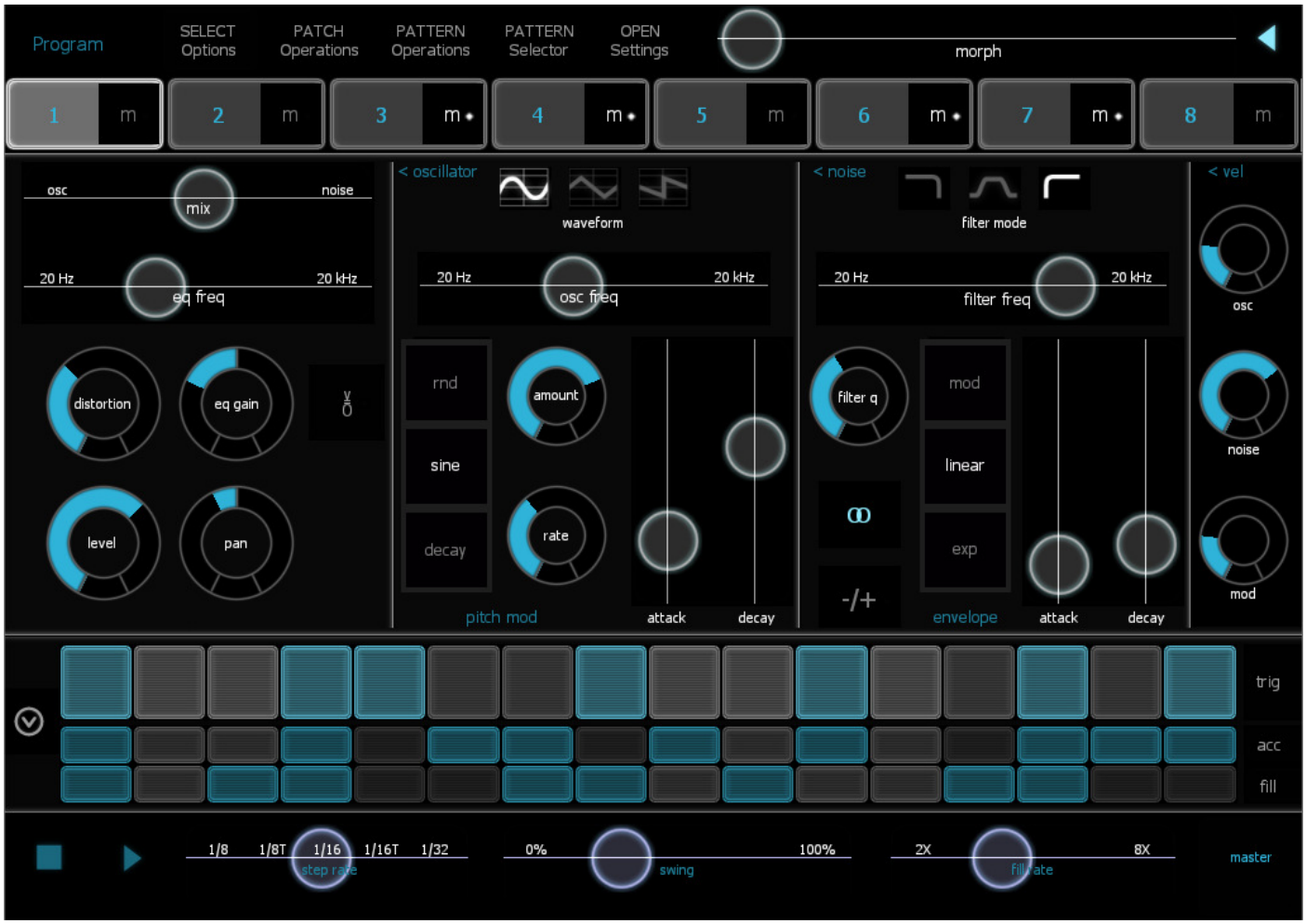
mTonicLemurized v2 - User Guide

by Softcore – www.softcore.net.gr



CONTENTS

Contents 2
Intro 3
What's Included in the Download Content 3
Requirements 3
Setting up 4
Using MTonicLemurized 8
Select Options 8
Patch Operations 9
Pattern Operations 11
Internal Pattern Presets Bank 12
Fine Adjustment (-/+) Mode 13
Quick Pattern Operations 14
Pattern Selector 14
Settings Page 15
The Morph slider and the Tilt-To-Morph feature 16
Technical Details - Limitations 17
The "Running Light" – Midi Clock Sync details 19
Acknowledgements 20
Version History 21
Final words 22

INTRO

mTonicLemurized is a Lemur designed control surface for full hands on control of mTonic (µTonic, Microtonic) plug in by Sonic Charge. It specifically takes advantage of the native midi mapping system of the plug in combined with the scripting – coding power of Lemur thus, providing functions similar to the plug-in options (initializing sounds and patterns, randomizing sounds and patterns, etc, etc) otherwise not possible with other midi controllers.

*mTonicLemurized does **not** produce any sounds by itself nor is it a standalone iOS application. It is a template which can only be used by users of Liine Lemur (http://liine.net) and owners of mTonic, a VST/AU plug-in*
By Sonic Charge (http://soniccharge.com)

WHAT'S INCLUDED IN THE DOWNLOAD CONTENT

- 3 folders named "target0", "target1", target2" containing Lemur templates, each one for a specific Lemur target.
- Inside each target folder, the typical mTonicLemurized template, working in most host applications and a specially designed *"AbletonLive-compatible"* template. This template is to be used, when mTonic is used as a plug-in with Ableton Live as a host.
- Inside each "pixelskin" folder, both typical and *"AbletonLive-compatible"* templates in pixel skin for the specified target.

If you are using a different host and you notice the mute buttons, step sequencer buttons and some Drum Patch operations not correctly updating mTonic in the host, try the AbletonLive template. (Also see "Technical Details – Limitations" section of this manual).

REQUIREMENTS

In order to use mTonicLemurized the following are required:

- iPad running Liine's Lemur app
- A Mac or PC running a DAW application or any other Midi compatible software which can operate as a VST/AU plug in host
- mTonic plug in version 3.1 or higher, by Sonic Charge
- For Midi communication: Lemur Daemon installed on Mac - PC. On PC – Windows systems additional virtual midi cable drivers-software is required.

In Mac OS, the Daemon utility creates additional virtual midi ports in the system, which then need to be used in you preferred software – no third-party software is required.

In Windows you have to install a third-party software for such midi ports to be available – probably you already have one if you have been using Lemur with midi functionality in a Windows system.
I suggest the free LoopMidi (http://www.tobias-erichsen.de/software/loopmidi.html) by Tobias Erichsen or the paid LoopBE 30 (http://www.nerds.de/en/loopbe30.html)

SETTING UP

Setting up mTonicLemurized to communicate with your mTonic requires, initially, a connection between Lemur and your host application via procedures fairly familiar to you if you are a long-time Lemur user. Once this connection is established, you then need to connect Lemur's target0 (or target1 or target2 according to the template used) to the midi track that contains the mTonic plug in. Notice here, that you do not need to enable this target as a "control surface" as all the midi mappings are internally handled by the plug in itself (using the provided MidiConfig file).

Once connection between your host application and Lemur is established, load mTonic in your host, and set it so that it receives midi messages from Lemur and on midi channel 14. Depending on the host application, the procedures may vary so consult its user manual if you are not sure. To make bi-directional communication possible, and have mTonicLemurized provide visual feedback of the controls you also need to set up mTonic in your host software to send midi messages back into Lemur. In most of the cases, this is can be achieved with a secondary midi track, set to receive midi from mTonic plug in itself and then send the messages to the midi out port connected to target0 – again, on midi channel 14.

