41,
    7. locked: null
2. **pin dot** `670~30`
   The gray dot at the end of the Pin, it is important.
    1. pin dot x: 330
    2. pin dot y: 140
3. **mark string** `GND~#000080~319~97~0~start~0~Times New Roman~9pt`
    1. net flag string: GND
    2. color: #000080
    3. position x: 319
    4. position y: 97
    5. rotation: 0 [0, 90, 180, 270]
    6. text anchor: start (start middle end)
    7. visible: 1/0 show/hide the net flag string

8. font family: Times New Roman
9. font size:null default is 7pt

4. **shapes**

   All other items are shapes.

netflag's attributes and image looks like bellow image:



Wire

```
    "W~570 130 680 130 680 210~#008800~2~0~none~gge19"
```

`Wire` is similar with Polyline, `Wire` is start with `W`, polyline start with `PL`.

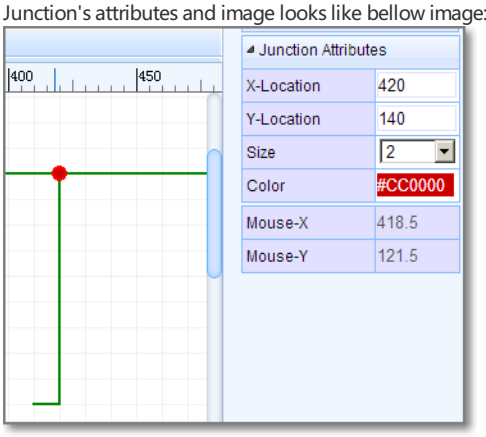
Junctions

```
    "J~420~140~2.5~#CC0000~gge18",
```

Format:

1. command: J
2. pin dot x: 420
3. pin dot y: 140
4. junction circle radius: 2.5 pixel
5. fill color: #CC0000
6. id: gge18
7. locked:null

Junction's attributes and image looks like bellow image:



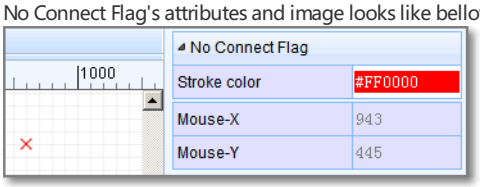
No Connect Flag

```
    "O~960~410~gge5~M956,406 L964,414 M964,406 L956,414~#FF0000"
```

Format:

1. command: O
2. pin dot x: 960
3. pin dot y: 410
4. id: gge5
5. pathStr: M956,406 L964,414 M964,406 L956,414
6. color: #FF0000
7. locked:null

No Connect Flag's attributes and image looks like bellow image:



SchLib

```
"LIB~220~140~package`C1`nameAlias`Value(F)`Value(F)`1u`spicePre`C`spiceSymbolName`Capacitor`~~0~gge66#@$T~N~214~129~0~#000080~Arial~~~~
```

A schlib has several shapes, join these shapes with #@$(Octothorpe Ampersat Dollar) as a string like above.

1. **configure** `LIB~270~140~package`DO35-7`nameAlias`Model`Model`1N4001`spicePre`D`spiceSymbolName`Diode`~~0~gge116`
    1. command: LIB
    2. position x: 270
    3. position y: 140
    4. custom attributes: *package`DO35-7`nameAlias`Model`Model`1N4001`spicePre`D`spiceSymbolName`Diode`*
    5. rotation: 0, can be ['null' or 0, '90', '180', '270']
    6. import flag: 0 just used in import from eagle
    7. id: gge116
    8. locked: null
2. **shapes**

    All other items are shapes.

strokeStyle

- 0 : solid
- 1 : dashed
- 2 : dotted

Q&A

1. Why don't save the Wire, Annotion, netlabel, netflag to Shape field.

These items will be used to create netlist, save them to separate field will make you spent more less time to do this. We don't need to traversal all the shapes.

EasyEDA PCB File Format

Note: PCB and PCB Library are used the same file format. Please check PCB JSON File Source out before keeping read.

