

---

# **Qucs Help Documentation**

***Version 0.0.18***

**Qucs Team (2014)**

20 January 2015



<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Getting Started with Digital Simulations</b>	<b>7</b>
2.1	VHDL File Component . . . . .	7
<b>3</b>	<b>Getting Started with Optimization</b>	<b>9</b>
<b>4</b>	<b>Getting Started with Octave Scripts</b>	<b>15</b>
4.1	Postprocessing . . . . .	15
<b>5</b>	<b>Brève description des man<sup>1</sup>/<sub>2</sub>uvres</b>	<b>17</b>
5.1	General Actions . . . . .	17
5.2	bouton gauche S . . . . .	17
5.3	“Insert Component”-Mode . . . . .	18
5.4	clic gauche I . . . . .	18
5.5	clic gauche C . . . . .	18
5.6	électionne le fichier . . . . .	18
5.7	Keyboard . . . . .	19
<b>6</b>	<b>Utiliser les sous-circuits</b>	<b>21</b>
6.1	Subcircuits with Parameters . . . . .	22
<b>7</b>	<b>Glossaire des fonctions mathématiques disponibles</b>	<b>23</b>
7.1	Operators . . . . .	23
7.2	Math Functions . . . . .	24
7.3	Electronics Functions . . . . .	27
7.4	Nomenclature . . . . .	28
7.5	Constants . . . . .	29
<b>8</b>	<b>List of Special Characters</b>	<b>31</b>
<b>9</b>	<b>Matching Circuits</b>	<b>33</b>
9.1	2-Port Matching Circuits . . . . .	33
<b>10</b>	<b>Installed Files</b>	<b>35</b>
10.1	Command line arguments . . . . .	35
<b>11</b>	<b>Schematic File Format</b>	<b>37</b>
11.1	Propriétés . . . . .	37

11.2 Symbol . . . . . 38

11.3 Composants . . . . . 38

11.4 Wires . . . . . 38

11.5 Diagrams . . . . . 39

11.6 Paintings . . . . . 39

Contents :



---

# Introduction

---

Qucs (prononcez : kiouks) est un simulateur de circuit muni d'une interface graphique. Il peut effectuer plusieurs types de simulations (ex : point de fonctionnement, calcul des paramètres s...). Ce texte constitue une introduction rapide au maniement de Qucs.

Quand vous démarrez Qucs la première fois, il crée un répertoire « .qucs » dans votre répertoire de base. Chaque fichier est ensuite enregistré dans ce répertoire ou l'un de ses sous-répertoires. Après que Qucs s'est chargé, il ouvre une fenêtre semblable à celle de la figure 1. À droite se trouve l'espace de travail (6) qui contient les schémas, les graphiques, etc. Avec les onglets (5) du dessus, vous pouvez choisir aisément n'importe quelle feuille ouverte. À gauche de l'espace de travail se trouve une autre zone (1), dont le contenu dépend de l'état de l'onglet qui la surmonte : « Projets » (2), « Contenu » (3) et « Composants » (4). Au lancement de Qucs, l'onglet « Projets » (2) est actif par défaut. Comme il s'agit de la première fois que vous exécutez Qucs, la fenêtre est vide, puisque vous n'avez encore jamais travaillé sur un quelconque projet. Cliquez sur le bouton « Nouveau » juste au-dessus de la zone (1). Une boîte de dialogue apparaît. Tapez-y le nom de votre premier projet, par exemple : « premier » puis cliquez sur « OK ». Qucs crée alors un répertoire projet dans ~/.qucs, « premier.prj ». Tous les fichiers inclus dans ce projet y seront par la suite enregistrés. Le nouveau projet est immédiatement activé (comme en témoigne la barre de titre de la fenêtre) et l'onglet actif passe alors à « Contenu » (3), ce qui signifie que les fichiers du projet courant sont affichés. Vous n'avez pas encore de fichiers, donc cliquez sur le bouton « Enregistrer » de la barre d'outils (ou utilisez l'entrée de menu : Fichier -> ; Enregistrer) pour enregistrer le document sans titre qui occupe l'espace de travail. Qucs vous demandera de baptiser ce nouveau fichier. Saisissez « premierschema » puis cliquez sur « Ok ».

