Version 1.01

midi
2themax

Table of Contents

Introducing MXL Pack

MXL Pack is a collection of **over 40 Max-for-Live devices** that that extend Ableton Live in many areas, most notably in the way it can process and transform MIDI messages. They work well with **Live Suite Edition version 10** and **11**, but do **NOT** work with other, less powerful versions of Ableton Live that don't support Max-for-Live devices.

Many MXL devices are truly unique and offer functionalities that you cannot find anywhere else. Moreover, often a single MXL device implements features that you may obtain only by combining two (or more) devices from other libraries.

MXL devices can be useful to all newbies and expert Live users, but were designed especially with these musicians in mind:

- live musicians and improvisers wishing to push the envelope of their MIDI instruments
- studio musicians looking for new ways to combine tracks and clips
- music composers intrigued by chaos, probabilities, and random melodies

The **MXL Pack** product includes all the **40+ M4L devices** described in this document, for a very convenient price: if you are serious about using MIDI with Ableton Live, it is surely the one to get. If you are not interested in all those devices, you might opt for slightly less expensive packs. We also carry the **MXL Free Pack** that – as its name implies – can be downloaded free and includes several devices that are useful in themselves or combined with devices in other packs.

You can find **MXL Pack** and **MXL Free Pack** here: http://www.gumroad.com/midi2themax

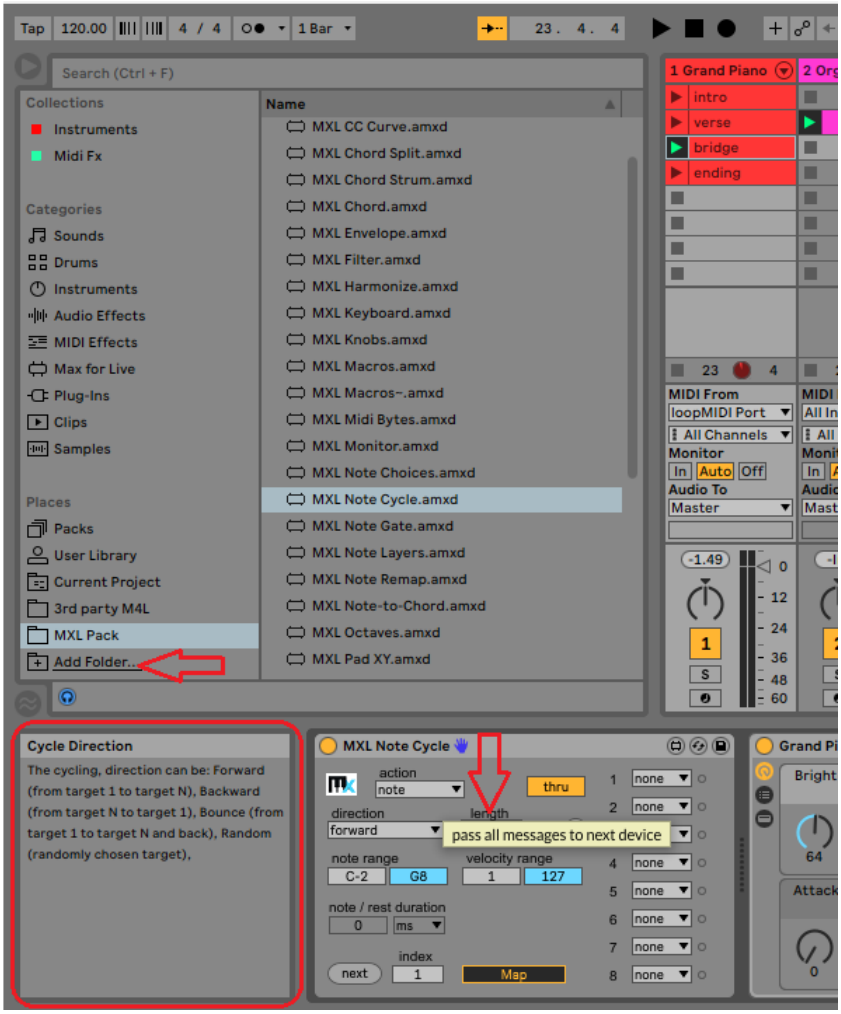
Setup and Documentation

Regardless of the MXL Pack version you have, this manual encompasses all MXL devices – included those that are distributed individually - because all of them share several common features, for example global buses, command groups and presets.

All MXL Pack versions are distributed as a compressed ZIP file containing one or more MXL devices. We recommend that you create a folder on your hard disk named **MXL Pack** and uncompress the ZIP file in that folder.

Inside Ableton Live, click in the **Add Folder** option in the left sidebar, and select the folder you just created. You can now drag MXL devices to your current track, as you would do with Live's native ones.

In addition to the PDF manual, you can learn how devices work by hovering the mouse on a field and look at Live's Help window (press the "?" key if the Help window is currently hidden). If you keep the mouse still for a few seconds, you can also read a short description in the yellow tooltip.



***NOTE**: a few customers reported that some fields do NOT show text in the Help window. We are working with Cycling74 support to fix it.*

Support

If you need to report problems, have suggestions on how to improve them, or proposals for MIDI devices that might complete the **MXL Pack** collection, please contact us at midi2themax@gmail.com.

How to

This section illustrates the most important features of MXL Pack devices and shows the portion of the documentation that discusses each of them in more detail.

Keyboard Enhancements

Keyboard Enhancements	
add the features that your MIDI keyboard is missing, including Aftertouch and Polyphonic Pressure, Bank and Program Change, and more	MXL Knobs
customize your MIDI keyboard with **split points and overlapping layers**, velocity scaling or custom **velocity curves**, velocity fade-in and fade-out near split points, and more	MXL Note Layers MXL Velocity

generate release velocity for MIDI noteoff messages – you can control release velocity with a knob on your MIDI keyboard, or by applying a curve to attack (noteon) velocity or note duration	MXL Release Velocity
implement a 4-state **hold & sustain** feature that no hardware MIDI keyboard offers – for example, sustain only notes whose attack (or release) velocity falls in a given range, or that played for at least the time interval you specify	MXL Sustain
never play the wrong note again thanks to the most advanced scale quantization device ever built for Ableton Live – select one of the **400 (four hundreds!) scales**, define how non-scale notes are processed, transpose them by the desired scale interval, or use white keys (e.g. the C major scale) to play any scale in any key	MXL Scale
quickly repeat incoming notes, either with a specified period or by providing the "tick" yourself	MXL Repeat
change the response curve of Control Change, Aftertouch or Pitch Bend messages, or selectively disable Pitch Bend up or down messages	MXL CC Curve

Buses, Monitoring and Filtering

Buses, Monitoring and Filtering	
filter incoming MIDI messages, route notes to different destinations depending on their pitch and velocity	MXL Filter MXL Note Layers
transform incoming MIDI messages into a different message, e.g. Control Change into Aftertouch or Pitch Bend	MXL Knobs MXL Pitch Bend
create flexible "note gates" where you dynamically decide which notes pass the gate and how their velocity is affected	MXL Note Gate
send MIDI messages from one track to another track using 16 buses	MXL Send MXL Receive
monitor incoming and outgoing MIDI messages, including advanced chord recognition and notes from buses	MXL Monitor MXL Bus Monitor MXL Keyboard
use feature-packed on-screen MIDI keyboard to test your setup when you don't have the real thing available – including PitchBend wheel, Aftertouch support, knobs that send Control Change, etc.	MXL Keyboard MXL Knobs

Chords and Harmonizers

Chords and Harmonizers	
make your melodies "thicker" by generating up to six transposed octaves, three above and three below the base note	MXL Octaves
create chromatic or diatonic (scale-based) harmonizers – for each note you can select the interval, the velocity offset, the destination track/instrument and the delay amount, so that the harmonizer can also work as an arpeggiators – you can even play fewer or more notes depending on the incoming note's velocity	MXL Chord MXL Scale Chord
stay away boring harmonization schemas based on fixed intervals – MXL offers both **chromatic and diatonic (scale-based) harmonizers** that react to notes coming from MIDI keyboard or from another track	MXL Harmonizer
create a customized harmonizer that converts any incoming pitch into a 4-note chord, each note with a different velocity – a great way to play chords using any portion of your MIDI keyboards, including the pads that you would normally use for finger-drumming	MXL Note-to-Chord
split incoming chords into distinct notes, process them individually or send them to different tracks and instrument, e.g. play a string quartet with a single hand or clip!	MXL Chord Split

easily create “rotating” harmonizers by sending each note to a different chromatic or diatonic harmonizer, in a cyclic or random fashion	MXL Note Cycle with MXL Chord or MXL Scale Chord
“strum” incoming chords, to simulate fast arpeggios or guitar	MXL Chord Strum

Note Generators and Processors

Note Generators and Processors	
transpose notes by the same number of semitones in multiple tracks	MXL Transpose with command groups
delay all MIDI messages or just notes	MXL Delay
control the minimum and maximum duration of notes, sustain notes for a time that depends on the original note’s velocity or duration	MXL Note Length
use portamento to move from one note to the next one with a glissando effect	MXL Portamento
detune all notes in a track by a given pitchbend offset, e.g. play with strings and horns tuned at 442 Hz!	MXL Pitch Bend
remap note to any other note – great for drum kits!	MXL Note Remap
send incoming notes to different tracks/instrument using a cyclic or random schema – for example, you can use two or more tracks with same instrument and different audio effects, and process incoming notes in slightly different ways to add variety to recorded or live melodies	MXL Note Cycle
quantize incoming notes to the correct scale, even if your song modulates to other scales and keys – select the current scale and key by means of notes stored in Live’s clips and tracks, so that your notes are always “right” even over the most harmonically intricate chord progressions	MXL Scale with remote preset recall
generate notes using CC messages: great for playing melodies with wearable MIDI devices, Bluetooth rings, etc	MXL Make Note

Randomization

Randomization	
randomly drop notes with a given probability while ensuring that resulting musical phrases are as short (or as long) as you wish	MXL Random
create in-tune variations of a given melody by transposing incoming notes by a random scale interval - you decide how similar or different the new melody is by assigning a distinct probability to each negative or positive scale interval	MXL Scale Random
randomize melodies by transposing notes and change their velocity, using the ranges you provide and optionally avoiding large intervals that quickly reveal the random nature of the result	MXL Random MXL Note Choices MXL Repeat
delay all MIDI messages by a random time interval	MXL Delay

Parameter Mapping

Parameter Mapping	
map Live’s parameters to velocity of incoming notes, pitchbend wheel, number and average velocity of active (playing) notes, and more	MXL Note Bindings MXL Pitch Bend MXL Chord Split
use pitch or velocity of incoming notes – either with specific pitch or whose pitch is in a range - to control Live parameters	MXL Note Bindings
create macro commands, to control up to **8 Live parameters with a single knob**, which you can control via Automation or from an external MIDI controller, map two or more	MXL Macros MXL Macro Multi

distinct MIDI message to the same Live parameter or combine two incoming messages using basic math operations	MXL Macro Pairs
detect pattern in musical phrases and use them to change Live parameters – for example, change scale when you play a given note three times	MXL Patterns
use an XY pad to control Live parameters or send any MIDI message – you can even record your mouse gesture and play them back in loop mode	MXL Pad XY
create ADSR or DAHDSR envelopes for each incoming note, and use those envelopes to control a Live parameter, send Control Change, Aftertouch or any other MIDI message	MXL Envelope

Miscellaneous

Miscellaneous	
create a group of "linked" MXL devices that behave in the same way even if they are located in different track, e.g. transpose two or more tracks by the same number of semitones by rotating a single knob on your keyboard	see Command groups
save current configuration in 100+ presets, assign it a name and quickly recall it by hitting a note on your MIDI keyboard, or send a note from a clip in same or different track	see Presets
control external hardware with custom Sysex messages that can contain numeric and text parameters	MXL Midi Bytes
add a description to tracks and devices	MXL Scratchpad

Common Features

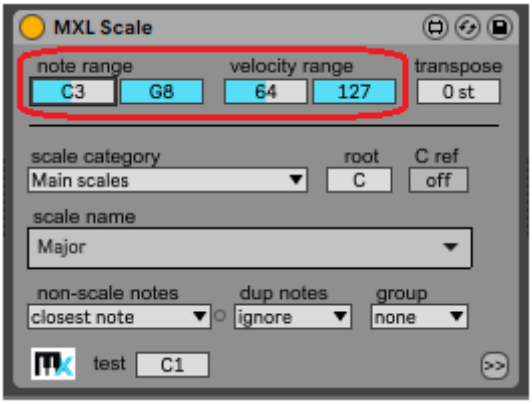
All MXL devices share the same “philosophy” and the same approach to MIDI processing. Consequently, many fields and parameters work in the same way across the entire collection of devices, which in turn simplifies learning how to use them.

Note Filters and Durations

Many MXL devices have been designed with the live performed in mind, as opposed to studio musicians. One of the key differences between these two roles is that the former might need or wish to follow the on-the-spur inspiration, without being framed in rigid rules.

Take “scale quantization” as an example, a.k.a. the ability to ensure that incoming notes fit a given scale. While this feature is surely welcome in most cases, live performers might desire that *not all the notes they play* be quantized. For example, they might want to play “passing chromatic notes” to join two notes in the melody they play with their right hand, or play non-scale notes to add tensions in the chords they play with their left hand. These exceptions to the rule cannot be enforced if you use Live’s Scale MIDI effect.

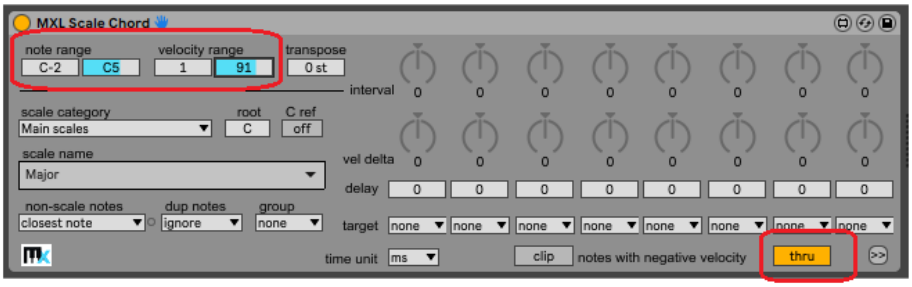
Quite conveniently, the MXL Scale device offers four fields labeled as **note range** and **velocity range**, which allow you to specify which notes the quantization effect must be applied to. You can use these fields to exclude the effect for notes below C3 (middle C), which is the area of the keyword where you presumably play chords, as well as to notes played with low velocity, which allows you to freely play “passing notes” if you don’t hit the key too hard:



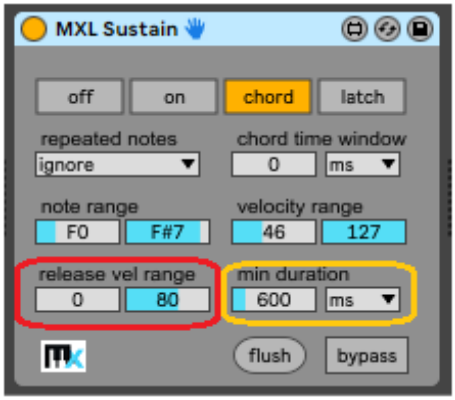
It is important to bear in mind that the filter only specifies which notes are processed by the device and to which the MIDI effect is applied. All the notes that do ***not*** match the filter are allowed to pass unaltered.

Many other MXL devices offer this filter, including MXL Delay, MXL Note Cycle, MXL Note Length, MXL Note-to-Chord, MXL Octaves, MXL Portamento, MXL Random, and MXL Tranpose.

