CLAVIA NORD MODULAR G2 PATCH CONVERTER PROJECT

Gleb G. Rogozinsky[1, 2], Michael Chesnokov[1]
gleb.rogozinsky@gmail.com
chesnokov.inc@gmail.com
[1]The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, St. Petersburg, Russia
[2]Saint-Petersburg State Institute of Film and Television, St. Petersburg, Russia

One of most remarkable hardware synthesizers of the late 90s, the Clavia Nord Modular G2 was discontinued in 2009. It inspired a whole new generation of modular synthesizer fans by combining the immediacy of dedicated hardware with the power and flexibility of computer-based programming. The suite of components comprising the G2 project included an extensive list of modules (from oscillators to effects), an attractive software graphical interface and on the hardware side, keyboard and rack options. The website Vintage Synth Explorer rates the G2 as “awesome” and it has been used extensively by artists such as *Astral Projection*, *Autechre*, *The Chemical Brothers*, *Somatic Responses*, *Junkie XL*, *Mouse on Mars*, *Nine Inch Nails* and *Covenant* amongst others.

What really makes the G2 unique and sustains its relevance is the rich archive of patches that exists. The NMG2 community remains active, publishing new patches ranging from sound synthesizers to algorithmic generators.

It is difficult to compare this synth to commercial synths like the Access Virus or Roland JP8k, but its remarkable features are algorithmic patches and drones. Alongside their inherent use to musicians, NMG2 patches can also incorporate a valuable educational aspect, inspiring people to develop their own patching skills.

Where the challenge starts...

Like Csound, the Clavia system has two sampling rates for audio and control signals. Unlike Csound however, those rates are fixed. The audio rate is fixed at 96 kHz and the control rate is fixed at 24 kHz. In the software patching environment there are different types of virtual cables to reflect these different signal types: red for audio, blue for control signals, yellow for impulses at k-rate and orange for impulses at a-rate. Instead of complex waves with an analog behavior, Clavia uses near ideal saw and pulse waveforms (similar to what could be generated using GEN07). It actually uses waves that are band-limited for 96 kHz so if the synth is run at 44.1 kHz, aliasing becomes a problem. This fact provides a challenge for accurate modelling of most of its modules. The adders and logic modules have a mathematically straightforward design, as do envelope generators. There is also nothing unusual about most of Nord’s filters and waveshapers.

Csound is a unique language; like Noah's ark, it carries the sounds (and patches) from the past, which were originally compiled on old mainframes, but which have been continually re-compilable throughout the intervening years, even as hardware and the program itself evolved. Csound has the potential to provide a solution for (re)implementing the sound engine of the Clavia NM G2, facilitating it to escape from its closed source software and providing it with a continuing existence.

We began this project at the beginning of the summer 2015 with the decoding of the *pch2* patch format. The first Clavia Nord Modular had an open text-based patch format, but the NM2 uses coded binary data after a short text header. Fortunately the format was more or less correctly parsed by Michael Dewberry [1].

There are several issues to be noted regarding the patch format decoding. Firstly the data fields are not byte aligned. This produces additional difficulties while writing the parser. Secondly and unlike Csound, it is possible in some cases to connect an output to another output, as shown in Figure 1. In this patch, the output of the noise generator, *Noise1,* is connected by a red cable (an audio type) to the input of *2-Out1* (which is the equivalent to the *outs* opcode in Csound), but instead of patching the output of the generator to another *2-Out1* input, the outputs *L* and *R* of the latter module are interconnected. It can also help if we examine the patch as a schematic.

Thirdly Clavia uses two separate patches: one for voice (VA) and another for global effects (FX). You can use FX space to build the synth, just like in the VA part. The real purpose of such a division becomes clear once we start playing several notes simultaneously.

Finally the Clavia Nord Modular system supports MIDI; this means that every parameter is stored as a 7-bit integer; although it is seen as a float value of the corresponding parameter (frequency, volume, effect level etc.) in the editor program. This means that a special mapping table should be constructed for any value type to convert the MIDI number into the expected and usable value.

The issues described here made our work far more complex than initially imagined.

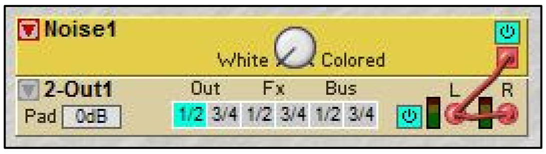

Figure 1 An example of Clavia’s output-to-output patching

Our approach to convert the *pch2* format into *csd* was as follows. First we built a library of user-defined opcodes, each of which accurately simulated the behavior of a corresponding Clavia module.