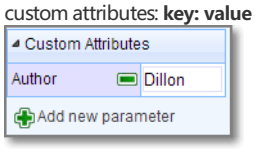
Head

Head information for PCB.

```
"head":"3~1.7.5~Author`Dillon`"
```

Format:

1. document type :`3`
2. document version: `1.7.5`
3. custom attributes: **key: value** pairs, separate with `, added via **Add new parameter**



Head information for PCB Library

```
"4~1.7.5~400~300~`pre`U?`Contributor`Dillon"
```

Format:

1. document type :`4`
2. document version: `1.7.5`
3. origin x position. **Reserved field, can't be changeded**
4. origin y position. **Reserved field, can't be changeded**
5. custom attributes: **key: value** pairs, separate with `, added via **Add new parameter**.

pre:U?, when place to PCB, will be marked as U1, U2. *Contributor:Dillon sourceId:xxxxxxxx* (just for PCB lib)

Parameters Dimensions

EasyEDA support millimeter, inch and millimeter, but when these items are stored to a file, all of them will be expressed as 10X mil. Taking line lengths or widths for examples, stroke width equal 1, stands 10mil.

Canvas

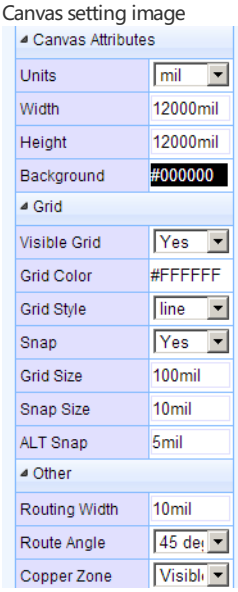
```
"CA~2400~2400~#000000~yes~#FFFFFF~10~1200~1200~line~1~mil~1~45~visible~0.5~400~300"
```

Format:

1. command: CA

2. view box width: 2400(24000 mil), View Box Width / Canvas width = scaleX = 2
3. view box height: 2400(24000 mil),View Box Height / Canvas Height = scaleY = 2
4. back ground: #000000
5. grid visible: yes/none
6. grid color: #FFFFFF
7. grid size: 10(100 mil)
8. canvas width: 1200 (12000 mil)
9. canvas height: 1200 (12000 mil)
10. grid style: line/dot
11. snap size: 1 (10 mil)
12. unit: mil(inch, mil, mm)
13. routing width: 1 (10mil)
14. routing angle: 45 degree(45 90 free)
15. copper area: visible/invisible
16. ALT snap size: 0.5 (5 mil Snap Size when pressing the `ALT` Key)
17. origin x position
18. origin y position

Canvas setting image



System Color

```
"#000000~#FFFFFF~#FFFFFF~#000000~#FFFFFF"
```

Format:

1. future use: #000000
2. future use: #FFFFFF
3. future use: #FFFFFF
4. hole Color: #000000
5. DRC error: #FFFFFF

Layers config

layers is an array, each layer is an item of the layers.

```
 "layers":[
    "1~TopLayer~#FF0000~true~true~true",
    "2~BottomLayer~#0000FF~true~false~true",
    "3~TopSilkLayer~#FFFF00~true~false~true",
    "4~BottomSilkLayer~#808000~true~false~true",
    "5~TopPasterLayer~#808080~false~false~false",
    "6~BottomPasterLayer~#800000~false~false~false",
    "7~TopSolderLayer~#800080~false~false~false",
    "8~BottomSolderLayer~#AA00FF~false~false~false",
    "9~Ratlines~#6464FF~true~false~true",
    "10~BoardOutline~#FF00FF~true~false~true",
    "11~Multi-Layer~#C0C0C0~true~false~true",
    "12~Document~#FFFFFF~true~false~true",
    "21~Inner1~#800000~false~false~false",
    "22~Inner2~#008000~false~false~false",
    "23~Inner3~#00FF00~false~false~false",
    "24~Inner4~#000080~false~false~false"
]
```

Format:

1. layer id: 1
2. layer name: TopLayer
3. layer color: #FF0000
4. visible: true, hints the objects in this layer show or hide

5. active: false. active layer
6. config: true. if be set false, you can't see it on the layer toolbar.