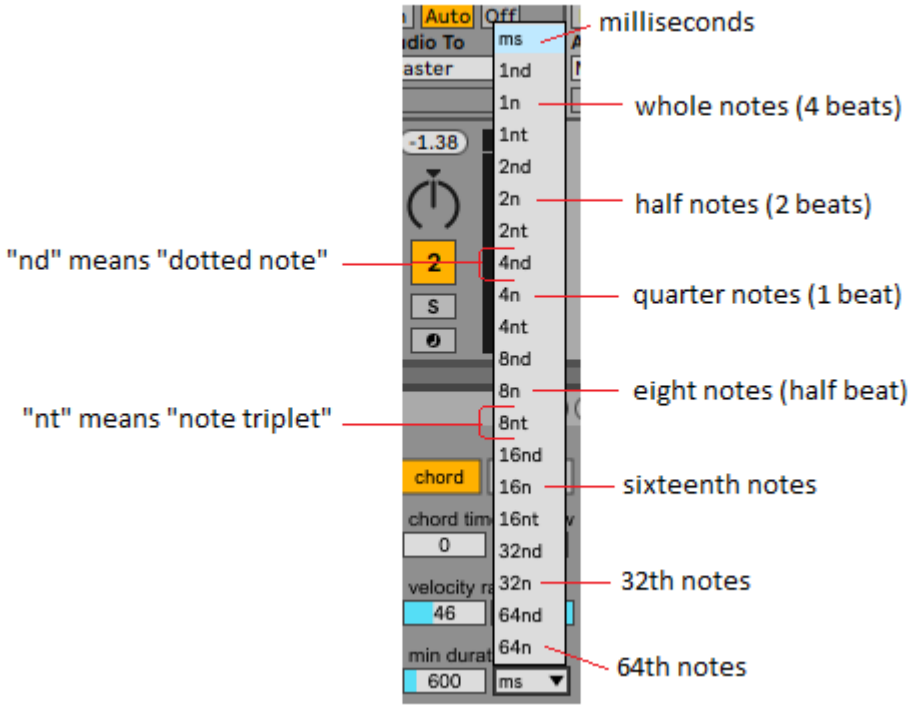
Some devices offer you one more option, the **thru** button. When this button is enabled, all incoming notes are passed verbatim – regardless of whether they match the filter – even though the specific MIDI effect is applied only to notes that match the filter. For example, the following MXL Scale Chord device let all notes pass unmodified, but generate a chord only for those below C5 with a velocity equal to or lower than 91:



The majority of MIDI effects must be applied when the note arrives, therefore their pitch and velocity are the only values available to accept or ignore them. In a few cases, however, the effect can be applied when the NoteOff message is received, in which case it is possible to evaluate also the release velocity (if your MIDI keyboard emits this information) and the note duration. For example, the following MXL Sustain device sustains only notes that have played for at least 600 milliseconds and that were “softly” released:



Speaking of time intervals, you can select between ***unsynched*** (or absolute) intervals (in milliseconds) and ***synched*** time units, which depend on current BPM. In all cases, the menu that allow you to make the selection is the same:

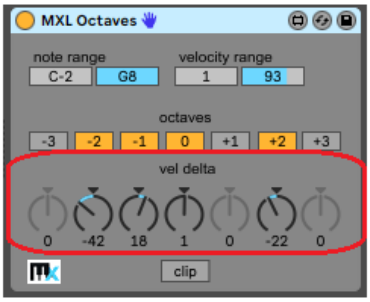


When working with absolute time intervals, you can select any interval between zero and 4000 milliseconds, with increments of 50 milliseconds. When working with a synched time unit, you can select a value between 0 and 16

Notice that some effects that use synched time units might not work properly if Live's transport is not enabled (i.e. if no clip is playing).

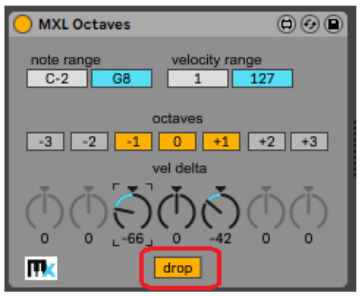
Velocity Clipping

Several MXL devices can increase or decrease the velocity of incoming notes. In most cases, the fields in question are labeled **vel delta**, as in the case of the MXL Octaves device:



The indicated value is added to – or subtracted from, if negative – to the original velocity and the result is the velocity of the generated note. By default, the result of the operation is constrained between 1 and 127, which means that the outgoing note always plays, even if it might be barely audible if velocity is 1.

However, all the devices that offer this velocity-clipping feature display a **clip/drop** button: if this button is enabled and the resulting velocity is zero or negative, then the note is note played at all:



For example, by default the device above plays two additional notes, one octave above and one octave below the original note. However, if the incoming note has velocity equal to 66 (or lower), then the lower octave doesn't play; moreover, if velocity is 42 or lower then the upper octave doesn't play, too.

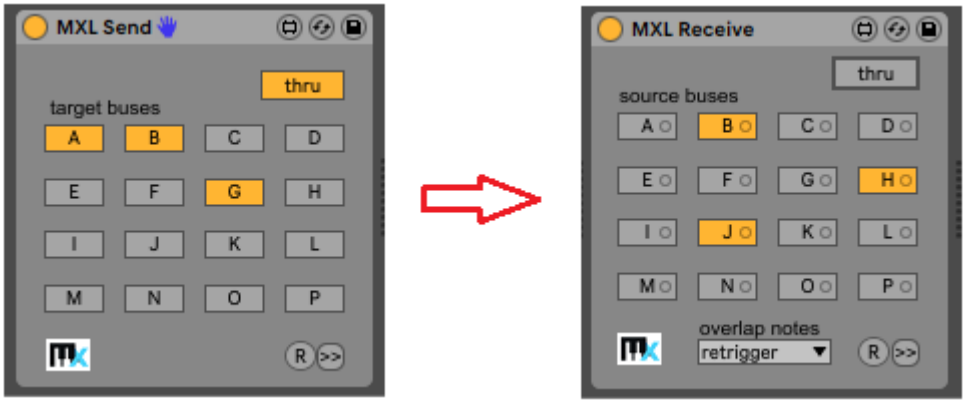
Many other MXL devices offer this **clip/drop** feature, including MXL Chord, MXL Note-to-Chord, MXL Random, and MXL Scale Chord.

Global Buses

The majority of MXL devices can send MIDI messages to a "global" MXL bus and/or receive MIDI messages from a global bus. In fact, much of the unique features that MXL devices offer are based on buses, because they allow devices to communicate with other devices in the same or different track.

The concept of MIDI buses isn't new in Live. For example, the "Max 7 Pitch and Time Machines" pack includes the Max MIDI Sender and Max MIDI Receiver devices, which build on the same concept. As you read on, you can realize that the MXL Pack expands on this mechanism to implement features that would not be possible otherwise. (Notice that MXL buses are distinct from the buses used by the abovementioned Live devices, thus you can use both buses in a Live Set without them interfering with each other.)

For starters, MXL includes two M4L devices that allow you to send and receive messages to/from MXL buses. These devices are named – quite predictably – MXL Send and MXL Receive:

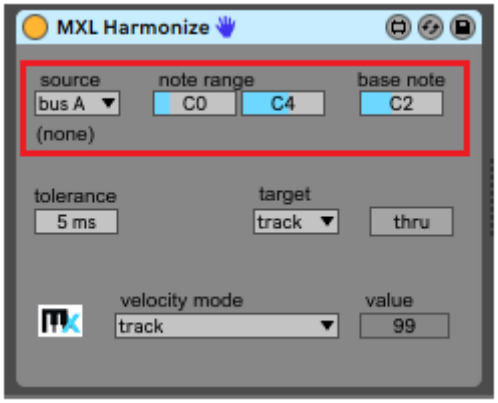


The above figure shows the most important feature of MXL buses: one single device can send to (or receive from) two or more buses at the same time. This detail adds a tremendous degree of flexibility to your Live sets:

- a given track can receive notes from two or more clips in other tracks, even if they are sending to different buses – it is as if Live tracks supported multiple clips that run together
- a given clip can send notes to multiple tracks, where they can be processed in different ways, for example by transposing them by a different number of semitones or by delaying them by a different time interval
- any combination of the above

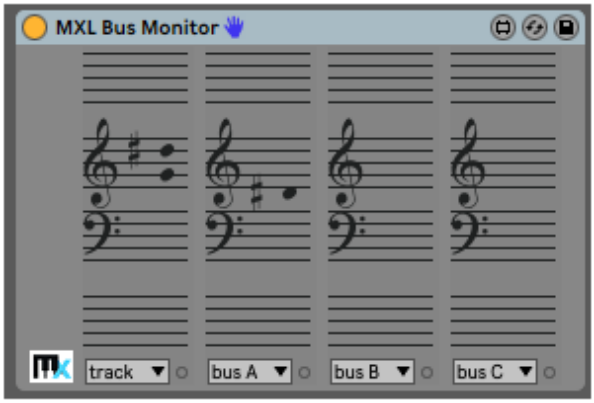
Notice that also Live's own Max MIDI Sender and Max MIDI Receiver devices allow you to merge notes from different tracks. However, they were *not* designed to handle complex scenarios and – in fact – they do not behave correctly when a device receives the same note from two or more sources.

MXL buses are not useful just to merge notes from different sources. In fact, many MXL devices can be controlled by means of MIDI notes received on a bus. For example, the MXL Harmonize device allows you to control harmonization by sending note to a bus:



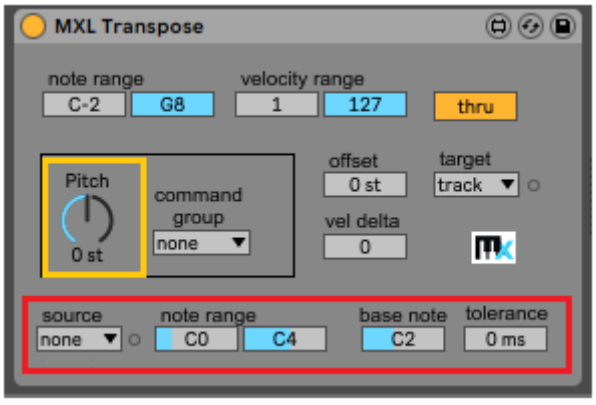
Another important feature of MXL Send and MXL Receive devices is the ability to store current configuration using the preset panel. This feature – which is common to many MXL devices – is described later in this document.

In some complex scenarios that involve multiple buses, you might find the MXL Bus Monitor device very useful to understand how different sources interact with each other.



Remote control

All MXL devices support Live's parameter mapping mechanism that allows you to bind their fields, dials, etc. to incoming MIDI Control Change (CC) messages. In addition to such standard feature, several MXL devices offer an alternative way of controlling their parameters, called *remote control*. To illustrate this concept we will use the MXL Transpose device, which allows you to transpose incoming notes by the number of semitones indicated by the **Pitch** dial:



> ***Note:*** *For a complete discussion of all the features of the MXL Transpose device see its specific section. The paragraphs below only explain the remote control feature.*

Like any Live user-interface element, the **Pitch** dial is automatable and can be bound to a MIDI parameter, for example a knob on your MIDI keyboard. However, precisely controlling the pitch using a knob during live performances can be tricky, thus MXL Transpose offers an alternative mechanism based on notes received from one of the buses A-P:

The **source** and **note range** fields indicate which notes are intercepted and used for remotely control the transposition dial. The **base note** is the note that corresponds to "no transposition": in the example above,

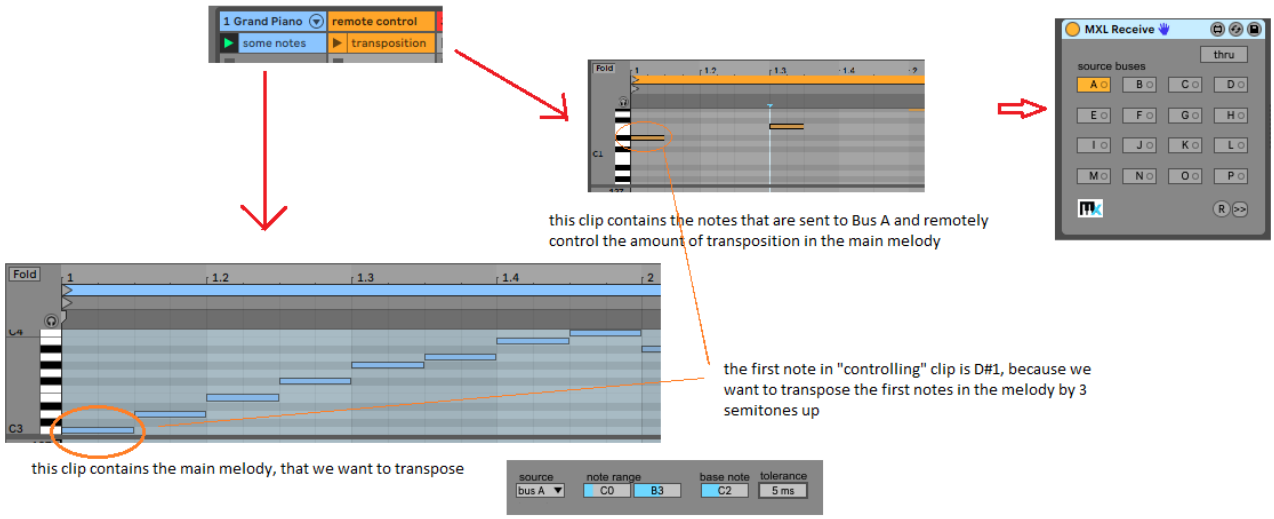
the C2 note received from Bus A resets the Pitch knob to **+0 st**, the note C#2 sets the knob to **+1 st**; the D2 note sets the knob to **+2 st**, and so forth. Notes below the base note correspond to negative values: the note B1 sets the **-1 st** value, the note Bb1 sets the **-2 st** value, and so on.

The **note range** and **base note** fields determine the range of values you can set remotely using this mechanism. For example, if the base note is equal to the lowest note in the range then you can only set positive values for the **Pitch** control; likewise, if the base note is equal to the highest note in the range then you can only set a negative number of semitones. In most scenarios, the base note sits halfway between the lowest and highest note in the range, so that you can set both negative and positive intervals.

The **source** menu can point to the current track or one of the A-P buses. You typically use the "track" setting when performing live, so that you can change the amount of transposition using a dedicated area of your MIDI keyboard; conversely, you use a bus as a source when controlling the amount of transposition using notes in a clip or in the arrangement section of another track.

> ***Note**: as mentioned in the [Parameter Mapping] section, all MXL parameters are automatable, therefore in general, the remote control feature is more useful to control parameters in real time from a MIDI keyboard. The following description uses remote control from a clip mostly to "visualize" the relationship between the notes that must actually play (*source notes*) and the notes that remotely control the device (*remote control notes*).*

When the notes that control the transposition come from another track (i.e. from a bus A-P), you might need to set the **tolerance** field appropriately, because you have no control on the order in which Live triggers notes inside clips. To understand why this can be important, look at this scenario, where the clip in the "remote control" track remotely controls the **MXL Transpose** device in the "Grand Piano" track:



Here is the potential issue: we cannot be sure that the D#1 note in the "controlling" clip arrives before the C3 note in the main melody. In fact, the relative order in which Live emits these notes should be considered as random, and one may arrive a few milliseconds before or after the other, depending on how many chores Live has to attend in the current set. If these two notes were meant to emit sounds this little delay would be negligible to even the most trained human ear; in our case, however, this discrepancy matters, because if the D#1 note arrives after the C3 note, then the C3 note won't be transposed as desired.

The problem may affect all the notes in the melody, not just the first one. For example, the second note in the controlling clip is F1, because we want to transpose all notes in the melody 5 semitones up, starting with the G#3 note at position 1.3. However, if the F1 note arrives after the G#3 note, the latter will not be

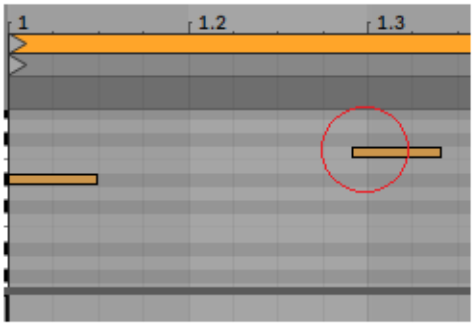
transposed correctly. (In this example, the 5-semitone transposition will be applied to the next note in the melody, namely A3.)

In some scenarios, for example when you are generating random melodies, if a note is not transposed as expected isn't a big deal; in other scenarios it is crucial that all notes be transposed by the intended number of semitones. In these cases, you have to ensure that the MXL Transpose device processes the "controlling" note *before* the note from the main melody. There are two ways to do so.

The first solution to the problem is that you manually shift notes in either the controlling clip or the main melody by a very small time interval:

slightly delay notes in the melody, to ensure that the controlling note is processed first

OR



slightly anticipate notes in the controlling clip, so that Live emits them before the corresponding notes in the melody.

(this approach cannot be applied to first note in the controlling clip)

In practice, you can select all notes in the controlling clip (right) and move them slightly to the left. However, this mechanism cannot be applied to the very first note in the clip, therefore if the melody starts on downbeat you have to move the very first note in the melody slightly to the right.

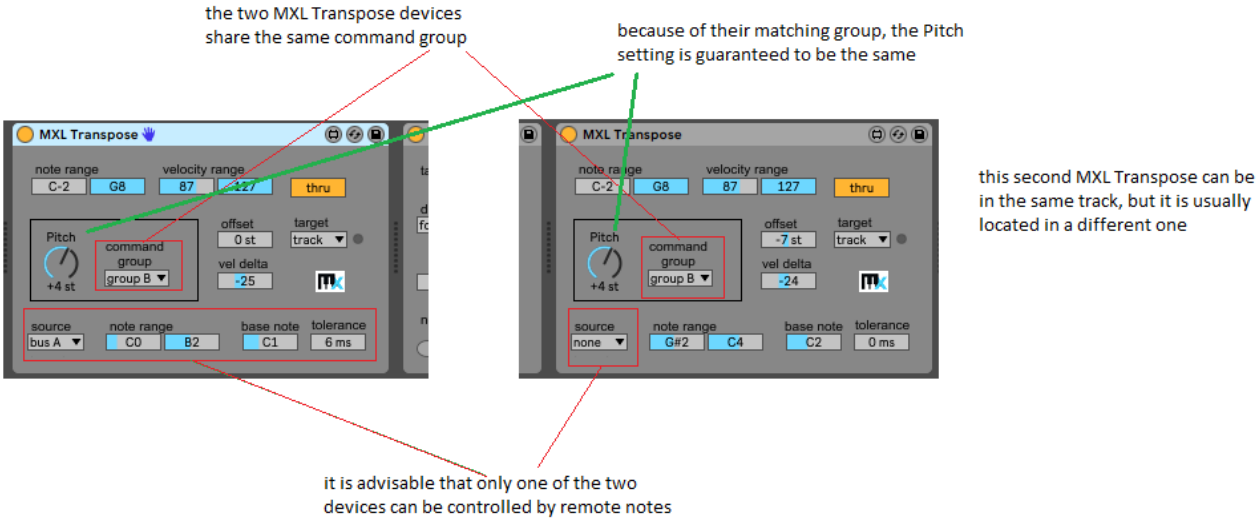
To avoid these manual adjustments, all the MXL devices that use remote control also include a special **tolerance** field. This field tells the device to wait for the specified number of milliseconds before transposing notes from the current track, so that the device has a chance to process notes from the **source** bus and change the **Pitch** setting by the intended number of semitones.

