SCOPE /SDK

Version 4.0

Chapter 4: *Circuit Design*

CreamWare Datentechnik GmbH
Wilhelm-Ostwald-Strasse 0/K2
53721 Siegburg
Germany

Tel.: (+49) 2241-5958-0

Fax: (+49) 2241-5958-5

Hotline: (+49) 2241-5958-12

Main Table of Contents

Contents

Circuit Design

The SCOPE /SDK offers the ability to build custom modules and devices by combining basic modules. The latters can be those of the *Module Library* or the ones you have programmed yourself - making use of the programming interfaces.
In either case the module is made by laying out a circuit of signal generating nodes and signal processing nodes.

This chapter deals with the functionalities available for *Circuit Design*. The procedures will be described and some ideas for optimizing the work-flow are given.

> ! *It has to be pointed out that there is no 'optimal work-flow'. As expected for a product of such an orientation and complexity there are countless possibilities to achieve a specific task. You shall find your own solutions within SCOPE but nevertheless there are some hints that you might find helpful.*

Circuit Design is about connecting the *Pads* of different modules to make routings and to create more sophisticated processing networks. It is best to begin with the basics again.

Usually you would use the *Project Window* for *Circuit Design*. However as the *Project Explorer* provides a 'list' view of the current project it is also possible to rely on that one. You could use the *Project Explorer* and the *Pad List* window to make connections.
There are situations for both of these approaches. A typical window set for *Circuit Design* includes the *Project Window*, the *Project Explorer*, the *File Browser*, the *Pad List* windows, the *Connections* window. In some situations the *Var Attribute* dialog box and the *Module Attributes* dialog box might provide some helpful details.
According to your needs the *Help Window* may be a useful addition to this list. It provides information on the module itself as well as on its *Pads*.
If you are not sure about the functionality of a module or the characteristics of a Pad the *Help Window* is a fast and convenient way to get a short description on it.

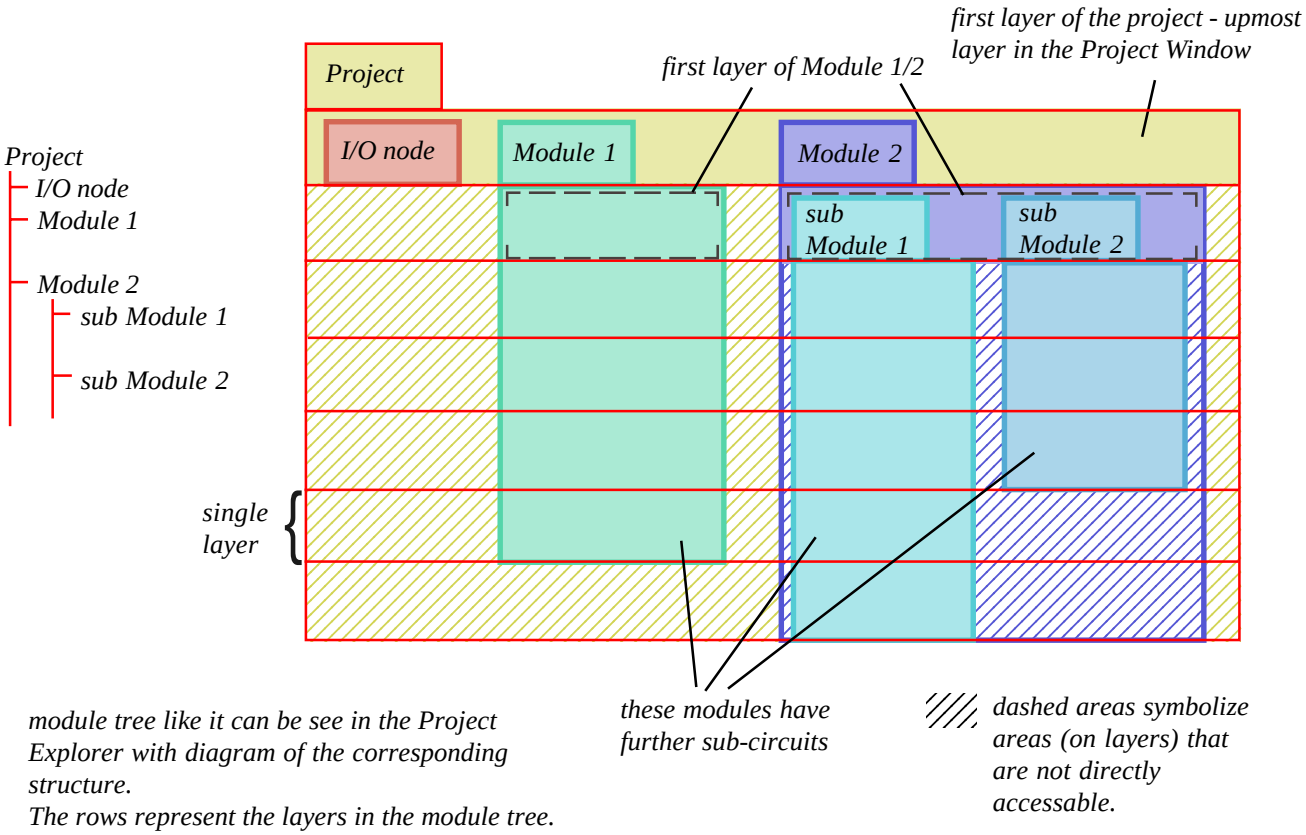
> ! *Doing Circuit Design the **Move** mode is most often your preferred working mode. Neverthelessa lot of operations can also be accomplished in **Use** mode. If not otherwise mentioned you might want to operate the **Project Window** in **Move** mode.*

Hierarchies

SCOPE is a sophisticated DSP development environment that allows you to build complex modules. For such an environment it is very important to provide the user with the capacity to structure complex circuits efficiently.
In the first chapter the *module tree* with its *parent-child-relation* and the *inheritance of specific attributes* was already discussed. Its hierarchical tree structure depicts that there are different levels within a module - the so called **layers**.

Layers

Modules can be organized in layers. A layer is a new level in the *module tree*'s hierarchy. It contains one or more modules. Each of the modules can add further layers to the evolving hierarchy.



module tree like it can be see in the Project Explorer with diagram of the corresponding structure.
The rows represent the layers in the module tree.

A SCOPE project is a special kind of a module. The *Project Window* when first opened shows the first layer of the project's *module tree*. The first layer of a hierarchy is always the upmost layer - e.g. that which is directly below the module node.

Normally you start building your modules on the first layer of the project - this layer will normally never be part of your module. It will only host the nodes of the modules.
In the diagram above only the name of the modules (*Module 1* or *Module 2*) inside a smaller box is on the first layer of the project. This should visualize that modules may be hosted on the first layer but this layer is not a part of the module itself.
In subsequent steps you keep pushing the first nodes of your module deeper and deeper in the hierarchy of the evolving module. Of course you can start new sub-circuits directly on a lower layer.

It is important that you do not create new layers on top of the current layer but underneath the current one. So the nodes that were formerly on the current layer are then in the first underlying layer. Although the project has one layer more than the 'deepest' module in it, lower layers can only be accessed inside the modules. Accordingly lower layers in modules can only be accessed via the children. That is why there are dashed areas in the diagram.
Of course not every module (like *Module 1*) has to reach the lowest layers of the project. Also not every sub-module needs to have as many layers as the parent module (like *sub Module 2*)

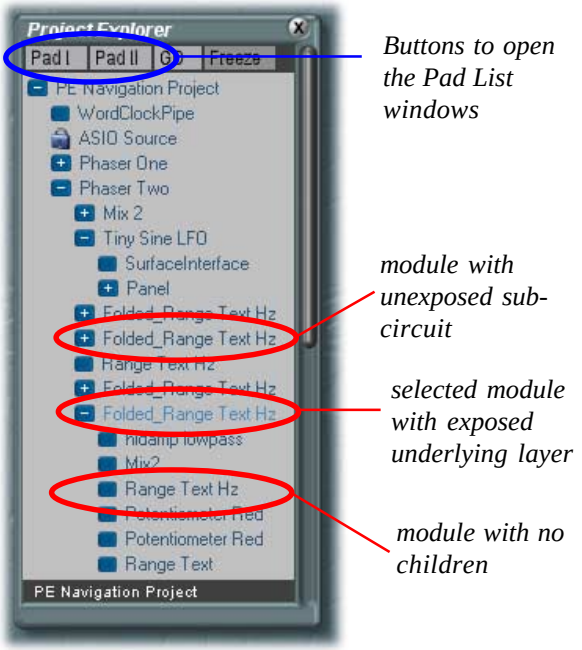
New layers are created by combining nodes into a new module. The new node is placed in the current layer. Alternatively you could use any module and put others inside of it.
More accurately you could say that each module can open a subsequent layer. You can access this layer by double-clicking

the node. As a consequence each node can act as a container for other nodes. So the lowest layers of a hierarchy are always empty.

Still, the idea behind this is to structure your circuit without losing the possibility to access and manipulate every node of it. So no matter how you generated the hierarchy, the content is not changed by generating the hierarchy. To be able to access and manipulate the nodes you haveto be able to navigate through the layers conveniently.

Navigation

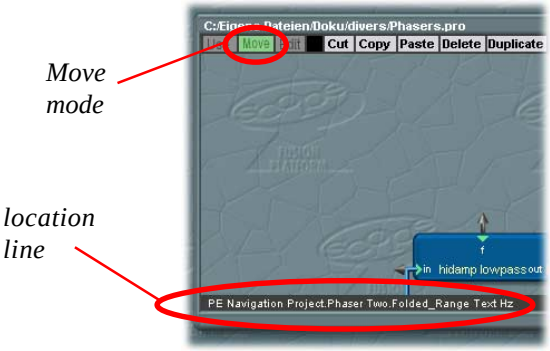
You can easily step through the hierarchy of your module. You can browse the *Project Explorer* which represents the project's *module tree* to make any changes to *Pads* of a node that are accessible via the *Pad List* window.



The *Project Explorer* behaves like most browsers relying on a hierarchical tree representations. Each node is displayed as a blue box with its name next to it. If the node is a parent module that hosts other modules a '+'-sign is inside the blue box. Clicking the box will unfold the next underlying level of the hierarchy.
If necessary scroll bars are automatically added to the *Project Explorer* which enable you to scroll horizontally and vertically. You do not enter the layers but rather than that you expose them in the same view. So the scroll bars are the only auxiliary means you need to navigate through the entire hierarchy.
In the *Project Explorer* you can select any module and change *Pad* values in the *Pad List* window or edit connections in the *Connections* window. The *Pad List* window can be accessed from the buttons at the top of the *Project Explorer*.

Using the *Project Explorer* is the fastest way to navigate through the hierarchy. However, the *Project Window* provides the better reproduction of the relations, connections and dependencies.

Whereas you can browse with the *Project Explorer* like with the standard Windows Explorer the navigation in the *Project Window* is different.
You really step through the layers. To see if there are any modules inside a node and which ones you have to enter the underlying layer. Nevertheless you can immediately overview how the modules are connected and if *Pads* are exported.



To retain the orientation through all the layers the *location line* at the bottom of the *Project Window* shows the path of your current location.

The navigation is fast and convenient:

- Make sure you are in *Move* mode - press <Ctrl>-<M> if necessary.
- To step down the layers double-click a node and you enter the underlying hierarchy.
- To navigate to a higher layer double-click onto the background of the *Project Window*.

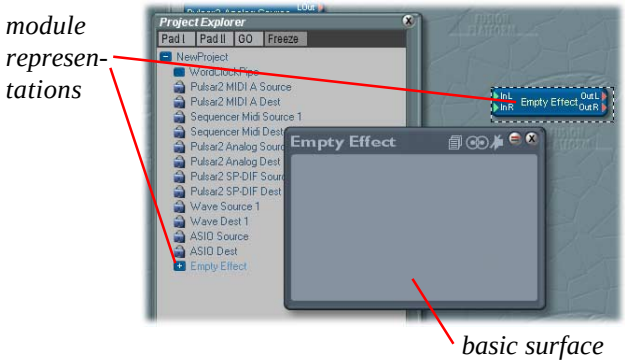
To navigate in the *Project Window* you should always be in *Move* mode. In *Use* mode you can move up the hierarchy but you cannot move to a lower layer.