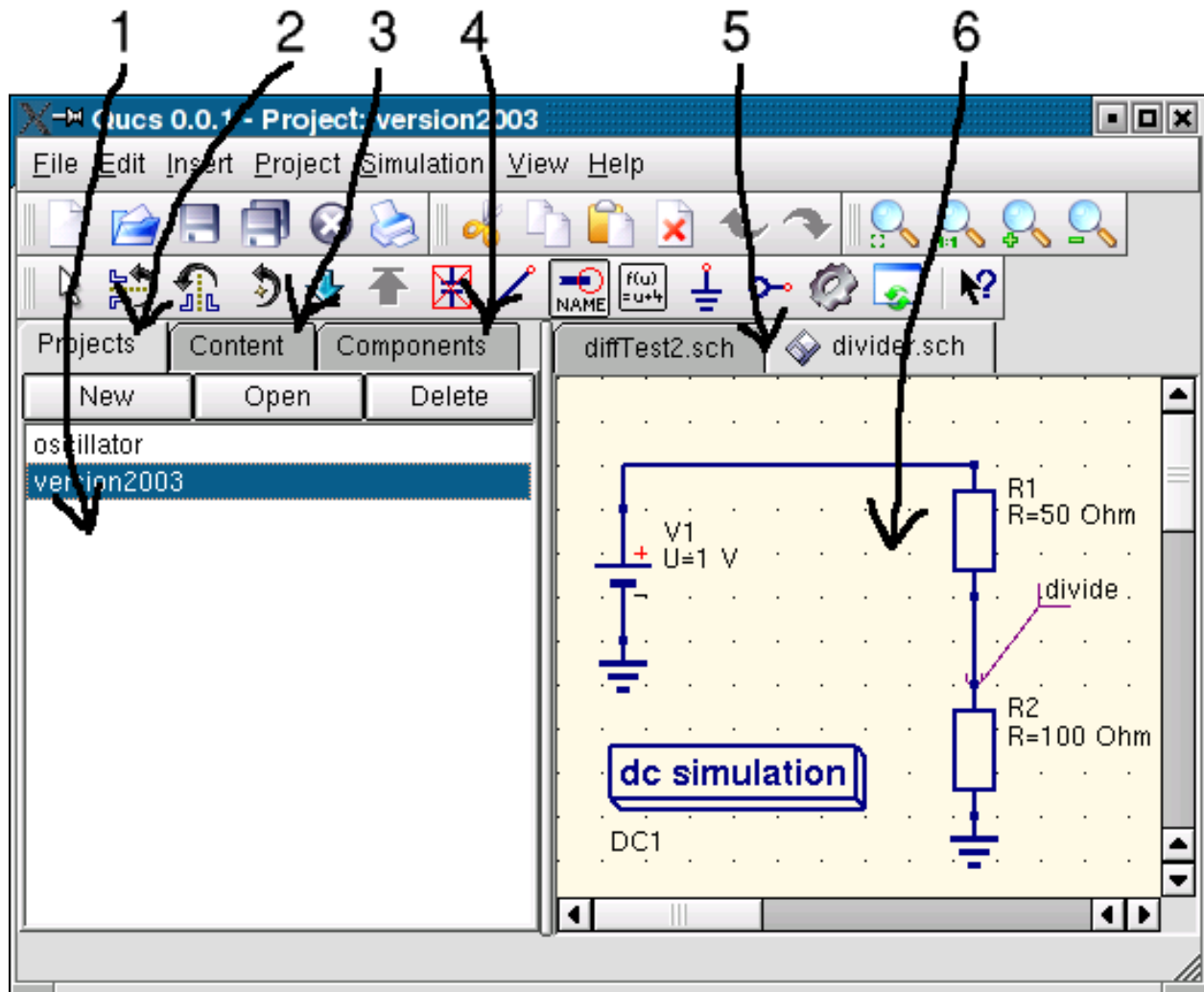


Figure 1 - La fenêtre principale de Qucs

Maintenant, nous allons simuler le point de fonctionnement du circuit de la figure (1). Activez l'onglet « Composants » ((4) dans la figure 1). S'affiche alors une réglette de choix intitulée du groupe (type) de composants à choisir, et, en-dessous, tous les composants faisant partie de ce groupe. Choisissez « Éléments localisés » et cliquez sur le premier symbole, « Résistance ». Lorsque vous amenez le curseur sur la zone de travail, vous voyez un fantôme de symbole de résistance accroché à la flèche. Un clic gauche fait pivoter ce fantôme d'un quart de tour, un clic droit place le symbole du composant sur la feuille. Saisissez ainsi tous les composants visibles figure 1. La source de tension se trouve dans le groupe « Sources », la masse dans le groupe « Éléments localisés » ou directement dans la barre d'outils ; la simulation désirée est définie par l'un des métasymboles qui se trouvent dans le groupe « Simulations ». Pour modifier la valeur de la seconde résistance, double-cliquez la. Cela ouvre une boîte de dialogue où vous pouvez saisir la nouvelle valeur : ici 100  $\Omega$ . Cliquez sur « Entrée » une fois terminé.

Enfin, vous devez nommer l'équipotentielle dont vous voulez calculer la tension. Cliquez sur le bouton « Nommer » de la barre d'outils (ou, par le menu : Insérer -> Nom d'équipotentielle), puis sur le fil voulu. Une boîte apparaît où vous pouvez saisir le nom. Saisissez « divide » et cliquez sur « Ok ». Votre circuit devrait ressembler plus ou moins à celui de la figure 1 (sauf pour les intitulés en français !).

Pour relier les composants entre-eux, cliquez le bouton représentant un fil (équipotentielle), ou passez par le menu : Insérer -> Équipotentielle. Déplacez le curseur sur une broche en l'air (repérée par un petit cercle rouge à son extrémité). Cliquez une fois pour démarrer le tracé du fil, déplacez-vous vers le point d'arrivée du fil et cliquez une nouvelle fois. Si vous souhaitez entrer explicitement des points de passage, cliquez à chaque point. Vous pouvez aussi laisser le



fil se finir en l'air en double-cliquant.

Pour démarrer le calcul, cliquez le bouton « Simuler » (ou le menu : Simulation -> ; Simuler). Une fenêtre s'ouvre pour vous indiquer le déroulement de la simulation. Si celle-ci se termine correctement, le programme enchaîne immédiatement sur la feuille de résultat. Ici, tout s'exécute si vite que vous ne devriez rien voir, à part une bref scintillement. Maintenant, il vous faut placer un graphique pour connaître les résultats. À gauche, le groupe « graphiques » a été automatiquement activé. Choisissez le type « Table », déplacez la souris dans l'espace de travail et cliquez. Une boîte de dialogue s'ouvre, vous pouvez y choisir les valeurs à afficher dans ce nouveau graphique (ici une table). À gauche, vous devriez voir apparaître le nom d'équipotentielle que vous avez saisi : « Divide ». Double-cliquez dessus, il se copie dans la partie droite. Validez avec « OK ». Maintenant, devant vos yeux ébahis, vous lisez : 0,6666667 V. Quelle puissance !



---

## Getting Started with Digital Simulations

---

Qucs is also a graphical user interface for performing digital simulations. This document should give you a short description on how to use it.

For digital simulations Qucs uses the FreeHDL program (<http://www.freehdl.seul.org>). So the FreeHDL package as well as the GNU C++ compiler must be installed on the computer.

There is no big difference in running an analog or a digital simulation. So having read the Getting Started for analog simulations, it is now easy to get a digital simulation work. Let us compute the truth table of a simple logical AND cell. Select the digital components in the combobox of the components tab on the left-hand side and build the circuit shown in figure 1. The digital simulation block can be found among the other simulation blocks.

The digital sources *S1* and *S2* are the inputs, the node labeled as *Output* is the output. After performing the simulation, the data display page opens. Place the diagram *truth table* on it and insert the variable *Output*. Now the truth table of a two-port AND cell is shown. Congratulation, the first digital simulation is done !

**image0\_FR!**

Figure 1 - La fenêtre principale de Qucs

A truth table is not the only digital simulation that Qucs can perform. It is also possible to apply an arbitrary signal to a circuit and see the output signal in a timing diagram. To do so, the parameter *Type* of the simulation block must be changed to *TimeList* and the duration of the simulation must be entered in the next parameter. The digital sources have now a different meaning : They can output an arbitrary bit sequence by defining the first bit (low or high) and a list that sets the times until the next change of state. Note that this list repeats itself after its end. So, to create a 1GHz clock with pulse ratio 1 :1, the list writes : 0.5ns ; 0.5ns

To display the result of this simulation type, there is the diagram *timing diagram*. Here, the result of all output nodes can be shown row by row in one diagram. So, let's have fun...

### 2.1 VHDL File Component

More complex and more universal simulations can be performed using the component "VHDL file". This component can be picked up from the component list view (section "digital components"). Nevertheless the recommended usage is the following : The VHDL file should be a member of the project. Then go to the content list view and click on the file name. After entering the schematic area, the VHDL component can be placed.

The last entity block in the VHDL file defines the interface, that is, all input and output ports must be declared here. These ports are also shown by the schematic symbol and can be connected to the rest of the circuit. During simulation the source code of the VHDL file is placed into the top-level VHDL file. This must be considered as it causes some limitations. For example, the entity names within the VHDL file must differ from names already given to subcircuits. (After a simulation, the complete source code can be seen by pressing F6. Use it to get a feeling for this procedure.)



---

## Getting Started with Optimization

---

For circuit optimization, Qucs uses the ASCO tool (<http://asco.sourceforge.net/>). A brief description on how to prepare your schematic, execute and interpret the results is given below. Before using this functionality, ASCO must be installed on the computer.

Optimization of a circuit is nothing more than the minimization of a cost function. It can either be the delay or the rise time of a digital circuit, or the power or gain of an analog circuit. Another possibility is defining the optimization problem as a composition of functions, leading in this case to the definition of a figure-of-merit.

To setup a netlist for optimization two things must be added to the already existing netlist : insert equation(s) and the optimization component block. Take the schematic from Figure 1 and change it until you have the resulting schematic given in Figure 2.

**limage0\_FR!**

Figure 1 - Initial schematic.

**limage1\_FR!**

Figure 2 - Prepared schematic.

Now, open the optimization component and select the optimization tab. From the existing parameters, special attention should be paid to 'Maximum number of iterations', 'Constant F' and 'Crossing over factor'. Over- or underestimation can lead to a premature convergence of the optimizer to a local minimum or, a very long optimization time.

**limage2\_FR!**

Figure 3 - Optimization dialog, algorithm options.

In the Variables tab, defining which circuit elements will be chosen from the allowed range, as shown in Figure 4. The variable names correspond to the identifiers placed into properties of components and **not** the components' names.

**limage3\_FR!**

Figure 4 - Optimization dialog, variables options.

Finally, go to Goals where the optimization objective (maximize, minimize) and constraints (less, greater, equal) are defined. ASCO then automatically combines them into a single cost function, that is then minimized.