Preference

```
"preference":{
    "hideFootprints":"gge118~gge221~gge227~gge233",
    "hideNets":"BSYNC~DREQ~GPIO0~MICP~GND"
}
```

`hideFootprints` : when the id of the footprints in here, you can't see them on canvas. `hideNets` : when the net name in here, you can't see them on canvas, you can hide the ratline at here too. There are some guys would like to hide then GND ratline, then use copper area to connect all the GND pad.

DRC Rule

```
 "DRCRULE":{
    "trackWidth":0.7,
    "track2Track":0.7,
    "pad2Pad":0.8,
    "track2Pad":0.8,
    "hole2Hole":1,
    "holeSize":1.6
}
```

`trackWidth`: 0.7 (7 mil) track width `track2Track`: 0.7 (7 mil) track to track distance `pad2Pad`: 0.8(8 mil) pad to pad distance `track2Pad`: 0.8(8 mil) track to pad distance `hole2Hole`: 1(10 mil) hole to hole distance `holeSize`: 1.6(16 mil) hole diameter

This is a simple DRC, more later.

Shapes

The shape is an array. EasyEDA store various shape in this field, they are different with a command which locate at the begin of the string.

```
"shape":[
    "TRACK~1~1~S$19~311 175 351 175 352 174~gge18",
    "PAD~ELLIPSE~329~185~6~6~11~~1~1.8~~0~gge20",
    "VIA~329~202~3.2~~0.8~gge23",
    "COPPERAREA~2px~1~GND~349 247 492 261 457 314 339 329~1~solid~gge27~spoke~none~[]",
    "SOLIDREGION~1~~350 146 483 146 447 228 371 220~solid~gge26"
]
```

Unit

EasyEDA takes **10 mil** as a basic factor, when a stroke width is 1, we can take it as 1*10mil = 10mil, is 2, we can take it as 2*10mil = 20mil

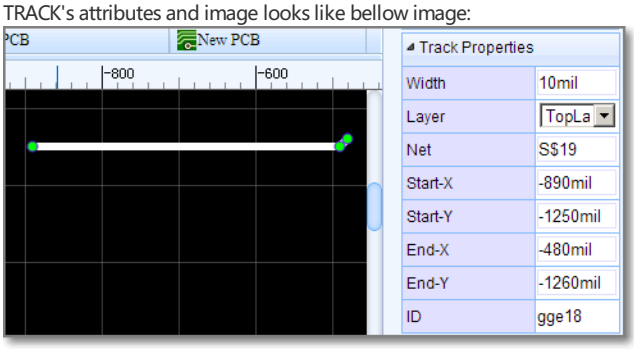
TRACK

```
"TRACK~1~1~S$19~311 175 351 175 352 174~gge18"
```

Format: Check Polyline element of SVG out.

1. command: TRACK
2. stroke Width: 1 (10 mil)
3. layer id: 1 (TopLayer)
4. net: "S$19"
5. points: 311 175 351 175 352 174
6. id : gge18
7. locked: null

TRACK's attributes and image looks like bellow image:



COPPERAREA

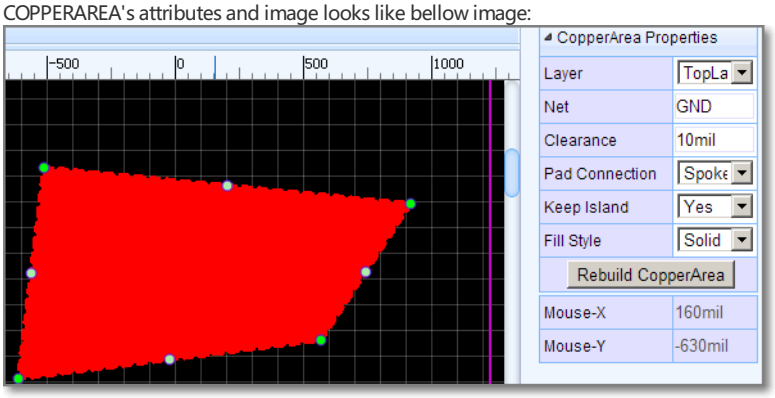
```
"COPPERAREA~2px~1~GND~349 247 492 261 457 314 339 329~1~solid~gge27~spoke~yes~[[\"M339,329 349,247 492,261 457,314z\"]]"
```