Modules

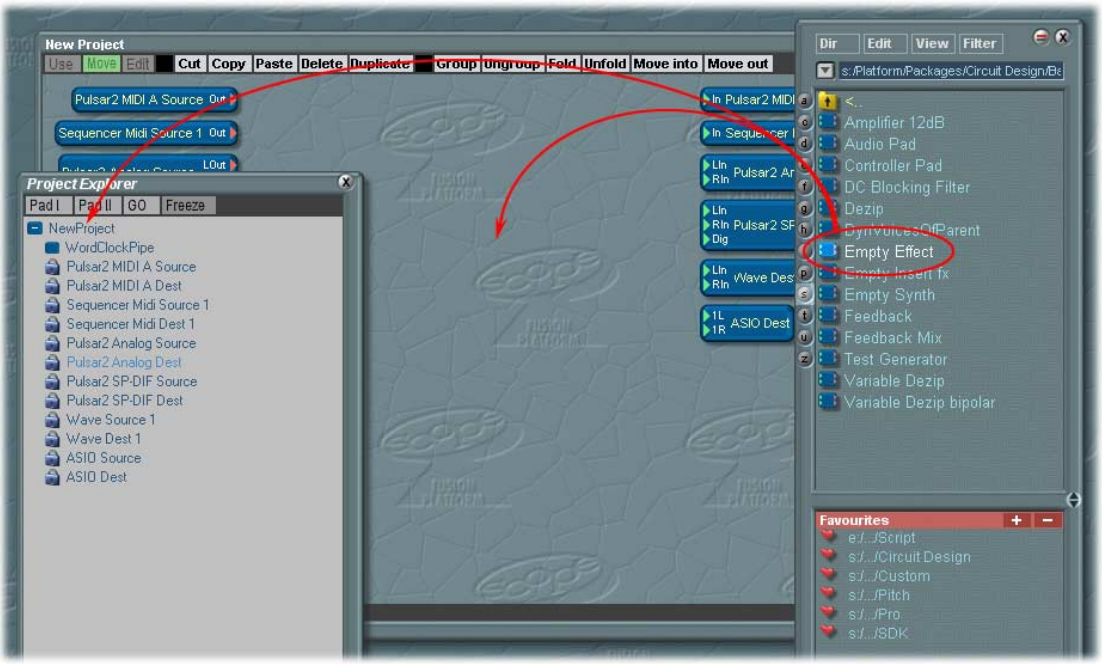
When starting to build a circuit you add a module to your project. You could use one of the basic modules that comes with the *Module Library*.

Adding modules

Drag the module from the *File Browser* into the *Project Window* or the *Project Explorer*. While dragging the module the mouse changes to in insert pointer.
In the *Project Window* drop it onto the background. In the *Project Explorer* you have to drop it into another node - like the project's node. While moving the mouse over the tree the node into which the module will be dropped gets highlighted. Release the mouse button to drop the module.
In the *Project Window* it will appear with its representation and in the *Project Explorer* as another leaf of the tree.



You can open and close the surface of a module from its context menu in the Project Window (Surfaces -> Open Panel)



Most modules from the *Module Library* have basic surfaces. The surface will show up at the location where it was when the module was saved the last time.

> ! *Be aware of that fact when working in teams. Not every member might use the same screen resolution. The surface of the module can get placed outside the visible area of the screen.*

You can drag modules to any layer of your project. This can be achieved by dragging it from the *File Browser* into the destination module in the *Project Explorer.* This makes it extremely easy to do so.
Alternatively you can navigate to the destination in the *Project Window* and drop it to this window. It is not supported to drop a module from the *File Browser* into another module in the *Project Window*. However, most often you want to connect the module after adding it to the project. So it cannot be considered cumbersome that you have to be on the layer where you want to drop module.
The newly dropped module gets automatically selected.

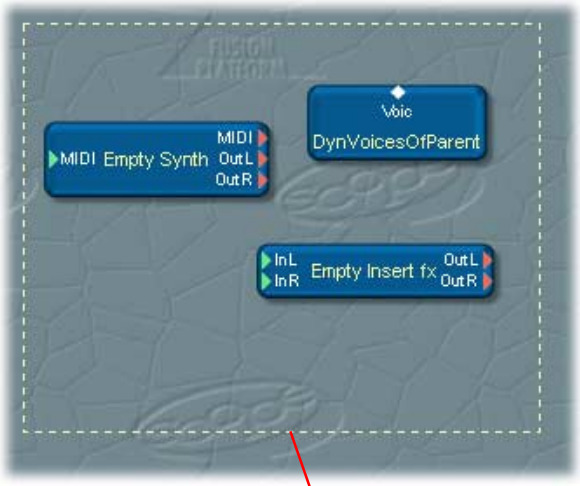
Selecting modules

When you want to make changes to a module or move it you first have to select it. A module can be selected by picking it with the mouse - in the *Project Window* or in the *Project Explorer*. These two representations of the project are synchronized so that the same modules are selected. In the latter the nodes get highlighted to indicate that they are selected.
In the *Project Window* it depends upon the settings you can make in the *Tool Bar* if the selected nodes are highlighted or not. In either case the selected module is displayed with a black and white dashed rectangle surrounding it.

To select more then one node at a time hold down the <Shift> key and pick multiple nodes subsequently. The node that was selected last is the current node and it is surrounded by a black and white dashed rectangle.

It is important to distinguish between the selected nodes and the current node.
The operations you perform in the *Project Window* will affect all select nodes. However, the *Pad List* window only displays the *Pads* of the current node and therefore all operations on the *Pads* only affect the current node.

Instead of <Shift>-clicking for selecting mutliple nodes you can also drag a rectangle over the modules. With the mouse click in the background of the *Project Window* and while holding the LMB pressed drag a rectangle over the modules you want to select.

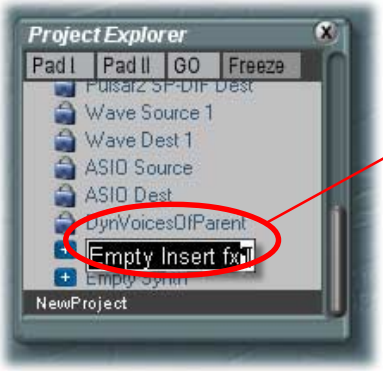


selection rectangle

To perform an operation on the selected modules you first have to pick one of them. This module will get the current module and it will be surrounded by a black and white dashed rectangel as it was described previously.
A module has to be totally within the rectangle to select. If one edge of a module is outside the rectangle it will not be included in the selection.

Renaming modules

You can easily rename any module. To do this open the *Project Explorer* (function key <F11>) if it is not already open. Locate the node in the tree, select it and press <F2>. An edit box will pop up and you can change the name of the node. Press <Return> to confirm your changes.



text edit box

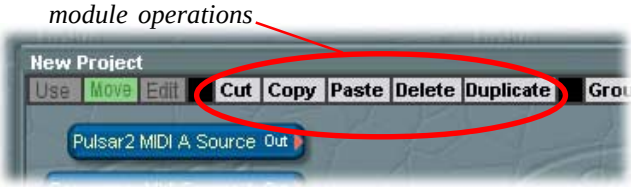
The module's name in the *Project Window* changes immediately - the node may resize to fit to the new name. All connections stay valid.

You should always use names that make it easy to determine the functionality and the use of the module. This helps you to orientate yourself even when making changes after several months.

Module operations

In the *Project Window* you can perform the standard operations (duplicate, copy, cut, paste) on the selected modules. You can find the commands in the *command bar* or you can use the established short-cuts (<Ctrl>-<D>, <Ctrl>-<C>, <Ctrl>-<X> and <Ctrl>-<V>).

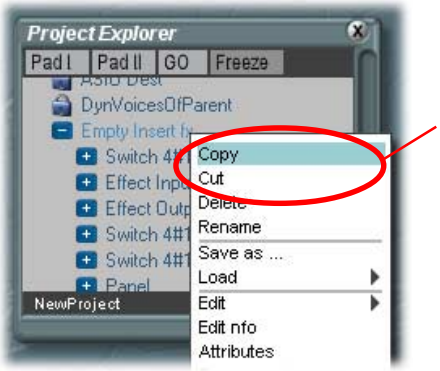
module operations



On the same layer a module is duplicated rather than copied. However from one layer to the other you copy and paste a node.

You can also cut, copy and paste modules in the *Project Explorer*. The procedure is as follows:

- In the *Project Explorer* select the module you want to copy.
- Right click it to open its *context menu* and choose 'Copy' or 'Cut'.
- Select the node to which you want to paste the formerly copied or cut module.
- From its *context menu* choose 'Paste'.
- The module will be pasted into the first layer of that node.



module operations

Paste is not yet available as no module was copied previously.

The methods from the *Project Window* and *Project Explorer* can be combined. This is great when you want to copy a module from a lower layer of another branch or module into a specifc place. In the *Project Explorer* you can navigate to the module you want to copy and paste it directly into the current layer in the *Project Window*. At once you begin with making connections.

! *You can also accomplish the same ease of use by working with two **Project Windows**. Each can be set to another layer so you can easily copy and paste between them.*

In the *Project Explorer* you only need a copy function because you always copy modules into others. Therefore a duplicate function like in the *Project Window* is not needed.

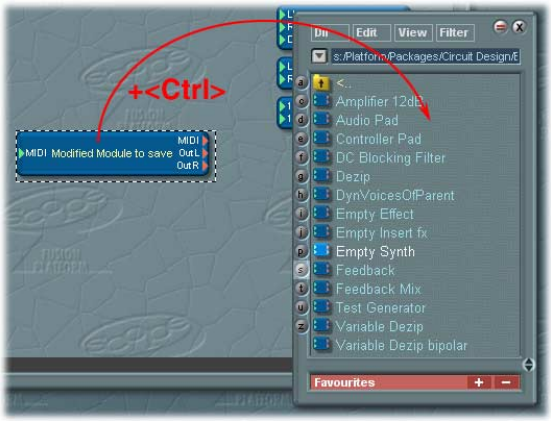
Positioning modules

To lay out yourprocessing network you can position the modules inside the *Project Window*. Select the module(s) and drag it offer the background. It will be positioned at the place where you release the LMB.
You have to be in *Move* mode though.

Saving modules

If you have modified a module and you want to make the changes permanent you have to save the module. There are two ways to save a module.

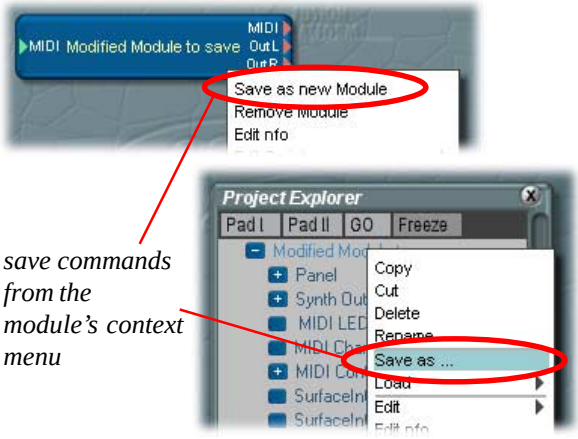
The fastest way is to drag the module from the *Project Window* into the *File Browser* while holding the <Ctrl> key down. The module is saved to the current folder. The module's name (as in the *Project Explorer*) automatically becomes the file name.



You should be aware that other files which have the same name - most probably older versions of this file - will be overwritten without a warning.

If there is an older file which you do not want to overwrite, but you also do not want to change the name of one of these modules you can change the name of the older file in the *File Browser* before <Ctrl>-dragging the current node to the *File Browser*.
To do so select the older file in the *File Browser* and press function key <F2>. Now you can change the file name without changing the module name. This is especially convenient for backing up incremental copies of a module.

The second procedure for saving a module is to call the 'Save as new Module' command from the module's *context menu* in the *Project Window*. This



is equivalent to the 'Save as' command of the module's *context menu* in the *Project Explorer*.
The command will open a standard Windows w'Save as' dialog box and you can choose the destination where you want to save the new module. Furthermore you can change the file name of the module or you can even save it as a device.

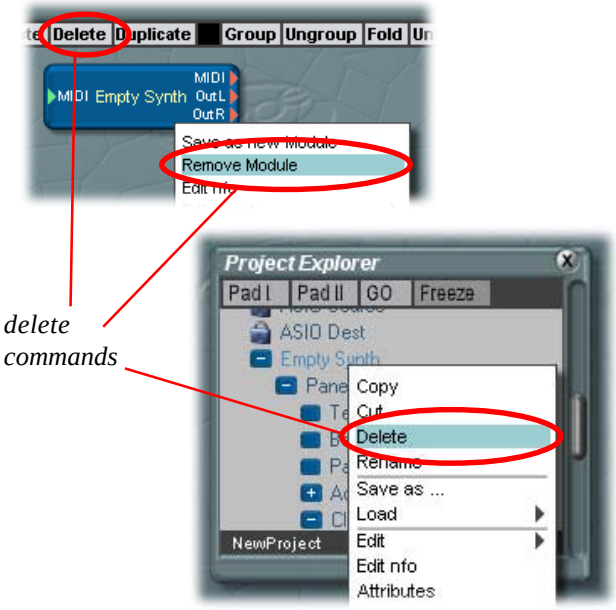
Although the second method includes some more steps it allows you to change the file name of the module. Additionally you do not have to change the current folder in the *File Browser* to choose a different location.

Removing modules

The concept of SCOPE encourages experimentations. Quite often you might try different modules to see which one provides the best results. After you came to a decision you might like to remove the unused modules.

To remove one or more modules from the SCOPE environment select them and press the <Del> key on your keyboard. A dialog box appears to assure that you really want to delete the module(s). Press 'Yes' if you want to delete the module(s) or 'No' to cancel the operation.