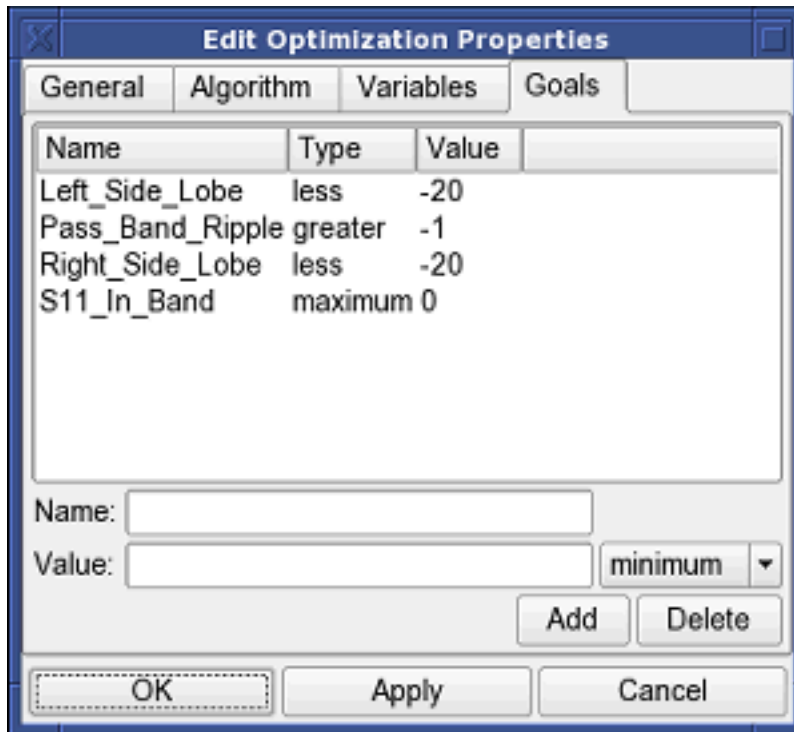


Figure 5 - Optimization dialog, goals options.

The next step is to change the schematic, and define which circuit elements are to be optimized. The resulting schematic is shown in Figure 6.

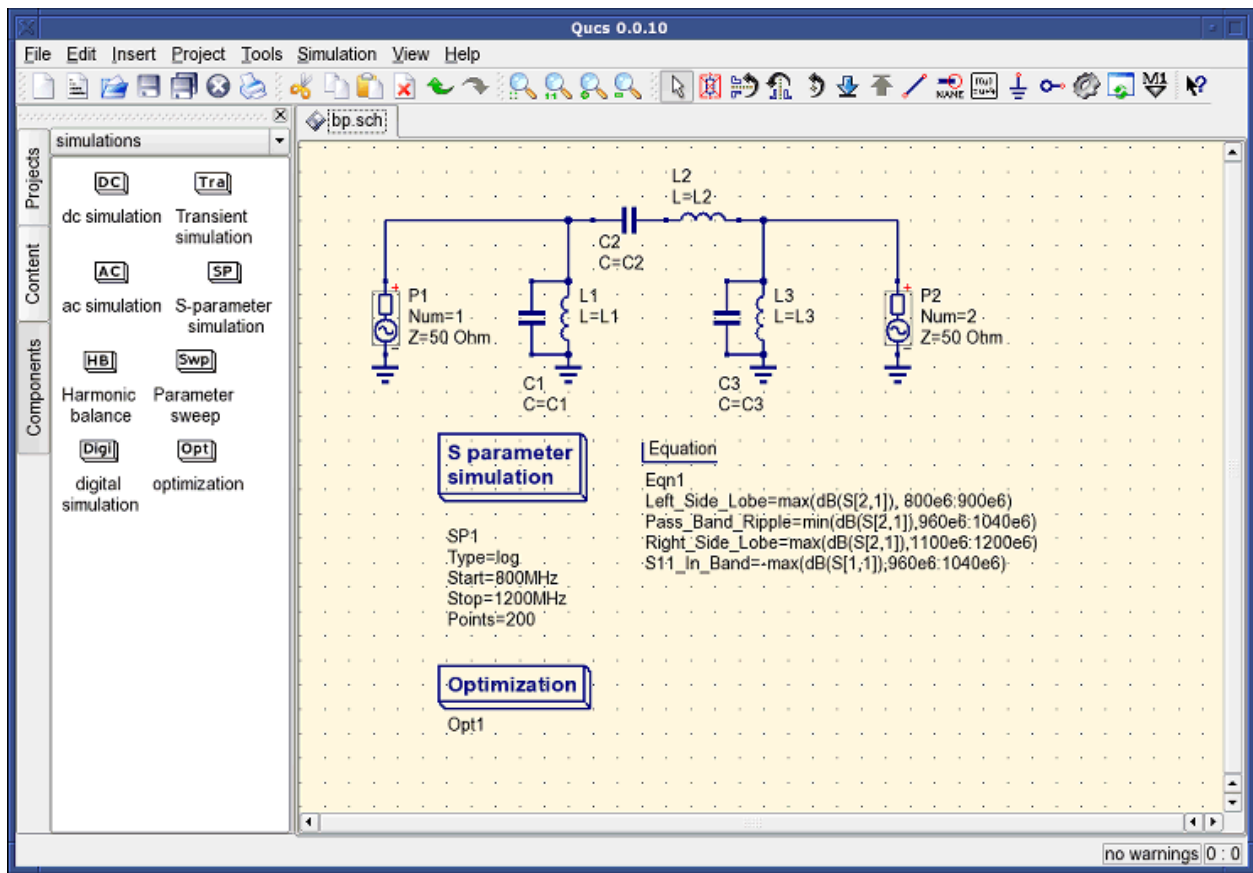


Figure 6 - New Qucs main window.

The last step is to run the optimization, i.e. the simulation by pressing F2. Once finished, which takes a few seconds on a modern computer, the best simulation results is shown in the graphical waveform viewer.