The reason, mTonicLemurized is configured to send and receive midi messages on channel 14 was to avoid possible conflicts with already-made midi mappings of your software. Whatever the case, we remind you again, that mTonicLemurized doesn't even have to be configured as a "midi controller" but merely as an "input" and "output" midi port for your "tracks" in your host application.

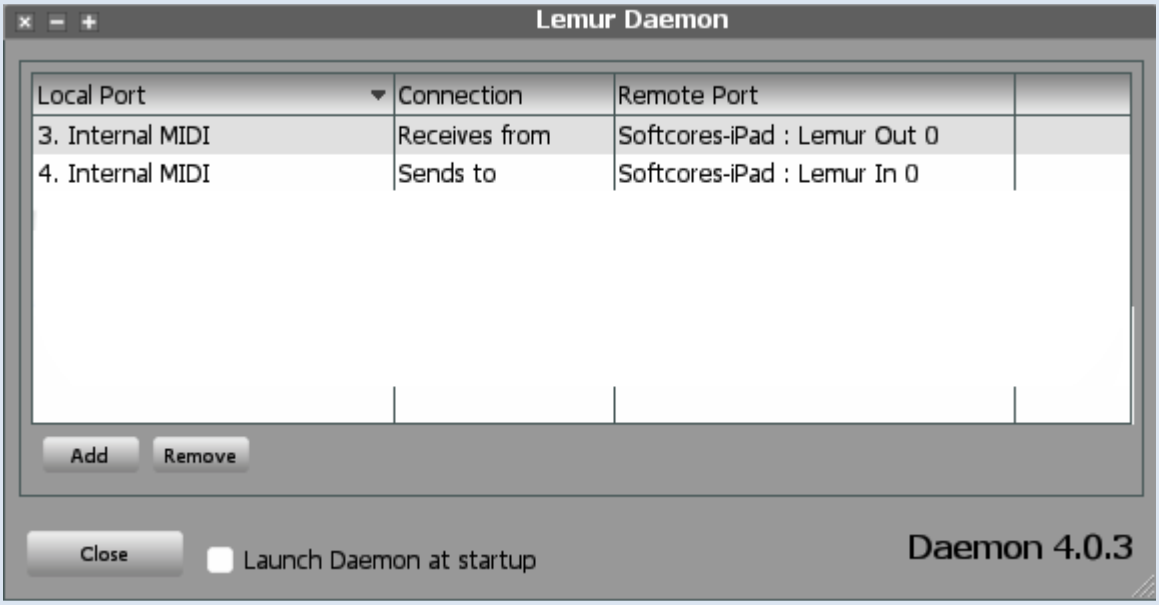
Also, starting from version 2, mTonicLemurized is capable of providing a visual feedback of the step sequencer's running light. In order to enable this feature, your host application needs to be configured in such a way so that it transmits "midi clock" messages to the mTonicLemurized daemon target. A limitation of this feature is that the running light is active regardless of the currently playing mTonic pattern *(Also read "Technical Details – Limitations" section).*

If you want to use a different midi target for the "midi clock" check out the Technical Details section of this manual for the necessary edits to the template file of your choice.

Let's see a specific example with one of the most popular DAWs, Ableton Live so that the above settings can be seen more clearly, while using Lemur's target0.
Specifically for the case of Ableton Live, you need to load the mTonicLemurized-AbletonLive template in Lemur, provided in the download content.

So, once you have loaded the template, and you have made sure that Lemur is indeed connected to Live, head over to Live's preferences – Midi Sync options. Here make sure, to enable the midi ports connected to Lemur target0 as track inputs and outputs – once again, notice that it is not required to enable the "remote" option but rather, the "track" option. If you also want to enable the sequencer "running light" then also activate the "Sync" option for the output port.

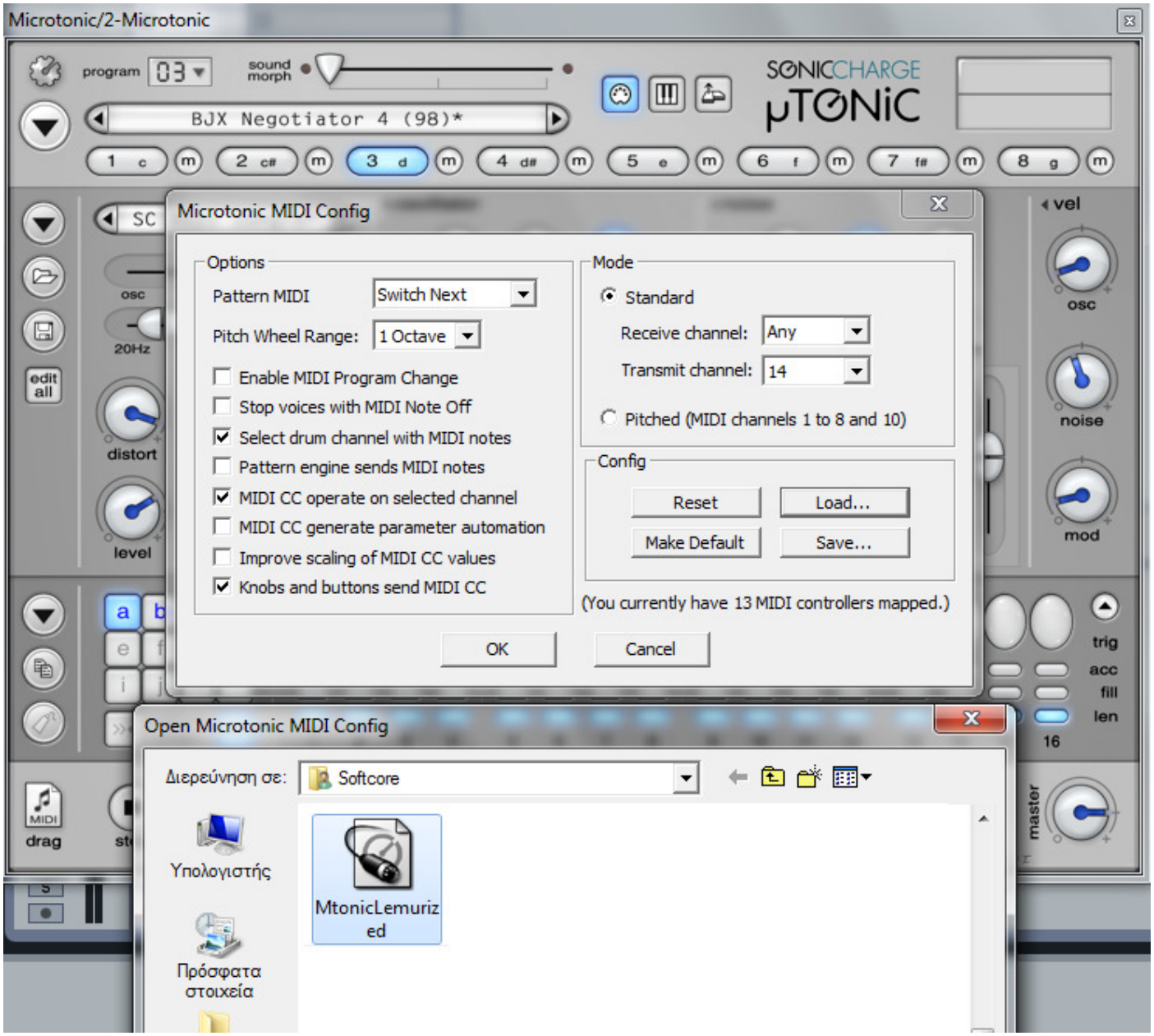
An example set up in Ableton Live in Windows OS