Alternatively to the <Del> key you can also press the <Delete> button in the *command bar* (in the *Project Window*) or you can use the 'Remove Module'/ 'Remove Selected Modules' command from the *context menu* of the module in



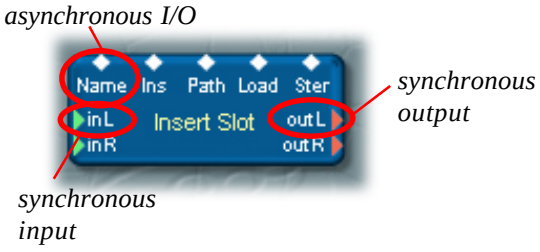
the *Project Window*. As described before the dialog box will appear. Make your choice and press the corresponding button.

You can also delete a module in the *Project Explorer* with the node's *context menu* or by hitting the <Del> key. Just select the module, bring up its *context menu* and choose 'Delete'. Alternatively select the module and hit the <Del> key. Both procedures will delete the module immediately without asking for confirmations. So be careful with the delete commands in the *Project Explorer*.

Modules are the most important elements in the SCOPE philosophy. Once, it has been pointed out that everything inside SCOPE is a module (of any form). So these basics in handling modules are absolutely important. They refer to a lot of situations as the underlying principal of SCOPE is to connect any numbers of modules.

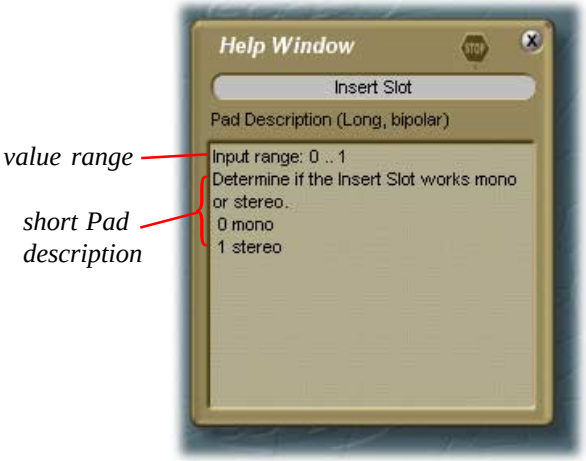
Pads

Having loaded several modules you can interconnect their *Pads*. Do you remember that *Pads* can have different processing rates and value ranges?
To make it easier to differentiate the Pads there are some conventions. Names of **synchronous** *Pads* begin with a **lower case** letter whereas names of **asynchronous** Pads begin with a **capital** letter.
Normally the signal flow has a direction - so there are outputs and inputs to a module. Inputs are indicated by a green triangle, outputs by a red one. Especially for control signals there are I/O-*Pads*, e.g. combined *Pads* that function as both inputs and outputs simultaneously. They are depict on the module by a white diamond.



Of course you can also consult the *Var Attributes* dialog box for detailed information on a specific *Pad*.

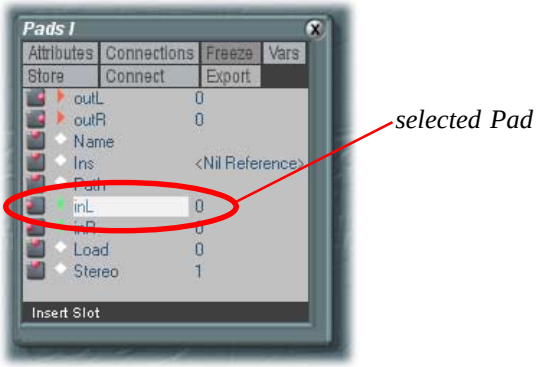
Another source for information is the *Help Window*, especially for the modules from the *Module Library*. Just position your mouse pointer over a *Pad* and the *Help Window* will display a short description of the *Pad* and its value ranges.



Selecting Pads

Pads can only be selected in the *Pad List* window. It is **not** possible to select a *Pad* from the *Project Window* or *Project Explorer* directly.
However in most cases, first, you have to select the module to which the *Pad* belongs in the *Project Window* or the *Project Explorer*. This will turn the focus of the *Pad List* window to this module. Now you can select any its of *Pads* in the *Pad List* window.

- Open the *Pad List* window (from the button in the *Project Explorer* or a *context menu* in he *Project Window*).
- Click on a *Pad*'s name to select it.
- The currently selected *Pad* gets inverted.



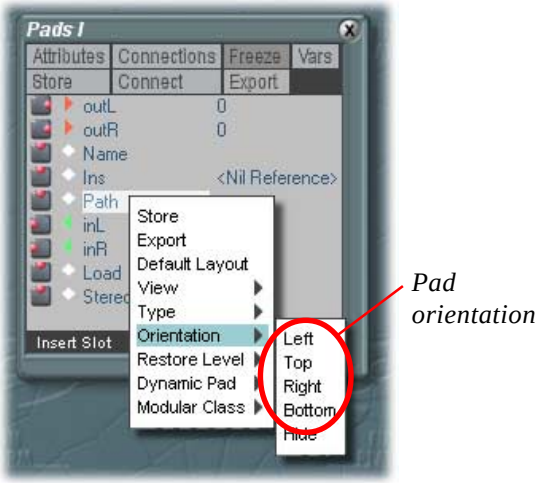
Changing the orientation

If you have a look at different modules you can see that the *Pads* are oriented to the different edges of the node. Inputs are not always on the left side and outputs are not always on the right side. Indeed the orientation of each *Pad* can be set separately. This is very practical for influencing how connections are displayed.
A *Pad* can be oriented towards any edge of the node - to the top, the left, the right or the bottom. To change the orientation to the following:

- Select the *Pad* (in the *Pad List* window)
- press <T> to set the orientation to the upper edge
- press <L> for the left edge
- press <R> for the right edge
- press <B> for the lower edge

Alternatively you can also change the orientation from the *Pad*'s *context menu* in the *Pad List* window:

- Right-click the Pad to open its context menu
- choose 'Orientation -> Left/Top/ Right/Bottom'



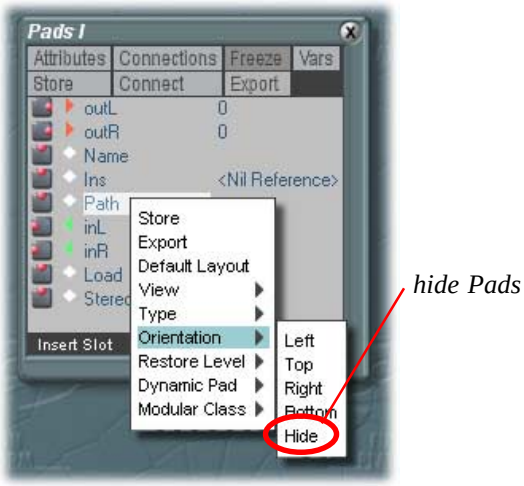
The *Pad* will flip to the corresponding edge on the node.

Hiding Pads

There is another relation between *Pads* and modules. The module node in the *Project Window* must be large enough so that all visible *Pads* fit onto it. This means the more *Pads* a module has the bigger is its size in the *Project Window*.

Within a complex layer you often want the modules to cover as few space as possible. Not all *Pads* of each module get connected. By hiding these *Pads* in the *Project Window* the node will shrink.

To hide a *Pad* select it and press the <H> key on your keyboard. Alternatively you can also use the *Pad*'s context menu in the *Pad List* window. Right-click the *Pad* and choose 'Orientation -> Hide'.



Although this reduces the number of visible Pads in the Project Window the Pad List will stay more or less the same. You can also hide the Pads from the Pad List by choosing 'View -> Hide hidden pads' from the Pad's or the Pad List's context menu.

However you will no longer be able to see the values of the hidden Pads that easily. You have to show them first or switch to the Var List to see it.

It is important to note that the *Pad* is still there, it can be connected to any other *Pad*, only, it is no longer visible on the node.

Show hidden Pad

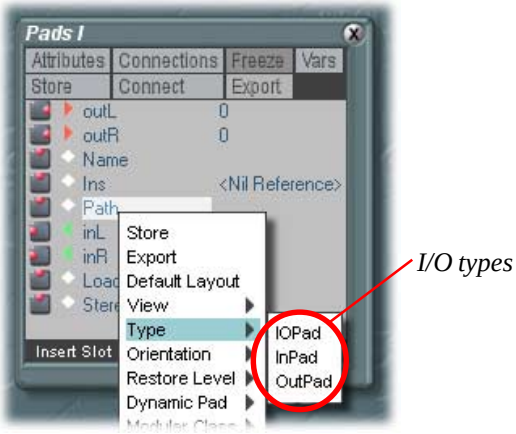
To show a hidden Pad again you just have to change its orientation from 'Hide' to another option ('Left', 'Top', 'Right' or 'Bottom'). The Pad will display again on the specified side of the node.

Changing the I/O type

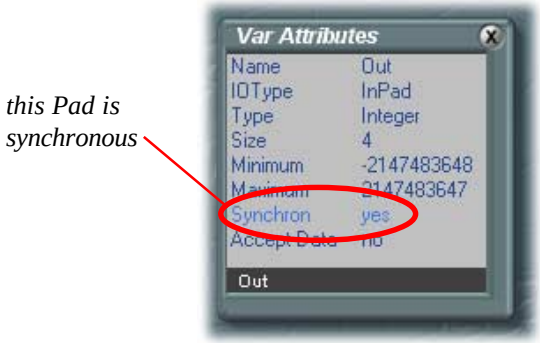
The I/O type of a *Pad* can also be customized. Changing the I/O type does not affect the functionality of the *Pad*. It just displays differently. SCOPE itself will always look on the I/O type of the referenced *Var*.

To depict the *Pad* as an 'input', select the *Pad* and press the <I> key on your keyboard. Pressing the <O> key will change the type to 'output' and <P> to 'input/output'.

Alternatively you could use the commands from the *Pad*'s context menu in the *Pad List* window. Press the right mouse button and choose 'Type -> IOPad/InPad/OutPad'. The Pad will update accordingly.



Synchronous *Pads* start with a lower case letter and asynchronous *Pads* with a capital letter. To look up if a *Pad* is synchronous or not you might have a look into the *Var Attributes* window of the *Pad*/*Var*.



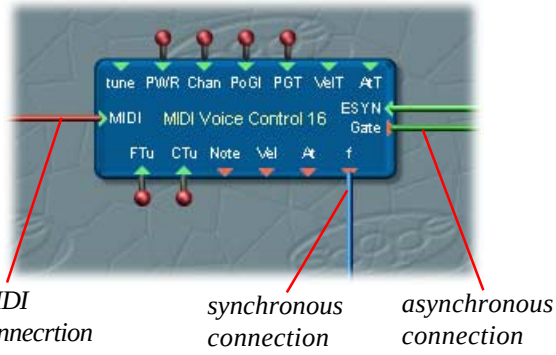
Renaming Pads

The functionality a *Pad* controls is expressed in its name. In some cases you would maybe prefer a more characterizing or a more meaningful name for a *Pad*.
You can rename virtually any *Pad*. To change the name open the *Pad List* window if necessary and select the *Pad* you want to rename. Press the function key <F2>. An edit box opens and you can change the name. If you are done press <Return> to confirm or <Esc> to discard your changes.

Whatever name you enter keep in mind that only the first four letters will be displayed on the node! Furthermore it is a ggod idea to pay attention ot the conventions of naming *Pads*.

Connections

Additionally to the spelling of the *Pads*' names there is a kind of color coding for the connections themselves.
Synchronous connections have blue virtual wires, asychronous connections have green ones and MIDI connections are red. This should make it easy to differentiate between these types of connections.



Concerning MIDI connections it has to be said that MIDI is handled as an asynchronous signal within SCOPE.
However due to the special format of these messages it is rather a category on its own than a derivate of a standard asynchronous signal.

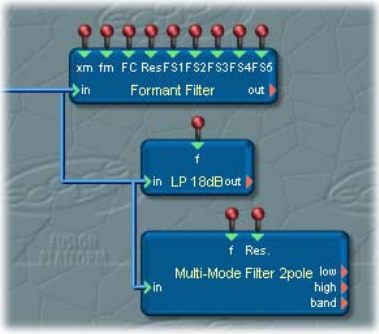
Making connections

Normally output *Pads* are connected to input *Pads* on other modules. *IOPads* can be connected to any kind of *Pads*. SCOPE will automatically prevent you from making connections between output *Pads*.

Please keep in mind that the I/O type of the Var is decisive, not that of the Pad!

However multiple inputs can be interconnected. These connections will operate like a bridge for the involved inputs - the signal that is fed to one of them will be fed to all of them.

bridged inputs



Such routings can be very comfortable when you want to route the same signal to adjacent modules and even more if you want to route to adjacent *Pads* on the same module - p.e. a mono signal from your circuit to a stereo pair of your physical audio outputs.
So you can always use the shortest possible connections.

As in a regular situation SCOPE will not allow you to connect more than one output to an input or a bridged group of inputs.