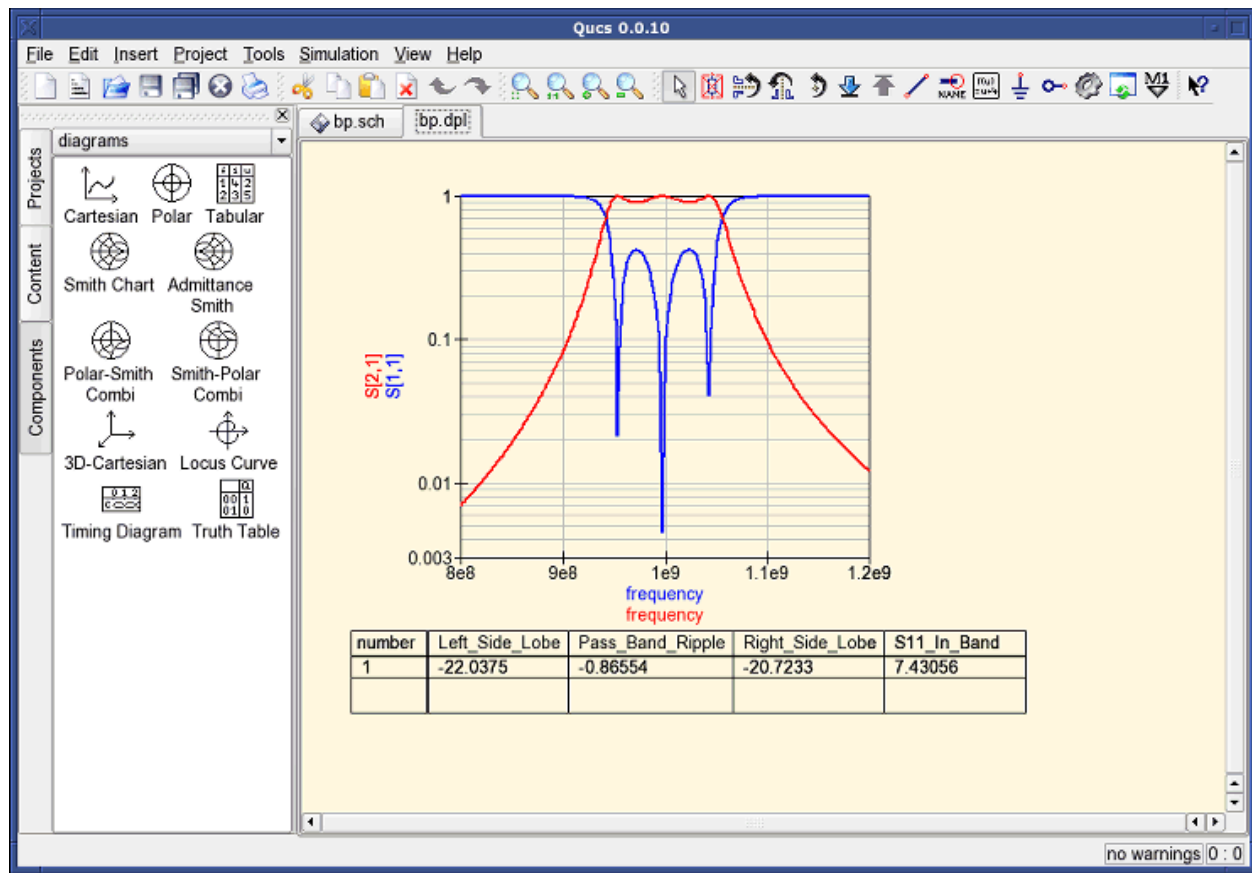


Figure 7 - Qucs results window.

The best found circuit sizes can be found in the optimization dialog, in the Variables tab. They are now the initial values for each one of introduced variables (Figure 8).



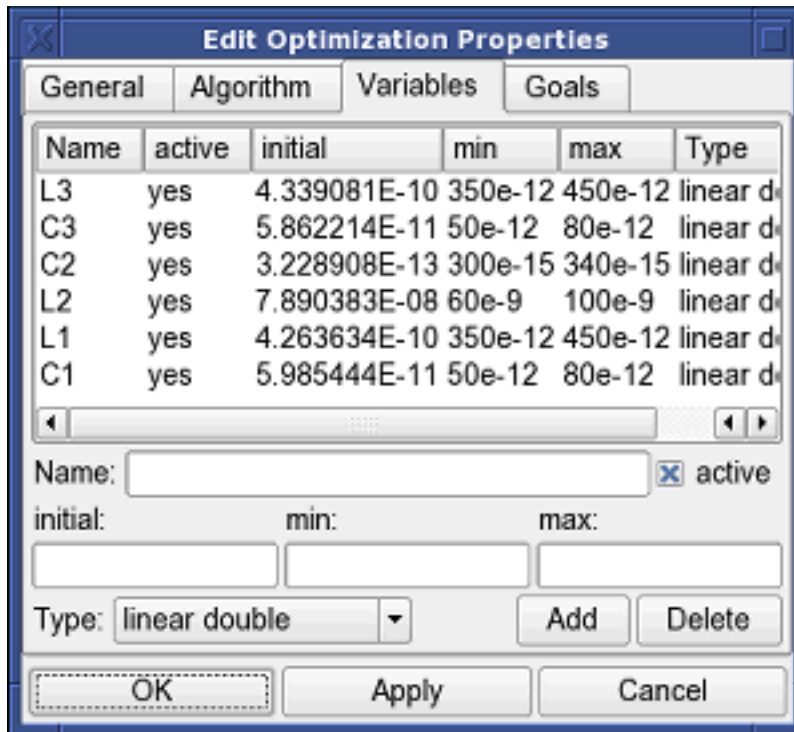


Figure 8 - The best found circuit sizes.



---

## Getting Started with Octave Scripts

---

Qucs can also be used to develop Octave scripts (see <http://www.octave.org>). This document should give you a short description on how to do this.

If the user creates a new text document and saves it with the Octave extension, e.g. 'name.m' then the file will be listed at the Octave files of the active project. The script can be executed with F2 key or by pressing the simulate button in the toolbar. The output can be seen in the Octave window that opens automatically (per default on the right-hand side). At the bottom of the Octave window there is a command line where the user can enter single commands. It has a history function that can be used with the cursor up/down keys.

There are two Octave functions that load Qucs simulation results from a dataset file : `loadQucsVariable()` and `loadQucsDataset()`. Please use the help function in the Octave command line to learn more about them (i.e. type `help loadQucsVariable` and `help loadQucsDataset`).

### 4.1 Postprocessing

Octave can also be used for automatic postprocessing of a Qucs simulation result. This is done by editing the data display file of a schematic (Document Settings... in File menu). If the filename of an Octave script (filename extension m) from the same project is entered, this script will be executed after the simulation is finished.



## Brève description des man1/2uvres

### 5.1 General Actions

(valid in all modes)

mouse wheel	Scrolls vertically the drawing area. You can also scroll outside the current size.
roulette + Shift T	Scrolls horizontally the drawing area. You can also scroll outside the current size.
roulette + Ctrl M	modification de l'échelle du dessin
drag'n'drop file into document area	Tries to open file as Qucs schematic or data display.

### 5.2 bouton gauche S



(Menu : Edit->Select)

clic droit P	Selects the element below the mouse cursor. If several components are placed there, you can clicking several times in order to select the wanted one. Keeping the mouse button pressed, you can move the component below the mouse cursor and all selected ones. If you want to fine position the components, press the CTRL key during moving and the grid is disabled. Keeping the mouse button pressed without any element below it opens a rectangle. After releasing the mouse button, all elements within this rectangle are selected. A selected diagram or painting can be resized by pressing the left mouse button over one of its corners and moving by keeping the button pressed. After clicking on a component text, it can be edited directly. The enter key jumps to the next property. If the property is a selection list, it can only be changed with the cursor up/down keys. Clicking on a circuit node enters the "wire mode".
left mouse button + Ctrl Button	Allows more than one element to be selected, i.e. selecting an element does not deselect the others. Clicking on a selected element deselects it. This mode is also valid for selecting by opening a rectangle (see item before).
double-clic droit T	uvre une boîte pour modifier les caractéristiques d'un objet (nom, valeur des paramètres...)
double-click right mouse button	Opens a dialog to edit the element properties (The labels of wires, the parameters of components, etc.).

## 5.3 “Insert Component”-Mode

Place a new instance of the component onto the schematic

clic droit P	Place a new instance of the component onto the schematic.
double-clic droit T	(Menu : Insérer -> Équipotentielle)

## 5.4 clic gauche I



(Menu : Insert->Wire)

clic droit P	Inverse la direction d'un angle du fil
double-clic droit T	Changes the direction of the wire corner (first left/right or first up/down).
double-click right mouse button	Ends a wire without being on a wire or a port.

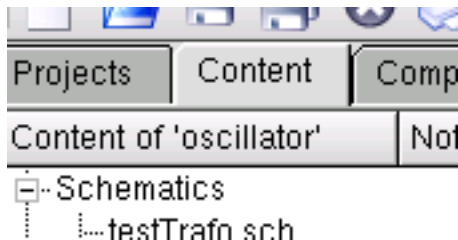
## 5.5 clic gauche C



(Menu : Edit->Paste)

clic droit P	Place the elements onto the schematic (from the clipboard).
double-clic droit T	Rotate the elements.

## 5.6 électionne le fichier



double-clic clic droit A	Ouvre le fichier Opens file. Displays menu with : “open”	— open selected file
right click	“rename”	— change name of selected file
	“delete”	— delete selected file
	“delete group	— delete selected file and its relatives (schematic, data display, dataset)

## 5.7 Keyboard

Many actions can be activated/done by the keyboard strokes. This can be seen in the main menu right beside the command. Some further key commands are shown in the following list :

Flèche gauche ou droite C	Deletes the selected elements or enters the delete mode if no element is selected.
Cursor left/right Tabulation	change la position des inspecteurs dans des diagrammes multi-variables Si aucun inspecteur n’est sélectionné, translate la courbe sélectionnée Si rien n’est sélectionné, sert d’ascenseur Changes the position of selected markers on more-dimensional graphs. If no marker is selected, move selected elements. If no element is selected, scroll document area.
Tabulator	Changes to the next open document (according to the TabBar above).





