Introduction

This manual is still under development, but it is enough to start using the SubModevZ, so i hope You will find it useful.

If You have any question, please contact me to micron@devz.org.

SubModevZ User's Guide

SubModevZ is a modular module that allows to load modular patches inside Modular 2 & 3, ModevZ and inside itself.

Creamware Modular is a great and versatile tool for the creation of synthesizers, effect processors and more, but until now all Modular users were forced to start patches everytime from scratch or by editing an already existing patch, but without having the option of save a patch, load it inside a new modular environment and interface it with other parts of the modular circuit or other saved patches and this is the main reason of the SubModevZ project.

I started some time ago with a module called Submod host, but it was unuseful for the majority of Scope users, because it was not possible to edit it outside the modular environment and it was not possible to save it if loaded inside modular, so just scope dp and scope sdk developers were able to make patches and share them.

After many days of testing i have finally found a solution that is user friendly and that allows every Scope user to make SubModevZ patches and of course i have named the project SubModevZ, because it will be used to launch my new web site dedicated to Scope dsp Systems that is called Devz and that is located@ www.devz.org

At the moment of writing this manual it's available the first example version of SubModevZ that can be interfaced to the modular environment by one midi in, two audio inputs and 4 audio outputs, but there will are several different versions with more connections so that users will are free to choose the more suitable one for any needs.

You could start by loading the example patch "3Osc Synt" translated from the already existing one in the scope modular 2 library.

Installation

SubModevZ can be opened from anywhere it is located on your hard drive, so You can place it in Your favourite folder.

The only file that needs to be installed is MXinSw02.dsp and it must be placed inside your Scope dsp folder, and usually it is something like this: c:/scope/app/dsp.

After the installation of the dsp file, restart scope and all should be fine.

On the next chapter i'll do an overall description of SubModevZ.

Chapter 1

Overall explanation

SubModevZ is composed by 3 interfaces:

The SubModevZ module interface (fig 1)

The SubModevZ modular window (fig 2)

The ModCon modular controller (fig 3)

Fig 1

Fig 2

Fig 3

Before of starting with the creation of SubModevZ patches, it's very important to understand some simple but basilar steps.

First of all, SubModevZ is not able to save patches by using the "save as" method available with modular 2 & 3, but it will overwrite the SubModevZ patch that we have loaded, so before of making a new patch, it's suggested to copy an empty SubModevZ file and create a new one (You are free to rename it, but it's important to keep the .dev extension at the end of the file name).
After this, You can load the SubModevZ device inside the Scope Routing Window as a standard non modular Scope device and this is the way we'll make new patches, because the SubModevZ is not able to save patches when it's loaded inside the modular environment.

Anyway You can change the SubModevZ content when it's loaded inside the Modular environment, but it will be saved as part of the entire modular patch, so it's not useful if You need a SubModevZ patch to be used in new Modular projects.

This is a resume of operations to do for the making of a SubModevZ patch:

1) Copy and rename a SubModevz file and rename it by keeping the .dev extension at the end of the name.
2) Open the file inside the Scope routing window as a standard Scope device and connect it (if You are not able to see the connectors, close and re-open the Scope routing window)
3) Open the SubModevZ modular window (double click on it) and start making your SubModevZ patch
4) Save it by opening the presets menu (use the button inside the modular window) and by clicking on the "save device" icon (a message will prompt You if you want to overwrite the device, click Yes)
5) Now Your patch is ready to be used inside Modular as a standard modular module

Known issues:

SubModevZ connectors are hidden when it's loaded inside the Scope routing window.
Solution: close and re-open the Scope routing window.

SubModevZ connectors are hidden after i change the display name when it's loaded inside the modular environment.
Solution: close and re-open the modular window

Chapter 2

Making connections

SubModevZ introduces a new way to make connections inside the modular window.

The connections can be done as in the standard modular environment or in a new way by using connection modules called "nodes".

The reason of why i have developed this new connection's method is that i never liked to see dozens of cables connected to the same connection point forcing us to use the solo function to hide all other cables making it very hard to solve problems or to reverse engineering a modular circuit.

When You open the SubModevZ modular window You can see on the right side many node points subdivided by colors.

Other than the coloured node points there can be other kinds of node points (it depends on the SubModevZ version) and they can be "outputs", "inputs", "midi in", "midi out" and so on.

All node points are connected to the modular circuit in the back end and this is the reason of why we need to have all of them pre-loaded inside the modular window.

In this explanation i'll describe the node points that are available inside the 3Osc Synth example.

The coloured nodes are good to connect all kind of pads except esync and midi.

The "esync & gates" nodes are good to connect esync, gates and all kinds of pads except audio and midi.