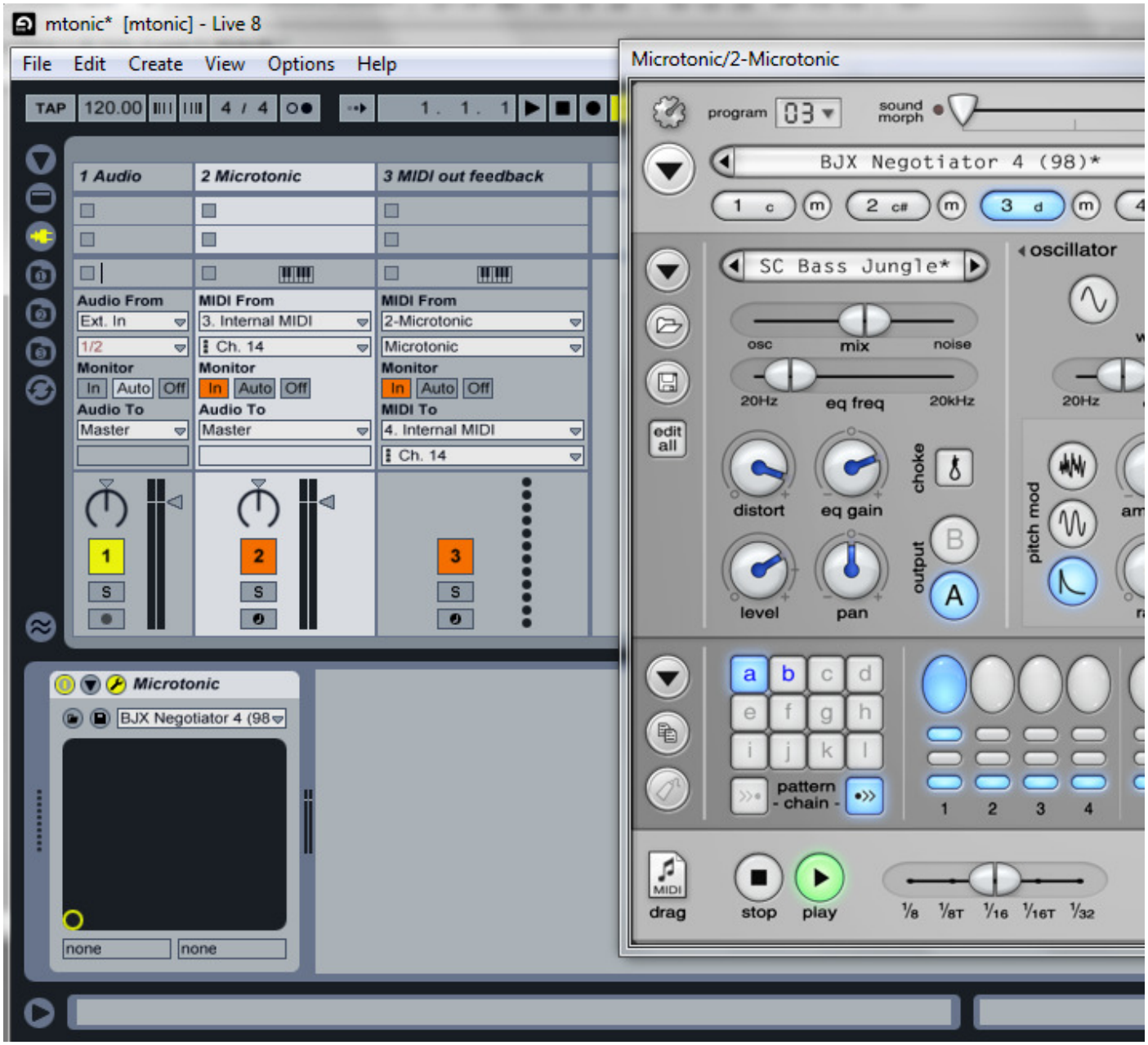
Keep in mind that the "Sync" option in the above image (midi clock messages transmitted) is only needed for the "running light" of the mTonicLemurized sequencer – in other words it doesn't affect the actual timing of either your Live application nor mTonic's timing, this is strictly for visual representation of the running sequencer. If your own host application is not capable of sending midi clock messages, mTonicLemurized will still be functional – you just miss out the "light show".

The next step is to load the Midi Config file provided with mTonicLemurized which will load all the required midi mappings and configurations to mTonic. You can also press “make default” so that mTonic will always be loaded with these settings to avoid repeating this process.



If the MidiConfig file has been successfully loaded, you will notice that the settings of mTonic are now similar to the ones displayed in the image above. Specifically make sure that *“Select drum channel with midi notes”* is enabled and that *“Knobs and buttons send MIDI CC”* is enabled. Also notice, that we have set mTonic to receive midi messages on “any” channel, as the channel 14 will be set inside Live as an input source, and that the transmit channel is set to 14. These settings should work with most host applications and not only Ableton Live.

Finally, let's see our track configuration so that we have a complete and working set up.



Notice in the above image, that mTonic plug in "resides" in track no2 – which is set to receive midi messages from our Lemur connected port and on channel 14. Also notice on our "3 MIDI out feedback" midi track, that we have set it to receive midi messages directly from mTonic track – mTonic plug in and that it is set to send the received messages back to our Lemur connected out port, on channel 14.

If all the steps are successfully done, a working bi-directional communication between mTonicLemurized and mTonic has been established. If you are not sure – try the morph fader. Not only mTonic will perform the morphing but also the controls on the iPad must be moving along. IF mTonic does not perform the morphing, we have incorrect input setting, if it performs it, but the controls on the iPad are not updated, then the output (feedback) is set incorrectly.

USING MTONICLEMURIZED

To quickly start up and "play" with your touch surface, tap the "program" button located on the top left corner and load a preset. To edit one of the 8 drum channels tap on the corresponding button located on the top part of the screen. You can also apply "swiping" gestures to the "mute" buttons – tap on one mute button and swiping over the rest will help you mute or unmute all channels at once.

Detailed usage of all the controls present on the mTonicLemurized is no different from using the actual plug in and is not in the scope of this user guide. What each of the controls provided does, is fully covered in the mTonic plug in user guide.

In this guide, however, we will cover the basics of more advanced controls which replicate functions of the plug in, which are not normally accessible via midi – and some useful additions which overcome some limitations of the midi communication implemented in mTonic. These advanced functions are mainly "Select Options", "PATCH Operations" and "PATTERN Operations".

SELECT OPTIONS



By default, every time you tap on a drum channel selector, the surface is updated to show the parameters of the selected drum channel – patch. At the same time, a midi Note on message is sent to mTonic (in order for the channel to be selected) thus, the sound is also triggered. Tapping on the "Select Options" button, a pop-up "toolbar" presents the following options:

Selection Trigger Velocity: this setting serves two purposes. The first, and obvious one, is to be able to adjust the velocity by which the drum channel is triggered upon selection thus being able to "monitor" the effect that the Velocity Sensitivity Section of the plug in has to the sound (the three far right knobs under the "< velocity" label). As an outcome though, if a sound is set to be full sensitive to velocity you can set the triggering velocity to 1, thus be able to select a drum channel without audibly triggering it. By default this setting is 64 – this means that when a sound is selected, it is triggered with a velocity value of 64.