## Utiliser les sous-circuits

Subcircuits are used to bring more clarity into a schematic. This is very useful in large circuits or in circuits, in which a component block appears several times.

In Qucs, each schematic containing a subcircuit port is a subcircuit. You can get a subcircuit port by using the toolbar, the components listview (in lumped components) or the menu (Insert->Insert port). After placing all subcircuit ports (two for example) you need to save the schematic (e.g. CTRL-S). By taking a look into the content listview (figure 1) you see that now there is a “2-port” right beside the schematic name (column “Note”). This note marks all documents which are subcircuits. Now change to a schematic where you want to use the subcircuit. Then press on the subcircuit name (content listview). By entering the document area again, you see that you now can place the subcircuit into the main circuit. Do so and complete the schematic. You can now perform a simulation. The result is the same as if all the components of the subcircuit are placed into the circuit directly.

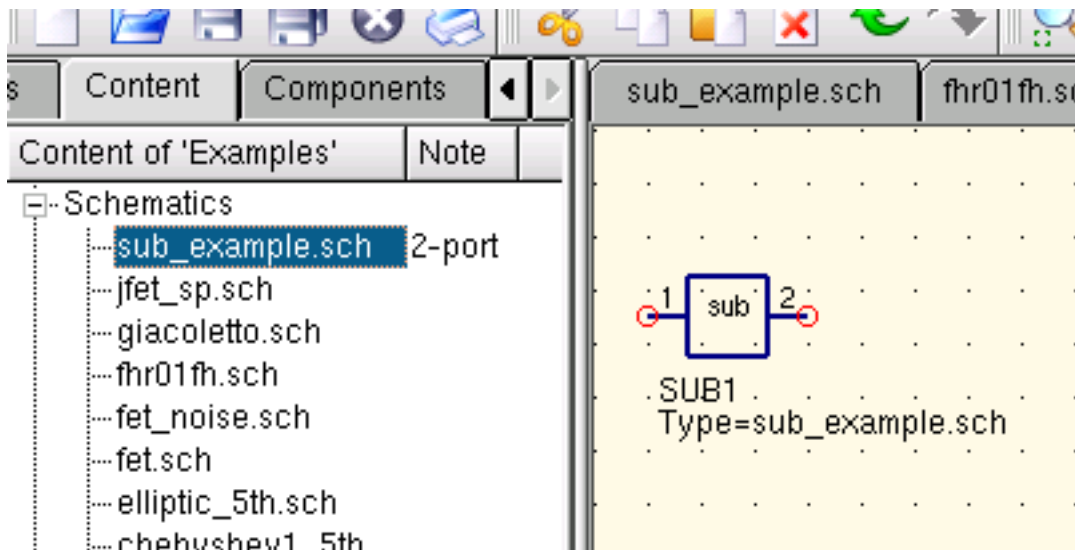


Figure 1 - Accessing a subcircuit

Lorsque vous sélectionnez le métasymbole d'un sous-circuit, vous pouvez en voir le contenu en tapant Ctrl-I (ou son équivalent dans la barre d'outil ou le menu). Pour revenir au schéma principal, tapez Ctrl-H.

If you do not like the component symbol of a subcircuit, you can draw your own symbol and put the component text at your favourite position. Just make the subcircuit schematic the current and go to the menu : File->Edit Circuit Symbol. If you have not yet drawn a symbol for this circuit. A simple symbol is created automatically. You now can edit this symbol by painting lines and arcs. After finished, save it. Now place it on another schematic, and you have a new symbol.

Just like all other components, subcircuits can have parameters. To create your own parameters, go back to the editor

where you edited the subcircuit symbol and double-click on the subcircuit parameter text (SUB1 in the Figure 1). A dialog apperas and you now can fill in parameters with default values and descriptions. When you are ready, close the dialog and save the subcircuit. In every schematic where the subcircuit is placed, it owns the new parameters which can be edited as in all other components.

## 6.1 Subcircuits with Parameters

A simple example using subcircuits with parameters and equations is provided here.

Create a subcircuit :

- Create a new project
- New schematic (for subcircuit)
- Add a resistor, inductor, and capacitor, wire them in series, add two ports
- Save the subcircuit as RLC.sch
- Give value of resistor as 'R1'
- Add equation 'ind = L1',
- Give value of inductor as 'ind'
- Give value of capacitor as 'C1'
- Enregistrer
- File > Edit Circuit Symbol
- Double click on the 'SUB File=name' tag under the rectangular box
  - Add name = R1, default value = 1
  - Add name = L1, default value = 1
  - Add name = C1, default value = 1
  - OK

Insert subcircuit and define parameters :

- New schematic (for testbench)
- Save Test\_RLC.sch
- Project Contents > pick and place the above RLC subcircuit
- Add AC voltage source (V1) and ground
- Add AC simulation, from 140Hz to 180Hz, 201 points
- Set on the subcircuit symbol
  - R1=1
  - L1=100e-3
  - C1=10e-6
- Simuler
- Add a Cartesian diagram, plot V1.i
- The result should be the resonance of the RLC circuit.
- The parameters of the RLC subcircuit can be changed on the top schematic.

---

## Glossaire des fonctions mathématiques disponibles

---

The following operations and functions can be applied in Qucs equations. For detailed description please refer to the “Measurement Expressions Reference Manual”. Parameters in brackets “[ ]” are optional.

### 7.1 Operators

#### 7.1.1 Arithmetic Operators

$+x$	Unary plus
$-x$	Unary minus
$x+y$	Addition
$x-y$	Subtraction
$x*y$	Multiplication
$x/y$	Division
$x\%y$	Modulo (remainder of division)
$x^y$	$\max(x)$

#### 7.1.2 Logical Operators

$!x$	Negation
$x\&y$	And
$x  y$	Or
$x^y$	Exclusive or
$x?y:z$	Abbreviation for conditional expression “if $x$ then $y$ else $z$ ”
$x==y$	Equal
$x!=y$	module
$x<y$	idem $\text{abs}(x)$
$x<=y$	carré du module
$x>y$	conjugué
$x>=y$	Larger than or equal

## 7.2 Math Functions

### 7.2.1 Vectors and Matrices : Creation

<code>eye (n)</code>	Creates $n \times n$ identity matrix
<code>length (y)</code>	Returns the length of the given vector
<code>linspace (from, to, n)</code>	Creates a real vector with $n$ linearly spaced components between <code>from</code> and <code>to</code> , both inclusively
<code>logspace (from, to, n)</code>	Creates a real vector with $n$ logarithmically spaced components between <code>from</code> and <code>to</code> , both inclusively

### 7.2.2 Vectors and Matrices : Basic Matrix Functions

<code>adjoint (x)</code>	Adjoint matrix of $x$ (transposed and conjugate complex)
<code>det (x)</code>	Determinant of a matrix $x$
<code>inverse (x)</code>	Inverse matrix of $x$
<code>transpose (x)</code>	Transposed matrix of $x$ (rows and columns exchanged)