There are different procedures to make connections within SCOPE. The most common procedure is the following:

- In the *Project Window* pick the *Pad* at which you want to start a connection. The *Pad* has a sensitive zone which is the area of it's pictogramm and it's name. The mouse pointer will change to a connector as soon as you move it over this area.

- Release the mouse button and drag the mouse over to the destination node. While the mouse is not over a *Pad* the mouse pointer is a red connector to indicate that you are currently dragging a virtual wire to another *Pad*.

- Position the mouse over the *Pad* you want to connect to. As soon as the pointer is over a *Pad* the connector will turn black again.

- Over the *Pad* or its name press the LMB to make the connection. A virtual wire will illustrate the connection.

In most cases the order you pick the *Pads* to connect them is not important. The direction of the signal flow is determined by the type of the *Pads*.
There is also a quick and easy way for multi-channel connections. To make connections of adjacent *Pads* from the one module to adjacent *Pads* on a second module make the first connection like describe above. For each additional connection press the <N> key once.

SCOPE will prevent you from making invalid connections. When trying to make such a connection the mouse pointer will change to a black connector plug with a red forbidden cross.

On the one hand this procedure is the fastest way to connect two *Pads*. On the other hand the modules to be connected have to be on the same layer. This way it is not possible to connect two *Pads* on different layers or a *Pad* of a circuit component with a *Pad* with a surface control element.

There are other procedures which allow you to do this. The idea is to select a *Pad* and remember it, then select another *Pad* anywhere and make the connection.
The first procedure uses the *context menu* of the *Pad* in the *Project Window*:

- In the *Project Window* open the *context menu* (RMB click on the *Pad*) of the first *Pad* you want to connect and select 'Store Pad'.

- Release the mouse button and move the mouse to the second *Pad*. As for the first procedure the mouse pointer changes to a red connector as you move it to the other *Pad*.

- Open the *context menu* of that *Pad* and select 'Connect to <name of the Pad>'. The *Pads* get connected.

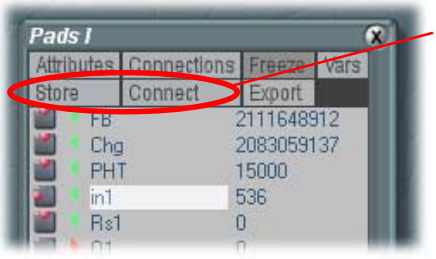
The third approach is similar but instead of the *context menu* the *Pad List* window is used:

- Select a module in either the *Project Window* or the *Project Explorer*.

- Open the *Pad List* window and within select a *Pad*. Press the 'Store' button on top of the *Pad List* window.

- Select another module in the *Project Window* or the *Project Explorer*. The *Pad List* will update to show the *Pads* of the currently selected module.

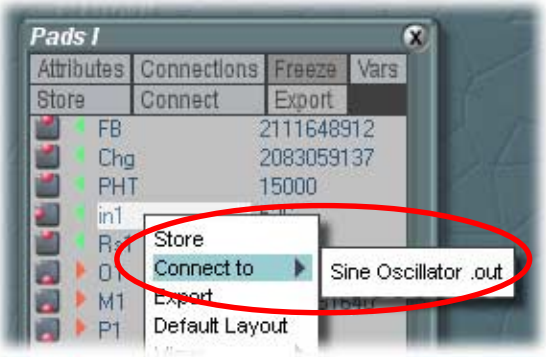
- Choose the *Pad* to which you want to connect the stored *Pad* and press the 'Connect' button next to 'Store' in the *Pad List* window. The connection is made.



Store and Connect commands in the Pad List window.

The fourth method allows you to store multiple Pads simultaneously. From the *Pads' context menus* in the *Pad List* choose the one to connect:

- Select a module in either the *Project Window* or the *Project Explorer*.
- Open the *Pad List* window and within select one or more *Pads*. From the *Pad*'s *context menu* use the 'Store' command.
- Select another module in the *Project Window* or the *Project Explorer*. The *Pad List* will update to show the *Pads* of the currently selected module.
- Open the context menu of the *Pad* which you want to connect. Select 'Connect to ->' and a list with the stored Pads opens. Pick one of the entries to make the connection.

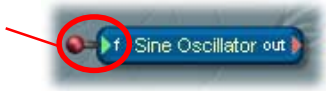


The last three procedures can be combined. So you can for instance use the *Pad List* to store a *Pad* and the *context menu* in the *Project Window* to make the connection.

Antennas

Connections that are not within the same layer will not be displayed as virtual wires. Instead the so-called **antenna**, which indicates that the *Pad* is connected to another one, is displayed.

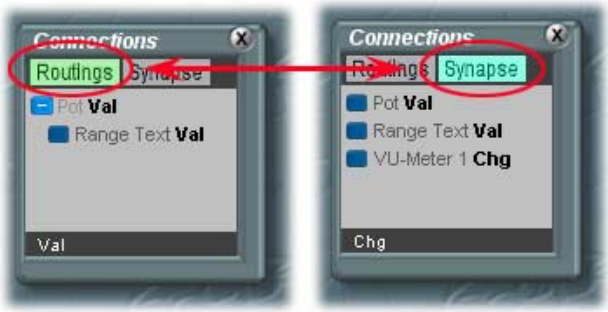
Antenna which indicates a connection



Antennas enable you to connect modules from different sub-trees. Even the connection between surface control elements and circuit components is possible.
However the character of the connection is not visible from the antenna. There is only one antenna for the different connections. So you can only guess from the corresponding *Pad* if the connection is synchronous or asynchronous.

Over viewing connections

To help you keeping an overview of the connections of a specific *Pad* there is the *Connections* window. It can display the routings or the synapse of the selected Pad.
Routings means that it will hierarchically list all *Pads* that are connected to the selected on - exactly the way the user extablished the routings. Hierarchies in this connection are visualized by a typical tree structure.
In contrast to that the **Synapse** view 'only' displays all connected *Pads* without considering the hierarchy. This means that also the selected *Pad* itself is listed.



Besides providing an overview over the connections it is also a fast way to select a module that is connected to the currently selected *Pad*. You can double-click an entry of the list and the

corresponding module and *Pad* will become selected - this means the *Pad List* and its sub-windows will get updated and the module will be highlighted in the *Project Explorer*.

Releasing connections

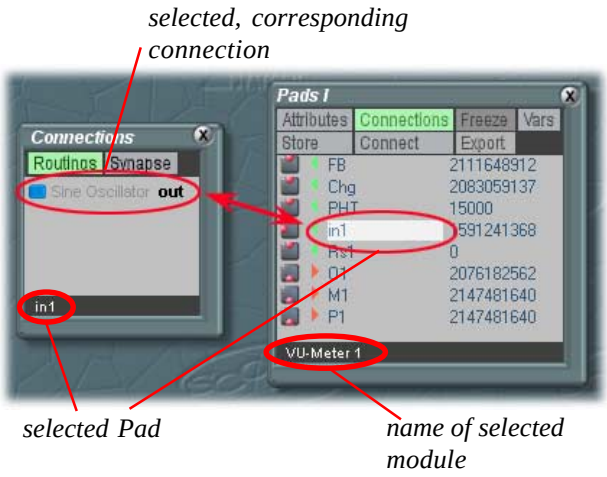
From time to time you might also need to disconnect *Pads*. So you can easily try how another module would alter the signal, or you feed the signal of another oscillator into your circuit.
You might try alternative modules or routings quite often.

To release a connection select the virtual wire with the mouse and press the delete key <Del> on your computer keyboard.The wire becomes highlighted if it is selected.

Alternatively you can either click on the 'Delete' button in the *command bar* or re-select the two connected *Pads*. When moving the mouse over the second *Pad* the pointer changes to become a scissor which enables you to cut the wire symbolically.

There is also the possibility to disconnect the *Pads* by deleting the connection from the *Connections* window. For connections that are not within a single layer of your circuit this is the most convenient method.

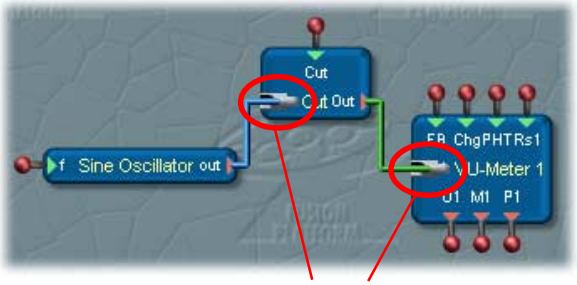
- In the *Pad List* window select a *Pad*
- The *Pad*'s connections are displayed in the *Connections* window
- Select the connection in the *Connections* window.
- Press the <Del> key to release the connection.



Converters - Connecting Pads of different types

As long as the *Pads* you want to connect are of the same type everything is fine. Of course the functionality does not end there. It was said more than a dozen times before - flexibility is one strength of the SCOPE environment.
What does connecting unequal *Pads* mean? Well, for analog audio it may require any type of adaption - like conversion or amplification. Imagine you wanted to connect the balanced, studio level outputs of your tool to your home stereo.
So if you want to connect a synchronous signal to an asynchronous *Pad* you would need a converter that allows this connection. A synchronous signal provides impulses with every word clock impulse. An asynchronous *Pad* expects impulse to occur only occasionally. So

the converter must thin out the impulse density of the signal.



converters

Accordingly a unipolar to bipolar converter has to adjust the value range of the signal so that it is compatible to the *Pad*.

In most cases connecting different *Pad*-types is possible and converters are inserted automatically. Nevertheless implying converters it is not possible to convert a MIDI signal to any other type of signal and vice versa. However this functionality is provided by dedicated modules which are included in the *Module Library*.
The *Pads* of the converter allow you to fine adjust the conversion. The converter is automatically removed when the connection is deleted.

A connection is also deleted when one of the modules on either side of it is removed.

The most frequently used converters are *synchronous to asynchronous* (sync to async) and double to long respectively.

synchronous to asynchronous converter

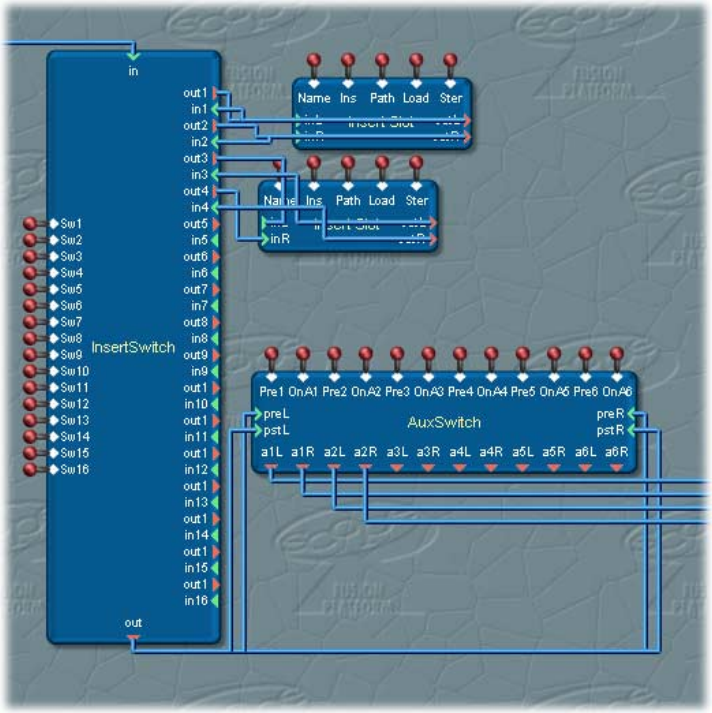
double to long converter

Their representation in the *Project Window* - as for all converters - does not differ from each other. However they have a different set of *Pads* that allows to adjust the conversion settings.
The most interesting *Pad* are those that allow to adjust the value range for the input and the output respectively. This range is set by the minimum and maximum values (Min and Max).

Connections and Pad orientation

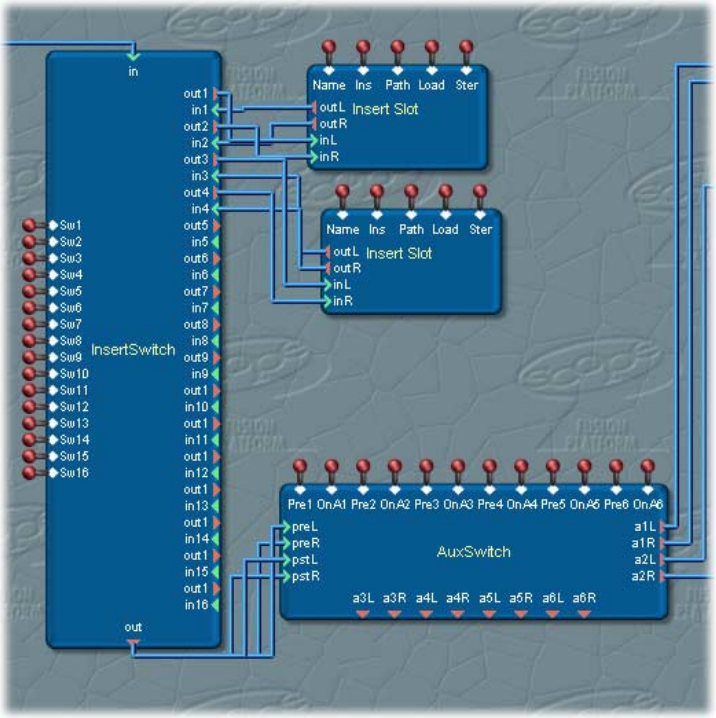
The more complex a circuit becomes the harder it gets to be able to overview it. A lot of connection and virtual wires that are pulled over modules does not augment the legibility. Therefore it is possible to reposition the *Pads* on a module.
As you already know a *Pad* can point towards any side of the module. So if a *Pad* that is currently oriented towards the top of the module but has a connection to a module that is underneath its own module you can re-orientate the Pad towards the bottom of the module. This would result in the shortest connection possible. Cluttering by virtual wires would be reduced to the minimum.