Format:

1. command: COPPERAREA
2. stroke Width: 2 (20 mil)

3. layer id: 1 (TopLayer)
4. net: GND
5. points: 349 247 492 261 457 314 339 329
6. clearance Width : 1 (10 mil)
7. fill style: solid/none
8. id: gge27
9. thermal: spoke/direct
10. keep island: none/yes
11. copper zone: [[\"M339,329 349,247 492,261 457,314z\"]] rings and holes
12. locked: null

COPPERAREA's attributes and image looks like bellow image:



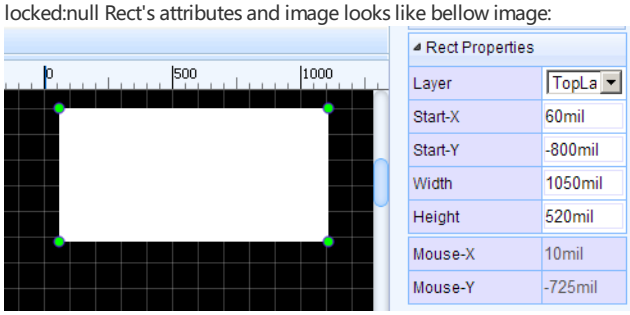
RECT

```
"RECT~406~220~105~52~1~gge32"
```

Format:

Check Rect element of SVG out.

1. command: RECT
2. x: 406
3. y: 220
4. width: 105
5. height: 52
6. layer id:1
7. id: gge36
8. locked:null Rect's attributes and image looks like bellow image:



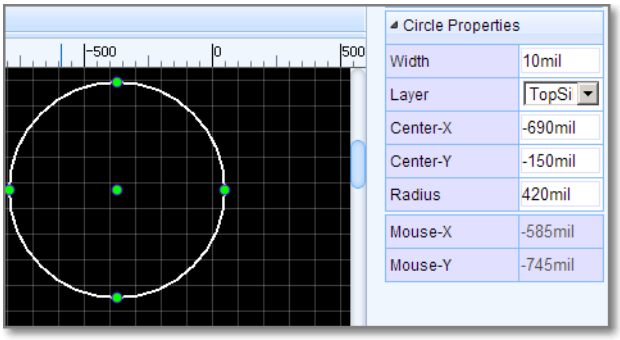
CIRCLE

```
"CIRCLE~363~273~42~1~3~gge33"
```

Format: Check Circle element of SVG out.

1. command: CIRCLE
2. cx:363 (3630 mil)
3. cy:273
4. r:42 (420 mil)
5. stroke width: 1 (10mil)
6. layer id: 3 (Top silk layer)
7. id: gge33
8. locked:null

CIRCLE's attributes and image looks like bellow image:



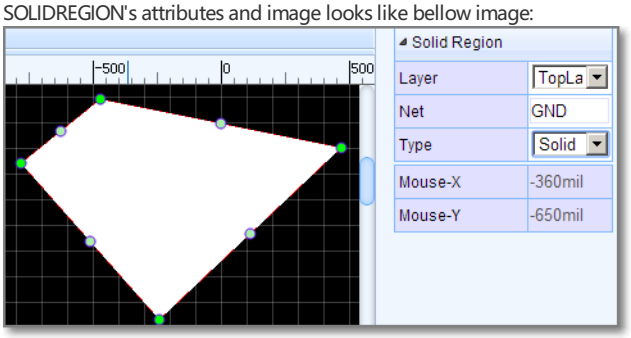
SOLIDREGION

```
"SOLIDREGION~1~GND~322 256 376 317 447 250 353 231~solid~gge34"
```

Format:

1. command: SOLIDREGION
2. layer id: 1 (Toplayer)
3. net: GND
4. points:322 256 376 317 447 250 353 231
5. type: solid/cutout/npth
6. id: gge34
7. locked:null