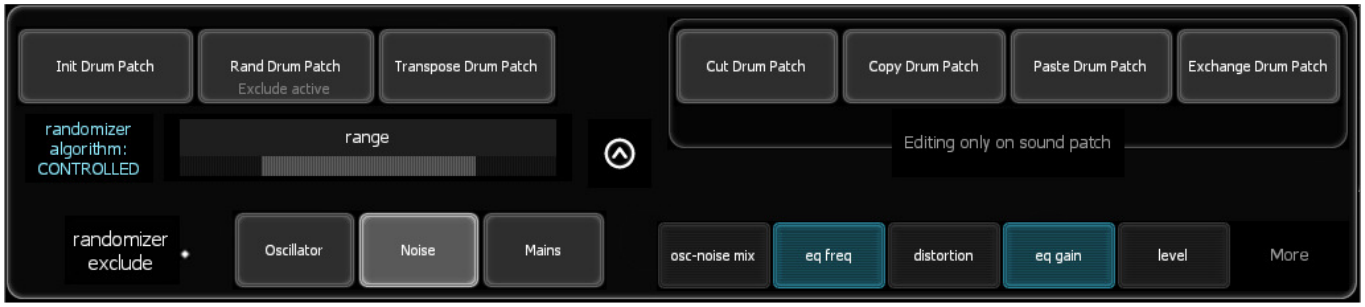
Mute on Selection on: there are times, where a drum patch – sound has both its "osc" and "noise" velocity sensitivity knobs set to 0 – this means that whatever velocity the patch is triggered with, it will still sound and in fact, this is the loudest it can get. If you still want to select a sound but not audibly trigger it you can enable this option.

With this option enabled, when a drum channel which is not muted, is selected – it is momentarily muted upon selection and then quickly unmuted. This prevents the selection of the drum channel to cause audible triggering of the sound. This is merely a "hack" in order to circumvent the fact that there is, currently, no way to select a drum patch without actually triggering it. *(Also read "Technical Details – Limitations" section)*. Notice that when this feature is active, when selecting a drum channel, its "mute" button lights up momentarily but without the LED. This provides a fast "reminder" that the feature is active.

To close the "Select Options" toolbar, just tap on the "Select Options" button again. By default, the toolbar will be automatically hidden after 10 seconds – you can change this behavior in the settings page.

PATCH OPERATIONS

Tapping on the "PATCH Operations" button opens up a pop-up window (covering the steps section) with various functions, always applicable to the selected drum sound – channel. A "minimize – maximize" button reveals even more options for the randomizer algorithms.



Init Drum Patch: Initializes the selected drum sound in the same way mTonic's native function does.

Rand Drum Patch: randomizes the selected sound based on two different configurations – algorithms. Using the **"free randomizer algorithm"**, all the controls of the drum patch are completely randomized – this yields more unpredicted results and sometimes not really useable due to extreme level, envelope and frequency settings.

You can also switch to **"controlled randomizer"** algorithm. With this setting the following happens:

- The drum level is randomized in a range of 40% to 80%
- The pan is randomized in a range of 30% to 70%
- The Noise filter frequency determines the probability of the Noise filter mode. A randomly generated filter frequency which is high will most likely cause a Low Pass filter mode while a lower filter frequency will most likely generate a Band Pass or High Pass filter mode.
- The rest of controls are randomized in a user specified range. You can set the lower and higher limits of this range using the "Range" slider.
- The envelope settings (attack, decay) for both the Oscillator and Noise generator circumvent the above rule – they will work in a range of 0 up to a limit defined by how big the range is set with the range slider. So if for example, the range slider spans across 25% to 75%, while all controls are randomized with values in that range (25% to 75%) , the envelope attack and release settings will be randomized in a range of 0 – 50%.

Tapping on the **"maximize-minimize"** button (arrow in circle) exposes the **"randomizer exclude"** options. Here, you can exclude parameters from being randomized either in groups or one by one. Specifically, tapping on **"Oscillator"** or **"Noise"** button will exclude all parameters related to the oscillator or noise section respectively being randomized while tapping on the **"Mains"** button will exclude the following parameters from being randomized:

- Oscillator – Noise mix
- Eq frequency
- Distortion
- Eq gain
- Level
- Pan

Tapping on the **“More”** button, located at the bottom right corner reveals all the parameters of mTonic so that you can exclude the desired ones, one by one. These buttons also act as displays that always show which parameters are currently excluded from the randomizer.



When any parameter is excluded, this is displayed on the **“Rand Drum Patch”** button and the **“randomizer exclude”** display lights up with a LED – tapping it will turn off all excluded parameters.

Transpose drum patch: Transposes the selected drum sound to a user selected amount. Upon tapping the transpose button, a slider pops up where the transposition amount is set and then, an **“OK”** button performs the operation. If **AutoHide** is enabled, after confirming the transposition the pop-up slider will close.



Cut Drum patch: copies the selected drum channel to the clipboard and after that, the selected drum channel is initialized.



Copy Drum Patch: copies the currently selected drum channel to clipboard

Paste Drum Patch: pastes the copied (or cut) drum patch to the selected drum channel

Exchange Drum Patch: after having copied (or cut) a drum channel you can use this function to paste the clipboard to another selected drum channel and replace the clipboard with the second drum channel. This way, you can re-select the first drum channel, tap paste and thus perform a “hot-swapping” of the two drum channels.

Editing only on sound patch – Include pattern: the **“cut”**, **“copy”**, **“paste”** and **“exchange”** functions are by default operating on the selected drum patch itself as all the rest of the functions. There are times though, where you might want to also operate on the patterns – for example if you completely want to “switch places” of two sounds and also switch their patterns. Tapping on this button you can choose to also cut, copy, paste and exchange the pattern together with the selected drum patch or operate exclusively on the drum patch

NOTES:

1. Besides the “Randomize” function, which is an algorithm surely less advanced than the native algorithm of mTonic (and possibly will not always result in interesting sounds) the rest of the functions do not differ in any way to the native equivalents of mTonic.

2. If you are using the “AbletonLive” compatible template you will notice that during some of the operations, the controls are momentarily randomized and then acquire the settings desired by the selected operation. This is by design and the purpose is to overcome some situations where Ableton Live will disregard incoming midi messages (which would therefore make the operations inaccurate). Rest assured, despite the in-between randomization, all the operations perform the tasks as designed and expected.

PATTERN OPERATIONS

CAUTION: Due to limitations of the midi communication and the way it is implemented in mTonic *(see also "technical details - Limitations" section)*, the Pattern Operations are inherently different to the native pattern functions found in the plug in for one reason: **the pattern operations are applied ONLY to the selected drum patch pattern and NOT globally on the patterns of all channels as is the case with the native functions.**



Clear Drum pattern: deletes all the active steps from the pattern edit lane (step sequencer) including the accents and fills.

Clear acc, Clear fill: delete all the accents and fills respectively

RAND Drum Pattern: randomizes the pattern, accents and fills. You can also set a somewhat guided randomization by using the probability sliders for the Triggers, Accents and Fills. With higher probability, more triggers-accents-fills will be occurring. Notice again, that the randomization algorithm is not the same with the native mTonic one but uses Lemur's randomizing algorithm.

RAND acc, RAND fill: randomizes only accents and fills. You can also set a somewhat guided randomization by using the probability sliders for the Accents and Fills. Notice again, that the randomization algorithm is not the same with the native mTonic one but uses Lemur's randomizing algorithm.

ALTER drum pattern: with this option, the current pattern will be "altered" with slight variations. The Alter amount slider defines how big of a "difference" the resulting altered pattern will have compared to the original. Once again, this is based on an algorithm designed strictly in Lemur, and as such, it may not yiled the same – desired results when compared to the native algorithm. That said however, it was tested and behaves quite nicely.