Usually a value of 4-6 milliseconds is adequate for most scenarios, and does not affect the timing of your song in a noticeable way.

Command Groups

A subset of MXL devices support command groups, a unique MXL feature that allows you to synchronize the configuration of multiple devices of same type. MXL supports eight groups, named **Group A** … **Group H**: if two or more devices share the same group, they will automatically share their settings.

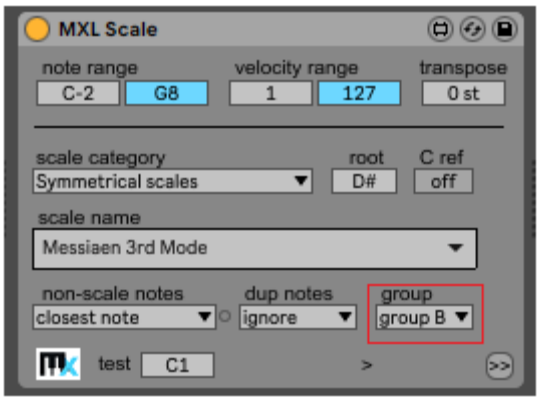
To illustrate how command groups work, we will use the MXL Transpose device as an example. As seen in previous section, this device is able to transpose incoming notes by a given number of semitones up or down, as indicated by the **Pitch** dial. When two MXL Transpose devices share the same command group (**Group B** in following figure), if you rotate the dial in one device, the new dial value is immediately applied to all other devices in the same group:



Notice that not all the settings in a given device are shared by other devices in same group: in the case of MXL Transpose, only the **Pitch** value is shared, yet note range, velocity range, offset, etc. are not.

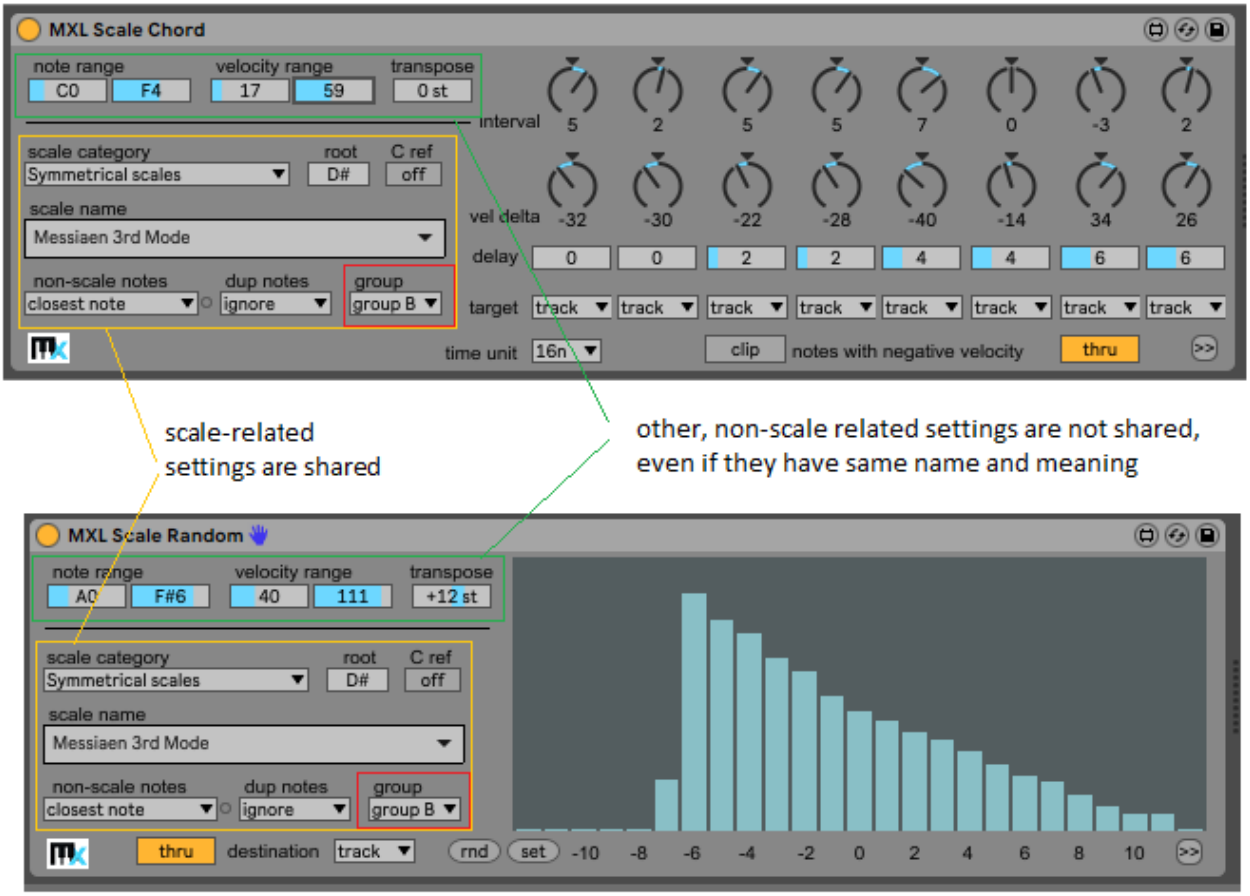
Belonging to a group establishes a relationship "among peers": there is no master device and there are no slave devices, and you can change the **Pitch** setting from any device in the group. However, if you plan to use remote automatic to control the amount of transposition – as explained in previous section – then the **source** field should be set to "none" for all but one device, which effectively becomes the "master" of the group (see figure above). Doing so reduces CPU usage and – above all – prevents infinite loops that might even crash Live in some extreme cases.

Let's see another command group example. As it name suggests, the MXL Scale device allows you to quantize incoming notes to a given scale, thus command groups offer a simple and effective way to set the same scale name, scale root and other scale-related settings for multiple tracks:



In general, the same command group can be shared by MXL devices of different type. For example, a MXL Scale device can communicate via Group B without interfering with MXL Transpose devices, because the commands that they exchange with other devices of their own group are different.

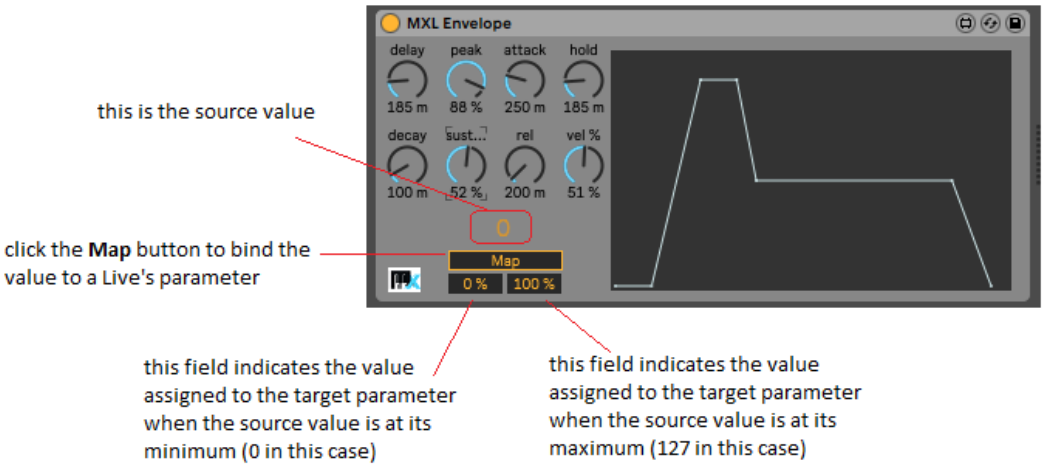
In some cases, however, devices of different type can affect each other if they share the same group. This happens, for example, with the MXL Scale, MXL Scale Chord and MXL Scale Random devices, all of which use command groups to share scale-related settings:



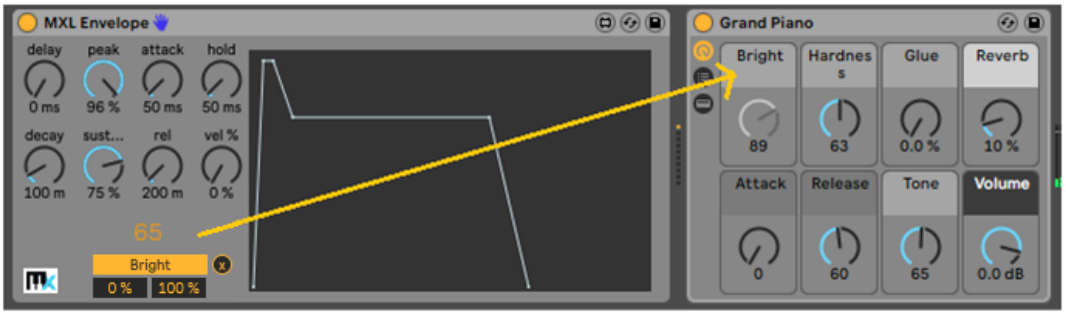
Parameter Mapping

As it happens with most Live devices, fields in MXL devices can be automated and can be controlled from MIDI or the computer keyboard.

In addition to this feature, a few MXL devices – for example MXL Envelope or MXL Pitch Bend - also allow you to control other parameters in Live, in same or different track. You can easily recognize the devices that support this feature because all of them show three yellow-on-black fields:



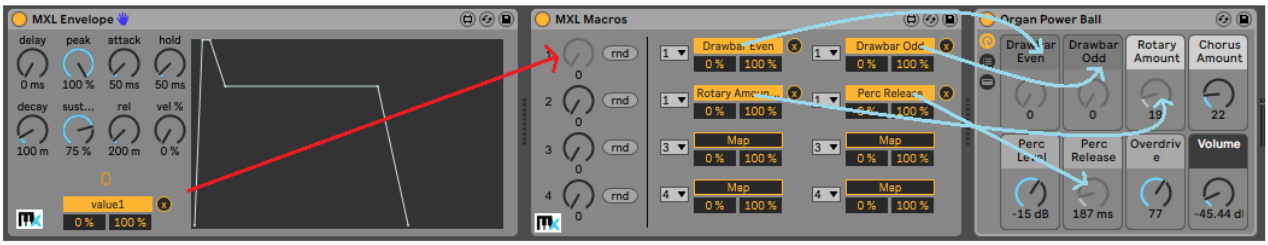
Regardless of the specific device, these fields always work in the same way: you click on the **Map** button, then click on the ***target parameter*** – for example, a dial or a slider – in another device, either in the same or in a different track. When you release the mouse button, the mapping is completed and the **Map** button displays the parameter name:



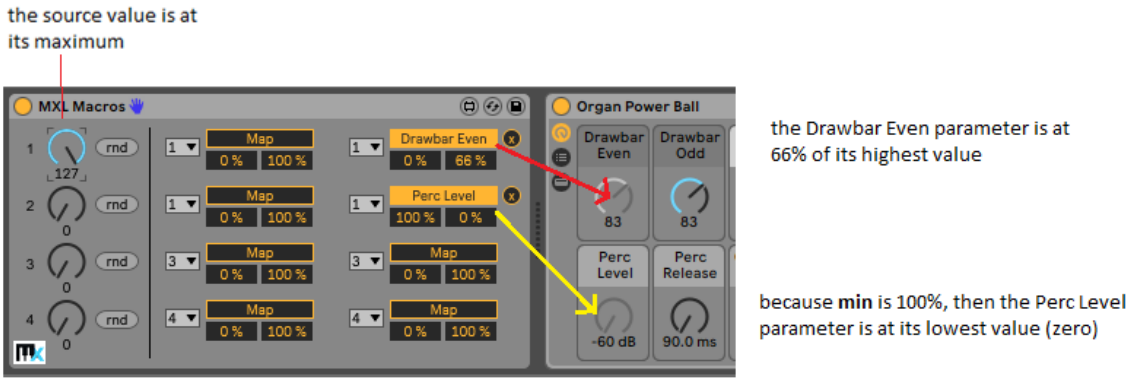
Notice that the target parameter now appears to be disabled, because it can only be operated indirectly through the source parameter. To unmap the parameter, click on the **"X"** circular button.

Apparently, you can map a value to only one target parameter. You can overcome this limitation with the MXL Macros device, which allows you to operate four dial and associate each of them to two or more target parameters (up to eight targets):

Each of the four dials can become the target parameter of another mapping operation, so that you can control up to eight target parameters with a single source value:



In most cases, you can leave the **min** and **max** fields at their default value (0% and 100%), but you might find it useful to reduce the **max** value if you don't want the target parameter to reach it highest value when the source value is at its maximum. You can even set **min** value higher than **max** value, if you want to decrease the target parameter when the source value goes up:



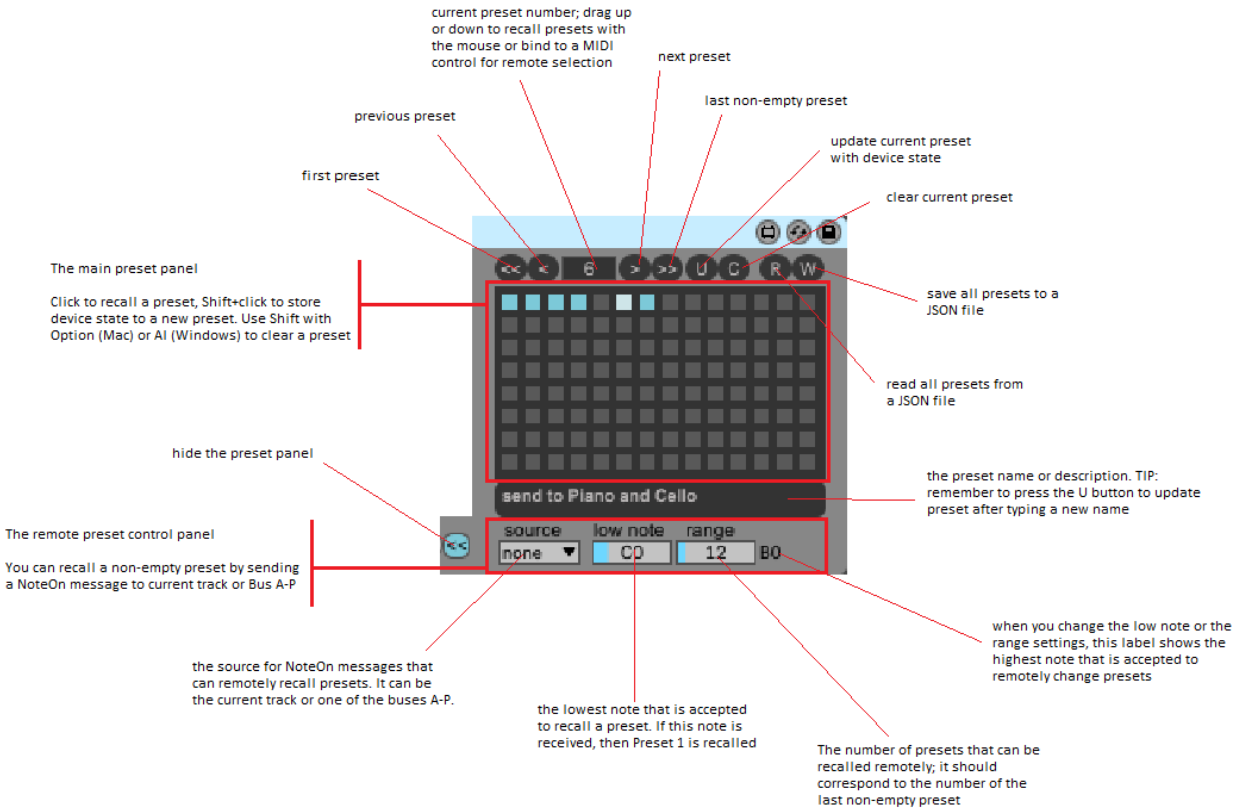
Presets

Many MXL Pack devices support a **local preset** mechanism that allows you to change quickly the value of all fields in the device. This mechanism is similar to Live's native presets, except it is faster and can be controlled in a variety of ways.

The preset panel

All the devices that support local presets have a **>>** button in their lower-right corner. By pressing this button, you expand the device to reveal the local preset panel:

The preset panel has same controls and functionalities regardless of the specific device:



You can select a non-empty preset by clicking on the corresponding gray square in the central area, use **Shift+click** to save the current device state in a preset, and **Shift+Option+click** (on Mac) o **Shift+Alt+click** (on Windows) to clear a preset.

The buttons in the top row allow you to select first, previous, next or last preset, or a preset with given number. These buttons support parameter automation and can therefore be bound to a key, knob or slider on your MIDI controller to remotely select presets.

It is crucial to bear in mind that changes in the device state are ***not*** automatically saved in the current preset. If you recall a preset and change some field values, you have to re-save explicitly to the same preset slot using **Shift+click** or, more simply, by clicking the **U** (update) button. The **C** button to its right clears the current preset and is therefore equivalent to the **Shift+Option+click** (on Mac) o **Shift+Alt+click** (on Windows) action on the current preset.