SOLIDREGION's attributes and image looks like bellow image:



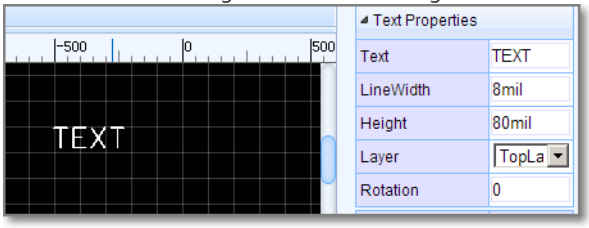
TEXT

```
"TEXT~L~351~252~0.8~0~none~1~~8~TEXT~M 352.55 250.64 L 352.55 258.27 M 350 250.64 L 355.09 250.64 M 357.49 250.64 L 357.49 258.27 M 357
```

Format:

1. command: TEXT
2. type: L/P (L = label, P = prefix)
3. position x: 351 (3510 mil)
4. position y: 252 (2520 mil)
5. stroke width: 0.8 (8 mil)
6. rotation: 0
7. mirror : none (not user now)
8. layer id: 1 (Toplayer)
9. net: ''
10. font size: 8 (80 mil in height)
11. string: TEXT
12. text path: M 352.55 250.64 L 352.55 258.27 M 350 250.64 L 355.09 250.64 M 357.49 250.64 L 357.49 258.27 M 357.49 250.64 L 362.22 250.64 M 357.49 254.27 L 360.4 254.27 M 357.49 258.27 L 362.22 258.27 M 364.62 250.64 L 369.71 258.27 M 369.71 250.64 L 364.62 258.27 M 374.65 250.64 L 374.65 258.27 M 372.11 250.64 L 377.2 250.64
13. display: '' (none = hide, other = show)
14. id: gge35
15. locked: null

TEXT's attributes and image looks like bellow image:



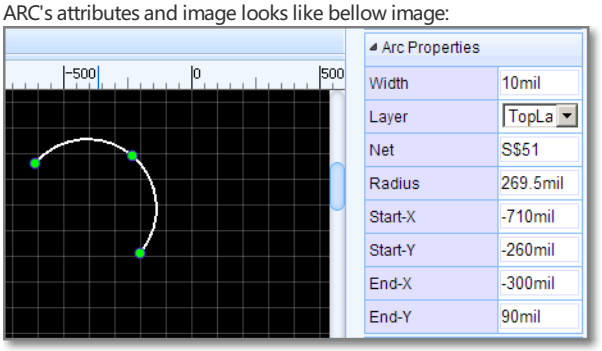
Arc

```
"ARC~1~1~S$51~M329,274 A26.95,26.95 0 0 1 370,309~~gge50"
```

Format: Arc is a **Path** element, Check Path element of SVG out.

1. command: ARC
2. stroke width: 1 (10 mil)
3. layer id: 1 (Toplayer)
4. net: S$51
5. path string: M329,274 A26.95,26.95 0 0 1 370,309
6. helper dots: the four green dots, no need in PCB, keep it blank
7. id: gge19
8. locked:null

ARC's attributes and image looks like bellow image:



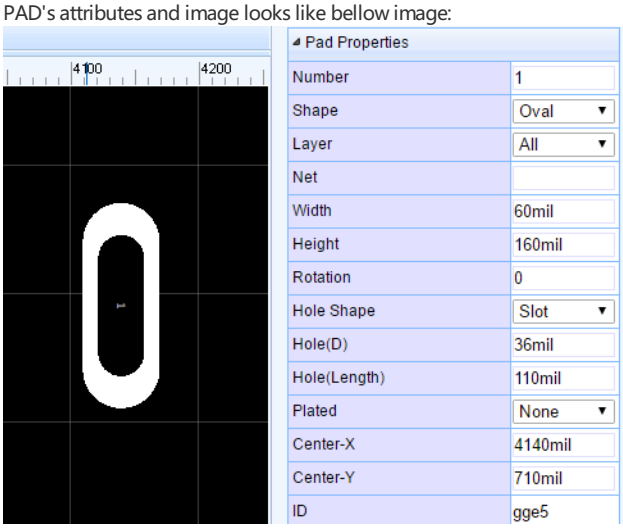
PAD

```
"PAD~OVAL~814~371~6~16~11~~1~1.8~814 366 814 376~0~gge5~11~814 374.7 814 367.3~N"
```