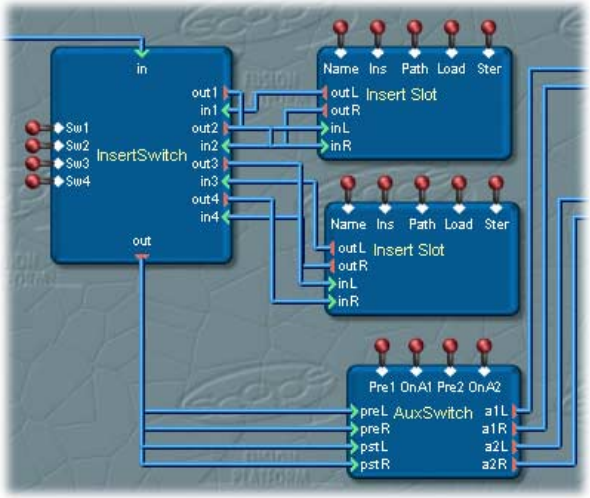
Additionally you can also hide a *Pad*. This is very practical if the module has a lot of *Pads* but you do not connect the *Pad* within this layer. It is important to point out that the *Pad* can be connect even if it is hidden it is just not displayed in the *Project Window*.



Connected mixer components. The routings of the virtual cables is not favorable.

The re-orientation of the Pads augments the legibility of the circuit.





Hiding unused Pads frees up the work space.

The first step on the way to your own device is to connect multiple modules. Due to the vast possibilities and the flexibility SCOPE provides to you there are several procedures to make and to release connections. Antenna connections even allow to connect two modules that are in totally different locations.
In most of the cases converters that adapt the different signals and their value ranges are place automatically. This improves the ease of use and allows the user to build circuits intuitively.

Additionally invalid connections are automatically prevented. Not allowing such connections helps minimizing the errors that can occur while inter-connecting modules. Also uncontrolled or unwanted feedback loops are not possible which helps protect your ears and your monitoring.

Organizing the circuit

Having added and connected several modules your circuit is growing in size. To avoid a lot of scrolling in the *Project Window* you can fold modules to build groups of processing entities.

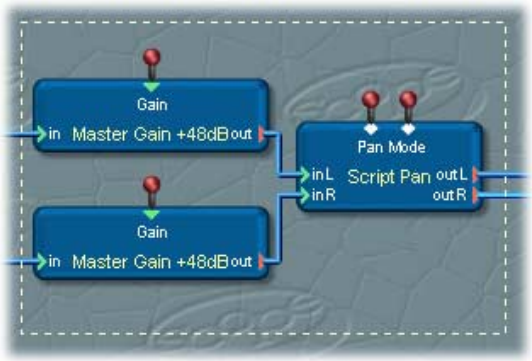
Folding

Folding modules to a logical entity is not only a way to minimize the needed display space but also to structure your circuit. As there is no restrictions in the number of layers a hierarchy is allowed to have inside SCOPE, the granularity of the structure can be set by the user.

Such a 'folder module' can be renamed and saved as a new module that can be re-used in a larger context. It is handled like every other module. Folding, saving and then reloading multiple times is a fast way to generate re-occurring structures within a circuit.

The procedure for folding several modules is straight forward:

- Select any number of modules in the *Project Window*.

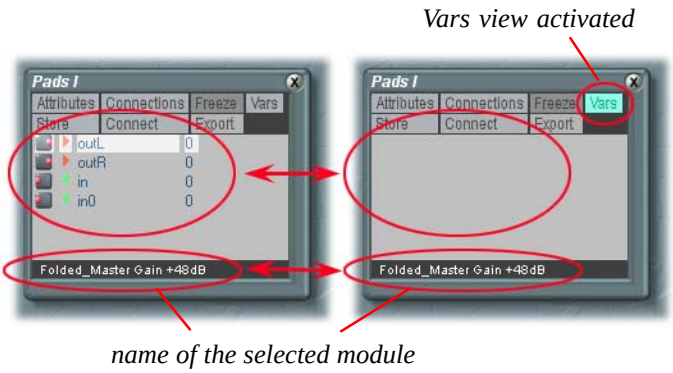


- Use the folding command by pressing the corresponding button in the *command bar* or by using the short-cut <Ctrl>-<F>.

A new module is generated that includes the formerly selected nodes. A new hierarchy layer for the folded modules is automatically added. However you will stay on the same layer in the *Project Window*.
If the selected modules were connected to others which were not folded SCOPE tries to preserve these connections. It will automatically export the involved *Pads*.

Exported Pads

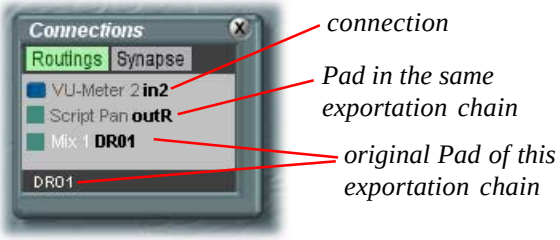
On the new module you can find these ***exported Pads***. If you have a look to the *Pad List* and the *Var List* you will find out that these *Pads* are only listed in the *Pad List*.



It was explained before that *Pads* are references to *Vars* and each *Var* can have several referencing *Pads*.
So the *exported Pads* on the new module are new references to the *Var* of their original *Pad*. The *Var* itself does not get exported and therefore cannot be derived from the algorithm of the folder module. So it will not appear in the *Var List* of that module.

However it is important to point out that the *exported Pad* is **referencing the *Var* directly** and **not** the *Pad* from which it was exported. This means that you can export *Pads* over multiple layers and if you only need the exported *Pad* on the upmost layer you can easily delete the *Pads* on the other layers without loosing the reference.
This might be confusing first but it reduces cluttering which is especially useful throughout complex projects.

The relations can be easily reproduced in the *Connections* window. The *Routings* view has a color-coding for connections and exportations. A blue item indicates a classical connection via a virtual wire or an antenna. Green items are members of an *exportation chain*. The module that holds the original *Pad* in this chain is printed with white letters.



In contrast to that the *Synapse* view only lists the original Pad - this time without a color differentiation - and the connected Pads. The selected Pad is not displayed as is none of the other exported Pads of the exportation chain.

You can change the orientation, the name and even the I/O-type of the *Pad* without affecting the reference or the functionality of the *Pad*. SCOPE will always take the attributes of the *Var* to evaluate a *Pad* for connections. These attributes can be looked up in the *Var Attributes* window which can be accessed from the *Pad List* window.

Exporting Pads

Pads get automatically exported if they are connected to other *Pads* while being folded. However it is also possible to export *Pads* manually to make connections on a higher layer.
While discussing how to make connections another method to connect modules that are not within the same layer was mentioned. So why should you export a *Pad* to make a connection if you could connect it via antennas?

It is often easier to understand the structure and the connections if *exported Pads* are used instead of antennas. The latter connections are only comprehensible by consulting the *Connections* window.
This however does not pay attention to the hierarchy of the modules (the *Routings* view shows the hierarchy of the connections, not necessarily that of the layers!). A connected module on the

same layer is displayed next to a connected module that is a part of a different sub-tree of the module tree. This does not facilitate the comprehension of the signal flow.

Exporting a *Pad* to make a connection on a higher level can be much more comprehensible. The *Pad* and its name is visible on the upper node. If you feel the need you can change the name to something that seems more meaningful or more characterizing to you.
Although you only see the *Pad* on the folding module most often it is not too difficult to associate the corresponding module.

To export a *Pad* there are two procedures - the first one makes use of the *Pad*'s *context menu* in the *Project Window* or the *Pad List* window respectively, the second one of the functionality provided by the *Pad List* window. These methods work very similar to the ones you came across wile making connections via the *context menu* or the *Pad List* window respectively.

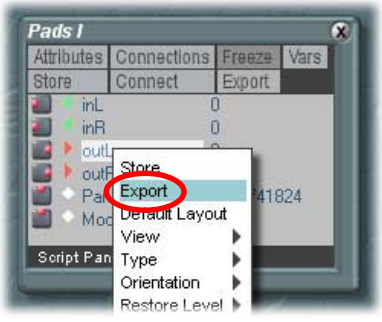
For exporting a *Pad* with the commands in its *context menu* you have to be in the *Project Window* and on the same layer as the corresponding module.

- Move the mouse pointer over the *Pad* so that the pointer changes to a connector
- Press the right mouse button to open the *Pad*'s *context menu*
- choose 'Export Pad' from the list that just popped up
- A little arrow will appear right next to the *Pad* indicating that the Pad is exported

arrow which indicates that this Pad is exported

The same can be achieved from the *Pad*'s *context menu* in the *Pad List* window. The only difference is that the command is only labeled 'Export'.

- Select the module in either the *Project Window* or the *Project Explorer*
- In the *Pad List* window select the *Pad* you want to export
- Open it's *context menu* and choose 'Export'
- The 'Export' button on top of the *Pad List* window gets highlighted



The equivalent procedure utilizing the *Pad List* window is as followed:

- Select the module in either the *Project Window* or the *Project Explorer*
- In the *Pad List* window select the *Pad* you want to export
- Press the 'Export' button on top of the *Pad List* window.
- The button will be highlighted and a little arrow will appear next to the *Pad* indicating that it is exported.

No matter which procedure you use to export *Pads* - the *Pad* that is exported always displays with an arrow in the *Project Window* and when select in the *Pad List* window the 'Export' button is highlighted.

To see the new *Pad* on the folder node navigate to the upper layer. A new *Pad* should be visible on the node - its name and orientation is the same as from the *Pad* it was exported form. The I/O-type should correspond to that form the *Var* of the original *Pad*.

There are situations in which it is better to use antennas to make connections and in others it is better to export the *Pads*. Exporting each and very Pad does not necessarily augment the comprehensibleness of the circuit but more likely results in cluttering the layers with *Pads* and connections.
Nevertheless it might be easier to start with using *exported Pads*. Normally a few weeks you are deeper into connections and you may have found situations where you want to have antennas instead of *exported Pads*.
Exported Pads might also be the better choice when working in a team as well as when looking into a module to modify something after several months.
Becoming more familiar you will get a feeling which technique to use in a specific situation.

Undoing exportations

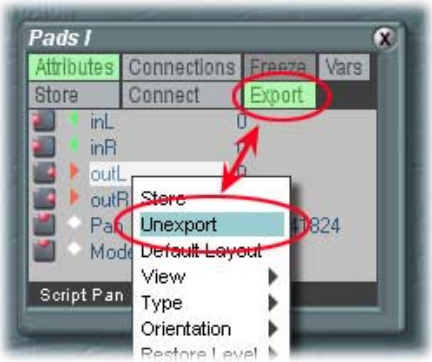
There are several scenarios when it may be required to delete formerly exported *Pads*. This may be because you do not find it appropriate the way SCOPE has automatically exported the connected *Pads*. Or perhaps you want to add another module at the end of a processing chain which is already folded and the *Pads* that connect to the rest of the circuit have already been exported. In either case you would have to undo the exportation of these *Pads*.

The methods to undo an exportation are analogous to those engaged to export a *Pad*. You can either make use of the *context menu* in the *Project Window* or in the *Pad List* window respectively, or you can deactivate the 'Export' button in the *Pad List* window.
Each procedure is briefly reviewed below. There is a point that is common to all of the procedures. To undo an exportation of a *Pad* you have to select that *Pad* which you have formerly exported. You cannot undo the exportation from the *exported Pad* itself.

- In the *Project Window* position the mouse over the *Pad* for which you want to undo the exportation
- The mouse pointer changes to a connector
- Press the RMB and select 'Undo exportation' from the *context menu*

Alternatively you can also do the same in the *Pad List* window.

- Select the corresponding module in the *Project Window* or the *Project Explorer*
- In the *Pad List* window select the right *Pad*
- The 'Export' button on top of the *Pad List* window should be high-lighted
- Click on the 'Export' button to deactivate it or use the 'Undo exportation' command form the *Pad*'s *context menu*



The arrow next to the *Pad* should disappear as should the *exported Pad* on the folding node.