The **R** (read) and **W** (write) buttons let you save the entire preset group to a JSON file and reload it sometime later. Saving presets to file is the easiest way to share presets among different Live sets.

The field immediately below the main preset panel lets you type a description, so that you can easily remember what each preset does. You can type any text inside it, but remember that you have to update explicitly the current preset, by clicking the **U** button.

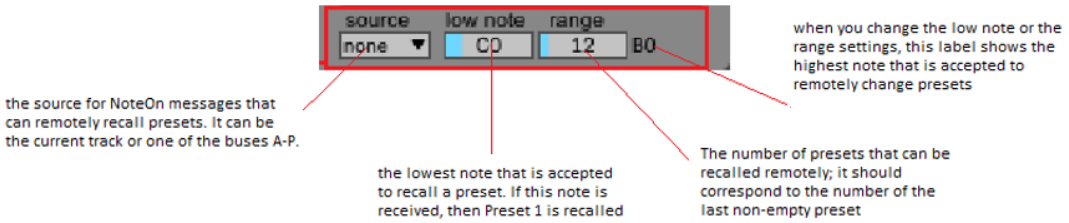
> ***NOTE****: while you can type any description, please bear in mind that the comma has a special meaning inside Max and M4L (it is used as a message separator). For this reason, if you type a comma it is translated into the backslash-comma (\,) sequence to be properly stored. If you don't like this compound character, just avoid using commas inside preset descriptions.*

All the presets you create are saved in the current Live set. In addition, all current presets are correctly copied when you copy or move a device to another track, using copy-and-paste or drag-and-drop. As mentioned above, you can save the current preset group to a JSON file so that you can easily reuse it inside a different Live set or project.

Remote preset recall

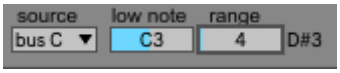
In addition to using the standard Live mapping mechanism to change the current preset number remotely, you can change a preset by sending note to current track or to a bus, using a mechanism similar to the one described in the Remote Control section, earlier in this document.

You can setup remote control method using the fields at the bottom of the preset panel.



This feature is especially interesting for live musicians, because it allows you to recall a preset by just pressing a key on your MIDI keyboard, without having to reach out the mouse.

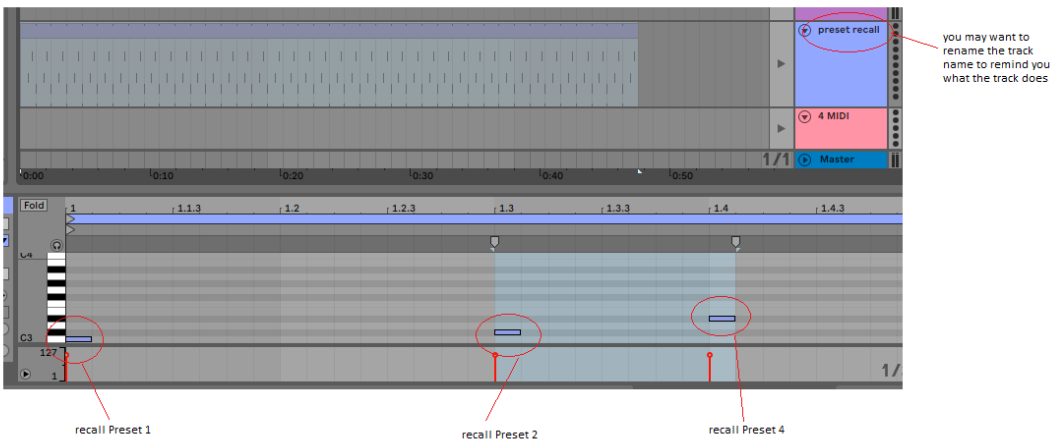
You first need to select a bus in the **source** menu, and then define the proper pitch range using the two fields to its right. The **low note** field specifies the lowest note that is recognized by the remote preset mechanism, whereas the **range** field specifies how many presets can be selected in this way. For example, the following settings allow you to select presets 1-4, using the notes in the range from C2 (middle C) to D#3 when they are received from Bus C:



You can use this mechanism only to select non-empty preset slots. For this reason, it is recommended that you do not leave “holes” in the preset panel and that the **range** field corresponds to the number of non-empty slots. This arrangement allows you to select any non-empty preset by playing a note and – at the same time – it doesn’t intercept more notes than strictly necessary.

The remote preset recalling mechanism is especially useful in a few common scenarios.

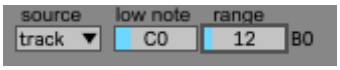
First, if you are playing back a song and want to change presets while the playing head advances, you can reserve a MIDI track for hosting all the notes that trigger the preset recall mechanism at exactly the right instant. In this scenario, the reserved track usually does not contain any instrument, because its notes are not supposed to generate any sound:



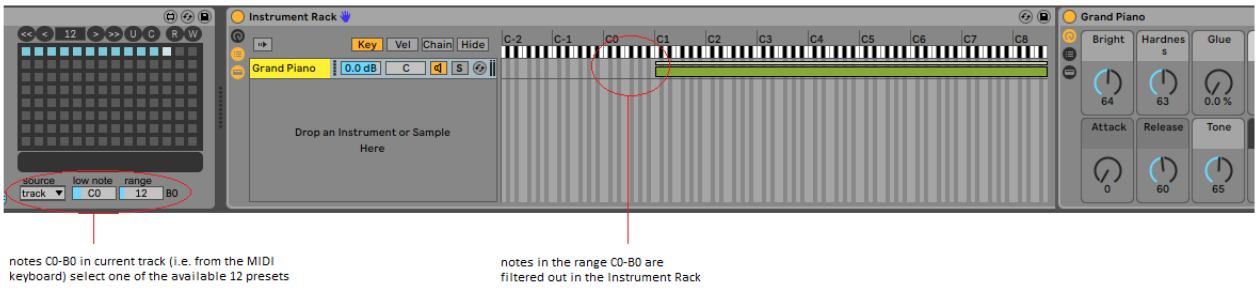
In this scenario, the track contains an MXL Send device that sends those notes to a bus (Bus C in this example). You can disable the **thru** button because these notes do not need to go anywhere else:

The second common scenario can be appealing to live musicians who want to recall presets by following their on-the-spur inspiration. This is especially effective, for example, when using an MXL Scale device to ensure that all played notes fix the current scale, and the current scale changes frequently during the performance.

In such live scenarios, you typically want to recall presets by playing notes on your MIDI keyboard, for example notes in an very low or very high octave, say from C0 to B0. Because you are using the same keyboard to both play your melodies/chords and to select presets, the **source** menu should be set to "track":



A minor problem when using a MIDI keyboard for both playing and selecting presets is that you usually do not want to hear the note used to select the preset (notes in the C0-B0 octave, in this example). You can solve this minor problem by inserting an Instrument Rack just after the MXL device whose presets are remotely controlled, and ensure that the notes used to recall presets are not received by the actual instrument:



Using note filters

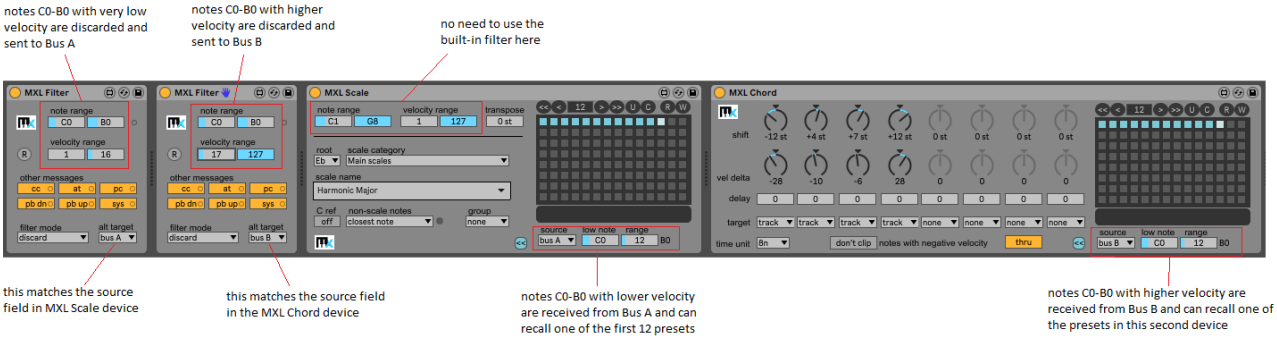
While using an Instrument Rack is fine in most cases, you can usually achieve more flexibility by inserting an MXL Filter before the device.

In the following example, MXL Filter prevents sending notes below C1 to devices to its right; however, these notes are not discarded, instead they are sent to Bus C and therefore are intercepted by the MXL Chord device and used to select one of the presets in the range 1 to 12:

The method based on the MXL Filter device offers several advantages:

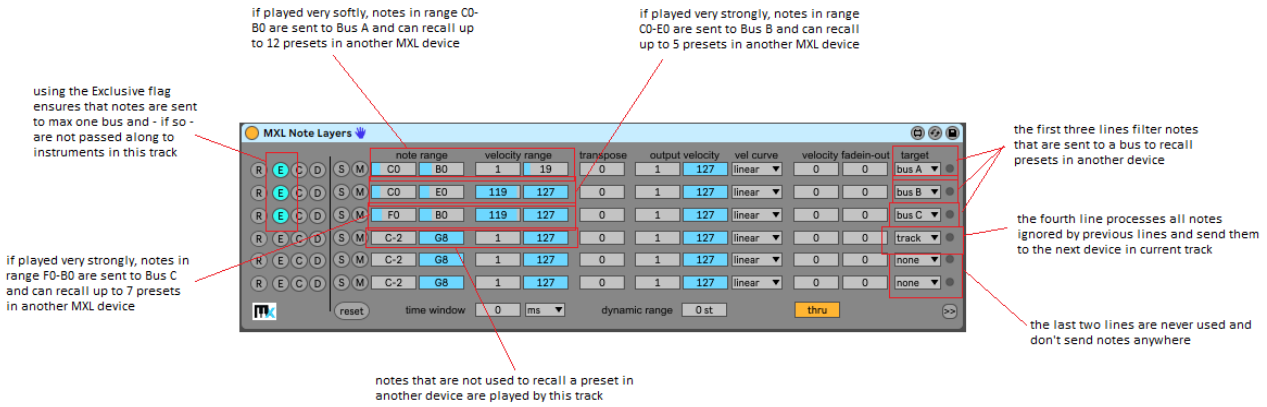
- A single MXL Filter can send notes and recall presets in multiple MXL devices of the ***same*** type, in same or different tracks. For example, if your Live Set contains two or more MXL Chord devices with the same group of presets – possibly loaded from the same JSON file – then one single note in the range C0-B0 selects the same chord for all those devices.
- A single MXL Filter can send notes and recall presets in MXL devices of ***different*** types, in same or different tracks. In other words, you can create multiple "scenarios", each one corresponding to a preset with given number. For example, preset 1 might correspond to "C Major" in MXL Scale and "a triad built on consecutive major thirds" in MXL Chord; preset 2 might correspond to "Eb Melodic Minor" in MXL Scale and "three consecutive perfect fourths" in MXL Chord, and so forth. If both the MXL Scale and the MXL Chord device respond to the same range of notes from the same bus – e.g. notes C1-C#1 from Bus D – then you can switch between the two scenarios by sending a single note to that bus.
- You can use a single MXL Filter device to send notes to a given bus and use multiple devices listening to that bus but with different settings for **low note** and **range** fields, as in the figure below:

- You can use two (or more) XML Filter devices to send the same note range to different buses, depending on the note velocity, as the following example illustrates:



Using note layers

In complex scenarios, you can use a single MXL Note Layers instead of multiple MXL Filter devices. For example, it is possible to send notes in same range – say, from C0 to B0 – to different buses, depending on their velocity or other factors:

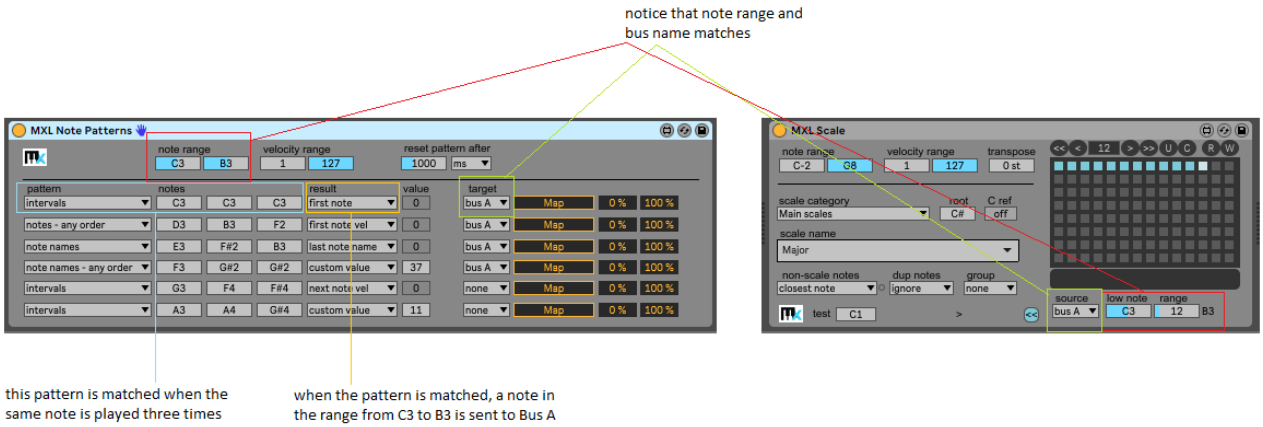


For more details, see the section devoted to the MXL Note Layers device, later in this document.

Using note patterns

The MXL Pack collection offers one more technique for selecting a preset. This approach is based on MXL Note Patterns, which allows you to detect specific patterns in the notes you play on your MIDI keyboard. A pattern is a series of maximum three consecutive notes: for example, it can detect when you play a chord in a specific key (e.g. C Major chord) or even in any key; it can detect when you play the same notes twice or three times in a row, etc.

When a pattern is recognized, you can use one of the notes' attributes – for example, its pitch or its velocity – to affect a remote Live parameter, or to send a given note to a remote bus. This note can be used to change a preset in another MXL device, as in the following example:

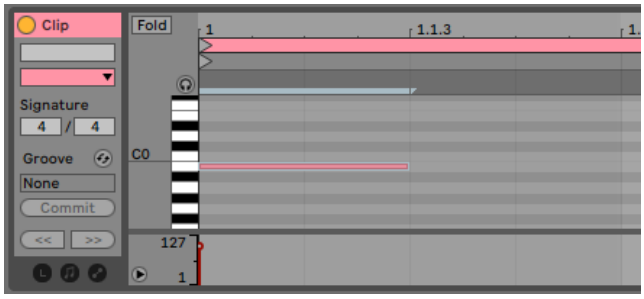


For more details, see the section devoted to the MXL Note Patterns device, later in this document.

Randomizing presets

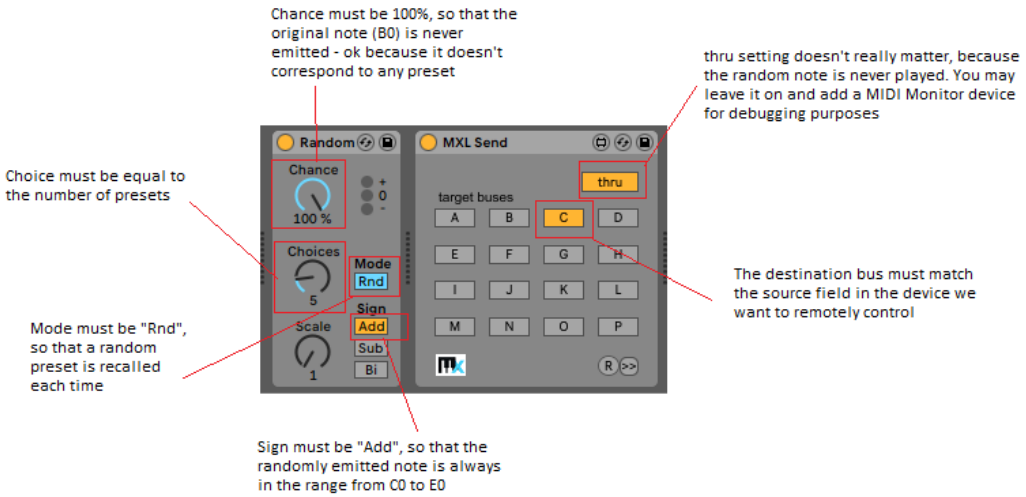
One non-obvious advantage of using notes to recall preset is that you can easily sequence or randomize presets. For example, say that you created five different harmonization in the following **XML Chord** device and you would like to choose randomly one of them every <N> measures.

Here's how to proceed. Create a new MIDI track, which we call the "controlling" track. Inside this track, create a simple clip that contains one single note, whose pitch is B-1:



The duration and the velocity of this note do not really matter, only the pitch does: it must be one semitone *below* the **low note** field in the MXL Chord device (C0 in this example). The length of the clip matters, however, because it dictates how often a new preset is randomly recalled.

