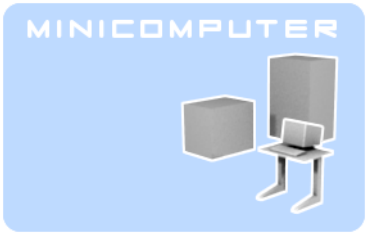
MANUAL rev. 1.0



linux sound synthesizer

by Malte Steiner, block 4

Minicomputer is an open source sound synthesizer for Linux, distributed under the GPL 3 license. It requires jack as an audioserver and Alsa for midi connection. Minicomputer is a standalone application, divided into two programs, the actual sound engine and the editor. Both independent programs communicate via OSC (open sound control) and it is possible to write different editors or control the engine from other applications which can talk OSC.

user part:

(not only) developer part:

You can reach me at steiner@block4.com

starting Minicomputer

Minicomputer is brand new so its unlikely that you received it from repositorys of the Linuxdestribution of your choice. At the moment there is no installation routine there so after compilation you can put the two applications which forms Minicomputer anywhere. Useful is for instance */usr/bin*

To start directly you can type at command line

./minicomputer

and

./miniEditor

to start. Depending on your kernelsettings for audio you might have to start the synthesis core as superuser to access realtime performance and Jack

sudo ./minicomputer

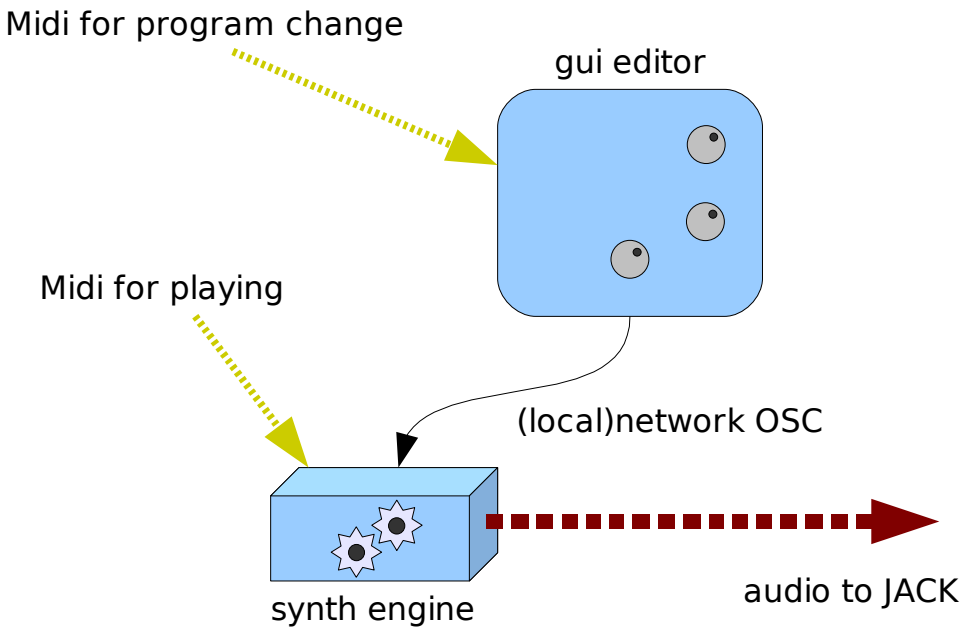
The editor should be started as normal user anyway, because it tries to find your home folder with the environment variable $HOME which is normally set in every Linux distribution. If it finds it, it creates are folder

./miniComputer

in which the setting files **minicomputerMemory.txt** and **minicomputerMulti.txt** are kept. Directly after the first start you might want to copy the memory file with the presets in the **factoryPresets/** folder to **./miniComputer** to get started but don't forget to restart the editor to access them. The presets contain one demo multi and 32 sound presets plus 1 init preset. The ladder is good to start creating sounds from scratch and there is a single init file too which you can import anytime.