Unfolding

Folding is extremely helpful for structuring your circuit but it is also a handy tool to pack a part of the processing network to save it or to copy it. After you have executed these operations you might want to use your modules as they where before - unfolded.

To unfold modules select the folding module and apply the 'Unfold' command from the *command bar* on top of the *Project Window*. Unfolding brings the modules from the underlying layer to the current layer and it will delete the folding node.

Alternatively to the *command bar* you can also use the key combination <Ctrl>-<Shift>-<F>.

Be aware that it deletes the folding node. It is also possible to use a normal processing module as a folder for other modules. This is often used if there are modules that extend a specific functionality of another module. The module that gets extended is then used as folder for the other modules.
Using the unfold command in this case will delete the outer module. Most often this is not desired.

Folding is a convenient way for encapsulating entities within a circuit. This is very helpful to organize the processing network of your device as well as to maintain and re-use specific composed units.
The capability to export Pads even enables you to use the originated module as a black box - with inputs and outputs.

This entity is a ordinary module and can be treaded that way. Cut, copy and paste

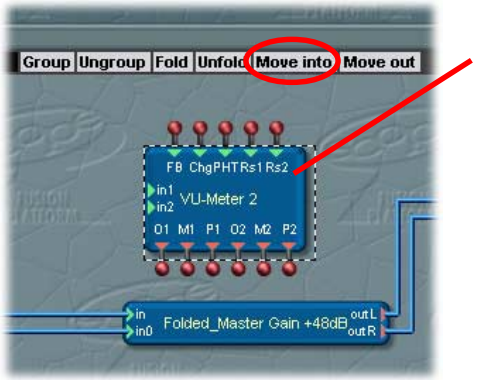
it to engineer your device. As there is no restriction in the number of nodes or layers such a module can have, you can maintain and re-use an entity of any complexity.
You can export *Pads* to any higher layer in the ***module tree*** of this circuit. You can hide or even delete the *exported Pads* on layers where they are not needed.

Folding and exporting *Pads* enhance the comprehensibleness and the usibility of complex circuits. They do not construct a hierarchy or a topography that could imply any limitations as the project is evolving.
The executed commands are fully reversible and you maintain full control over the structure.

Move into

So, what if you want to move an existing module into another module on the same layer? Besides the fold commands the move operations help to manage individual modules within a hierarchy.
The 'Move into' command allows you to move a module onto the first underlying layer of another module. As you would expect connections are preserved and connected *Pads* are automatically exported.

You can execute the move command from the *command bar* or by pressing a key combination.

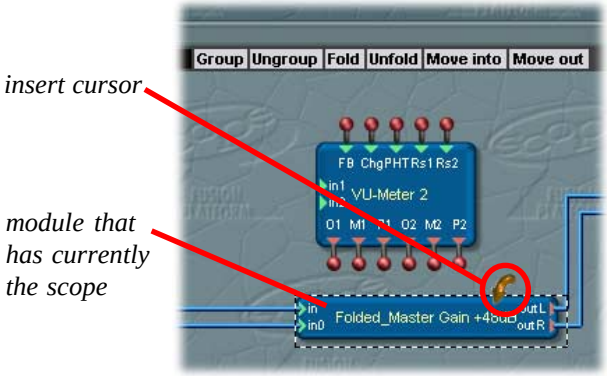


- In the *Project Window* select the module you want to move into another one.

- Press the 'Move into' button in the *command bar* on top of the *Project Window* - alternatively you can press the key command <Ctrl>-<I>. Note that the module is no longer the current module after pressing this button or key combination respectively.

- The mouse changes to an insert arrow.

- Pick the module which you want to host the other module. While moving the mouse over the other modules they are surrounded by a dashed rectangle.



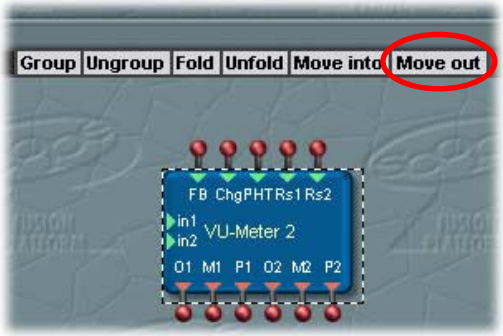
The first module is move onto the highest hierarchy layer of the picked module. By double-clicking that module you navigate to this layer.
There you have full access to the module as before you completed the move operation.

Move out

The complementary function is provided by the 'Move out' command. It pushes the selected node(s) up one layer in the hierarchy.
Connections to the module that are routed via *exported Pads* stay intact. The

exported Pads on the former parent node are delete and the connections are directly routed to the *Pads* on the moved node.

To perform this command select the node that you want to move up one layer and press the 'Move out' button in the *command bar* on top of the *Project Window.* Alternatively you can also use the short cut <Ctrl>-<Shift>-<I>.



The module is immediately moved to the upper layer.

The 'move into' and the 'move out' commands only move the selected modules over one layer up or down. To move a node to another branch in the hierarchy these commands are not convenient.
However there is a way to accomplish this in the *Project Explorer*.

Dragging modules in the Project Explorer

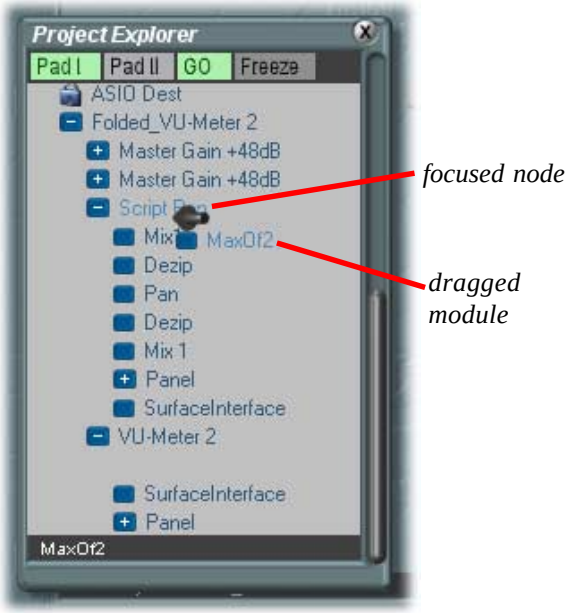
The *Project Explorer* shows the *module tree* of the project. You are able to expose selected branches of the tree by clicking on the blue boxes of the parent nodes. This way you can expose level per level of the hierarchy.

Inside this structure you can drag any node and drop it into a different one. The really time saving feature of this operation is that all connections of the module are preserved.

To drag a node onto the same layer as another one hold down the <Shift> key while dragging. This can be combined with the <Ctrl> dragging.

The procedure is as follows:

- In the Project Explorer select a module - the module can have any number of layers and it can be located anywhere within you circuit.
- Drag it over another module. As you drag it over other modules one aftr the other becomes highlighted to indicated which module is currently focused.
- If you drop the module (by releasing the LMB) you drop the module into the first layer of the currently highlighted node.
- Hold down the <Shift> key while dragging to drop the module onto the same layer as the currently highlighted node.



Dragging in the *Project Explorer* is really a very practical function. You can move modules in a quick and uncomplicated manner.

These are the basic functions for *Circuit Design*. The way you work with modules and their *Pads* - how to select or rename them. How to duplicate, copy and paste modules.
To structure and organize your circuit you can fold modules. Therefore you can export *Pads* to generated a processing unit. Nevertheless all modules on any layer of your circuit stay accessible.
You can even move modules from one layer to an upper or lower one with the move commands in the *Project Window* or quasi freely in the Project Explorer. Connections can always be maintained.

These are the basic steps that allow you to build the processing network for sophisticated devices.

Besides these basic technics there are a couple of advanced functionalities.

Circuit Controls

The *Library Modules* comes with their basic surface that allows instant control over the module's control inputs. To get an idea of the modules functionalities and charateristics this is very helpful.

However in more complex circuits these surfaces might rather complicate the handling than to provide intuitive control. There would be that much surfaces that you cannot manage all of them on your computer screen.
On the one hand you might have to close and reopen the surfaces of specific modules to alter their controls. As a complex circuit normally is set up with layers this would demand a lot of navigation and would not be that convenient.
On the other hand the controls could be combined on a new surface. Building a surface for a module that is not already finished is not the ideal solution though. You would have to change the surface with each module you add to or remove from the circuit. Furthermore you might have to re-assign the surface to another module if the upmost module changes.

While going through the *Quick Start* manual you might have already noticed that it is also possible to place control elements directly within the circuit. You can connect them to the Pads of the modules to control them.
So you would not need the surfaces to control the values. You navigate to the layer of the module and find control elements to adjust the values of the Pads. You can immediately overlook the connections and intuitively make setting.

It was pointed out that you normally navigate in *Move* mode. When selecting control elements in the circuit in *Move* mode you can position them but you cannot change their values. To do so you have to switch to *Use* mode.
Although this seems to be a little drawback it is not that bothering. You can easily switch between *Move* mode and *Use* mode with the short-cuts <Ctrl>-<U> (*Use* mode) and <Ctrl>-<M> (*Move* mode). Additionally differentiating between these modes assures that modules are handled the same way.

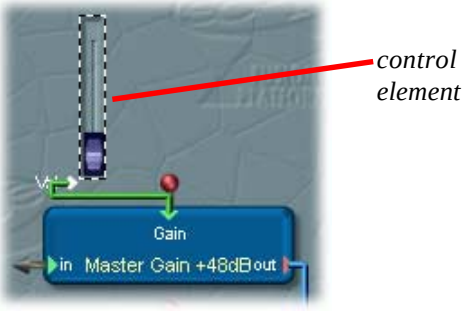
Use mode to adjust the values of circuit controls

Control elements used in circuits are also called circuit controls - to distinguish them from control elements on surfaces (surface controls).

The procedure to control the *Pads* with control elements that are in the circuit involves multiple steps. You have to place and connect the control elements in the circuit and then you have to adjust the value ranges.

Placing and connecting circuit controls
In the File Browser navigate to the 'Circuit Controls' folder within the 'Circuit Design' directory in the *Module Library*. There you can find a collection of control elements for use in the circuits.
You can find the same control elements and more of them in the 'Surface Design' directory, but these are ways faster to access while doing *Circuit Design*.

Drag the controllers you plan to use into the *Project Window*. The controllers have exactly one *Pad*. Connect this Pad to the desired *Pad* on the module.



Adjusting the control ranges
Now that you have connected the circuit control elements with the appropriate *Pads* it might be necessary to adjust the value ranges of the control elements.
By default the control elements have a value range over the whole positive range with a linear characteristic. Range Text fields that indicate other calibrations (p.e. Range Text Hz) are an exception to this.

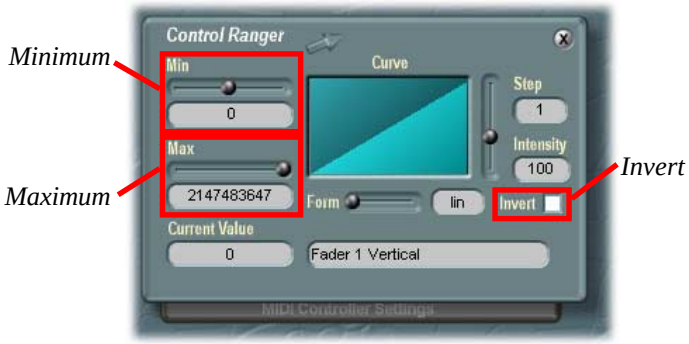
You can use the *Control Ranger* window or the *Vars* of the control element to adjust the value range. This is obviously useful for faders and potentiometers, but also for range text fields.
To use the *Control Ranger* select 'View -> Control Ranger' from the *menu bar* to open it. Then make your settings.
To adjust the value range with the *Vars* switch to *Vars* view in the *Pad List* window.

Adjusting control ranges
It is neither always desired that a control element goes over the full range nor is a linear characteristic always applicable. If you want to control a logarithmic scale like the level of a signal for instance, a linear control range will not provide satisfying results. You can easily customize the controls.

Adjusting the value range
The value range is determined by a minimum value and a maximum value. The widest range possible is from -Max to +Max. All ranges inbetween that are possible. The maximum has to be bigger than the minimum.
However there is an extra 'Invert' flag that allows to invert the range.

To set the values in the *Control Ranger* open it from the *View* menu and adjust the settings of the **Min**, the **Max** and the **Invert** fields. For the **Min** and the **Max** values you can type the values directly

into text field or you can use the sliders above the text fields.



Invert is a flag that can be set by clicking into the check box.

The Step value

Having declared the boundaries of the range you might prefer to jump stepwise between the upper and lower limit. The **Step** value allows to specify how many steps it takes to cycle between the limits. For values greater than '1' the smallest possible increment/decrement is the quotient of the range divided by the Step value.
As an exception for a value of '1' the step-size is one.

The **Step** function provides a convenient method to have stepped potentiometers or faders for switches.