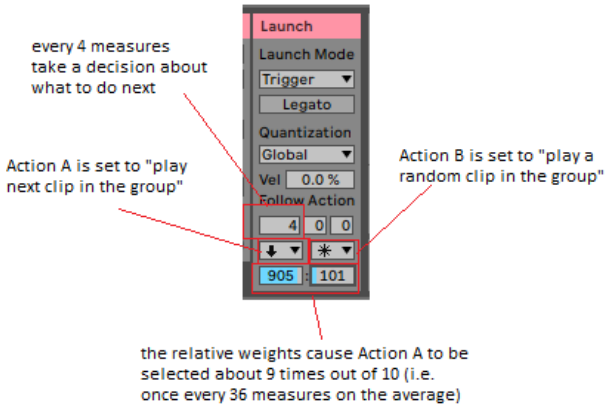
Next, insert a standard Live **Random** device and an MXL Send device in the "controlling" track:



With these settings, the controlling track generates a random note in the range C0-E0 and send this note to Bus C; these notes match the **low note** and **range** fields in the MXL Chord device, and therefore they select a random preset each time the clip plays its first measure.

You can modify and refine this mechanism in many ways. For example, the clip in controlling track can match the **low note** field (instead of being one semitone below) and the **Chance** field in the Random device can be lower than 100 percent. For example, if this field is set to 33%, preset 1 will be randomly selected about twice more frequently than presets 2-5 (whose individual probability is 67/4 = 16.75%)

Another example: you use two or more clips in the “controlling” track and switch among them with Follow Action settings:

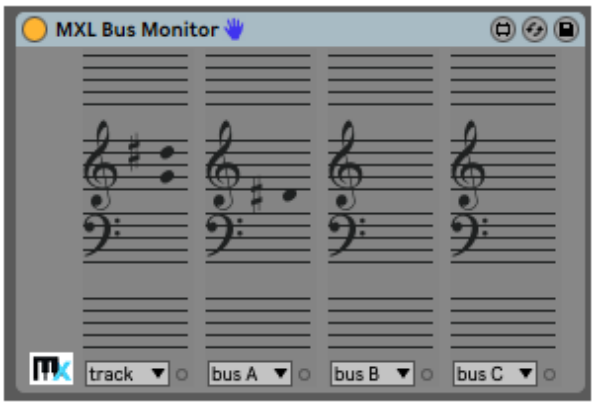


Many other methods for sequencing notes – and indirectly recalling presets in an MXL device – are available. For example, you can use an arpeggiator device, or one of the countless step sequencers that are available for Live.

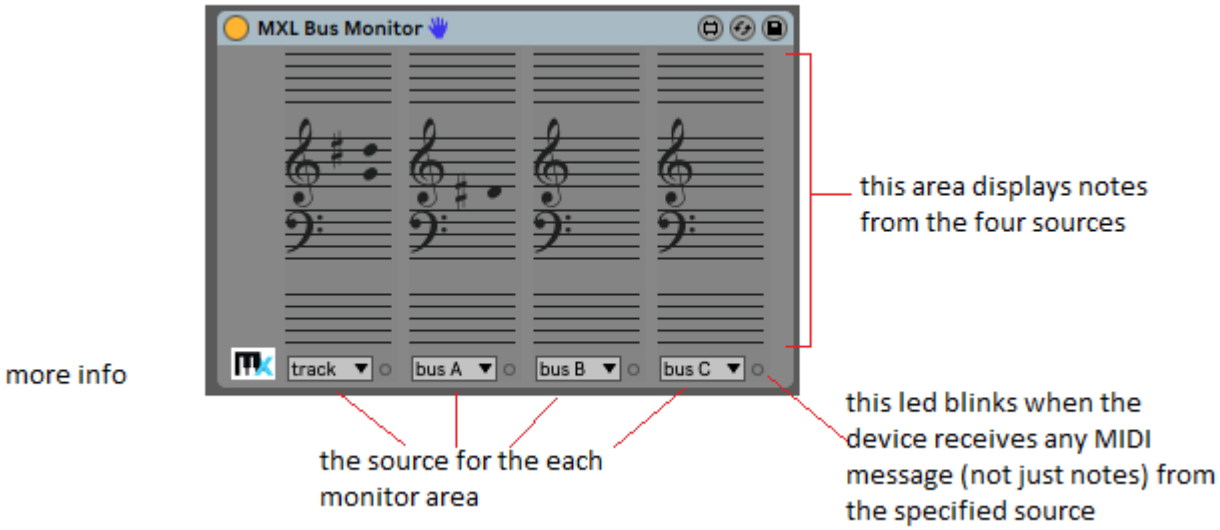
MXL Devices

This section of the documentation describes individual MXL devices.

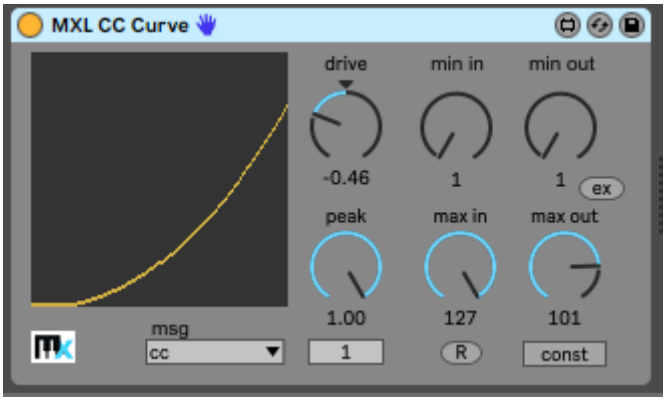
MXL Bus Monitor



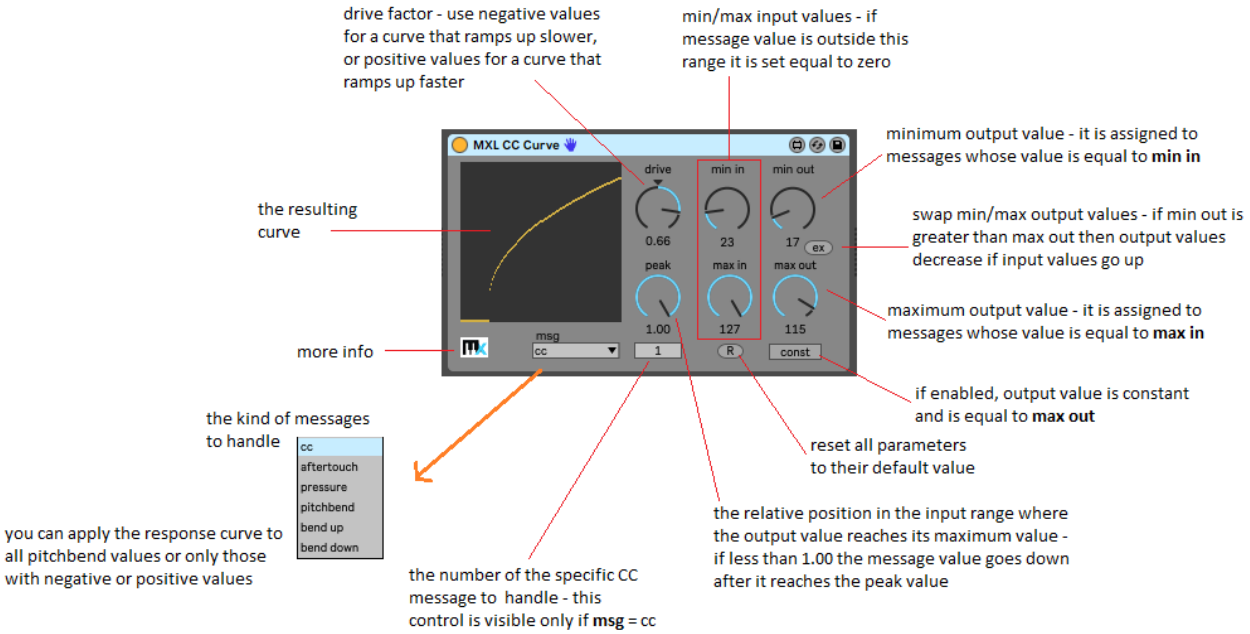
This simple device allows you to monitor notes coming from four different sources (track of a bus A-P). MXL Bus Monitor is especially useful to debug scenarios where multiple buses are involved.



MXL CC Curve



This device allows you to define a response curve for Control Change (CC) messages and other MIDI messages.



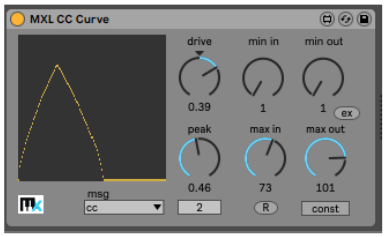
In spite of its name, MXL CC Curve can be used with MIDI messages other than Control Change (CC) messages, namely Aftertouch, Polyphonic Pressure and Pitchbend. It works in an intuitive way, once you understand how to shape the curve to meet your needs. The curve only applies to messages with a nonzero value: messages with zero values are ignored by this device. For examples of how you can use the six knobs, see the section about MXL Velocity.

The **msg** menu offers three ways to process Pitch Bend (PB) messages: you can specify the same curve for all PB values, only for positive values or only for negative values. If you select the **bend up** option then PB messages with negative values are passed verbatim to the track; likewise, if you select the **bend down** option then PB messages with positive values are passed unmodified. Like for other message types, PB messages with zero values (i.e. the pitchbend wheel in rest position) always pass unchanged.

Notice that selecting the **pitchbend** option causes a loss of precision, because most MIDI controllers emit PB messages with 14-bit precision yet output from this device only has 8-bit precision. This problem is slightly less severe when selecting **bend up** or **bend down** options, because input data has 13-bit precision (i.e values from 0 to +/-8191).

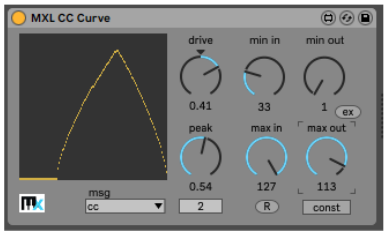
The MXL CC Curve device can be useful when you want different tracks to respond differently to the same MIDI message. For example, say you are sending MIDI from a breath controller to three tracks that host three different instruments, and you want the three instrument to fade-in and fade-out depending on how hard you blow. Here's how you can arrange the three tracks:

track 1



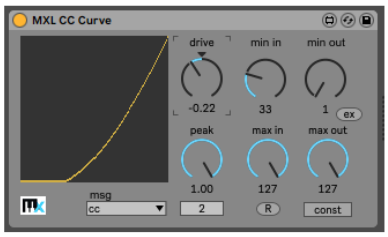
this instrument fades in quickly even if you blow softly, reaches its peak when CC2 is about 1/4 of its range, then quickly fades out and disappears when you are in the middle range

track 2



instrument 2 fades-in when instrument 1 reaches its peak and continues to increase its volume at about 2/3 of the full range, then quick fades-out so that its level is zero when the CC2 value is highest (127)

track 3



instrument 3 fades-in when instrument is already fading-out, and initially grows slower (**drive** is negative). It then ramps up more quickly and reaches the peak when you blow very hard. When this happens the other two instruments have already faded-out completely

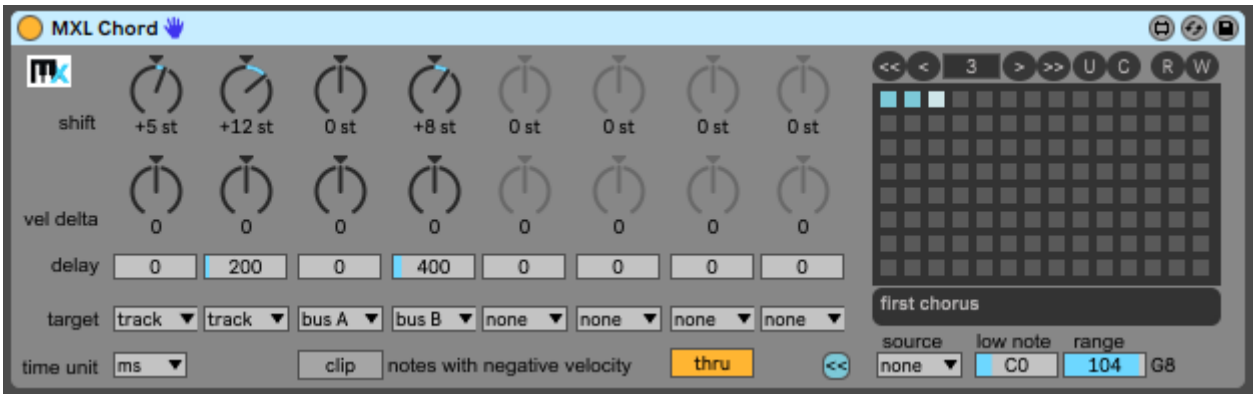
MXL CC Filter

This device can block some common Control Change MIDI messages – namely modulation wheel (CC1), breath (CC2), foot controller (CC4), channel volume (CC7), pan (CC10), expression (CC11), sustain (CC64) and portamento on/off (CC65) – plus two CC messages with any specified number. All other messages flow unchanged.

Notice that switching off one of the messages that support high-resolution 14-values disables both the MSB and the LSB control change message. For example, switching off the **volume** button blocks both CC7 and CC39 messages.

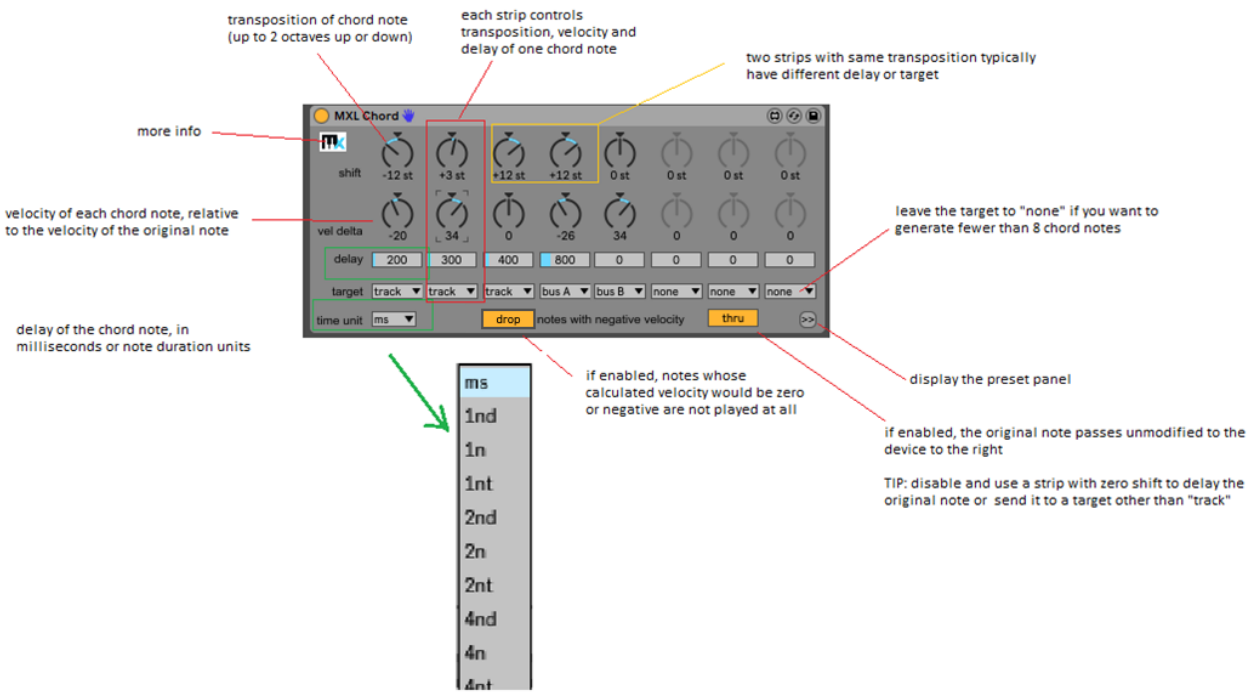
This device has several uses; arguably, the most interesting ones is the ability to enable the sustain pedal only for selected note ranges, when you use keyboard splitting or a device such as MXL Note Layers.

MXL Chord



This device intercepts incoming notes and generates chords of up to eight notes. Unlike Live's Chord device, you can control the velocity, the delay and the destination of each note of the chord.

This device offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.