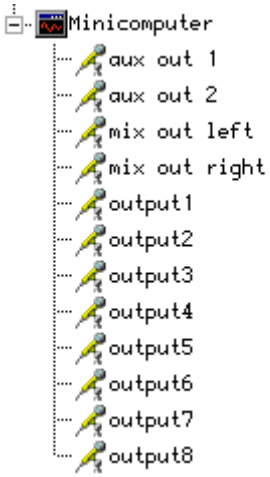
hookup diagram

Well, its software but under Linux with Alsa and Jack its rather similar to studio environments so a diagram makes sense. The editor talks to the sound core via Open Sound Control (OSC) protocol over localhost network. These both applications don't need to be connected manually and the order of launching doesn't matter, they should find each other. They communicate over the port 7770 which can be changed during compilation time in the ***common.h*** file (which is necessary when you want run several instances of Microcomputer, see OSC implementation). Normally they reside on the same machine but it could be possible, with minor changes in the source code, that they run on separate computers, connected via network.
Both provide a midi port. The editor accept Midi program changes on channel 1 – 8 for switching between the first 128 (0 – 127, a limitation of Midi specification) of the 512 sound settings. This limitation is not that bad, because on channel 9 you can switch between the 128 possible multi setups so all 8 voices can be changed at once, for instance at the beginning of a new song.



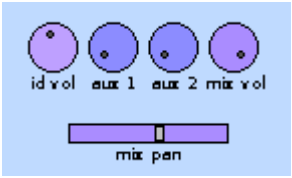
The synth engine recieves note and performance data through it Midi port, channel 1 – 8 for the individual voices. It provides several audio ports for JACK for the output of sound.

There is a stereo mixed output for mixing voices together, aux 1 and aux 2 as a separate two channel output, for instance for effects, and individual outputs for each voice.



audio ports as seen on QJackctrl

Each voice has a knob for individual, aux 1, aux 2 and mix output volume, the mix is augmented with a dedicated panoramaslider while the auxiliary panning need to be done with the two volume knobs if desired. Otherwise they ment to be separate outputs but its up to the user what to do with it.

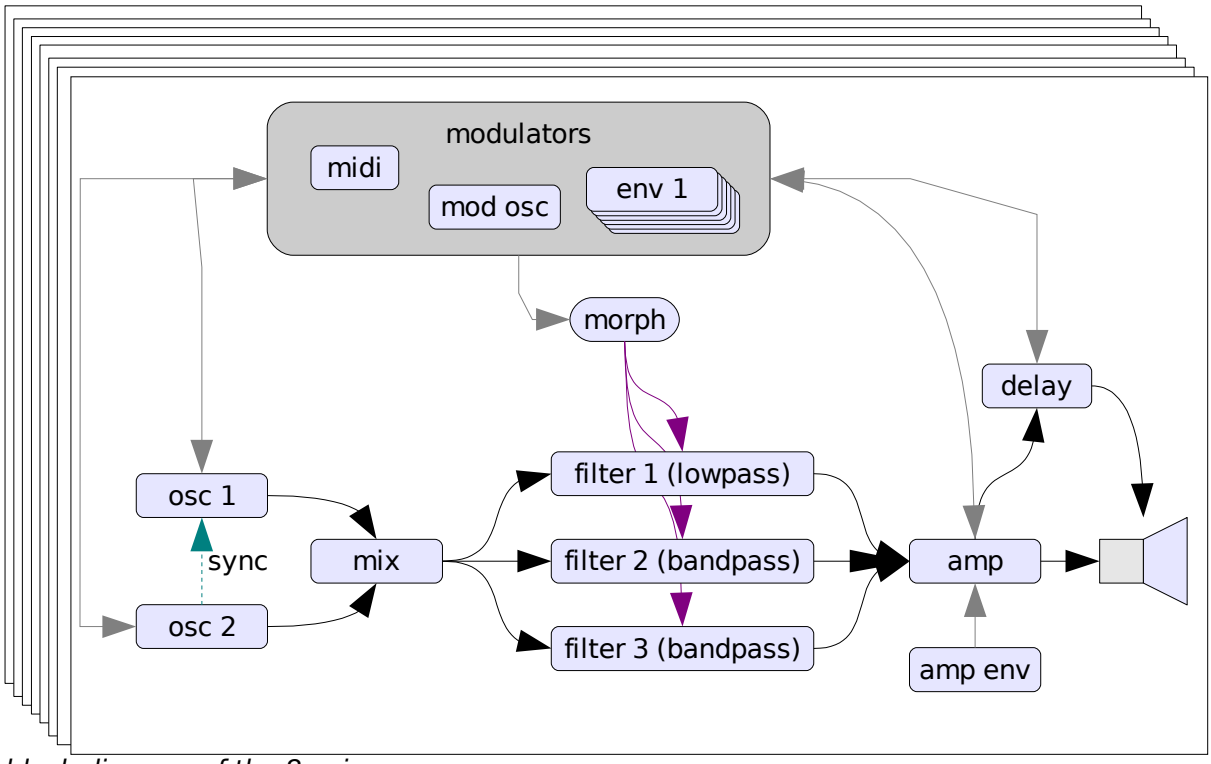


the volume section on the gui editor

The volume settings are stored in the multisettings, not in the sound data.