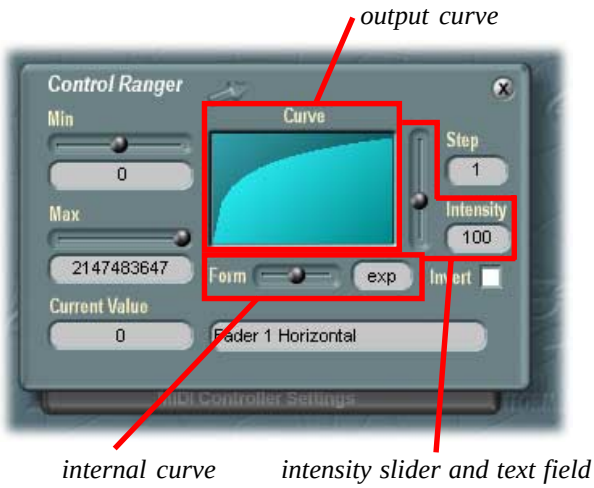
Adjusting the characteristic

According to the scale of a specific *Pad* it might be smarter to use a logarithmic or exponential characteristic for the control element. As an example the control for the level of a signal was already mentioned.
The control element has a **Curve** *Var* which enables you to specify its characteristic. However as the name suggests it determines the internal curve of the control and not the output.

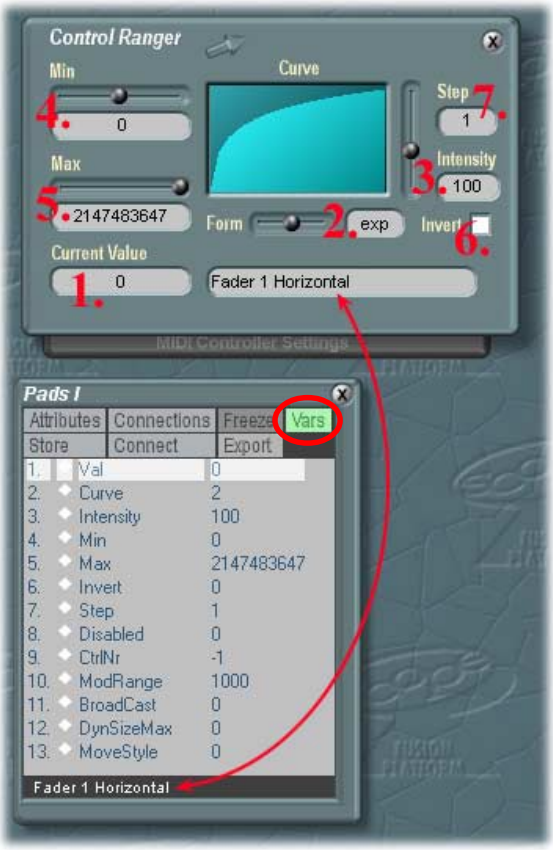
How to understand that? The control element gets its 'commands' from the display representation of the module - p. e. the slider image you see on the screen. As typical for a screen the info is always linear - like one pixel up/down/left/right.

Now this information has to be transformed to another scale. The linear input values interpolate the output values by scanning the internal curve.
All three curves do have the same amount of segments - the count is determined by the value range.
Now the control element receives a specific number of counts from the user interface - this is the input curve. To obtain the output value that corresponds to this position of the control element it takes the x-axis and y-axis values of the internal curve at this segment. Then it calculates the output by swapping these values.
You can say that the internal curve is axially symmetrical to the output curve and the linear input values define the axis.

So if you want a logarithmic output the internal curve should be exponential. In the *Control Ranger* the **Form** slider controls the internal curve and the **Curve** display shows the output curve.
The intensity slider and text field enable you to further adjust the characteristic. Drag the slider or enter value into the text field directly.



The *Var List* view of the *Pad List* window provides *Vars* for each of these options. So you can adjust the settings either in the *Var List* or in the *Control Ranger*.



Using control elements in the circuit instead of relying on the basic surfaces of the modules is a fast way to alter settings in your processing network.
You may find it ways faster and more concise then managing the basic surface of all the modules.

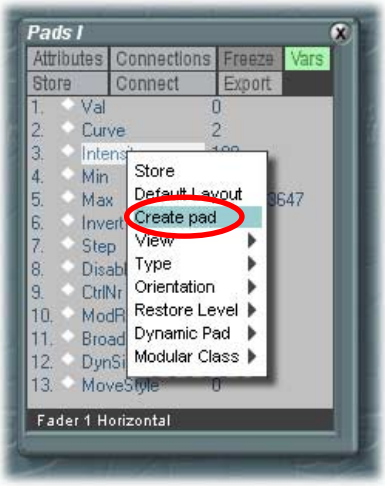
Creating Pads from Vars

You might like to access some *Vars* from other modules - like to control the minimum value of a control element. However *Vars* cannot be accessed from other modules.
To control the value of a *Var* from another module you have to create a *Pad* for this *Var*. This can be done by using the 'Create pad' command from the Var's context menu.

Creating Pads

To create a Pad from a Var do the following:

- In the *Var List* open the *context menu* of the *Var*.
- Select 'Create pad' from the menu.



A *Pad* is created for this *Var*. Switch to the *Pad List* view to see the new *Pad*. It can be connected like every other *Pad*.

Deleting Pads

Analogously you can also delete *Pads*. The corresponding *Var* will remain, only its external reference - the *Pad* - will be deleted.
There are two procedures for deleting the *Pad* of a *Var*.

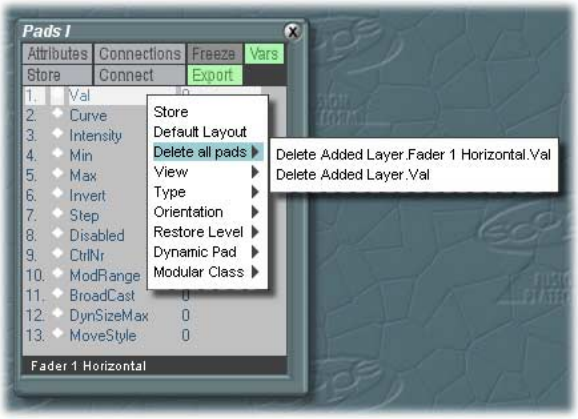
You can either:

- In the *Pad List* select the *Pad* you want to delete
- Hit the <Del> key.

Instead of selecting the Pad you want to delete you can also select the Var and choose the Pad you want to delete. Especially if the Var has more than one Pad this might be the smarter way to accomplish this.

- In the *Var List* open the *context menu* of the Var from which you want to delete one or more Pads.

- Select 'Delete all pads' (the command in the main menu without entering the sub-menu) or select a specific Pad from the submenu.



Perhaps it is not quite clear at this point why deleting *Pads* is a great feature. It was already mentioned that it is possible to hide *Pads* in the *Project Window* and to hide hidden *Pads* in the *Pad List*.
There are modules with lots of *Pads*. Now imagine that you are not only using one of these modules in a circuit but several on multiple layers. To keep things clear in the *Project Window* it is often good to hide *Pads* that are not connected within this layer.
Please keep in mind that hidden *Pads* can be connected!
By hiding these *Pads* you clear up the view in the *Project Window*. Nevertheless, most often you would like to see them in the *Pad List* - they are connected and their values may keep changing. As a result your *Pad List* is still quite long.

However there are also Pads that do not get connected. Their only purpose is to allow to make different settings - like you have just seen it for the control elements. Normally you set such a value once for a module as long as it is in a specific circuit - at least you will not change it frequently!
By deleting these *Pads* you can clean your *Pad List*. So you can hide *Pads* that are not connected within the same layer and with hide hidden *Pads* you can hide or show them in the *Pad List*. The *Pads* you do not change frequently and which do not get connected can be deleted. To see them switch to the *Var* view in the *Pad List* window.
This is a convenient way to handle or hide complexity. This can improve your work-flow.

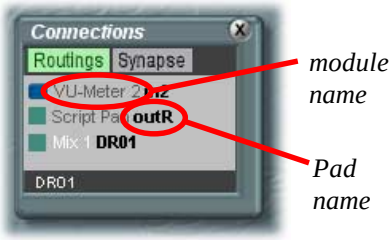
Connection-centered navigation

Until now when talking about navigation, we mainly considered navigation inside the *Project Window* or the *Project Explorer*. This navigation is related to the hierarchy of the module.
However the navigation related to connections was alluded before while discussing connections.

There are situations when you want to select the module or the *Pad* that is connected to a specific *Pad* of the current module. It is not always the case that the other module is on the same layer rather than being anywhere in the hierarchy.
Assuming the scenario that you are working on a quite completed circuit. You are making some fine adjustments on a module. The modulation on one *Pad* is too exaggerated so you want to change the settings. Therefore you have to change some settings of the module that is connected to this *Pad*. As often in a complex circuit the other module is somewhere in the circuits hierarchy.

Instead of having to browse to the other layer you can also select the *Pad* (in the

Pad List window) and open the *Connections* window. There all connected *Pads* are listed, each one with the name of its module.



Double click on one of these entries. The module of this *Pad* will be highlighted in the *Project Explorer* and the *Pad List* window will update to show the *Pads* and the one is selected. As you have access to the *Pads* and *Vars* now, you can make most changes to the settings of the module.
Furthermore by holding the <Ctrl>-key while double-clicking the entry in the *Connections* window you can navigate the first *Project Window* to the layer of the corresponding module. The module becomes highlighted in the *Project Explorer* and the *Pad List* window updates to show its *Pads*.

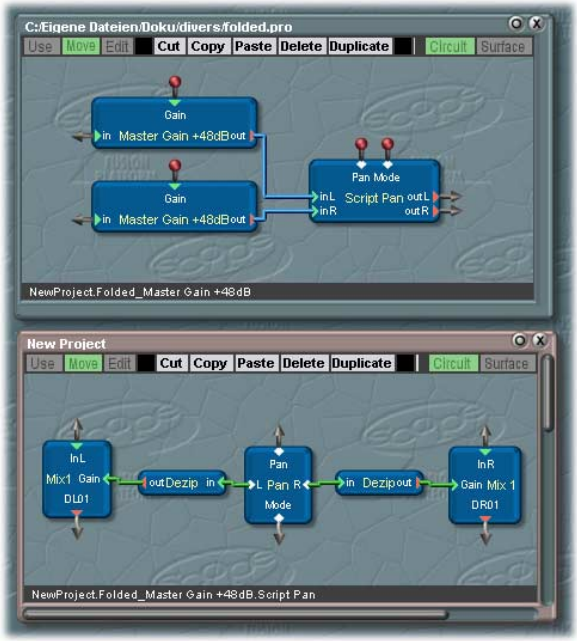
This is a fast way to jump between connected modules. It is also good to browse circuits that you did not build yourself. If you have understood the connections you know how the module works.

The second Project Window

Circuit controls facilitate instant access to the settings of your module without clustering the computer screen with basic surfaces. Nevertheless there are only a few controls accessible at a time.

However, with the second *Project Window* you can have direct access to two different layers. The two *Project Windows* work independently and provide two different views of the project.

Each *Project Window* has ist own set of commands - like cut, copy and paste. You can set each of the them to a different view mode (*Circuit* view or *Surface* view) - that is especially interesting for simultaneous design of the circuit and the surface of a module. However, the work modes are synchronized so that modules behave the same way in both windows.
Setting them to different location within your project you can easily make connections, control nodes that influence each other, compare two circuits etc.
You could also create a new layer that holds all the circuit controls for the module and open it in the second *Project Window*.



In most situations and for most operations both windows work the same. However it was already mentions that for connection-centered navigation **only the first** *Project Window* is affected when you double-click an item in the *Connections* window while pressing <Ctrl>. You might want to take this into account.

Freezing the content of windows

When working with two Project Windows or when applying the connection-centered navigation the Pad List window and the Project Explorer become updated each time a new module is selected. For the Project Explorer tthis might include that deeper levels of the hierarchy are exposed and that it scrolls to in any direction to display the newly selected module.
As a result the view in the Project Explorer gets more complex and you might lose orientation.

To prevent such confusions the *Project Explorer*, the two *Pad List* windows and the *GO List* window can be frozen to refuse any updates of their current focus. This means that after having frozen one of these windows you can select any other module and the window will not update to display the current selection.
To freeze one of these windows click on the 'Freeze' button at the top edge of the window. The button is highlighted to display that this window will not update to correspond to the current selection.

Nevertheless you can navigate manually in the *Project Explorer* and the *Pad* values in the *Pad List* windows are still updated.

Having two *Pad List* windows with the possibility to freeze one of them is of great use in lots of situations. In this case the *Connections* window and the *Var Attributes* window will update with the *Pad List* window that is not frozen.
This combined with the connection-centered navigation and the two *Project Windows* speeds up the work-flow significantly.