### 7.2.3 Elementary Mathematical Functions : Basic Real and Complex Functions

<code>abs (x)</code>	Absolute value, magnitude of complex number
<code>angle (x)</code>	Phase angle in radians of a complex number. Synonym for <code>arg ()</code>
<code>arg (x)</code>	Phase angle in radians of a complex number
<code>conj (x)</code>	Conjugate of a complex number
<code>deg2rad (x)</code>	Converts phase from degrees into radians
<code>hypot (x, y)</code>	Euclidean distance function
<code>imag (x)</code>	Imaginary part of a complex number
<code>mag (x)</code>	arg cosinus hyperbolique
<code>norm (x)</code>	Square of the absolute value of a vector
<code>phase (x)</code>	Phase angle in degrees of a complex number
<code>polar (m, p)</code>	Transform from polar coordinates (magnitude $m$ , phase $p$ ) into complex number
<code>rad2deg (x)</code>	Converts phase from radians into degrees
<code>real (x)</code>	Real part of a complex number
<code>sign (x)</code>	Signum function
<code>sqr (x)</code>	conversion paramètres $s$ vers $Y$
<code>sqrt (x)</code>	Square root
<code>unwrap (p[, tol[, step])</code>	Unwraps the angle $p$ (in radians – default <code>step</code> is $2\pi$ ) using the optional tolerance value <code>tol</code> (default is $\pi$ )

### 7.2.4 Elementary Mathematical Functions : Exponential and Logarithmic Functions

<code>exp (x)</code>	Exponential function to basis $e$
<code>limexp (x)</code>	Limited exponential function
<code>log10 (x)</code>	Decimal logarithm
<code>log2 (x)</code>	Binary logarithm
<code>ln (x)</code>	Natural logarithm (base $e$ )

### 7.2.5 Elementary Mathematical Functions : Trigonometry

<code>cos (x)</code>	Cosine function
<code>cosec (x)</code>	Cosecant
<code>cot (x)</code>	Cotangent function
<code>sec (x)</code>	Secant
<code>sin (x)</code>	Sine function
<code>tan (x)</code>	Tangent function

### 7.2.6 Elementary Mathematical Functions : Inverse Trigonometric Functions

<code>arccos (x)</code>	Arc cosine (also known as “inverse cosine”)
<code>arccosec (x)</code>	Arc cosecant
<code>arccot (x)</code>	Arc cotangent
<code>arcsec (x)</code>	Arc secant
<code>arcsin (x)</code>	Arc sine (also known as “inverse sine”)
<code>arctan (x[, y])</code>	Arc tangent (also known as “inverse tangent”)

### 7.2.7 Elementary Mathematical Functions : Hyperbolic Functions

<code>cosh (x)</code>	Hyperbolic cosine
<code>cosech (x)</code>	Hyperbolic cosecant
<code>coth (x)</code>	Hyperbolic cotangent
<code>sech (x)</code>	Hyperbolic secant
<code>sinh (x)</code>	Hyperbolic sine
<code>tanh (x)</code>	Hyperbolic tangent

### 7.2.8 Elementary Mathematical Functions : Inverse Hyperbolic Functions

<code>arcosh (x)</code>	Hyperbolic area cosine
<code>arcosech (x)</code>	Hyperbolic area cosecant
<code>arcoth (x)</code>	Hyperbolic area cotangent
<code>arsech (x)</code>	Hyperbolic area secant
<code>arsinh (x)</code>	Hyperbolic area sine
<code>artanh (x)</code>	Hyperbolic area tangent

### 7.2.9 Elementary Mathematical Functions : Rounding

<code>ceil (x)</code>	Round to the next higher integer
<code>fix (x)</code>	Truncate decimal places from real number
<code>floor (x)</code>	Round to the next lower integer
<code>round (x)</code>	Round to nearest integer

## 7.2.10 Elementary Mathematical Functions : Special Mathematical Functions

besseli0(x)	Modified Bessel function of order zero
besselj(n, x)	Bessel function of first kind and n-th order
bessely(n, x)	Bessel function of second kind and n-th order
erf(x)	Error function
erfc(x)	Complementary error function
erfinv(x)	Inverse error function
erfcinv(x)	Inverse complementary error function
sinc(x)	Sinc function ( $\sin(x)/x$ or 1 at $x = 0$ )
step(x)	Step function

## 7.2.11 Data Analysis : Basic Statistics

avg(x[, range])	Arithmetic average of vector elements ; if a range is given then x must have a single data dependency
cumavg(x)	Cumulative average of vector elements
max(x, y)	Returns the greater of the values x and y
max(x[, range])	Maximum value in vector ; if a range is given then x must have a single data dependency
min(x, y)	Returns the lesser of the values x and y
min(x[, range])	Minimum value in vector ; if a range is given then x must have a single data dependency
rms(x)	Root Mean Square of vector elements
runavg(x)	Running average of vector elements
stddev(x)	Standard deviation of vector elements
variance(x)	Variance of vector elements
random()	Random number between 0.0 and 1.0
srandom(x)	Give random seed

## 7.2.12 Data Analysis : Basic Operation

cumprod(x)	Cumulative product of vector elements
cumsum(x)	Cumulative sum of vector elements
interpolate(f, xval)	Equidistant spline interpolation of real function vector f(x) using n equidistant datapoints ; the latter can be omitted and defaults to a reasonable value
prod(x)	Product of vector elements
sum(x)	Sum of vector elements
xvalue(f, yval)	Returns the x-value which is associated with the y-value nearest to a specified y-value yval in a given vector f ; therefore the vector f must have a single data dependency
yvalue(f, xval)	Returns the y-value of the given vector f which is located nearest to the x-value xval ; therefore the vector f must have a single data dependency

## 7.2.13 Data Analysis : Differentiation and Integration

ddx(expr, var)	Derives mathematical expression expr with respect to the variable var
diff(y, x[, n])	Differentiate vector y with respect to vector x n times. If n is omitted it defaults to n = 1
integrate(x, h)	Integrate vector x numerically assuming a constant step-size h

## 7.2.14 Data Analysis : Signal Processing

<code>dft(x)</code>	Discrete Fourier Transform of vector $x$
<code>fft(x)</code>	Fast Fourier Transform of vector $x$
<code>fftshift(x)</code>	Shuffles the FFT values of vector $x$ in order to move the frequency 0 to the center of the vector
<code>Freq2Time(V, f)</code>	Inverse Discrete Fourier Transform of function $V(f)$ interpreting it physically
<code>idft(x)</code>	Inverse Discrete Fourier Transform of vector $x$
<code>ifft(x)</code>	Inverse Fast Fourier Transform of vector $x$
<code>kbd(x[, n])</code>	Kaiser-Bessel derived window
<code>Time2Freq(v, t)</code>	Discrete Fourier Transform of function $v(t)$ interpreting it physically

## 7.3 Electronics Functions

### 7.3.1 Unit Conversion

<code>dB(x)</code>	dB value
<code>dbm(x)</code>	Convert voltage to power in dBm
<code>dbm2w(x)</code>	Convert power in dBm to power in Watts
<code>w2dbm(x)</code>	Convert power in Watts to power in dBm
<code>vt(t)</code>	Thermal voltage for a given temperature $t$ in Kelvin

### 7.3.2 Reflection Coefficients and VSWR

<code>rtoswr(x)</code>	Converts reflection coefficient to voltage standing wave ratio (VSWR)
<code>rtoz(x[, zref])</code>	Converts reflection coefficient to admittance; by default reference $zref$ is 50 ohms
<code>rtoz(x[, zref])</code>	Converts reflection coefficient to impedance; by default reference $zref$ is 50 ohms
<code>ytor(x[, zref])</code>	Converts admittance to reflection coefficient; by default reference $zref$ is 50 ohms
<code>ztor(x[, zref])</code>	Converts impedance to reflection coefficient; by default reference $zref$ is 50 ohms

### 7.3.3 N-Port Matrix Conversions

<code>stos(s, zref[, z0])</code>	Converts S-parameter matrix to S-parameter matrix with different reference impedance(s)
<code>stoy(s[, zref])</code>	Converts S-parameter matrix to Y-parameter matrix
<code>stoz(s[, zref])</code>	Converts S-parameter matrix to Z-parameter matrix
<code>twoport(m, from, to)</code>	Converts a two-port matrix from one representation into another, possible values for $from$ and $to$ are 'Y', 'Z', 'H', 'G', 'A', 'S' and 'T'.
<code>ytos(y[, z0])</code>	Converts Y-parameter matrix to S-parameter matrix
<code>ytoz(y)</code>	Converts Y-parameter matrix to Z-parameter matrix
<code>ztos(z[, z0])</code>	Converts Z-parameter matrix to S-parameter matrix
<code>ztoy(z)</code>	Converts Z-parameter matrix to Y-parameter matrix