the sound

Minicomputer doesn't ape any existing synthesizer but has a rather traditional, subtractive approach and thus is easy to operate. Its made of 8 parallel voices forming a multitemperal setup, suited for example as a drumsynth. Each voice is monophonic in itself so it can play only one key per time, additional triggers might cut off a sound of the same voice which was already on or fading.



block diagram of the 8 voices

The synthesis based on the idea to create first a rather complex waveform via two oscillators which then got shaped with an complex morphing filter, actually made of three filters. A lowpass filter (for not losing ground) parallel to two bandpass filters which shapes further formants. The oscillators are not bandlimited which might lead to aliasing but give on the other hand a rich tone, important for the FM and amplitude modulation which can happen in both directions, even in the same time enabling feedback. The second oscillator has slightly different features: it can be hard synced to the first one and the second amplitude modulator only modulates the FM output while the first one shapes the outputvolume.

Additionally there are more modulation options like another oscillator with a range going much into audio, the only difference that its not controlled by keyboard or pitch bending. Six envelopes of the traditional ADSR format are there for a lot of parameter changing, they can be even turn into additional oscillators with the repeat button and reach also in the audio frequency - range when speed up.

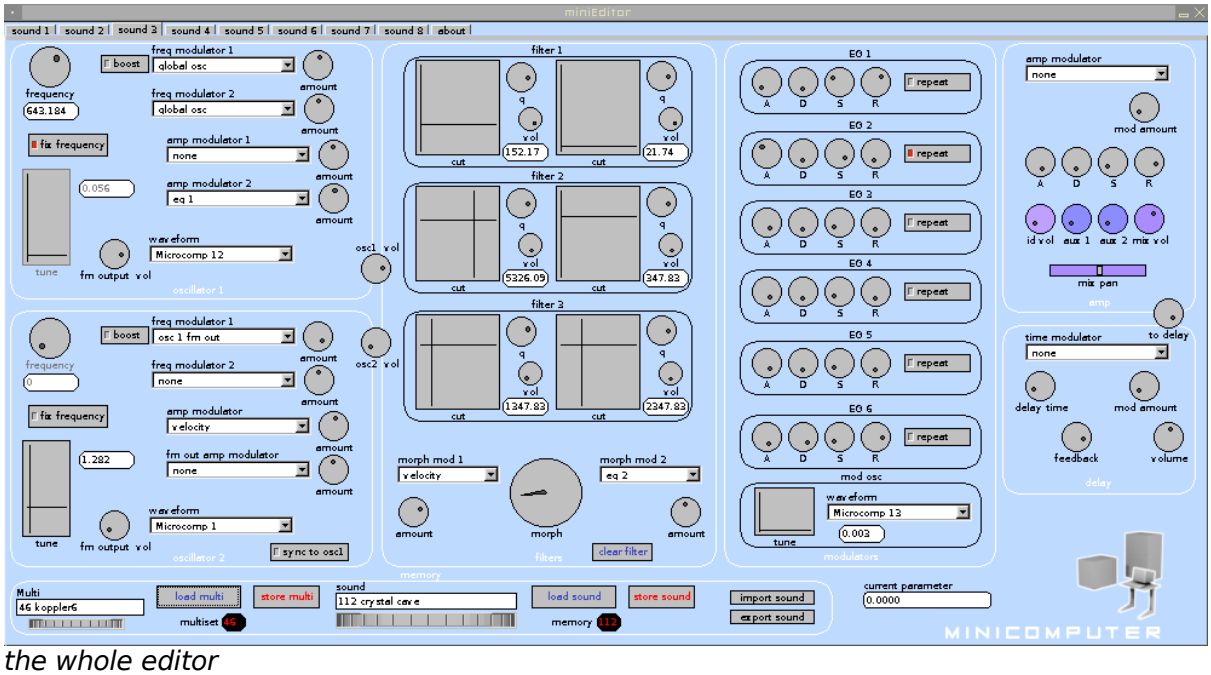
A seventh envelope generator is routed to the amplitude of the output and is not available on other modulation destinations.