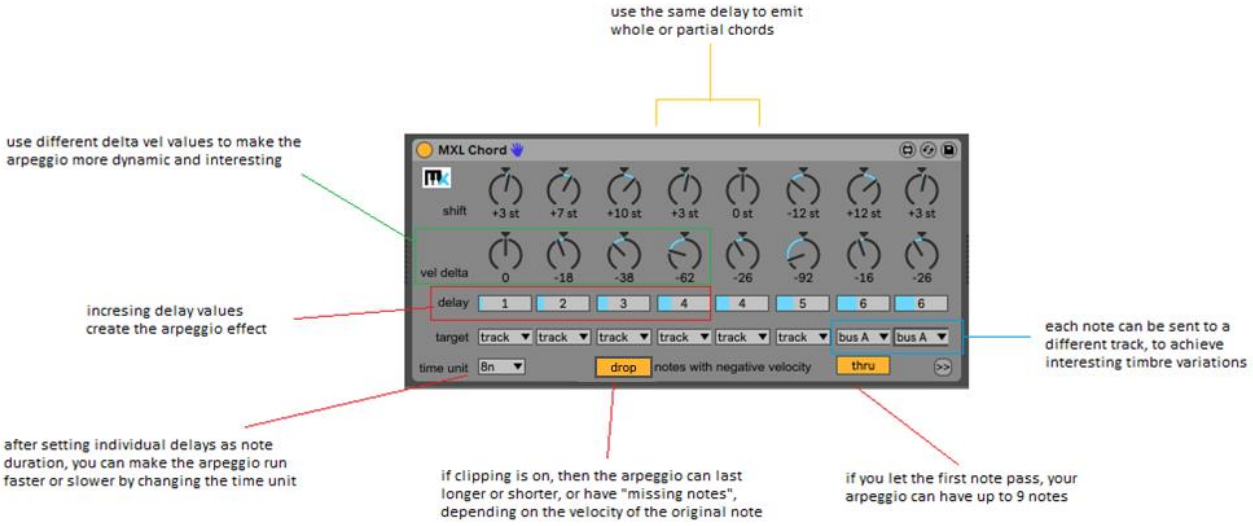
When you first load the device, all **target** fields are set to "none" by default: your first action is deciding how many chord notes you want to generate and change their destination either to "track" or to one of the A-P buses. If you want to send the same chord note to two destinations, you have to configure two vertical strips with same **shift** value but different target.

After setting the desired number of semitones (max two octaves up or down), you can use the **vel delta** dial to increase or decrease the chord note's velocity. If you use a negative value and the resulting velocity would be zero or negative – for example, if you set the value -90 and the velocity of the incoming note is 80 – then the **clip/drop** switch near the bottom edge comes into play. If this switch is enabled ("drop") then the note is not played at all; if the switch is disabled ("clip") these notes are emitted with velocity set to 1.

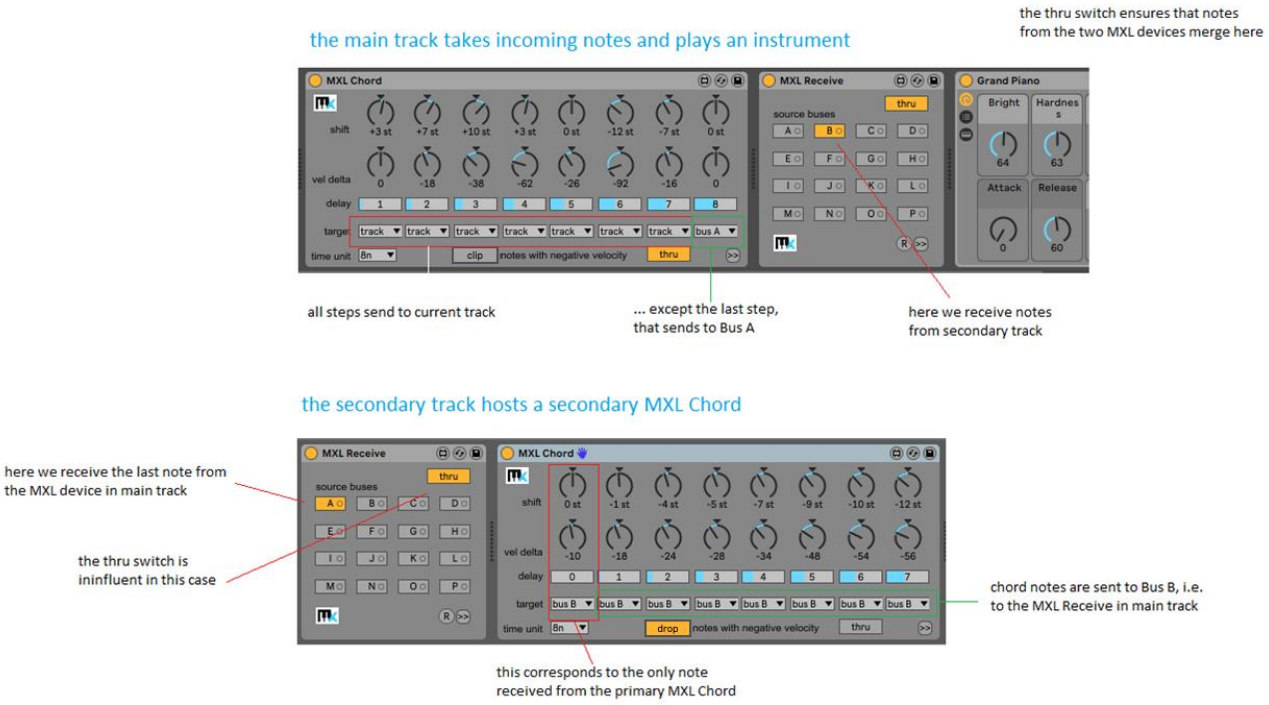
The **delay** field in each vertical strip allows you to select the delay of each note, where the measurement unit is selected the **time unit** menu: it can be **ms** (milliseconds) or a note duration symbol (e.g. **4n** for a quarternote).

All these features provide a lot of flexibility and allow you to use MXL Chord as a replacement for many other devices, such as MIDI delays, arpeggiators or even steps sequencers:

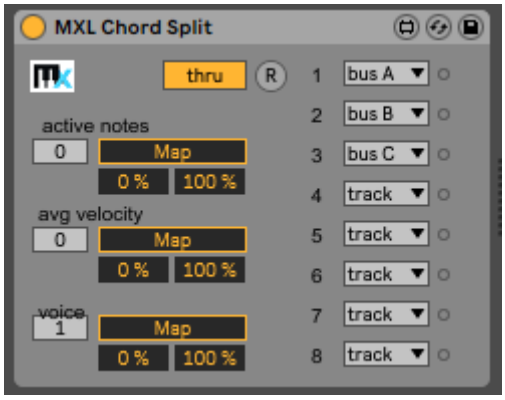
- change notes' "timbre" by sending the same note (in same or different octaves) to different targets, where each target corresponds to a different instrument
- create a MIDI delay effect by sending the same note to same target but with different delays (and maybe decreasing velocities to simulate the echo effect)
- create interesting effects by sending delayed notes to different buses, where each bus corresponds to slight variations of the same instrument, optionally processed in different ways (for example, with different chorus or reverb settings)
- create arpeggios by transforming a single note into a chord and sending each note with a different delay and different velocity settings:



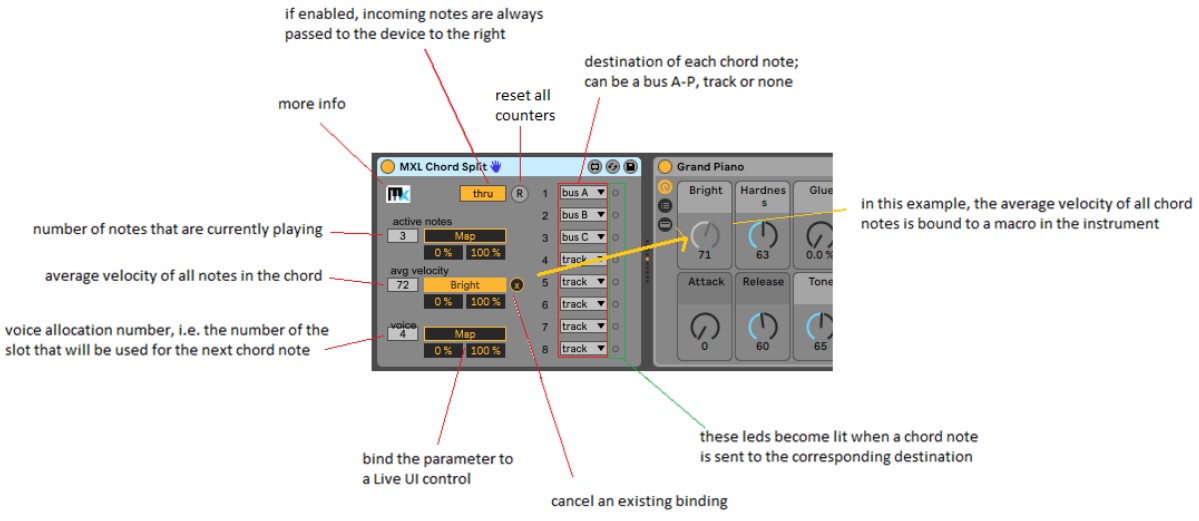
The main purpose of MXL Chord is emitting chords rather than compete with more sophisticated arpeggiators or step sequencers. However, the ability to send to different sources and clip low-velocity notes can be intriguing in some scenarios. Moreover, you can combine two MXL Chord devices for extra power. For example, you can overcome the 8-note limit by "chaining" two devices in different tracks:



MXL Chord Split



This device allows you to send individual notes of a chord to different destinations. Moreover, you can map some of its parameters to another Live's control, in same or different track:



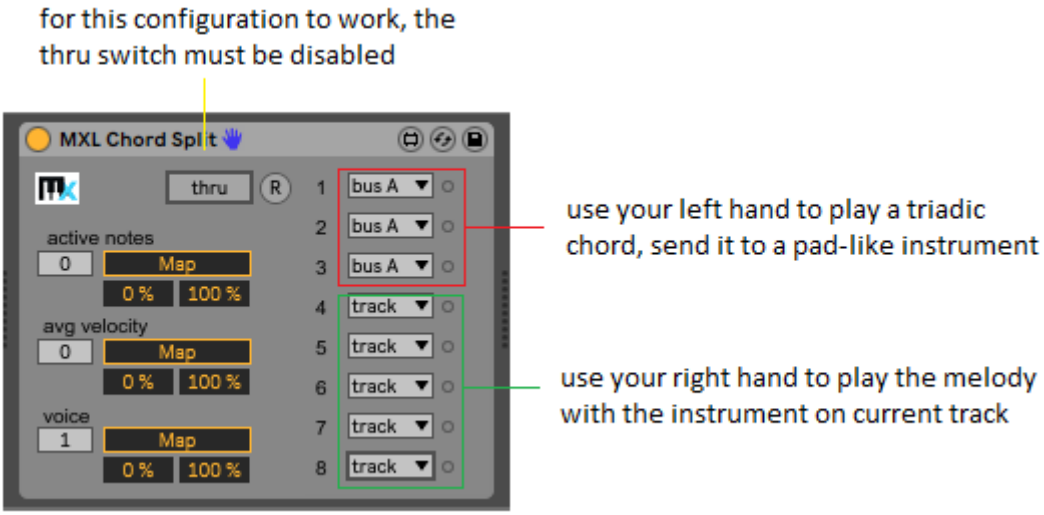
The device uses a simple voice-allocation algorithm: the first incoming note goes to target 1, the second note goes to target 2, etc. When the NoteOff message arrives, the note is silenced and the corresponding voice is made available to the next incoming note.

The ability to send individual chord notes to different buses allows to you implement some interesting tricks, especially during live perfomances. For example, you can mimic a string quartet, by sending the first note to a double bass, the second note to a cello, and all other notes to a virtual violin. Same for a woodwind quartet.

Another simple application of this device is sending only the first chord note to a bus that applies some specific audio effect. For example, you might route only the first note so that it is doubled at the lower octave, or is echoed, etc.

The MXL Chord Split device becomes especially useful when working with monophonic instruments and VSTs. By sending each chord voice to a different VST instance, you can effectively work around the single-voice limitation of those virtual instruments.

Here is a configuration that allows you to play two (or more) different instruments with your MIDI keyboard, without splitting the keyboard in multiple regions. More simply, the first three notes play a chord with one timbre and – while the chord is playing – any additional note is played with a different instrument:



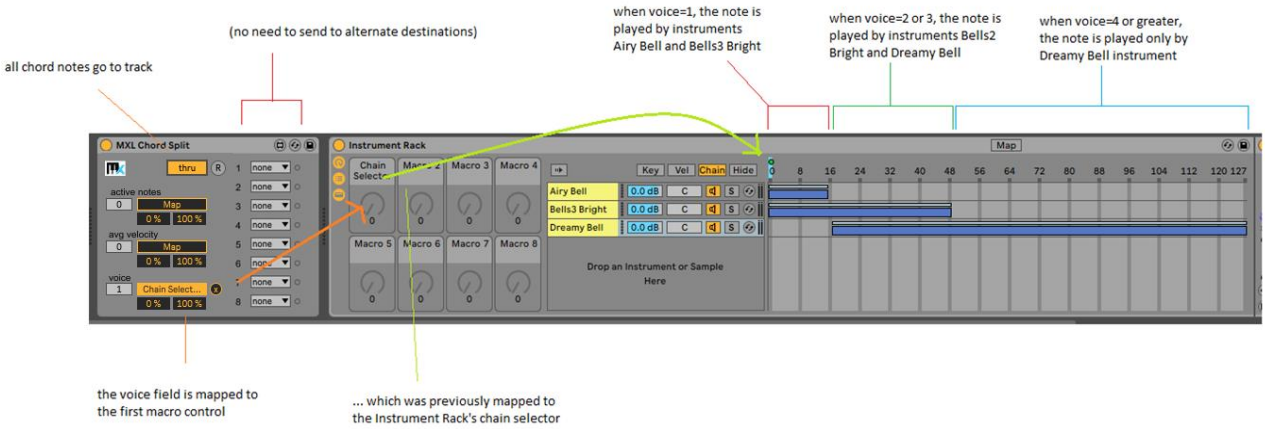
The MXL Chord Split device gives you even more flexibility thanks to its ability to map its fields to a Live parameter. You enable this feature by simply click on the corresponding **Map** button and then click on any Live element, in same or different track. The name of the corresponding parameter then replaces the "Map" caption and an "x" button to the right allows you to delete the mapping.

The **active notes** and **voice** fields can be a number between 0 and 8. When these fields are mapped to a Live element, their value is automatically scaled to the range 0-127 – in practice, the field value is multiplied by 16. When the **avg velocity** field is mapped, its raw value is used because it is already in the range 1-127 and goes down to zero when no note is playing. You can also manipulate these values using the **min-max** percentages just below the Map buttons.

Mapping these values to a Live parameter provides a degree of flexibility that you can hardly obtain otherwise. For example, several M4L devices offer the ability to map note velocity to a Live parameter, thus you can control the sound or the effect applied to incoming notes. However, if the track is polyphonic, having the sound of all chord notes be affected by the velocity of the last played note rarely makes sense,

and using the **avg velocity** value is often a better choice. The same consideration applies when using the **active notes** value.

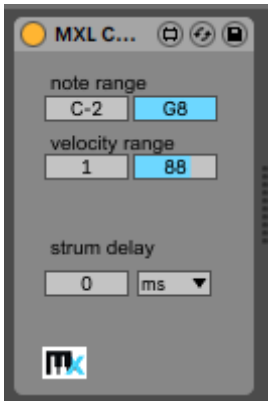
Because **active notes** and **voice** fields are automatically scaled to the 0-127 range, they can be easily mapped to most Live parameters. For example, you can map them to the Chain Selector parameter of a Midi Rack or Instrument Rack, as in the following example:



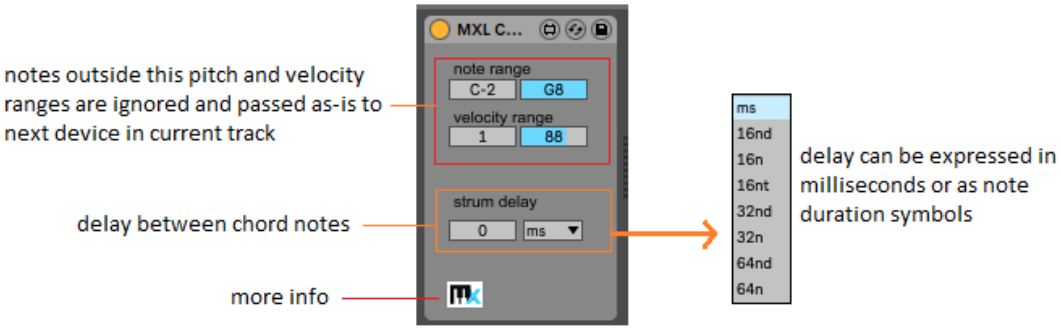
Notice that mapping the **active notes** field or the **voice** field to the Chain Selector of a rack provides slightly different behavior if you play a chord and then lift one or more fingers to play other notes. Try it out yourself to learn more about these subtleties.

For more information, read the Parameter Mapping section, earlier in this manual.

MXL Chord Strum

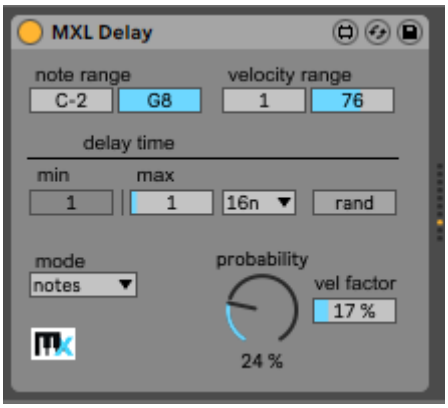


This simple device “strums” incoming chord, to simulate fast one-time arpeggios or guitar strumming. You can specify which chords should be strummed (in terms of pitch and velocity range) and you can specify the delay between chord notes.