Building groups

You have already got to know that you can fold several modules to save or copy them. However it is not always desired to first fold the modules, thencopy the modules and in the next step unfolding them because you can them in the same layer as other modules thatalready exist. Grouping provides a faster way for those tasks.
However it might be more confusing first. It packs the selected module into a folder that is transparent in the *Project Window*. Instead of a folder - like when folding modules - you still see the modules. Nevertheless they are grouped and by clicking nearby a module you select the group. Now you can move, copy, delete, save etc. this group of modules. Indeed the group behaves like a folder that is transparent in the *Project Window*.
Having a glance at the module tree in the *Project Explorer* you see that the modules have been moved into a 'Group' node. You can also rename this group node.

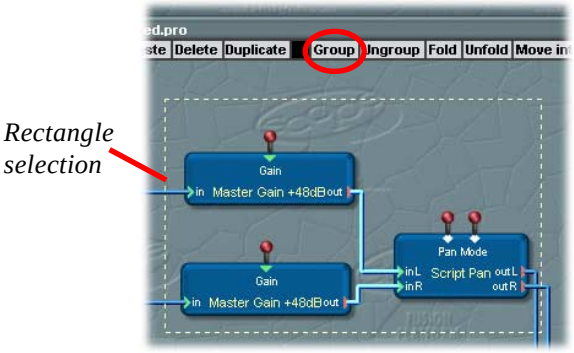
Another feature of groups is that they can be hidden.Of course this is not something you need everyday but in some situations it might be helpful.

! *Be careful with this functionality if you are working in a team. It might be hard on your tem members if you hide groups. The circuit becomes harder to understand.*

Grouping modules

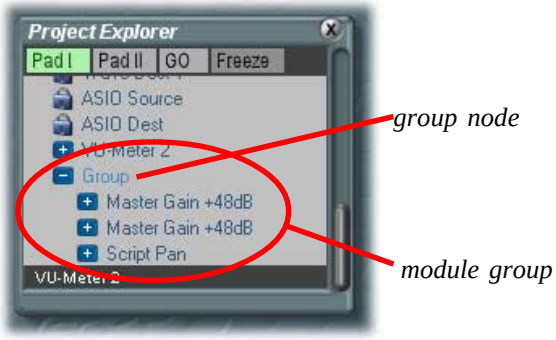
Modules can be combined to build a group by selecting them in the Project Window and teb executing the group command. The exact procedure is as follows:

- In the *Project Window* select the modules you want to group.
- Press the 'Group' button in the *command bar* or the equivalent short cut <Ctrl>-<G>.



Once you have completed the command the grouped modules still are visible in the layer. However if you click nearby a node the group selects. Now you can perform any command you can perform on a module.
You can even move modules into or out of a group with the equivalent commands.

If you have a look to the *Project Explorer* you see that the grouped modules are combined into a 'Group' node. You can rename this node like any other module.



Ungrouping modules

You can easily ungroup formerly grouped modules by using the corresponding command:

- In the Project Window select the group.
- Press the 'Ungroup' button in the command bar or the short cut <Ctrl>-<Shift>-<G>.

The group is removed as is the 'Group' node in the Project Explorer. The modules behave like you are used to it.

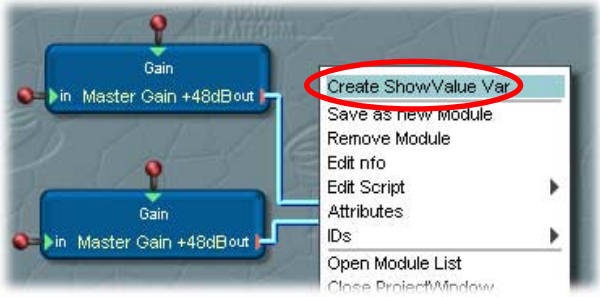
Hiding groups

If you select a group its *Pads* become visible in the *Pad List* window. There is a 'Show'-*Pad* which value is '1'. This Pad will allow you to switch the visibility of the group on (value=1) or off (value=0).

Because you may use multiple groups in a circuit you also have the possibility to specify for which value the group should be visible.
It is important to point out that the visibility of the group does not affect its operation. The group will process the signal no matter if it is visible or not.

To change the value for which the group is visible you have to create the group's ShowValue *Var*. This is a special feature of the group and not found on other modules. To create that Var do the following:

- Select the group in the Project window by clicking nearby a included module.
- Open the groups context menu.
- Select 'Create ShowValue Var'.



This *Var* is created and can be access in the *Var List* view of the *Pad List* window. The value you assign to this Var will be the value for which the group is visible.
By connection any circuit control that is ranged appropriately to the Show-*Pad* you can switch the visibility of the group on or off.

You can use this to display grouped circuit controls only when they are needed; or for text descriptions inside the circuit.
Although you could also use this for processing entities to reduce the needed place it can not be recommended to do that. It might get hard for other team members to compehend the circuit. The same might be the case if you have to make some changes to the circuit after some time.

The ShowValue *Var* can be delete like it was create - from the *context menu* of the group in the *Project Window*.

Routing via dummy Pads

Connecting modules is the underlying principle of the SCOPE development enviroment. This way you construct processing circuits and add control elements. To keep your circuit manageable you can organize your circuit by folding sub-networks.
This does not restrict the flexibility as you can easily access very module or even move it to another location within your project - without affecting existing connections. A connection-centered navigation even simplifies the navigation between connected modules.
However connections are constraints and lead to further restrictions. Perhaps you do not agree to this right away.
Connections are made between two *Pads*. The more completed a module becomes the more difficult it gets to add or replace specific nodes. Adding a module might mean having to delete existing connections, to undo exportations, to make new connections and, finally, to export *Pads* again.

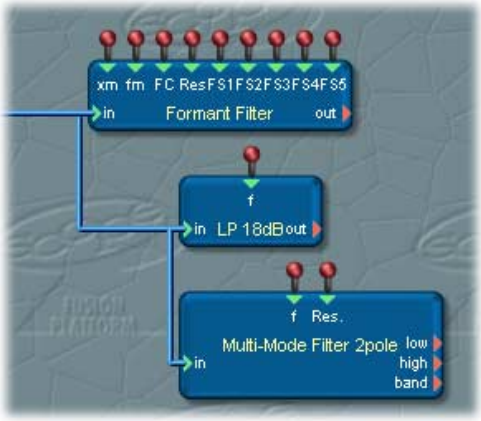
Taking into account that any two *Pads* in a circuit can be connected makes clear that it can be time consuming to insert a module in specific situations.

Things get even worse when talking about replacing modules. There is the *Module Exchanger*, which could do a pretty good job in some situation. In others you have to accomplish it manually.

A much smarter way to get around these constraints is to use *dummy Pads*.
In **/Circuit Design/Basics/** you can find the CONTROLLER PAD and the AUDIO PAD modules. These modules do not process the signals in any way but function as tie-points which are routing the signal.

synchronous dummy Pad like found in the Module Library

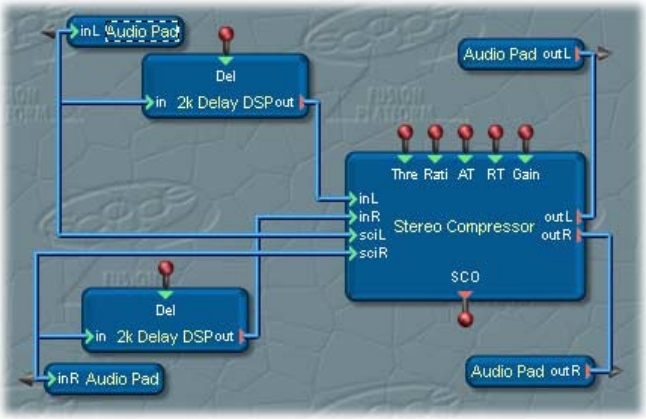
How to understand that?
It was mentioned before that input *Pads* can function as bridges when interconnection multiple inputs. They simultaneously transmit the received signal to the connected input *Pads* on the same or on other modules.



To be even more precise you have to consider that a *Pad* is a reference to a *Var*. So the signal is not passing through the *Pad* itself but rather than that it is directly routed to the algorithm. The *Pad* only influences the routings that are established between the *Vars*.

So if you now take a module with only one input *Pad* you get that functionality without the expense for extra processing. It could be used in a very flexible manner - it does not affect the signal itself and does not consume processing power.
Nevertheless it is not the perfect solution to insert *dummy Pads* into connections before each and every *Pad*. For *Circuit Design* it is advisable to use them as input bridges and output bridges of layers.

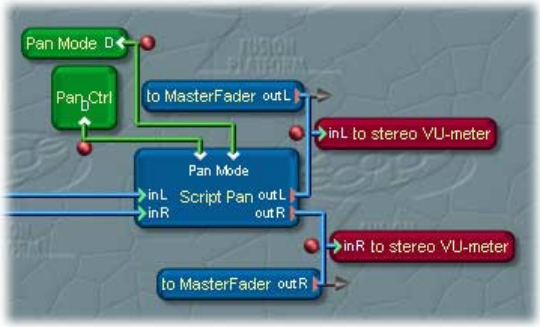
Instead of exporting *Pads* on processing modules that may get exchanged in a subsequent step, you might want to add *dummy Pads* and export them. The signal is then routed via the *dummy Pad*.



for the dummy Pads that function as outputs the I/O type and the orientation were changed

This way it gets much easier to add modules to the processing network of a layer. Having inserted such *dummy Pads* before the formerly first module or after the previously last module of the processing chain and you do no longer need to delete exported *Pads* and to export a couple of other *Pads*.
Just break the connection from and to the *dummy Pads* and reconnect the modules as desired. The *dummy Pads* will stay the inputs and outputs of the layer.
Perhaps it is important to point out that the *dummy Pads* are all of the I/O type 'InPad'. As for any *Pad* you can adjust the I/O type and the orientation of the *Pad* freely. So on the outputs you could say that you mirror the input of the next module on the higher layer into this layer.

Another scenario where it is preferable to insert *dummy Pads* is a connection via an antenna. These connections are the most critical ones when replacing a module.



Instead of using them directly on the module itself insert a *dummy Pads* to make such connections. So the antenna is not directly hung up to a specific *Pad* of a module but to a *dummy Pad* next to it. This facilitates the replacement of modules because critical connections are bound to *dummy Pads*.
Additionally you can rename the *dummy Pad* modules to reflect the module on the other end of the connection.

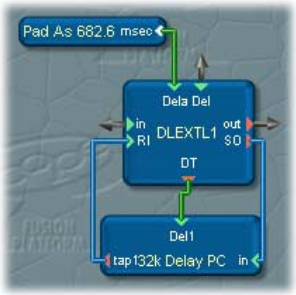
In the *Module Library* there is a *dummy Pad* for synchronous signals (the AUDIO PAD module) and one for asynchronous signals (the CONTROLLER PAD module).

Switches

Dummy Pads are bridges that can be used with expense. Although switches (**/Circuit Design/Switches/**) seem to be totally different they can be considered as more complex bridges.
Especially the signal switches - those which route synchronous signals - do not pass the signals themselves. Instead they alter the routing on the SCOPE hardware. Therefore switches do not introduce latency to the signal because they only manage the routing on the hardware.

Assigning static values

Dummy Pads are also needed in another situation. *Vars* of modules that are based on DSP-algorithms or Scopefx-algorithms are often initialize with their default value. In most situation this is the preferred behavior. This way output *Pads* do not output any signal after a module is loaded.
If you want to assign a static value to such a *Vars* you should connect a *dummy Pad* to its *Pad* and set the *dummy Pad* to the value. This way the value is send to the *Var's Pad* whilst the module is loaded.

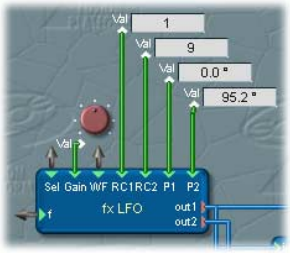


Dummy Pad that sends static value after loading.

Using *dummy Pads* for static values has also the advantage that you can rename the *dummy Pad* module to reflect the given value.

Alternatively you could also use circuit controls like a RANGE TEXT field or a potentiometer.

potentiometer and RANGE TEXT fields are used to assign a static value.



Color Coding

Another way to make it easier to comprehend the basic structure of your circuit while investigating it roughly is to use different colors for the sections. You can set the color for every individual node.
Developing and applying a color scheme for your circuits can make your life much easier. There are different starts for the definition of such schemes. It can be related to stages processing (like input, first processing stage, second processing stage, output), the designation of the signal (main audio signal, frequency modulation, different modulation controls,etc.) and so on.

Also when using the AUDIO PAD or CONTROLLER PAD to make routings via antennas you can use colors to associate the source module and destination module respecitvely, or the designation of the signal.

After having completed this chapter you are now prepared to construct even the most sophisticated circuits. Due to the various procedures to set your circuit up for your specific tasks you might want to have the manual in reach to look up some details.
As it was pointed out at the beginning of this chapter - there is not '**one**' work-flow. It is within your possibilities to customize the work-flow so that it is optimized for your projects and for your personal way of solving problems.

The next chapter will discuss where to start to build your own surface, how to add your surface controls and all the other techinques you need to build user-friendly surfaces.

Index