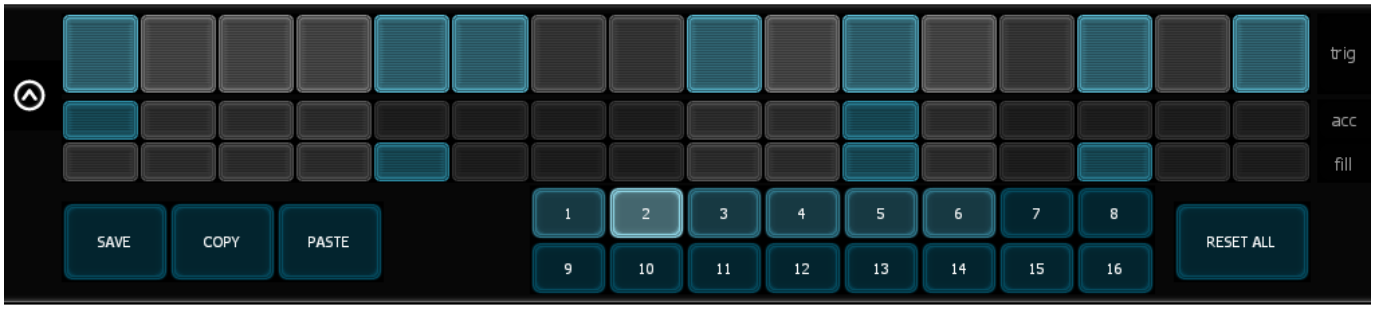
SHIFT left – right: shifts the current drum pattern one step to the left or to the right respectively.

REVERSE drum pattern: reverses the order of the pattern (accents and fills included)
INVERT Drum Pattern: this is a function not existing natively in the plug in. It inverts the triggers of the pattern – an active step becomes inactive, and an inactive one is deactivated. Coupled with the "Clear" function you can also quickly activate – deactivate all the triggers. The accents and fills are not included in this function.

INV acc, INV fill: same as INVERT Drum Pattern but for the accents and fills respectively.

Cut, Copy, Paste, Exchange Pattern: these behave as the native equivalents, again with the difference being they are applied to the selected drum sound pattern and not globally on the patterns of all sounds.

INTERNAL PATTERN PRESETS BANK



mTonicLemurized features an internal pattern presets bank capable of storing up to 16 patterns which can serve as a quick way to change between patterns for a drum patch and provide some sort of real time variety. The difference between the native mTonic patterns and this bank is that it only affects the currently selected drum patch/sound.

Using the internal presets bank is pretty straightforward – tap on the **"maximize – minimize"** button (arrow in circle) located on the left of the steps lane to access it. When you are satisfied with a pattern, tap and hold on **"save"** button and then tap on a preset number to store it. To recall a saved pattern just tap on the desired preset number. To copy and paste stored patterns tap and hold the **"copy"** and **"paste"** buttons and then the desired preset number. Keep in mind that any of these functions will not result in any change of the currently playing pattern – so in fact, you are free to use them while a pattern is playing in order to prepare your next steps in your performance.

To copy a preset slot to clipboard tap and hold the **"copy"** button and then tap on the desired preset slot. If you tap on an empty slot, an empty pattern (no steps activated) is assumed as the pattern to be pasted.

After having copied a preset to clipboard the **"copy"** button appears brighter to indicate that the clipboard is not empty. You can cancel the **"copy"** operation by either tapping **"paste"** or **"copy"** without any subsequent preset slot tapping.

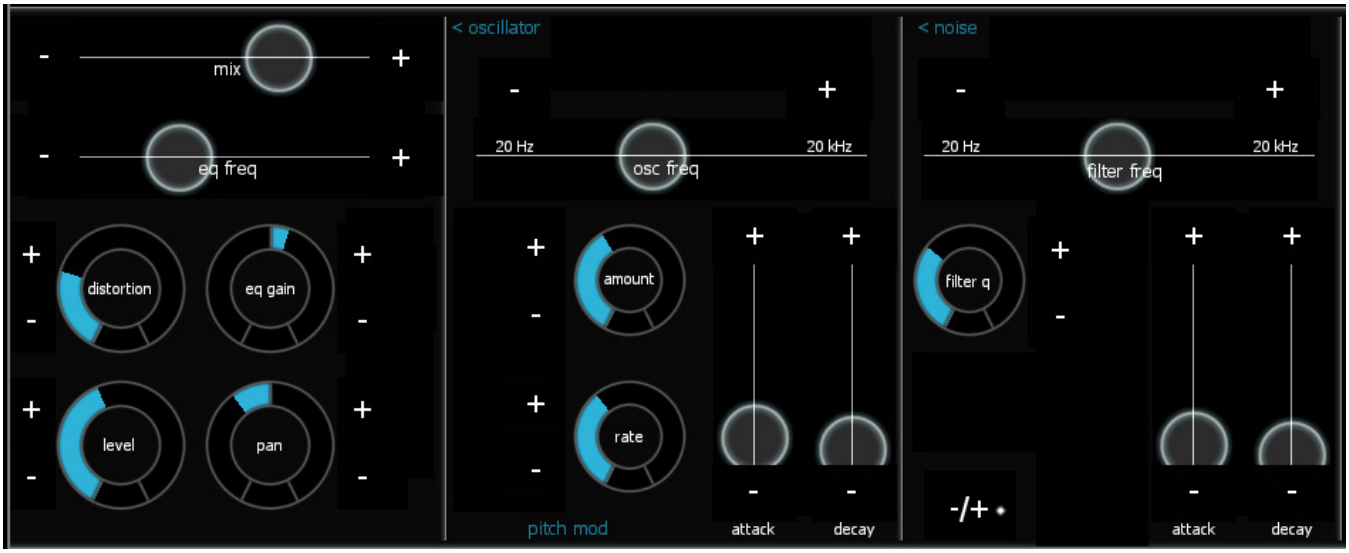
After having copied a preset to clipboard you can paste it onto one or multiple preset slots by keeping the **"paste"** button pressed and tapping on the desired slots.

The currently selected pattern preset is always displayed brighter –the preset numbers where a pattern is indeed stored also appear slightly brighter.

Tapping on the **"Reset All"** button deletes all the stored presets in the internal bank but has of course no effect on the actual currently playing pattern.

Finally, keep in mind that the 16 internal pattern presets are not drum patch dependent – so for example if you save a pattern on slot "1", even after selecting a different drum patch to edit – slot "1" will still have the saved pattern stored into memory.

FINE ADJUSTMENT (-/+) MODE



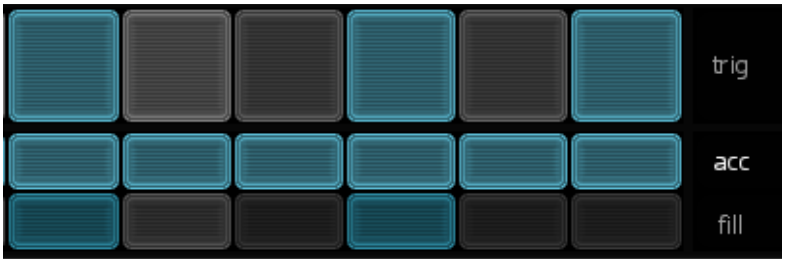
Sometimes delicate settings must be dialed in for specific parameters of mTonic, such as for example the Oscillator frequency which essentially affects the tonality - pitch of your drum sound. By tapping on the **"-/+"** button, located at the bottom left corner of the *"noise"* section of the interface the **"Fine Adjustment Mode"** is enabled.