The sound choice of the eight mono voices can be stored into the multisetup which is handled seperatly. Its important to save your soundsettings first and than the multisetting to make sure everything can be recalled correctly.

Minicomputer stores the sounds in a file called **minicomputerMemory.txt** which can be opend in a texteditor. Same goes for the multis which are stored in **minicomputerMulti.txt**. When the editor stores something to disk it writes it first to temporary files ending with .temp, then make a backup of the former files to .bak and then finally commit the changes be renaming the temp files to the actual names. This is done for your convenience and savety and you shouldnt have files with same names like mentioned above in the minicomputer directory, otherwise they got overwritten.

the editor

The 8 voices are arranged each on one individual tab and can be reached by mouseclick or keys **F1** – **F8**. The oscillator section features for each the frequency setting which can be set to a fixed value or course/ fine tuned, varied by the pressed Midi key. Tune is done with the X-Y cross-slider, the horizontal is course in octaves while the vertical one is for fine tuning. Additionally for more precise control there are a display field showing the frequency value and accepts input from keyboard and mouse. Click on it and click-drag to left and right for changing the value further. The three possible mousebuttons, left, right and middle click, change the value with a different factor. While all tuning cross-sliders have such a numeric field, all other values can be fine tuned with the **current parameter** field near the logo. It changes the last tuned knob with the exception of these X-Y fields. This seems the best compromise between quick access and total control at the moment although I am thinking about a different approach to user interfaces.



the whole editor

Waveforms can be chosen from a dropdownbox. There are traditional forms like sawtooth and square, special ones for adding crackle noise (bit and spike), waveforms created with additive methods and 8 from Microcomputer, a non free Plugin I programmed and offered some years ago for Windows and Mac OS9.

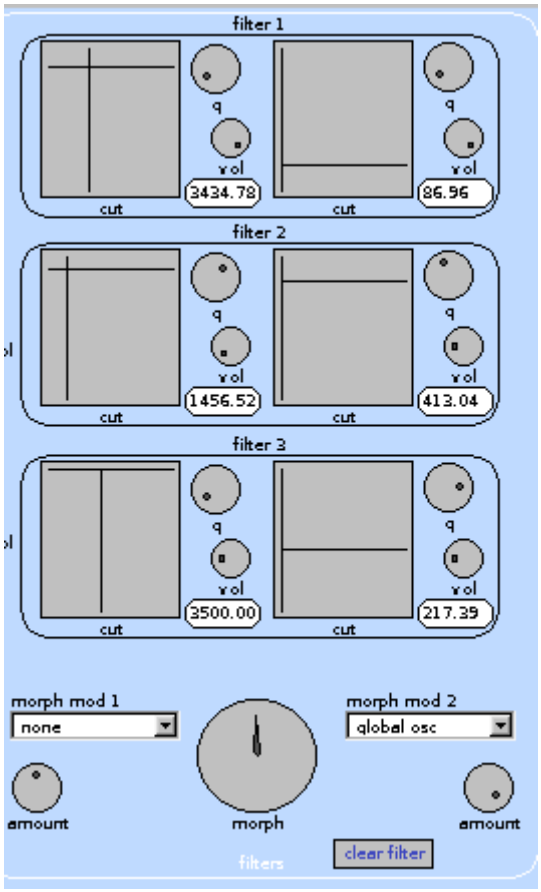
Modulation settings on Minicomputer consist of two gui elements, a dropdownbox for choosing the actual modulator and an amount dial going from -1 to 1. The negative amount changes the polarity of the modulator. There are two slots for frequency modulation with the first one bearing a boost button to create extra harsh metallic sounds by boosting the modulation level 100 times. Two amplitude modulationslots shapes the volume. At the first oscillator they both forms the volume at the output to the mixer and the separate FM output which becomes a modulator while at the second oscillator only the first slot do so, the second forms only the FM output so more complicated blending can be done. Also only at the second one there is a sync button to hardsync it to the first one, so the phase got restarted when the first oscillator resets its phase. The oscillators can also modulate themself, creating interesting chaotic feedback behaviour. In case of some modulation, particular the boosted FM, go out of control, you can reset the internal signal path with the **clear filter** button in the filter section. It doesn't change settings but effect the whole internal signal path for that voice.

list of available modulators

The morphing filter consists of three distinct filters in parallel, each with settings for cutoff frequency, q which means resonance and volume with positive and negative polarity. These settings are there two times and you (and/ or two modulators) can blend between them with the big morph knob for all 3 filters at once. When you use modulators, the morph dial acts more as an offset so experiment with the modulation amounts and morph setting to get the desired effect like filter sweeps or formant morphs.