## 7.3.4 Amplifiers

GaCircle(s, Ga[, arcs])	Circle(s) with constant available power gain Ga in the source plane
GpCircle(s, Gp[, arcs])	Circle(s) with constant operating power gain Gp in the load plane
Mu(s)	Mu stability factor of a two-port S-parameter matrix
Mu2(s)	Mu' stability factor of a two-port S-parameter matrix
NoiseCircle(Sopt, Fm, NoiseFigure(s), F, Arcs)	Generates circle(s) with constant Noise Figure(s) F. Arcs specifies the angles in degrees created by e.g. linspace(0, 360, 100). If Arcs is a number it specifies the number of equally spaced circle segments, if it is omitted this number defaults to a reasonable value
PlotVs(data, dep)	Returns a data item based upon vector or matrix vector data with dependency on a given vector dep, e.g. PlotVs(Gain, frequency/1e9)
Rollet(s)	Rollet stability factor of a two-port S-parameter matrix
StabCircleL(s[, arcs])	Stability circle in the load plane
StabCircleS(s[, arcs])	Stability circle in the source plane
StabFactor(s)	Stability factor of a two-port S-parameter matrix. Synonym for Rollet()
StabMeasure(s)	Stability measure B1 of a two-port S-parameter matrix

## 7.4 Nomenclature

### 7.4.1 Ranges

LO:HI	Range from LO to HI
:HI	Up to HI
LO:	From LO
:	No range limitations

### 7.4.2 Matrices and Matrix Elements

M	The whole matrix M
M[2, 3]	Element being in 2nd row and 3rd column of matrix M
M[:, 3]	Vector consisting of 3rd column of matrix M

### 7.4.3 Immediate

2.5	Real number
1.4+j5.1	Complex number
[1, 3, 5, 7]	Vector
[11, 12; 21, 22]	Matrix



### 7.4.4 Number suffixes

E	exa, 1e+18
P	peta, 1e+15
T	tera, 1e+12
G	giga, 1e+9
M	mega, 1e+6
k	kilo, 1e+3
m	milli, 1e-3
u	micro, 1e-6
n	nano, 1e-9
p	pico, 1e-12
f	femto, 1e-15
a	atto, 1e-18

### 7.4.5 Name of Values

S[1,1]	S-parameter value
<i>nodename.V</i>	DC voltage at node <i>nodename</i>
<i>name.I</i>	DC current through component <i>name</i>
<i>nodename.v</i>	AC voltage at node <i>nodename</i>
<i>name.i</i>	AC current through component <i>name</i>
<i>nodename.vn</i>	AC noise voltage at node <i>nodename</i>
<i>name.in</i>	AC noise current through component <i>name</i>
<i>nodename.Vt</i>	Transient voltage at node <i>nodename</i>
<i>name.It</i>	Transient current through component <i>name</i>

Note : All voltages and currents are peak values. Note : Noise voltages are RMS values at 1 Hz bandwidth.

## 7.5 Constants

i, j	Imaginary unit ("square root of -1")
pi	$4 \cdot \arctan(1) = 3.14159\dots$
e	Euler = 2.71828...
kB	Boltzmann constant = 1.38065e-23 J/K
q	Elementary charge = 1.6021765e-19 C



---

## List of Special Characters

---

It is possible to use special characters in the text painting and in the text of the diagram axis labels. This is done by using LaTeX tags. The following table contains a list of currently available characters.

**Note :** Which of those characters are correctly displayed depends on the font used by Qucs !

### Small Greek letters

LaTeX tag	Unicode	Description
<code>\alpha</code>	0x03B1	alpha
<code>\beta</code>	0x03B2	beta
<code>\gamma</code>	0x03B3	gamma
<code>\delta</code>	0x03B4	delta
<code>\epsilon</code>	0x03B5	epsilon
<code>\zeta</code>	0x03B6	zeta
<code>\eta</code>	0x03B7	eta
<code>\theta</code>	0x03B8	theta
<code>\iota</code>	0x03B9	iota
<code>\kappa</code>	0x03BA	kappa
<code>\lambda</code>	0x03BB	lambda
<code>\mu</code>	0x03BC	mu
<code>\textmu</code>	0x00B5	mu
<code>\nu</code>	0x03BD	nu
<code>\xi</code>	0x03BE	xi
<code>\pi</code>	0x03C0	pi
<code>\varpi</code>	0x03D6	pi
<code>\rho</code>	0x03C1	rho
<code>\varrho</code>	0x03F1	rho
<code>\sigma</code>	0x03C3	sigma
<code>\tau</code>	0x03C4	tau
<code>\upsilon</code>	0x03C5	upsilon
<code>\phi</code>	0x03C6	phi
<code>\chi</code>	0x03C7	chi
<code>\psi</code>	0x03C8	psi
<code>\omega</code>	0x03C9	omega

### Capital Greek letters

LaTeX tag	Unicode	Description
<code>\Gamma</code>	0x0393	Gamma
<code>\Delta</code>	0x0394	Delta
<code>\Theta</code>	0x0398	Theta
<code>\Lambda</code>	0x039B	Lambda
<code>\Xi</code>	0x039E	Xi
<code>\Pi</code>	0x03A0	Pi
<code>\Sigma</code>	0x03A3	Sigma
<code>\Upsilon</code>	0x03A5	Upsilon
<code>\Phi</code>	0x03A6	Phi
<code>\Psi</code>	0x03A8	Psi
<code>\Omega</code>	0x03A9	Omega

### Mathematical symbols

LaTeX tag	Unicode	Description
<code>\cdot</code>	0x00B7	multiplication dot (centered dot)
<code>\times</code>	0x00D7	multiplication cross
<code>\pm</code>	0x00B1	plus minus sign
<code>\mp</code>	0x2213	minus plus sign
<code>\partial</code>	0x2202	partial differentiation symbol
<code>\nabla</code>	0x2207	nabla operator
<code>\infty</code>	0x221E	infinity symbol
<code>\int</code>	0x222B	integral symbol
<code>\approx</code>	0x2248	approximation symbol (waved equal sign)
<code>\neq</code>	0x2260	not equal sign
<code>\in</code>	0x220A	“contained in” symbol
<code>\leq</code>	0x2264	less-equal sign
<code>\geq</code>	0x2265	greater-equal sign
<code>\sim</code>	0x223C	(central european) proportional sign
<code>\propto</code>	0x221D	(american) proportional sign
<code>\diameter</code>	0x00F8	diameter sign (also sign for average)
<code>\onehalf</code>	0x00BD	one half
<code>\onequarter</code>	0x00BC	one quarter
<code>\twosuperior</code>	0x00B2	square (power of two)
<code>\threesuperior</code>	0x00B3	power of three
<code>\ohm</code>	0x03A9	unit for resistance (capital Greek omega)

---

## Matching Circuits

---

Creating matching circuits is an often needed task in microwave technology. Qucs can do this automatically. These are the necessary steps :

Perform an S-parameter simulation in order to calculate the reflexion coefficient.

Place a diagram and display the reflexion coefficient (i.e.  $S[1,1]$  for port 1,  $S[2,2]$  for port 2 etc.)

Set a marker on the graph and step to the desired frequency.

Click with the right mouse button on the marker and select “power matching” in the appearing menu.

A dialog appears where the values can be adjusted, for example the reference impedance can be chosen different from 50 ohms.