While enabled, all the faders and knobs of the interface are presented with "**-**" and "**+**" buttons – tapping on them adjusts the selected parameter by one step of typical midi resolution (1/128 of the whole range). Of course the parameters which are presented with buttons (Oscillator waveform type, noise filter mode etc) are hidden as there is no point for fine adjustment to these.

By tapping both the "-" and "+" buttons of a knob or fader, the selected control will be set to exactly 50% of its range.

By default, when the "-/+" is tapped, the "Fine Adjustment Mode" is enabled in a "toggle" fashion: you need to tap "-/+" again to exit the mode. If you prefer to momentarily have "Fine Adjustment Mode" enabled you can change this behavior in the "Settings" page. The "Fine Adjustment Mode" will also be disabled automatically when navigating to other pages of the interface.

QUICK PATTERN OPERATIONS



It's also possible to make some simple pattern edits without even having to open the "Pattern Operations" page. Tapping on the **"trig"** label found right besides the steps lane, all triggers are disabled.

Tapping on the **"acc"** label all accents are activated – this is handy if you are planning to set the velocity sensitivity knobs all right up and keep the sounds audible with one move. On a second tap the opposite happens: all accents are disabled. Finally, tapping on the **"fill"** label, all fills are de-activated.

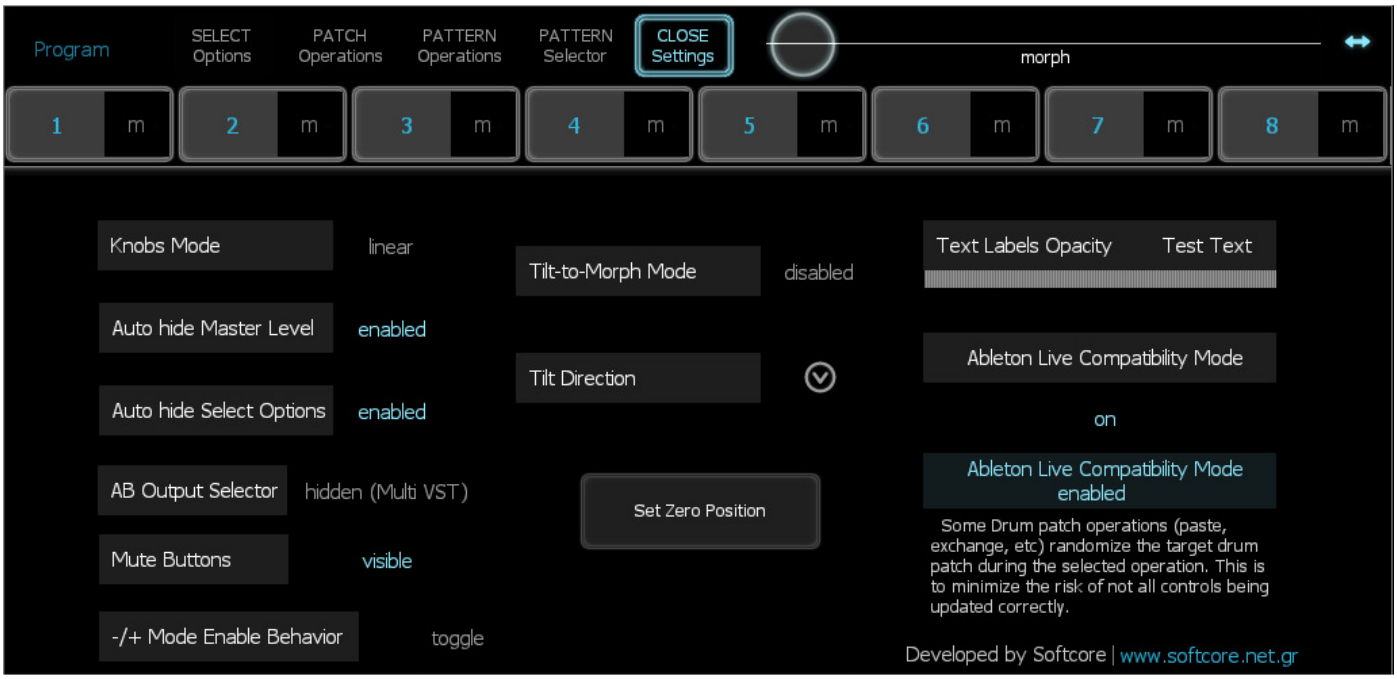
PATTERN SELECTOR



Tapping on the "Pattern selector" button opens a pop up window with buttons to select the available patterns of mTonic named from "a" to "l". Notice however that there is currently no way to activate-deactivate the chaining of patterns or have visual feedback of what is the currently playing pattern when chained patterns are enabled. *(Also see "Technical details – limitations" on this manual).*

SETTINGS PAGE

Tapping on the “Open Settings” page reveals a page with some settings to better configure mTonicLemurized to your workflow.



Knobs Mode Linear or polar: with linear mode, you can set the knobs by tapping on them and dragging up or down. With polar mode, you need to interact with the knobs in a circular fashion. This setting is similar to the native option found in the plug in preferences “Knob Behavior mode” although it does not really change the behavior in the plug in UI – just in mTonicLemurized.

Auto hide master level: due to ergonomics and space saving, the master level is controlled by a pop-up slider (the knob idea was tested and it was fiddly to adjust precisely the level so it was abandoned) which opens when you tap the “master” button located on the bottom right corner of the surface.
By default, after ten seconds the master level is hidden again or you can hide it by tapping on the “master” button. You can choose to have the “master level” slider hide only manually by disabling Auto hide here.

Auto hide Select Options: By default, after ten seconds the “Select Options” toolbar is hidden or you can hide it by tapping on the “Select Options” button. You can choose to have the “Select Options” toolbar hide only manually by disabling Auto hide here.

AB output selector: mTonic comes in two different VST configurations - .dll files. When in basic mode, it provides to the host 2 output channels A and B where all the 8 drum sounds can be routed by using this selector. When using the “Microtonic Multi” VST however, the plug in exposes to the host 8 different audio outputs with each drum channel internally routed to these outs so the A-B selector becomes redundant. Here you set the visibility of this selector in MTonicLemurized to suit your needs. By default, the selector is hidden as I really believe it’s more appropriate mixing-wise to use the multi out .dll version of the plug in and I think most users use this dll version.

Mute Buttons: By default, next to each channel select button, its mute button is located. You can choose to hide the mute buttons however, if for example you control the muting of mTonic channels with your host’s mutes.

Fine Adjustment Mode Enable behavior: by default, the "Fine Adjustment Mode" is enabled when the "-/+" button is tapped, disabled when tapped again (toggle). You can change this behavior to "momentary" – tapping the "-/+" button will enable the "Fine Adjustment Mode" and releasing it will exit the mode.

Text labels Opacity: if you are into uncluttered, minimalistic, elegant interfaces drag it all the way down and observe ;)

Ableton Live Compatibility Mode: (available only in "AbletonLive" template):
By default the "AbletonLive" template loads up with this option enabled. It's mainly there as a way to debug various functions. Feel free to disable it and observe how Live starts to "miss" or rather "ignore" some of the incoming midi messages, resulting in inaccurate Drum patch operations.

THE MORPH SLIDER AND THE TILT-TO-MORPH FEATURE



By default, the **"Morph"** slider located on the top right corner of the surface is controlled manually. An added **"Left Arrow"** button provides an alternate function. When lit, the morph slider will return to zero when released.