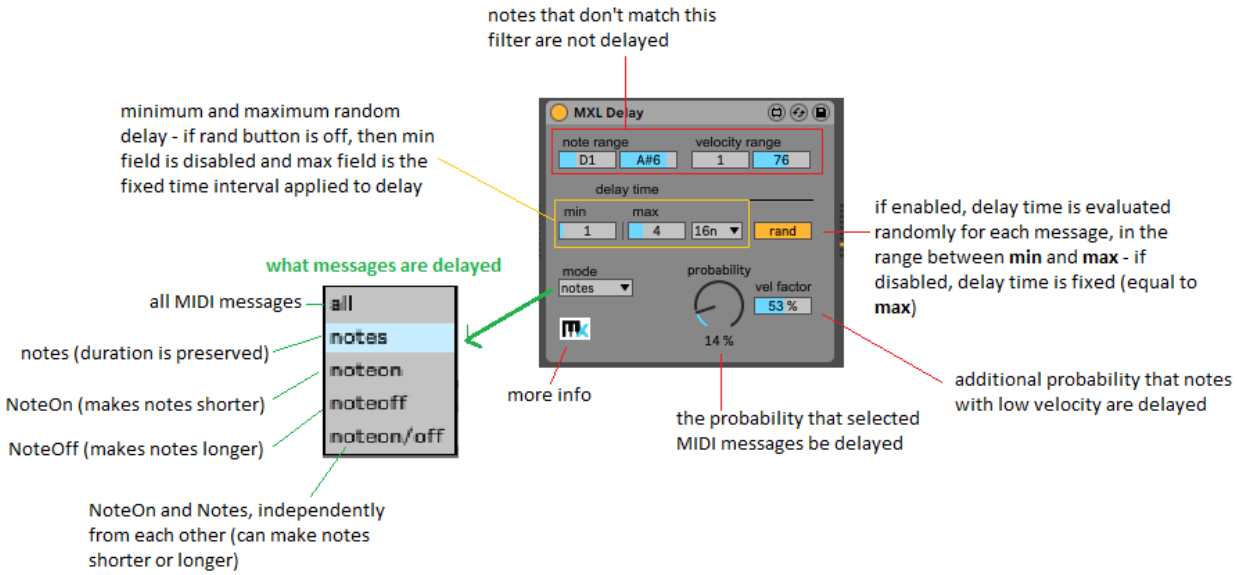


Regardless of the unit selected in rightmost menu – milliseconds or a note duration symbol – you can select zero in the leftmost field to disable strumming.

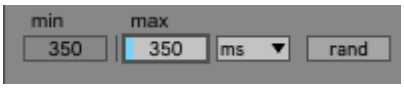
MXL Delay



This device can delay MIDI messages, either by a specified time interval or randomly in a range. It can apply to all messages or only to notes, and you decide whether notes preserve their duration or should become longer or shorter.



To begin with, you use the **rand** button to decide whether you want a fixed or random delay time. In the latter case, you specify the random range using the **min** and **max** fields, and select the proper time units (which can be milliseconds or a note duration such as quartertones, 8ths, 16ths, etc.); in the latter case, the **min** field is disabled and you use the **max** field to select the fixed delay:



When **mode** menu is set to **all**, every MIDI message is delayed independently from each other, except NoteOn and NoteOff messages for the same note are delayed by the same time interval, so that the note preserves its duration. When mode is set to **notes**, NoteOn and NoteOff messages are always delayed by the same time interval (even if **rand** is enabled), so that notes are delayed but preserve their original duration.

When mode is set to **noteon** only the note attack is delayed, therefore note duration is shorter; conversely, when mode is set to **noteoff**, note release is delayed but note attack isn't, therefore note duration is longer.

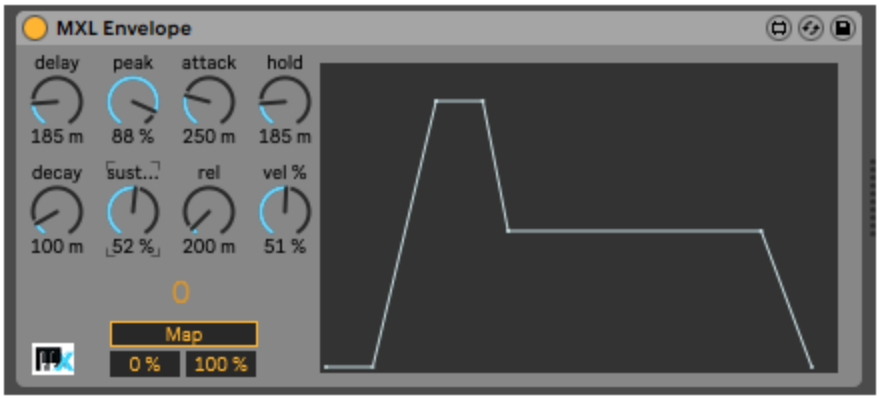
Finally, when mode is set to **noteon/off** both attach and released are delayed and the effect depends on whether **rand** setting. If disabled, the same delay is applied to both NoteOn and NoteOff messages, therefore the result is the same as **mode** = **notes**. If enabled, NoteOn and NoteOff messages are delayed by a random (and possibly different) time, therefore the resulting note can be shorter or longer than the original note.

Notice that in two cases – namely (1) mode is set **noteon** or (2) mode is set to **noteon/off** and **rand** is enabled – it may happen that the delayed NoteOn message would be emitted ***after*** the corresponding NoteOff message, which would produce a "stuck" note. In such cases, MXL Delay detects the problem and doesn't play the note at all.

Two fields control the probability of MIDI messages being delayed. The **probability** dial applies to all messages and is self-explanatory; the **vel factor** field only applies to NoteOn messages and specifies the additional probability that notes with lowest velocity (=1) are delayed. Let's make a practical example.

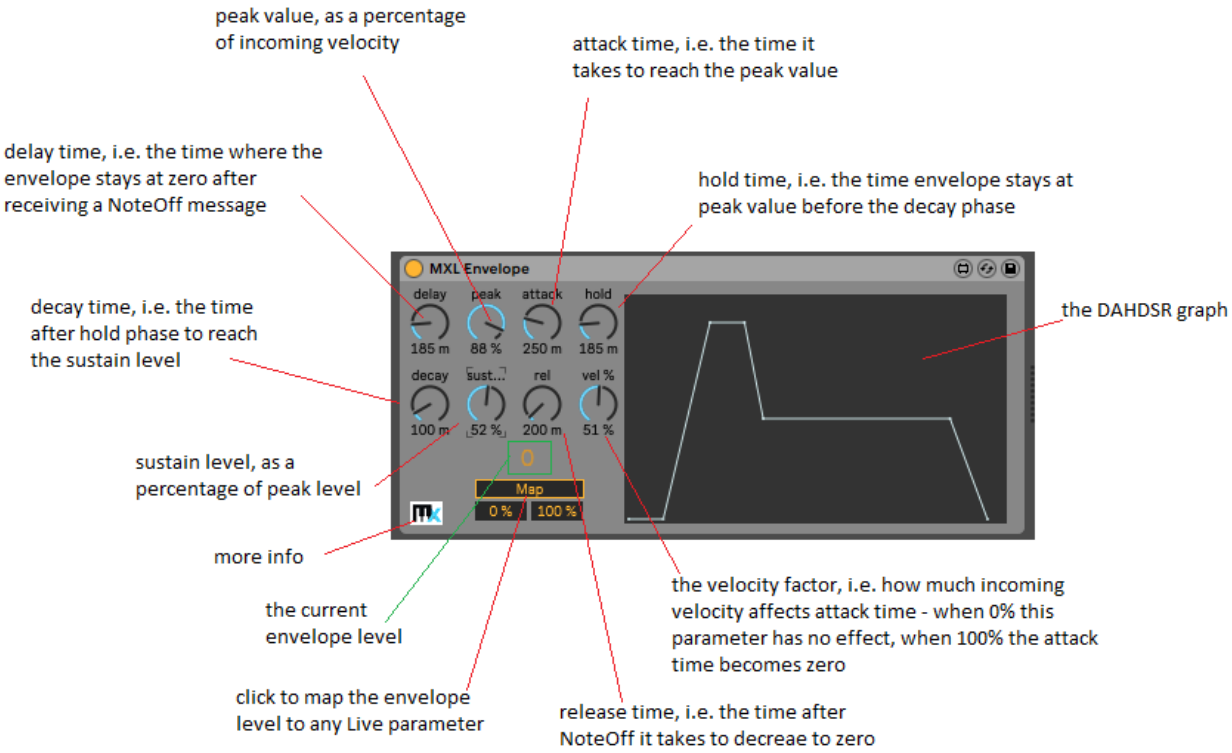
Say that **probability** is set to 20% and **vel factor** is set to 50%: this means that 1/5 of all MIDI messages (on the average) are delayed, and this probability applies also to notes with maximum velocity (=127), whereas notes with minimum velocity (=1) are delayed by 20% + (100-20)*50% = 60%. To evaluate probability of notes with velocity inside the min-max range the device interpolates between these values. For example, for notes with velocity = 64 (median value between 1 and 127), the delay probability is 40% (median between 20% and 60%).

MXL Envelope

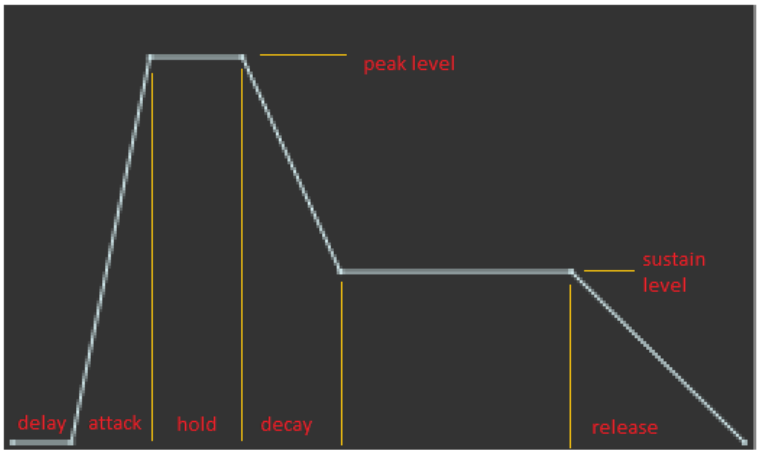


This device generates a DAHDSR envelope curve any time it detects an incoming note and allows you to map the envelope value to a Live parameter, in same or different track. DAHDSR is an envelope shape richer than the more common ADSR, and stands for

- **Delay**: delay after note inception
- **Attack**: time to reach the peak value
- **Hold**: time interval when the envelope stays at peak value, before starting the decay phase
- **Decay**: time to reach the sustain level
- **Sustain**: the value where the envelope stays while the note is still playing (as a percentage of peak value)
- **Release**: the time it takes to reach zero after the note key is released

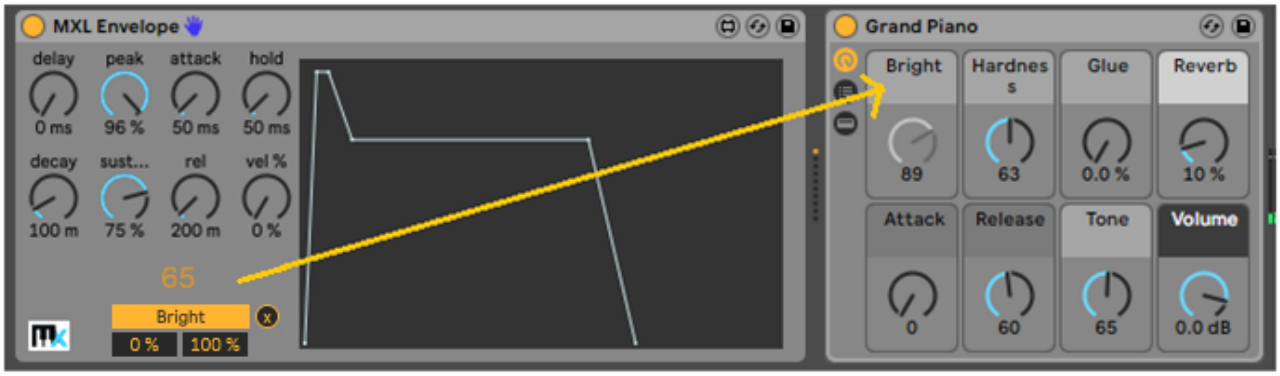


The following figure illustrates the relations between these parameters:



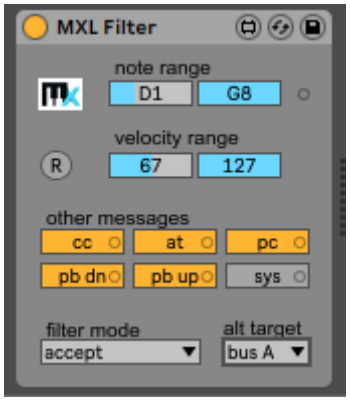
The **velocity factor** provides a way to make the attack time shorter than it should when the NoteOn velocity is higher. By default, this parameter is zero and velocity doesn't affect attack time; if nonzero, the parameter specifies how much shorter the attack time becomes when velocity is highest (127). For example, if **attack time** is 200 ms and **velocity factor** is 50%, then the actual attack time for a note with velocity equal to 127 decreases by 50% and becomes 100 ms; if velocity is equal to 89 (roughly 70% of its maximum value) then attack time is reduced by 50% * 70% = 35% and becomes 130 ms.

The only purpose of the MXL Envelope device is mapping the envelope value to a Live parameter, which you do by clicking on the **Map** button. To unmap the parameter, click on the "x" button:

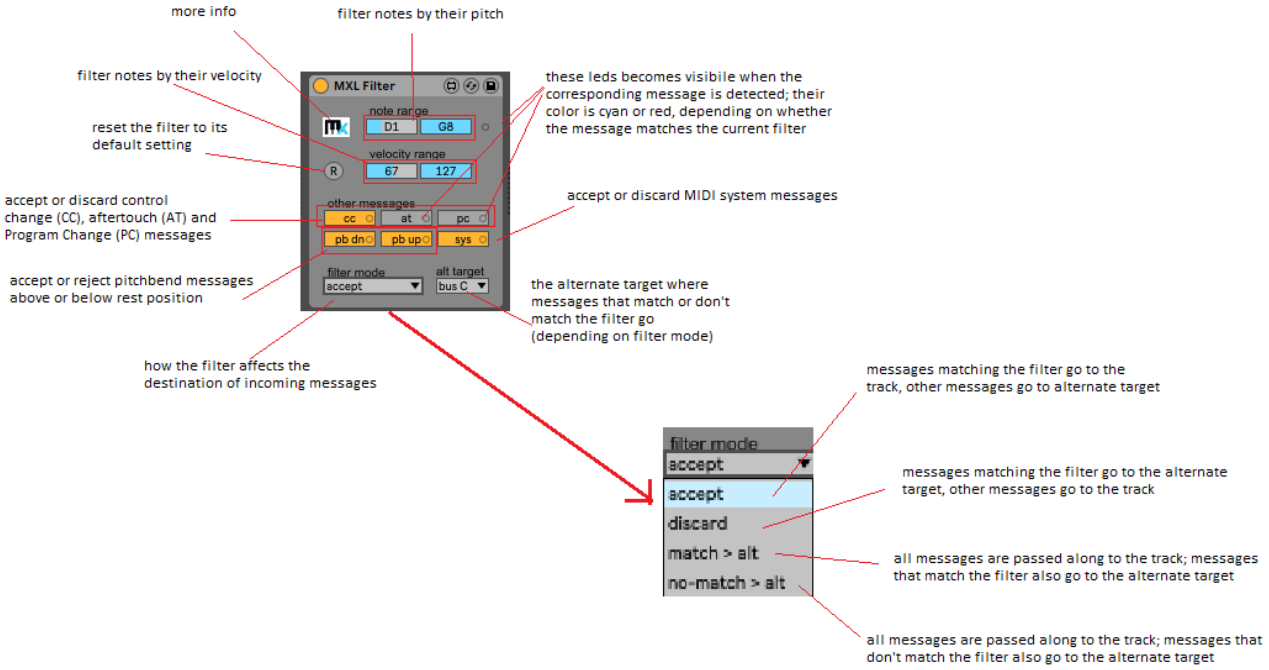


For more information, read the Parameter Mapping section, earlier in this manual.