Figure 2 shows one of Clavia’s basic oscillators, *OscD*. It has one input for pitch modulation (this defaults to k-rate), an on/off toggle switch for keyboard tracking (KBT), rotary knobs for coarse and fine tuning of frequency, a drop-down selector for wave-type, an on/off toggle switch for the entire module and an audio output.

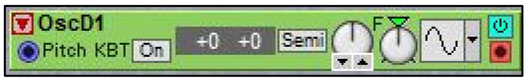

Figure 2 *OscD* generator

The corresponding Csound code for this module is shown below.

```
opcode OscD, 0, iKKKKii
     ifn, kFreq, kFine, kDum1, kDum2, iMod, iOut xin
     kPitch zkr iMod ; CHANGE
     kfine = cent(kFine)
     aout oscili 0.5, kFreq*kfine+kPitch, 1
     zaw aout, iOut  ; CHANGE
endop
```

As we can see, the designed UDO has no outputs (in common with all of the UDOs used in our project). It has "iKKKKii" as its sequence of input types the purpose of which are clarified below:

ifn –	waveform type (pointer to Csound ftable)
kFreq –	frequency
kFine –	fine tuning
kDum1 –	not used
kDum2 –	not used
iMod –	zk channel for modulation input
iOut –	za channel for generator output

Wave-type could also be a topic of discussion since we can easily change it during performance; although in the case of the reviewed modules it will cause the patch to be rebuilt resulting in a click and a moment of audio interruption. For this reason we only treat it as an initialization time value. Some other Clavia oscillators can change their wave types seamlessly so obviously these will need to be k-time values.

We also defined all of the function tables used by Clavia's oscillators and waveshapers. For simplification we assume that the oscillators are not bandlimited and can exhibit aliasing. In the case of a 96 kHz rate, we have to use the vco2 opcode in order to provide an alias-free behavior. Secondly we created a number of mappings to convert MIDI values of different Clavia parameters to corresponding values for frequency, amplitude etc.

To connect the modules in a modular fashion we decided to use *zak* space. Zak provides a separate field for *a*- and *k*-rate communication. Clavia uses common numbering for all cables, so another issue was related to switching a connection between different types: *a* and *k*.

Further developments

Currently our patch converter is able to correctly parse the patch format and translate it into Csound *csd* format. There are a number of issues related to different module groups. Here is a list of our current status:

Group	Description	Status	Next TO-DOs
In/Out	Contains audio IO modules and some MIDI related modules	All outs (stereo, quad) are implemented	MIDI related modules
Osc (Oscillators)	Contains oscillators (basic waveforms, phase mod, shaping oscil, simple physical model), noise generators, DrumSynth and DX7 model.	only OscD and Noise are partly implemented	Modules with waveform morphing
Rnd (Random Generators)	Contains random generators and triggers	Probably the hardest section to be implemented. The random modules are using various random generators.	-
Filter	Contains filters of several types (from basic ones to equalizers, comb filter and wah-wah effect)	only basic ones (like LP and HP, which are very close to ordinary Csound tone/atone opcodes) are implemented	Find the closest opcode to *Nord* and *Classic* filters
Delay	Contains several delays (from single to multitap)	Completed	-
Level	Contains a lot modules related to value modulation, comparison etc.	Almost everything is implemented	*NoiseGate*
Switch	Contains a number of different switches	Completed	-
Seq	Contains several sequencers	Nothing is implemented yet	*Sequencer Event* is the easiest
Note	Contains note quantizers, gliders, zero crossing counter etc.	Nothing is implemented yet	-
LFO	Contains several LFOs and also a clock generator	several basic LFO generators are implemented	*Clock Generator* should be implemented ASAP. All sequencers depend on it.
Env	Contains several types of envelopes (H, D, ADSR, AHD etc.)	several basic envelope generators are implemented	-
FX	Contains typical set of digital effects (chorus, flanger, phaser, reverb, digitizer, pitch shifter, scratcher and compressor).	Nothing is implemented. This is probably the most controversial part. The quality of effects is poor compared to state-of-the-art plug-ins.	*Chorus* and *Flanger* are probably the easiest to model.
Shaper	Contains typical shaping modules	Only *Rectifier, Shape Static* and *Clip* are implemented.	*Overdrive, Saturate*

Mixers	Contains a number of mixers	Completed	-
Logic	Contains logic modules (from AND, OR, XOR etc to binary counter and ADC/DAC)	Completed	-
MIDI	Typical MIDI section (note on, program change etc).	Nothing is implemented yet	-

As this table shows, there are some groups which are completely finished, whereas others are still under development. Random number generators are proving to be a considerable challenge to model accurately as all of them generate new random values with some bearing on previous values. The effects section may prove difficult to model as each reverb or chorus has its own particular features. The other issue here is that the quality of the Nord reverbs is not great; certainly when compared to today's state-of-the-art *reverbsc* opcode. It may be worth considering breaking authenticity by offering improved effects units.