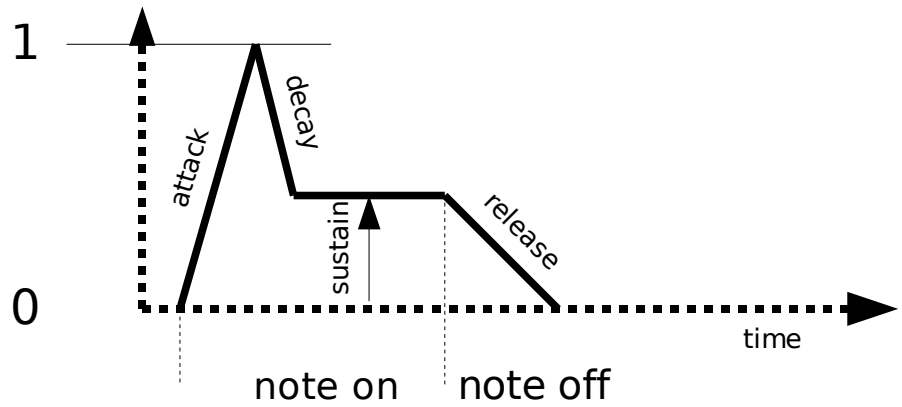
The frequency is here again to be set with the X-Y cross-slider like above mentioned and also features a separate display for fine tuning.

Be careful with the resonance settings because this can lead to harsh distortion. I ripped once a hifispeaker with a Korg MS 20 analog synthesizer in 1986 so I know what can happen. But I decided not to scale down that much the resonant signal because its important for creating formant peaks. Anyway there is some scaling involved to protect a little bit, otherwise the filters would roar, tear and rip much earlier.



the filtersection

Next to the filter there are the modulators with the 6 free routable envelopes and the independent modulation generator. The envelopes can get quite fast and act as modulation oscillators too when switched in the autorepeat mode with the button. Otherwise they are are pretty oldschool with a conservative attack, decay, sustain and release (a.d.s.r.) approach.



the classic: an a.d.s.r. envelope

Further to the right there is the amp section with one modulation slot (for instance modulating the volume by the velocity of the pressed key) and a dedicated envelope generator.

Also on the right there is a delay for each individual voice, whose settings are stored along the sound data. The delay time can be modulated which creates some interesting effect inclusive flanging and feedbacked pitch shifting. The output of the delay is also available as an modulator. I tried it also as a feedback audio source but the results were not that interesting.

On the footer there is the memory section which allows the storage and recall of sound- and multisettings. Multi stores only the sound choices of each voice and the volume section whose knobs have a different colorcode.
You can choose a sound or multiprogram with the large horizontal wheels underneath the name fields which are handy with the large list of settings. After choosing a setting you have to press the load buttons for actually loading. The store button write the current setting to that storage location and you can give it a new name in the field *before* pressing that button.

The digits show the memory location and are helpful for finding out the numbers for a remote program change.

To exchange sounds with others you can import or export individual sounds to separate textfiles, whose names you can choose freely. The in/export goes in and from memory directly, you can only export a saved sound or import to a certain memory location. The memory location didn't need to be loaded, it sufficient to have it dialed. After importing you need to load that sound with the load button to actually play it.

compiling Minicomputer

Dependencies include:

- Jack
- FLTK guitoolbox for editor
- Alsaseq for Midi communication
- pthreads
- liblo for OSC communication

and

- Scons for building

The engine is written in C and have their sourcecode in the folder **cpu/** and the editor in C++ with the source files in **editor/**. Parameters for customization are in the **SConstruct** and **commons.h** file.