You can, though, enable the **"Tilt-to-Morph"** feature in the settings page. When enabled, the morph slider will "react" to the tilting of your iPad either towards you or towards the audience ((hopefully you have one). The most usual case scenario is that the iPad is not completely flat on a surface so an amount of "tilt" already occurs. To counteract, or rather to disregard this initial tilt, every time the "Tilt – To – Morph" feature is enabled, the current position-orientation of the iPad is considered the "Zero position". If however, during your performance you need to change the position – orientation of your iPad you can tap the "Set Zero Position" to reset it.

Tip: even after tapping on the "Set Zero position" you may still observe some "flickering" of the morph fader. To eliminate it, **slightly** tilt your device, tap the "set zero position" and then place it to its initial position. Now the morph fader is not flickering and it will start moving with your tilting right after your slight tilted location. In a way, consider the "set zero position" as the "threshold" tilt after which, the morph fader will start moving.

When the "Tilt-to-Morph" mode is enabled you can still move manually the morph fader, although naturally after being released, it will return to the position specified by the amount of tilt – if no tilt is applied, this means it will return to zero. The "Left Arrow" button is also replaced with a **"DualArrow"** button – you can use this when performing to quickly bypass the "Tilt-to-Morph" mode without having to go to the settings page.

TECHNICAL DETAILS - LIMITATIONS

Do not read this section unless you are a Lemur template aficionado yourself and you are curious about some design choices I made with this template.

The first obstacle I had to overcome was the fact that selecting a drum channel cannot be mapped to CC messages but only to midi notes. Therefore, when selecting a drum channel it is also triggered. This cannot be avoided currently although I'm quite hopeful that after the developers see what a "Box of Pandora" they opened by adding the "knobs and buttons send CC messages" they might implement it in the future. I'm also planning to intrigue them by making a small video presentation of mTonicLemurized in action so that they can understand what kind of "monster" they have created. ;)

For the time being, you can somehow select without audibly triggering a channel by using either the "Selection Trigger Velocity" or the "Mute on Selection" features, both found in "Select Options" toolbar.

Another approach of course would be to use separate CC messages for **EACH** drum sound – in other words, do not enable the "CC messages operate on selected channel" and the "Select channel with Midi notes" features.

Unfortunately this would require a total of 128 CC numbers **only** for the step sequencer alone not to mention the rest of the controls. Doable? Yes, in Lemur it would be, but the problem is, currently mTonic doesn't discriminate the midi channels it receives CC from. In other words, if we wanted to use different CC numbers for different drum channels, we would need more than one midi channel and obviously some CC numbers would only differentiate by the midi channel.

In simple words, if you try to midi map CC 21 ch1 and CC 21 ch2, currently in mTonic these two mappings will be perceived as one and the same midi message therefore we are limited to a total amount of 128 CC values of **any** channel. Although they are more than enough when "CC operate on selected channel" they are simply not enough if you want to use different CC numbers for each mTonic channel and still control all the aspects of mTonic. With a rough estimation based on older efforts with midi mapping mTonic we need a total of around 128 x 3 messages – so that means 3 midi channels CC numbers, all nearly-completely "occupied" – 2 channels for the controls and 1 channel for the steps lane. (Remember if each drum-channel step lane is controlled independently we need 16 steps x 8 channels CC numbers = 128 CC numbers) – and oh, wait a minute, I still haven't calculated the 16 x 8 accent buttons and the 16 x 8 fill buttons. So we need 3 midi channels for the lane editor and 2 midi channels for the rest of the controls – rising up now our demands for 5 midi channels and a total of 128 x 5 CC messages.

I further on, tested the "pitched mode" which supposedly receives messages on different midi channels – although true, this is only applicable to Midi Notes and not to Midi CC so again I had hit a brick wall.

So, by using the "CC messages operate on selected channel" and the "Select channel with Midi notes" features we have the best possible implementation with one limitation: there is no way for Lemur, to know the pattern of a preset for all the channels. In fact it is unaware of the pattern up until you select a drum channel to edit it – that's when mTonic sends the data of the selected drum pattern. We could, set up an internal memory bank in Lemur to store all the channel patterns of a preset after we select them one by one, but to perform operations like copy, paste, randomize etc Lemur would have to somehow select channel after channel to perform the operation on each one of them.
The limitation arising from these facts is that the pattern operations, as described in the manual can only refer to one channel at a time.

Ideally, my opinion is that mTonic should also add CC control support for the pattern selection buttons, the chain buttons and the mute buttons and also feature some way to show with "midi feedback" which pattern is now playing – possibly a different set of CC messages (so a set to select a pattern, and a set to "display" which pattern is currently played". Another approach would be to discriminate the channels of

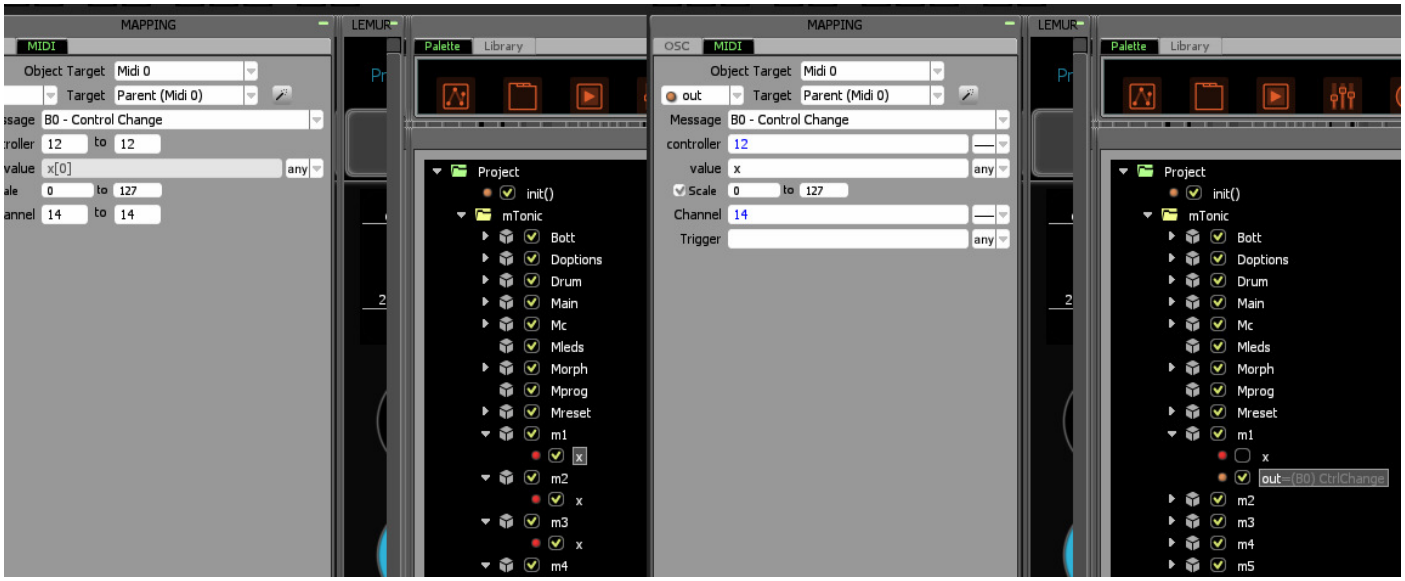
the CC messages it receives, although this would require, as noted above so many controls that frankly I don't think ANY other controller besides Lemur would be able to take advantage of the feature.