Conclusion

Finally we would like to provide an example of patch conversion. The original NM2 patch for this example is shown in Figure 3 and the corresponding code that was generated automatically by our converter, is listed below.

```
<CsoundSynthesizer>
<CsOptions>
</CsOptions>
<CsInstruments>

sr = 96000
kr = 24000
nchnls = 2
0dbfs = 1

;******************************
; Initialize the ZAK space
zakinit 3, 2

;******************************
; Opcode Definitions
opcode Out2, 0, iiiii
     iOn, iPad, iRoute, iLCh, iRCh  xin ; iRoute ignored
     aL zar iLCh
     aR zar iRCh
     outs aL*iPad*iOn, aR*iPad*iOn
endop


opcode OscD, 0, iKKKKii
     ifn, kFreq, kFine, kDum1, kDum2, iMod, iOut xin
     kPitch zkr iMod ;
     kfine = cent(kFine)
     aout oscili 0.5, kFreq*kfine+kPitch, ifn
     zaw aout, iOut  ;
```

```
endop


instr 1; VA Space
      Out2 1, 1, 0, 2, 2
      OscD 1, 880.00, 0, 0, 0, 0, 2
endin

instr 2; FX Space
      OscD 1, 440.00, 0, 0, 0, 0, 3
      Out2 1, 2, 0, 3, 0
endin

;*******************************
</CsInstruments>
<CsScore>

f1 0 16384 10 1
i1 0 [60*60*24*7]
i2 0 [60*60*24*7]

</CsScore>
</CsoundSynthesizer>
```

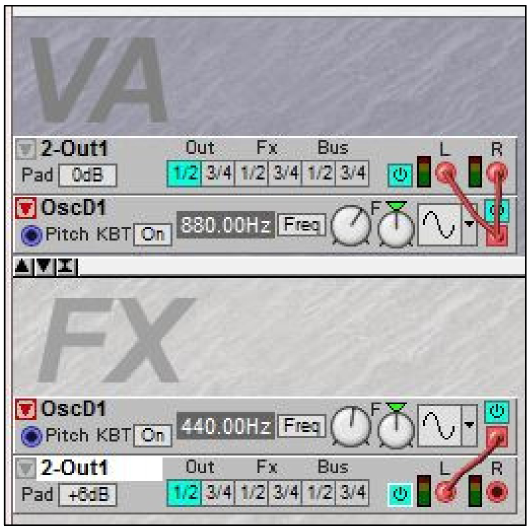

Figure 3 Clavia test patch

As can be seen, both VA (*instr 1*) and FX (*instr 2*) spaces are used. In our output csd there will always only be two instruments. Both of the spaces contain an OscD - 2-Out pair. Notice that for illustrative purposes in the VA space we first created the 2-Out unit and then created the OscD. When usinging zak space, the patching order of UDOs in our instruments becomes irrelevant; for example it can be seen that in instrument 1 that the Out2 is above the OscD.

We have already started the github page for the project. It is located at https://github.com/gleb812/pch2csd. We hope that some Clavia fans will join the project and assist us in developing our models. ICSC2015 showed us that there is interest in our project from our fellow Csounders which we found highly inspiring.

If you are a Nord Modular fan, this software will allow you to resurrect your beloved device in the world of Csound. You also can improve the precision of models and use the whole gamut of Csound possibilities to augment what was possible in the Clavia. If you are already a Csound person this could be a new avenue for exploration. Obviously there is great potential in having access to an existing library of digital synths available on Csound. Once the conversion project is done, you will be able to use hundreds of patches immediately on Csound. If you are interested in the world of modular synthesis and algorithmic composition, this system provides a great way to describe the graphical patches of Clavia. If you are a developer of an alternative Clavia Nord Modular G2 Editor, you could merge your graphical editor software with our system as the sound engine.

We are ready to continue our project and we are also keen to hear responses from other people.

References

[1] Michael Dewberry Page on Clavia http://www.dewb.org/g2/pch2format.html

[2] Clavia NM2 manual and Demo Editor http://www.nordkeyboards.com/downloads/legacy/nord-modular-g2

[3] Clavia NM2 official page http://www.nordkeyboards.com/products/nord-modular-g2

[4] Clavia NM2 page at SoundOnSound http://www.soundonsound.com/sos/jul04/articles/nord.htm