The midi nodes are for midi connections.

The audio nodes are for audio connections.

To take advantage of nodes, we have 8 modules for each color group and 8 node for each of the other groups (audio, midi and so on).

If You take a look at the 3Osc Synth modular window, You will see that there are several modules of the same color connected to different modules.

To make an example, if i connect the lfo out of a module to the connector #1 of the red node, then i can connect the connector#1 of another red node to the cutoff modulation input of a filter and this is the same as connecting the lfo out straight to the filter's cutoff modulation input.

In this specific example it is unuseful to use nodes if we need to connect the lfo out to just one module, but it is very useful if we need to connect the lfo out to several modules and it will make the modular circuit less confusing than as usual.

The "audio out" nodes are optimized in a way that we can connect several signals to the same outputs without the need of a signal merger, so if You want to connect several modules to the audio out #1, You can connect each of the modules to the connector #1 of a dedicated audio out node.

To avoid confusion, You can take advantage of the nodes settings panel, where You can type the details for each one of the connectors, so that if You have forgot what is connected to a specific node, You can open its settings panel to see it.

So, now we know what nodes are and we are ready to patch :)

Chapter 3

Modcon

The ModCon is a modular environment dedicated to the controller surface.

The aim of the ModCon is to have the ability of create a custom control surface to let users play on a more confortable interface, without having to see the modular circuit with connectors in front of the screen and watching on a control surface similar as the non-modular devices interface.

At the moment of writing this manual, the ModCon is at its nearly stage and it works exclusively with midi control change protocol, but in the upcoming version, it will be able to use integer values, so that it will be 100% accurate with the controllers settings.

By using midi cc controller messages, we can work in an acceptable way on all controls that need to be set by a pot value, such as filters, adsr envelopes, sends and so on, but we have some limitations when a specific numeric value is needed such as the milliseconds of a delay.

Setting up the ModCon is very simple and we need to enter the control change numbers for each controller and to plug the midi connectors.

The midi connectors are hidden, so all ModCon control modules have the midi input on the bottom left and the midi out on the bottom right and even if we can't see the standard connection plug, it is the same thing.

The ModCon modules does not use the standard Scope midi cc assignment procedure, but they are equipped with the Cyprox engine, so that to assign a control number, we must open the assignments surface by clicking the button at the top right of the module surface and manually enter the cc number for each of the available controllers.

Each ModCon module label is customizable and for example we can open a 4 pots module and enter the label ADSR to control an ADSR envelope.

Each midi out of ModCon modules must be connected to a dedicated midi out and all available midi out modules are merged in the backend circuit.

If we need the ModCon controllers able to receive midi, we must connect the midi input of each module to the midi input of the ModCon.

The midi input can be shared with all other modules, so there is just one midi input module inside the ModCon environment.

After we assigned control numbers, placed the ModCon modules in the right position and entered the names of the labels, we can choose to hide the ModCon routing (click on the routing button) and resize the ModCon window to keep just the working surface visible and to hide all the rest of the connection modules.

Now we have to assign the cc numbers (by using the standard scope right click) on the equivalent modules inside the SubModevz modular window.

After everything is done, we will be allowed to control our SubModevZ device and to make presets by working on the ModCon surface.

Chapter 4

Opening the SubModevZ inside itself

To make the SubModevZ compatible with the Scope standard routing window, i have had to make some settings on the device that are not usual for a modular module and even if it is loaded without problems inside Modular 2 & 3, there is a snapping issue if it is loaded inside SubModevZ modular environment, so that i have developed a module called "**SubModevZ Loader**" that can be loaded inside SubModevZ without problems and that can be used as a bridge.

To load SubModevZ inside another SubModevZ do as it follows:

1) Load "SubModevZ Loader" inside the SubModevZ modular window
2) Load SubModevZ inside the "SubModevZ Loader" modular window (to open the loader modular window, click on the devz logo)
3) Connect the coloured connectors located on the left of the SubModevZ module to the equivalent ones on the bridge module.
4) Close the Loader modular surface and use it in the same way of the SubModevZ opening the ModCon and ModevZ surfaces by using the buttons on the display.

Known issues:

After loading SubModevz inside the loader, it is necessary to scroll down to see it.

Solution: it is the reason of why i have developed the SubModevZ loader, so after we have connected it to the bridge we can work using the loader module.

What's next

In the next days i'll do a release of a stand alone version of the ModCon controller and of the ModevZ.

Who is interested in the release of custom ModCon surfaces, please let me know to micron@devz.org.

Please visit my new web site www.devz.org for news about devz devices.

Thank You

Salvatore Di Guido