Format:

1. command: PAD
2. shape: ELLIPSE/RECT/OVAL/POLYGON
3. center x: 814
4. center y: 371
5. width: 6 (60 mil)
6. height: 16 (160 mil)
7. layer id: 11 (All)
8. net: ''
9. number: 1
10. hole radius: 1.8 (18 mil)
11. points: '' (ELLIPSE = '', RECT = outline points)
12. rotation: 0 [0 - 360]
13. id: gge19
14. Hole(Length): 11 (110mil)
15. Hole Points: 814 374.7 814 367.3 // slot hole from to point
16. Plated:Y/N
17. locked:null

PAD's attributes and image looks like bellow image:



VIA

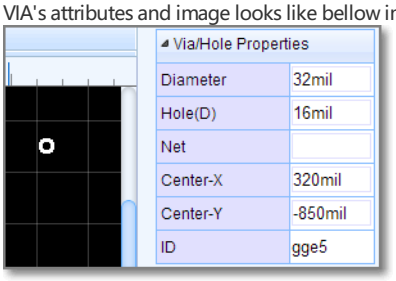
```
"VIA~432~215~3.2~~0.8~gge5"
```

Format:

1. command: VIA
2. center x: 432
3. center y: 215
4. diameter: 3.2

5. net : ''
6. hole radius: 0.8 (8 mil)
7. id: gge5
8. locked:null

VIA's attributes and image looks like bellow image:



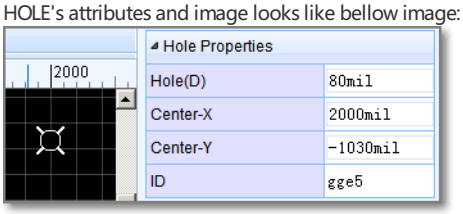
HOLE

```
"HOLE~284~255~4~gge5"
```

Format:

1. command: HOLE
2. center x: 284
3. center y: 255
4. diameter: 4
5. id: gge5
6. locked:null

HOLE's attributes and image looks like bellow image:



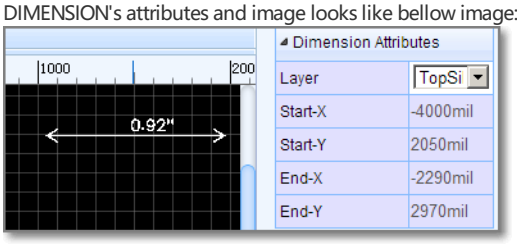
DIMENSION

```
"DIMENSION~3~M 301 217 L 442 217 M 306 220 L 301 217 L 306 214 M 437 220 L 442 217 L 437 214 M 369.5 209.82 L 370.05 209.55 L 370.86 208
```

Format:

1. command: DIMENSION
2. layer id: 3 (Top Silk layer)
3. path: M 301 217 L 442 217 M 306 220 L 301 217 L 306 214 M 437 220 L 442 217 L 437 214 M 369.5 209.82 L 370.05 209.55 L 370.86 208.73 L 370.86 214.45 M 372.94 213.09 L 372.66 213.36 L 372.94 213.64 L 373.21 213.36 L 372.94 213.09 M 377.74 208.73 L 375.01 212.55 L 379.1 212.55 M 377.74 208.73 L 377.74 214.45 M 380.9 209.82 L 381.45 209.55 L 382.26 208.73 L 382.26 214.45 M 384.06 208.73 L 384.06 210.64 M 386.25 208.73 L 386.25 210.64
4. id: gge5
5. locked:null

DIMENSION's attributes and image looks like bellow image:



DIMENSION just allows to change it layer id, if you don't accept this DIMENSION, delete it and redraw again.