After clicking “create” the page switches back to the schematic and by moving the mouse cursor the matching circuit can be placed.

The left-hand side of the matching circuit is the input and the right-hand side must be connected to the circuit.

If the marker points to a variable called “Sopt”, the menu shows the option “noise matching”. Note that the only different to “power matching” is the fact that the conjugate complex reflexion coefficient is taken. So if the variable has another name, noise matching can be chosen by re-adjusting the values in the dialog.

The matching dialog can also be called by menu (Tools->matching circuit) or by short-cut (<CTRL-5>). But then all values has to be entered manually.

### 9.1 2-Port Matching Circuits

If the variable name in the marker text is an S-parameter, then an option exists for concurrently matching input and output of a two-port circuit. This works quite alike the above-mentioned steps. It results in two L-circuits : The very left node is for connecting port 1, the very right node is for connectiong port 2 and the two node in the middle are for connecting the two-port circuit.



---

## Installed Files

---

The Qucs system needs several programs. These are installed during the installation process. The path of Qucs is determined during the installation (configure script). The following explanations assume the default path (/usr/local/).

- /usr/local/bin/qucs - the GUI
- /usr/local/bin/qucsator - the simulator (console application)
- /usr/local/bin/qucsedit - a simple text editor
- /usr/local/bin/qucshelp - a small program displaying the help system
- /usr/local/bin/qucstrans - a program for calculation transmission line parameters
- /usr/local/bin/qucsfilter - a program synthesizing filter circuits
- /usr/local/bin/qucsconv - a file format converter (console application)

All programs are stand-alone applications and can be started independently. The main program (GUI)

- calls qucsator when performing a simulation,
- calls qucsedit when showing text files,
- calls qucshelp when showing the help system,
- calls qucstrans when calling this program from menu “Tools”,
- calls qucsfilter when calling this program from menu “Tools”,
- calls qucsconv when placing the SPICE component and when performing a simulation with the SPICE component.

Furthermore, the following directories are created during installation :

- /usr/local/share/qucs/bitmaps - contains all bitmaps (icons etc.)
- /usr/local/share/qucs/docs - contains HTML documents for the help system
- /usr/local/share/qucs/lang - contains the translation files

## 10.1 Command line arguments

```
qucs [file1 [file2 ...]]
```

```
qucsator [-b] -i netlist -o dataset (b = progress bar)
```

```
qucsedit [-r] [file] (r = read-only)
```

```
qucshelp (no arguments)
```

```
qucsconv -if spice -of qucs -i netlist.inp -o netlist.net
```





---

## Schematic File Format

---

This document describes the schematic file format of Qucs. This format is used for schematics (usually with suffix `.sch`) and for data displays (usually with suffix `.dpl`). The following text shows a short example of a schematic file.

```
<Qucs Schematic 0.0.6>
<Properties>
  <View=0,0,800,800,1,0,0>
</Properties>
<Symbol>
  <.ID -20 14 SUB>
</Symbol>
<Components>
  <R R1 1 180 150 15 -26 0 1 "50 Ohm" 1 "26.85" 0 "european" 0>
  <GND * 1 180 180 0 0 0 0>
</Components>
<Wires>
  <180 100 180 120 "" 0 0 0 "">
  <120 100 180 100 "Input" 170 70 21 "">
</Wires>
<Diagrams>
  <Polar 300 250 200 200 1 #c0c0c0 1 00 1 0 1 1 1 0 5 15 1 0 1 1 315 0 225 "" "" "">
  <"acnoise2:S[2,1]" #0000ff 0 3 0 0 0>
  <Mkr 6e+09 118 -195 3 0 0>
</Polar>
</Diagrams>
<Paintings>
  <Arrow 210 320 50 -100 20 8 #000000 0 1>
</Paintings>
```

The file contains several section. Each of it is explained below. Every line consists of not more than one information block that starts with a less-sign `<` and ends with a greater-sign `>`.

### 11.1 Propriétés

The first section starts with `<Properties>` and ends with `</Properties>`. It contains the document properties of the file. Each line is optional. The following properties are supported :

- `<View=x1,y1,x2,y2,scale,xpos,ypos>` contains pixel position of the schematic window in the first four numbers, its current scale and the current position of the upper left corner (last two numbers).
- `<Grid=x,y,on>` contains the distance of the grid in pixel (first two numbers) and whether grid is on (last number 1) or off (last number 0).
- `<DataSet=name.dat>` contains the file name of the data set associated with this schematic.

- `<DataDisplay=name.dpl>` contains the file name of the data display page associated with this schematic (or the file name of the schematic if this document is a data display).
- `<OpenDisplay=yes>` contains 1 if the data display page opens automatically after simulation, otherwise contains 0.

## 11.2 Symbol

This section starts with `<Symbol>` and ends with `</Symbol>`. It contains painting elements creating a schematic symbol for the file. This is usually only used for schematic files that meant to be a subcircuit.

## 11.3 Composants

This section starts with `<Components>` and ends with `</Components>`. It contains the circuit components of the schematic. The line format is as follows :

```
<type name active x y xtext ytext mirrorX rotate "Value1" visible "Value2" visible ...>
```

- The `type` identifies the component, e.g. R for a resistor, C for a capacitor.
- The `name` is the unique component identifier of the schematic, e.g. R1 for the first resistor.
- A 1 in the `active` field shows that the component is active, i.e it is used in the simulation. A 0 shows it is inactive.
- The next two numbers are the x and y coordinates of the component center.
- The next two numbers are the x and y coordinates of the upper left corner of the component text. They are relative to the component center.
- The next two numbers indicate the mirroring about the x axis (1 for mirrored, 0 for not mirrored) and the counter-clockwise rotation (multiple of 90 degree, i.e. 0...3).
- The next entries are the values of the component properties (in quotation marks) followed by an 1 if the property is visible on the schematic (otherwise 0).

## 11.4 Wires

This section starts with `<Wires>` and ends with `</Wires>`. It contains the wires (electrical connection between circuit components) and their labels and node sets. The line format is as follows :

```
<x1 y1 x2 y2 "label" xlabel ylabel dlabel "node set">
```

- The first four numbers are the coordinates of the wire in pixels : x coordinate of starting point, y coordinate of starting point, x coordinate of end point and y coordinate of end point. All wires must be either horizontal (both x coordinates equal) or vertical (both y coordinates equal).
- The first string in quotation marks is the label name. It is empty if the user has not set a label on this wire.
- The next two numbers are the x and y coordinates of the label or zero if no label exists.
- The next number is the distance between the wire starting point and the point where the label is set on the wire.
- The last string in quotation marks is the node set of the wire, i.e. the initial voltage at this node used by the simulation engine to find the solution. This is empty if the user has not set a node set for this wire.

## 11.5 Diagrams

This section starts with `<Diagrams>` and ends with `</Diagrams>`. It contains the diagrams with their graphs and their markers. The line format is as follows (line break not allowed) :

```
<x y width height grid gridcolor gridstyle log xAutoscale xmin xstep  
xmax yAutoscale ymin ystep ymax zAutoscale zmin zstep zmax xrotate  
yrotate zrotate "xlabel" "ylabel" "zlabel">
```

- The first two numbers are x and y coordinate of lower left corner.
- The next two numbers are width and height of diagram bindings.
- The fifth number is 1 if grid is on and 0 if grid is off.
- The next is grid color in 24 bit hexadecimal RGB value, e.g. #FF0000 is red.
- The next number determines the style of the grid.
- The next number determines which axes have logarithmical scale.

## 11.6 Paintings

This section starts with `<Paintings>` and ends with `</Paintings>`. It contains the paintings that are within the schematic.

Technical description concerning the simulator

Available online at <http://qucs.sourceforge.net/tech/technical.html>

Example schematics

Available online at <http://qucs.sourceforge.net/download.html#example>