Another technical limitation that had me baffled for days was the way Ableton Live handles incoming midi messages. It appears that it filters out any incoming CC number if the sent value is not different to the one previously received (or 0 if none is received).
Let's consider this problem with an example. The mute buttons are mapped to CC numbers from 12 to 19. Let's say that we load a preset where all the mute buttons are off – so CC 12 to 19 are 0. Next, we load a preset where the mute button of channel 1 is on (the channel is muted). Naturally, our set up includes feedback so, the mute button lights up in our iPad, having received a CC 12 of value 127. Remember though, Live is still not aware of this – Live only knows that this button was previously off (0 value). So, upon tapping the mute button on our surface, it sends a CC 12 value 0 to Live – but Live, still thinking that 0 was the previous value, disregards this message and does not unmute the channel. Now we have the following – an unmuted button on our surface which is muted in MTonic inside Ableton Live. We tap on our surface again and the two buttons are both in "mute" state – this time though, Live received the CC 12 value 127 and it KNOWS that the button is in mute state. Upon a third tapping of the button on our surface everything works correctly.
The above mismatch is directly visible in buttons (mutes, pattern buttons) because we only have two states to differentiate although it also happens when any control is updated by the feedback – Live doesn't really "know" that the value has changed…..Unless……

Unless the connected surface, "re-sends" the values that are changed every time it receives the changes via feedback.
Here lies, in a nutshell the difference between the two templates provided with mTonicLemurized.

The typical template uses the native midi mapping system of Lemur, which naturally does NOT resend messages when it receives feedback. In the AbletonLive compatible template though, I have used custom midi outs which are triggered according to the x of the object. In other words, when the object receives feedback, and its value is changed – it resends the new value to Live. This feature proved to be invaluable – in fact, I don't think ANY midi controller will work correctly with ANY plug in which provides feedback and Live – while Live is good when using its native mapping system, there are the above limitations when using a plug in's native midi mapping.

ON THE LEFT, THE TYPICAL MTONICLEMURIZED – ON THE RIGHT THE "ABLETONLIVE" COMPATIBLE



Another limitation arising from the above characteristics of Live had to do obviously with the Drum Patch Operations. There were times where functions like cut, init, paste, and exchange didn't update all the controls correctly and of course again, this would happen when Live was "left thinking" there was no value change involved. I rectified these shortcomings by implementing a "timed" randomizer function in between these operations – before "pasting" the values of the controls, they are randomized, Live acknowledges this change of value and then will happily accept the subsequent changes when the operation finishes.

If you want to check the accuracy of my findings (and claims) you can easily do so with two ways:

1. Use the non AbletonLive compatible template with mTonic in Live. Don't be fooled however – at firsts it seems fully operational. And then you start noticing the little things that don't work correctly – some mute buttons refuse to work, some steps in the sequencer refuse to deactivate or inactivate. All for the first 3 presses - taps! ;)
2. Use the AbletonLive compatible template and perform an "Init Drum Patch"….Now go to the settings page, disable "AbletonLive compatibility" mode and try to initialize a different drum channel. You will immediately see that the two may not be the same – even though they should.

Finally, other limitations that were out of my control were the lack of feedback for the currently selected "program", the fact that you can only midi control – select out of a bank of 16 programs only, the lack to control chaining of the patterns and feedback of the currently playing pattern.

THE "RUNNING LIGHT" – MIDI CLOCK SYNC DETAILS



By default, each template provided in mTonicLemurized is also designed to receive midi clock sync messages in order to display a "running light" along the step sequencer of the interface at the same Daemon target where midi CC messages are sent and received. If for any reason you would like to edit a template and set a different Daemon target for the reception of Midi clock messages then open the selected template in the Lemur editor and locate the following object in the project hierarchy:

Project -> **mTonic** -> **Seq** -> **ClockLeds**

Make sure you have the object "ClockLeds" selected (it is a Lemur container) and in the "mapping" window of the Lemur editor select the midi tab. There, change the default "Object target" to your desired one and save the template – I advise you to save it with a different name, indicating that this template has a changed target for the midi clock to avoid future confusions.

Also, when the "running light" for the step sequencer is active (midi clock sync messages sent from host application to mTonicLemurized) there is no way for Lemur to know if the currently visible pattern on the interface is in fact the one being played – as a result, the running light will always be visible even though the currently selected pattern may NOT be the actual pattern being played in the mTonic plug in.

ACKNOWLEDGEMENTS

At this point I have to state that the “logic” of the midi clock is entirely derivative of the scripting found in Lemur’s factory template “iPad – LFO Studio” so a big thanks to the developer, Mark Strauss for making that template freely accessible.

Also, the internal preset bank system is hugely inspired by similar works of Antonio Blanca – a big thanks to him too for all the fantastic scripting found in his templates.

VERSION HISTORY

VERSION 2

Changes:
- Loading animation added – when mTonicLemurized is opened, version number and credits are displayed for a few seconds
-"Randomizer exclude" feature added
- Ability to include pattern in "cut", "copy", "paste" and "exchange" edits of drum patch operations
-"Internal Pattern Preset Bank" added
-"Fine Adjustment Mode" added
- Sequencer "Running Light" added
- Acknowledgements section and more images added in the user guide
- Minor aesthetic fixes (brightness of some labels, coloring, etc)

VERSION 1.2

Changes:
- "Select Options" toolbar added, ability to set desired velocity when drum channels are triggered by selection, ability to momentarily mute an unmuted drum channel when selecting it, so that it is not audibly triggered when being selected.
- "Auto Hide Select Options" added in Settings page
-"Quick Pattern Operations" added: the labels "trig", "acc" and "fill" are now essentially buttons to perform quickly deletion of triggers, all accents on – all accents off and all fills off
- Pattern selector buttons now light up to display the currently selected pattern
- a "thinning" mechanism for the output of "Tilt-to-morph" feature prevents unnecessary midi data being sent to the host application when no tilting is occurring.
- enabling the "Tilt-to-Morph" feature now automatically sets the initial "Zero Position".
- Corrected the knobs in "Velocity Sensitivity" section which were falsely presented as bipolar knobs (like pan) in classic skin
-Knobs in pixel skins are now brighter making their background (off range) more distinguishable
-a small line indicates the 0dB setting of master level fader

Bug Fixes:
- The "oscillator mix", "eq frequency", and the "attack-release" envelopes of both the "oscillator" and the "noise generator" were falsely being randomized with "range" even when the "free randomizer algorithm" was selected.
-the "master level" label was not becoming visible when touching the "master level" fader when "text Labels Opacity" was set to low values.

VERSION 1.1

Bug fixes:
fixed an issue with the bi-directional feedback of "oscillator waveform" and "noise filter mode" for the Target1 and Target2 templates

VERSION 1

Initial release

FINAL WORDS

As all software, perhaps bugs have slipped into mTonicLemurized although exhaustive testing has been performed on it - feel free to report to me any bugs you happen to find and rest assured I will do my best to solve them immediately.

A big “Thank you” to the developers of mTonic – for me, it’s the Roland x0x of year 2000 – original and great sounding electronic drums, with originality and correctly “ported” to the virtual world without being overly attached to the hardware nature of things.

A huge respect to all the Lemur community for the inspiration and help acquired when learning Lemur and specifically to developers who provide quality templates for the rest of the world without a fucking price tag on everything.

Also, special thanks to Antonio Blanca – most of my ideas have come from his works. In fact, any common element you might see in my templates and Antonio’s, be sure that he was the original creator and I just copied – got inspired from his ideas. As an example, the idea to use locked Multiball objects as faders was something I saw on one of his templates. Would have I thought it myself at some point? Possibly, but truth is, I saw it in his stuff first!

Greetings!

softcore – www.softcore.net.gr