PCBlib

```
"LIB~245~240~package`CK17-B`~~~gge15~1#@$TEXT~P~295~219.5~0.7~0~~3~~4.5~C1~M 298.07 218.07L297.86 217.66 L297.45 217.25 L297.05 217.05 L
```

A PCBlib has several shapes, join these shapes with #@$(Octothorpe Ampersat Dollar) as a string like above.

1. **configure** `LIB~245~240~package`CK17-B`~0~~gge15~1`

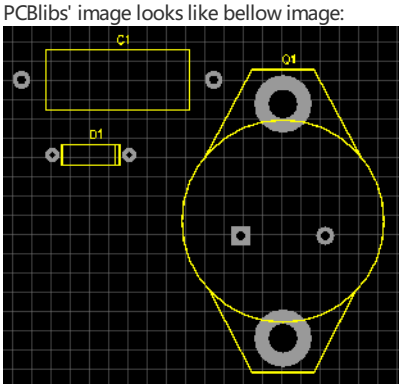
    1. command: LIB
    2. position x: 270
    3. position y: 140
    4. custom attributes: *package`CK17-B`*
    5. rotation: 0, can be [0 - 360]
    6. import flag: '', just used in import from eagle

7. id: gge115
8. locked: null

2. **shapes**

   All other items are shapes.

PCBlibs' image looks like bellow image:



EasyEDA Schematic File Object

Note: Schematic, Schematic Library, Spice Symbol, Subpart and Subckt use the same format.

EasyEDA Schematic File Object is a JSON Object which allows you to hack your designs via other languages, such as Javascript, Python, PHP, C, C++. The interesting thing is that your can control/modify your design in EasyEDA editor via Javascript language.

Rules

JSON Keys

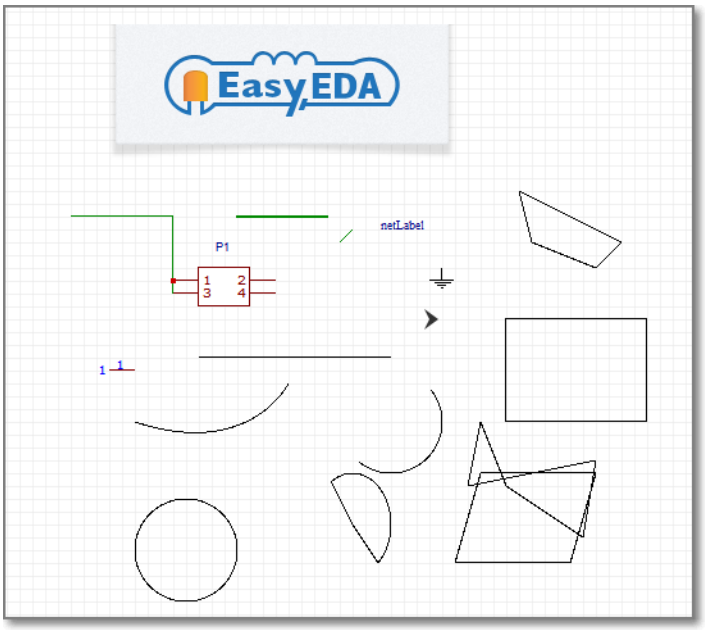
Every EasyEDA graph unit has an unique key, such as "**wire**", "**schlib**", "**junction**", "**bus**", "**busentry**", "**netlabel**", "**netflag**", "**pin**", "**polyline**", "**path**", "**arc**", "**rect**", "**polygon**", "**arrowhead**", "**ellipse**", "**image**"

itemOrder key

Because an object in EasyEDA is an unordered set of name/value pairs in JSON format, but EasyEDA's graphs are ordered. We need an array to store the order of these objects. Every schematic lib has an itemOrder key and the whole JSON object has an itemOrder key.

Example

File



Open Schematic Example

wire

```
"wire":{
    "gge48":{
        "gId":"gge48",
        "strokeColor":"#008800",
        "strokeWidth":"1",
        "strokeStyle":0,
        "fillColor":"none",
        "pointArr":[
            {
                "x":290,
                "y":430
```

```
            },
            {
                "x":370,
                "y":430
            },
            {
                "x":370,
                "y":490
            }
        ]
    }
    ..........
}
```

All wires will be stored to **wire** key, their id will be taken as the key such as `gge48`.