For building Minicomputer just type

scons

for building it. Yet missed is an installation routine so put the to files anywhere you need them.
Cleaning the buildfiles but the new created programfiles TOO, is done by typing

scons -c

The buildprocess accept parameters for refining architecture optimization:

- 64bit=1

forces a build for 64bit platforms with the -m64 option

- k7=1

optimizes for AMD Athlon-XP processors

- k8=1

optimizes for AMD K8 cpus, giving the compilers the -march=k8 -mtune=k8 settings

So a k8 64bit build looks like

scons 64bit=1 k8=1

The common.h file have more customizable parameters. You can switch on debugging messages by uncommenting the line
//#define _DEBUG

I tried a vectorized version for improving performance but it doesnt improve it at least on my hardware which is currently AMD cpus only. For experimentation you can uncomment
//#define _VECTOR

There are another switches like the 'old' binary file format but that use is discouraged now because it certainly will lead to wrong behaviour.

TROUBLESHOOTING:

- 'I have library xyz installed but scons complains about not finding xyz'

make sure that you have the developer version of the library installed which comes extra and additionally. Usually looks like libxyz.dev
Its my first project with scons and I experienced following bug: when I give wrong parameters for gcc, scons complains about not finding a library although its there. This could be a reason too.

OSC implementation

The core accepts messages from the editor in the OSC protokol. The port is 7770 as default which can be changed in the common.h file for building alternative versions. One port accept only editor so ***its not possible to run several instances*** of Minicomputer without building alternative versions.

Following messages are sent via OSC:

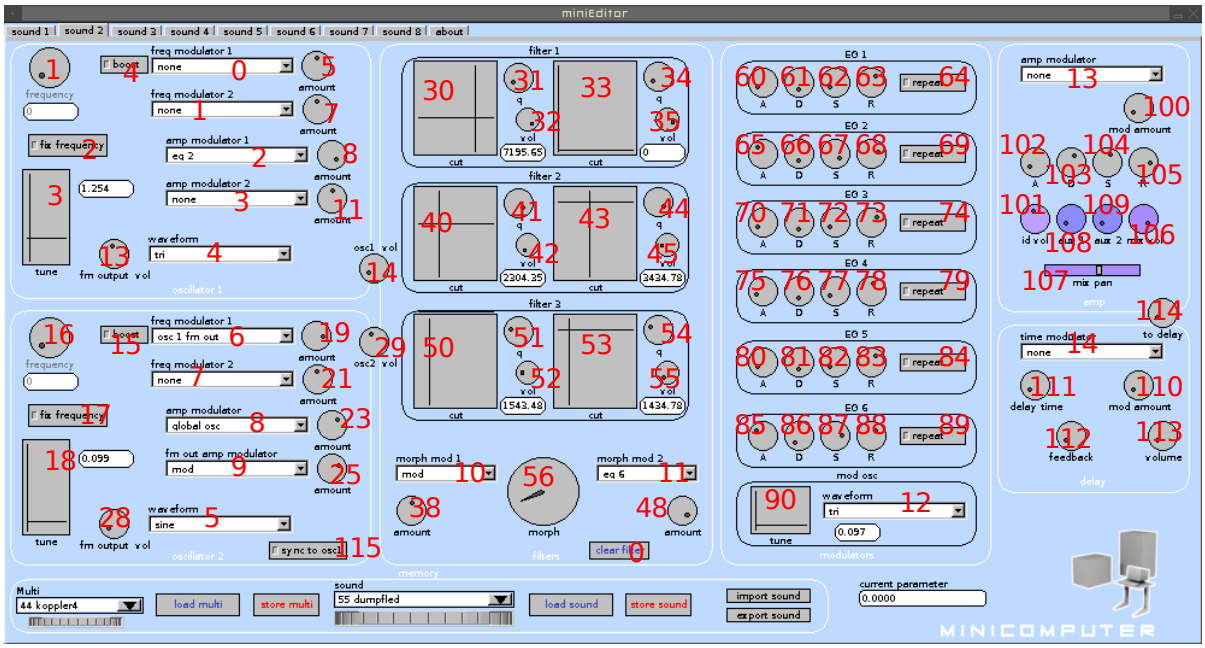
For choices the message is

/Minicomputer/choice followed by three integers, the first one denotes the voice number, the second the number of choice parameter and the last one the actual value.

Other parameters are sent as

/Minicomputer and two integers and one float. First integer says which voice, the second the parameter index and the last one the actual value as float.

The following diagram is overlaid with the OSC parameter numbers:



the picture shows an older version of the gui but the numbers and knobs are the same

Please bear in mind that the drop down boxes have their own numbers which are transferd as **choice** as stated above.