Schlib

All schematic components will be stored to **schlib**, their id will be taken as the key such as `gge7`. Schematic component JSON is a little bit complicated, it has lots of other **JSON Keys**, such as `polyline`, `image`, `path` etc.

Note: please check the other shapes format via below JSON example

JSON example

check the complete JSON object via github gist Schematic Json object

EasyEDA PCB File Object

Note: PCB and Package use the same format.

EasyEDA PCB File Object is a JSON Object which allows you to hack your designs via another language, such as Javascript, Python, PHP, C, C++. The interesting thing is that your can control/modify your design in EasyEDA editor via Javascript. So you can use codes to create your own outline.

Rules

JSON Keys

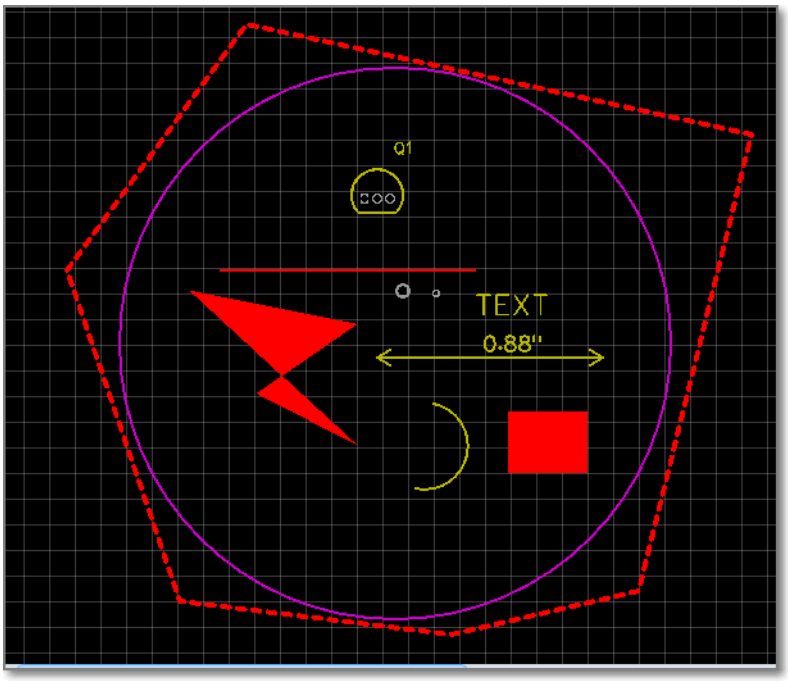
Every EasyEDA graph unit has an unique key, such as "TRACK", "PAD", "VIA", "TEXT", "DIMENSION", "FOOTPRINT", "ARC", "RECT", "CIRCLE", "HOLE", "COPPERAREA", "SOLIDREGION", "DRCRULE", "FABRICATION"

itemOrder key

Because of an object in EasyEDA is an unordered set of name/value pairs in JSON format, but EasyEDA's graphs are ordered. We need an array to store the order of these objects. Every package has an itemOrder key and the whole JSON object has an itemOrder key.

Example

File



Open PCB Example

TRACK

```
"TRACK":{
    "gge6":{
        "gId":"gge6",
```

```
        "layerid":"1",
        "net":"S$7",
        "pointArr":[
            {
                "x":357,
                "y":171
            },
            {
                "x":456,
                "y":171
            }
        ],
        "strokeWidth":1
    }
    ......
},
```

All tracks will be stored to **TRACK** key, their id will be taken as the key such as `gge6`.

SIGNALS

EasyEDA groups all of the objects with the same net name in one array.

FOOTPRINT

All packages will be stored to **FOOTPRINT**, their id will be taken as the key such as `gge7`. PCB package JSON is little bit complicated, it has lots of other **JSON Keys**, such as `TRACK`, `ARC`, `RECT` etc.

Note: please check the other shapes format via below JSON example.

JSON example

Check the complete JSON object via github gist PCB Json object