MXL Filter



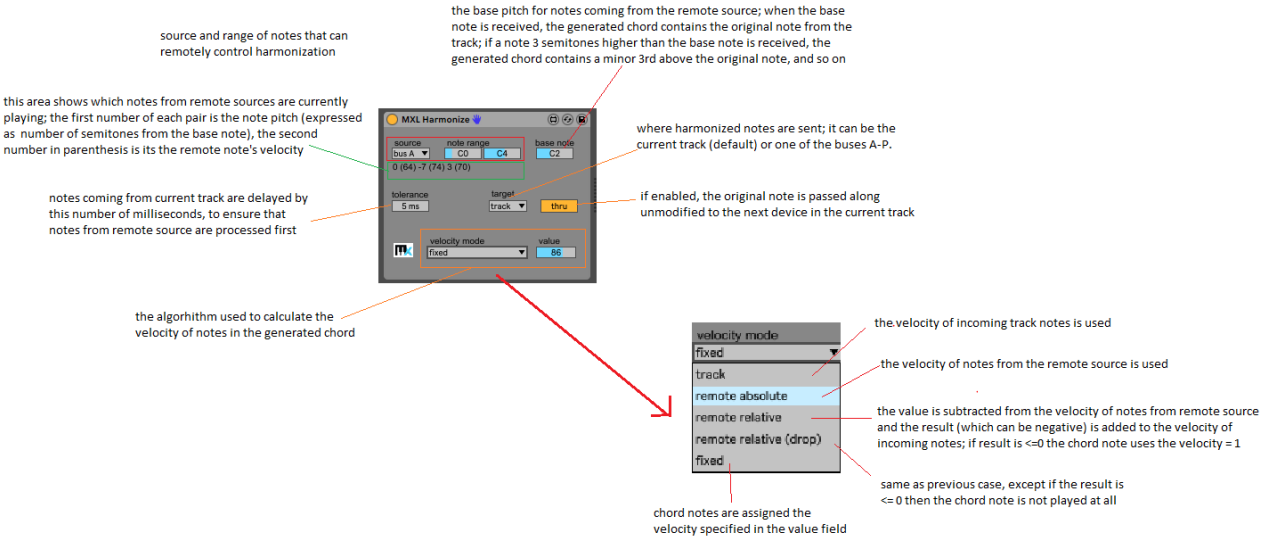
This device provides a simple and intuitive way to filter out notes or other MIDI messages, and optionally send discarded messages to one XML Receive device, through one of the buses A-P. You can filter notes by their pitch and velocity, and decide what to do with Control Change (CC), Aftertouch (AT), Program Change (PC), pitch bend (PB) and system messages. You can even send pitch bend up and down messages to different destinations:



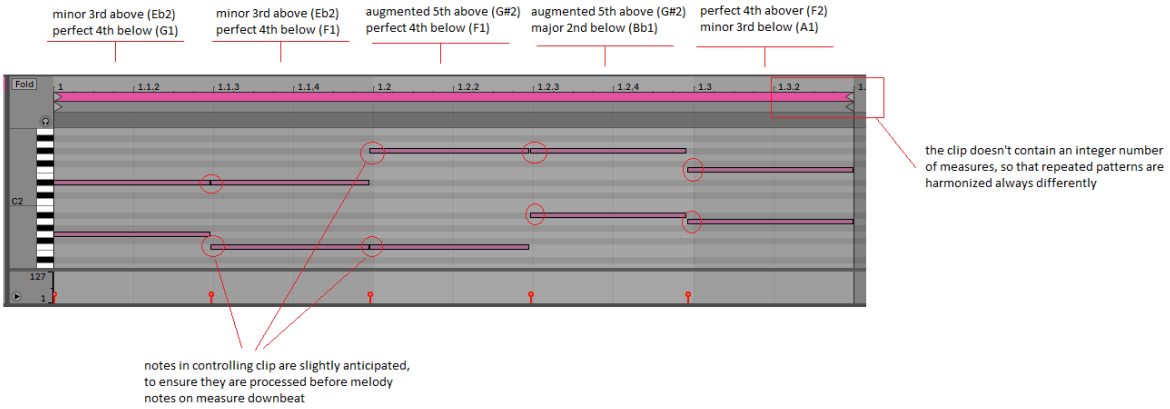
See the section “Presets – Using note filters” for more uses of the MXL Filter device.

MXL Harmonize

This device allows you to harmonize incoming notes using a chord voicing that reflects the notes received from the specified **source**. It can be considered as an expanded version of MXL Transpose, in that it can create two or more intervals and gives you more control on the velocity of outgoing notes:



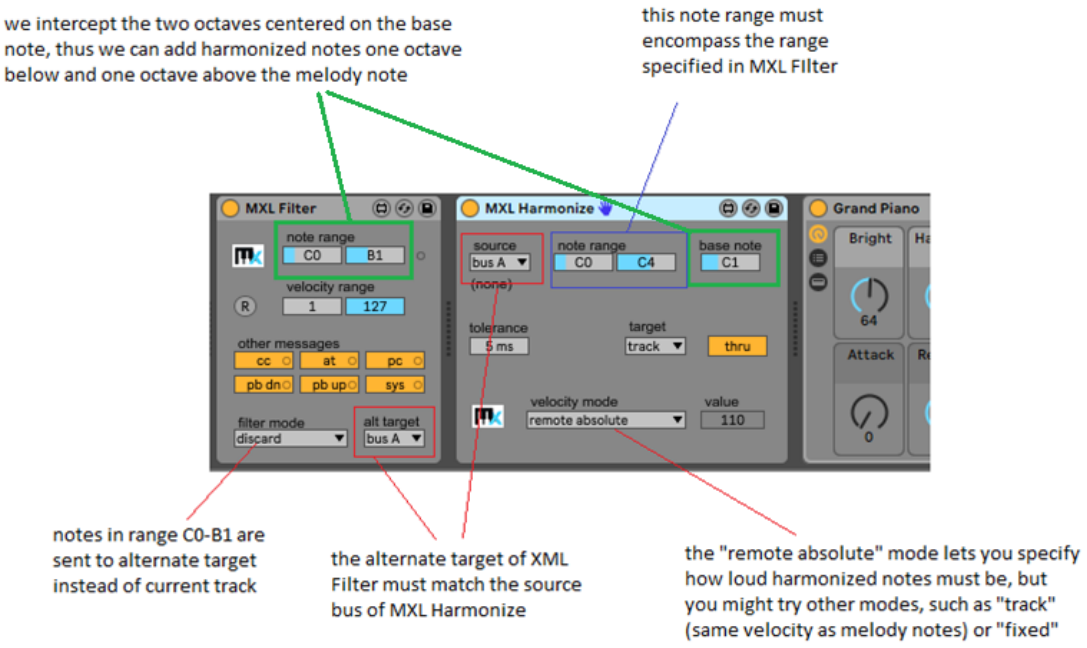
Depending on what sources you use for the main melody (track) and the remote sources, the MXL Harmonize device can prove useful in many scenarios. For example, you can use pre-recorded clips to set up the harmonization of a melody that you play on your MIDI keyword, as if you were using many fingers instead of just one:



The previous example illustrates a simple trick: all notes in the clip (except those at the very beginning) are slightly anticipated, to ensure that MXL Harmonize processes them before melody notes that are played right on the downbeat. Also, notice that the clip doesn't contain an integer number of measures, thus repeated melodic patterns in the main track will be harmonized in different and somewhat unpredictable ways.

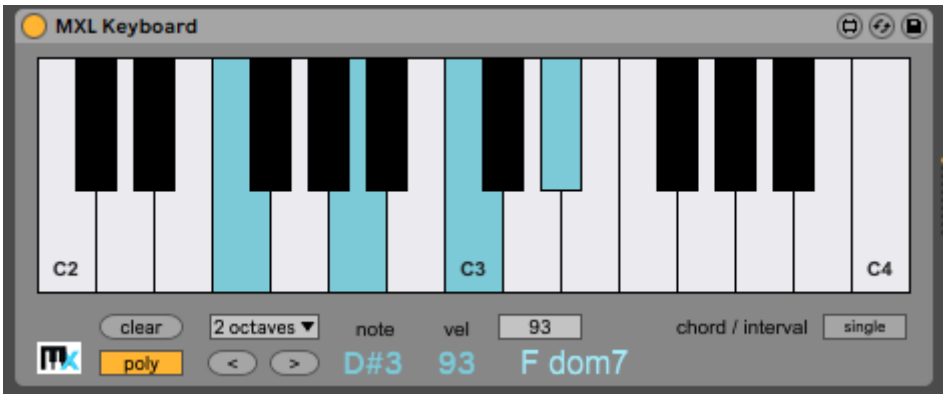
You can achieve interesting results by reversing roles, and have pre-recorded clips to provide the main melody, while you harmonize it in different ways by hitting keys on your MIDI keyboard.

If you are performing live and want full control on all harmonization parameters, you can split your keyboard and use one note range to play the melody and the other range to establish how the melody must be harmonized. In this case, you might need to insert an MXL Filter device just before MXL Harmonize, to "grab" notes that are intended to control chord voicing:

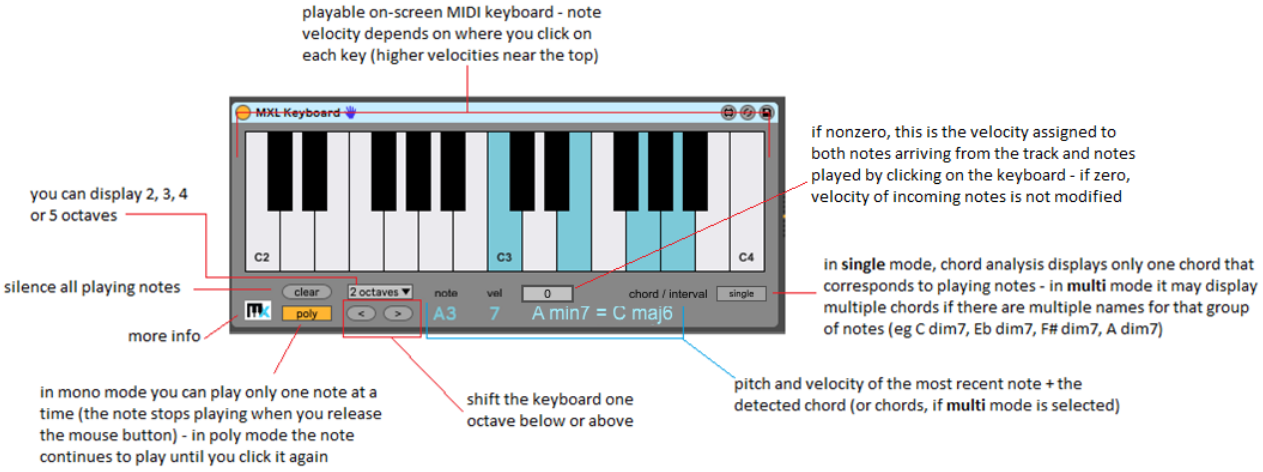


When not performing live, you typically use one clip for the main melody and another clip to provide the harmonization; this is the least interesting way of using the MXL Harmonize because it generates repetitive patterns. However, you can make this scenario less predictable by using clips of different length, so that the pattern repeats itself with a longer period.

MXL Keyboard



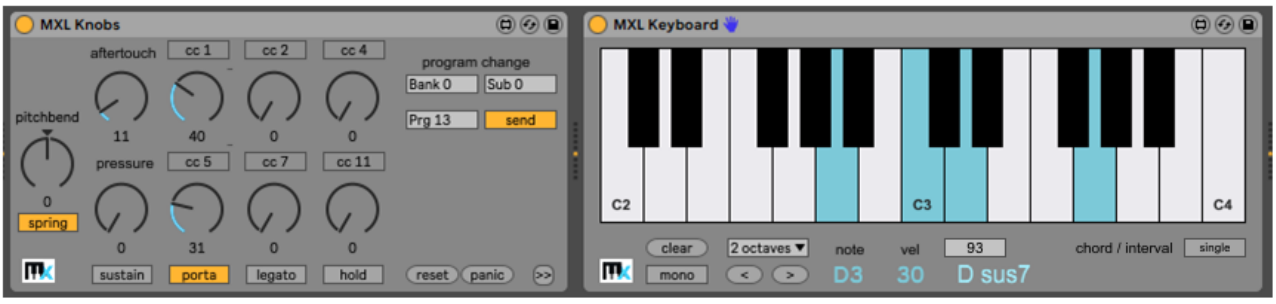
This device works as both a MIDI Monitor and a playable on-screen MIDI keyboard. It is similar to other Max-for-Live devices, but has some added features, such as the ability to display from 2 to 5 octaves and the best chord detection utility you can find anywhere.



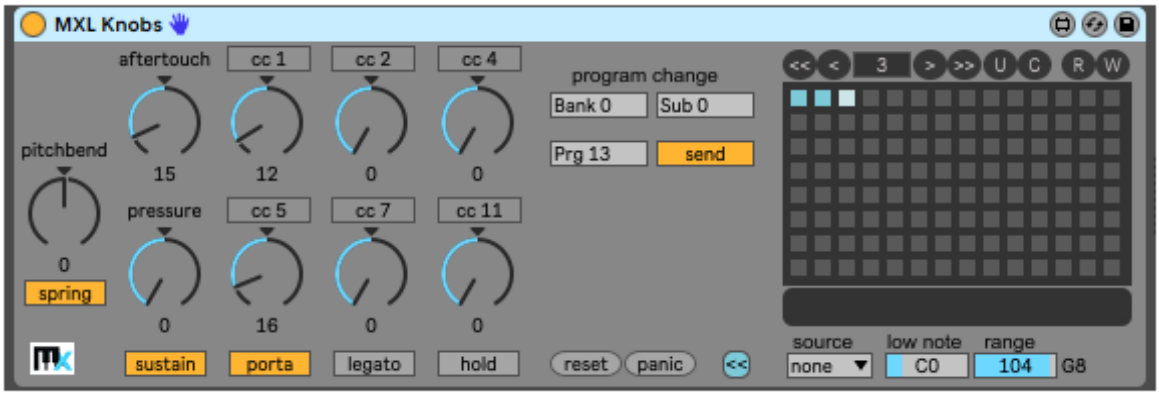
The chord detection mechanism identifies the interval between two notes as well as virtually any chord formed by incoming notes, from simple triads up to chords with added 7th, 9th, 11th and 13th, in regular or altered form. For example, it can detect “monster” chords such as C dom7/b9/#11/b13 or F# sus7/9/#11. It correctly identifies chords even when their third or fifth is missing.

In **single** mode, MXL Keyboard always displays one single chord at most, even if the notes that are currently playing can be interpreted in more than one way, as is the case of the C-E-G-A notes that form both Amin7 and Cmaj6 chords, or augmented triads such as C-E-G# that can be spelled as Caug, Eaug or G#aug. To display all name variants of current chord, enable the **multi** mode.

A final tip: MXL Keyboard works beautifully with MXL Knobs to provide all the controls you need to test your Live instruments.

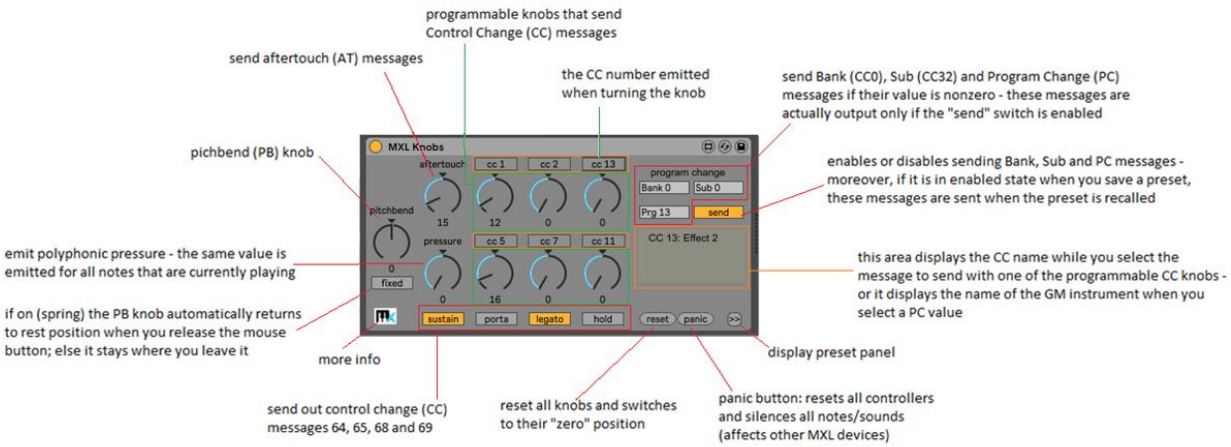


MXL Knobs



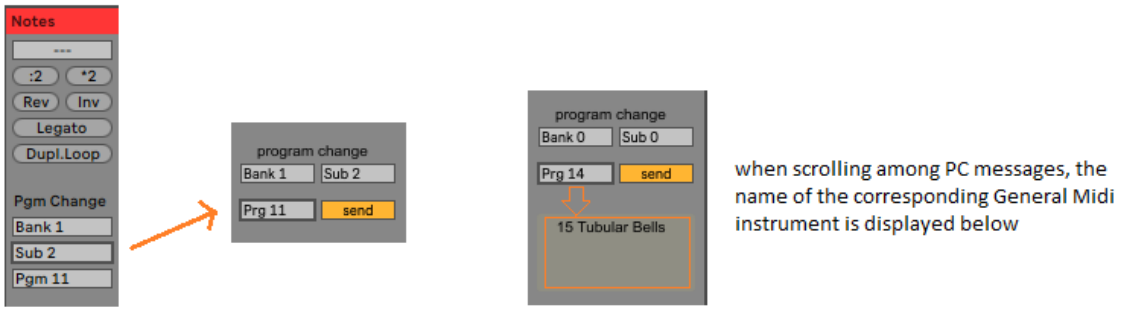
This device provides a simple dashboard to send channel MIDI messages other than note messages, namely Control Change (CC), Polyphonic Pressure, Aftertouch (or Channel Pressure), Pitch Bend, and Bank/Program Change. (It cannot emit system MIDI messages, such as Sysex or Clock).

Like other MXL devices, MXL Knobs offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.



The MXL Knobs device can both *monitor* incoming non-note messages and *send* them. The six rightmost knobs allow you to monitor and/or send CC1 (mod wheel), CC2 (breath), CC4 (foot controller), CC5 (portamento time), CC7 (volume) and CC11 (expression); however, they can be programmed to match any CC message.

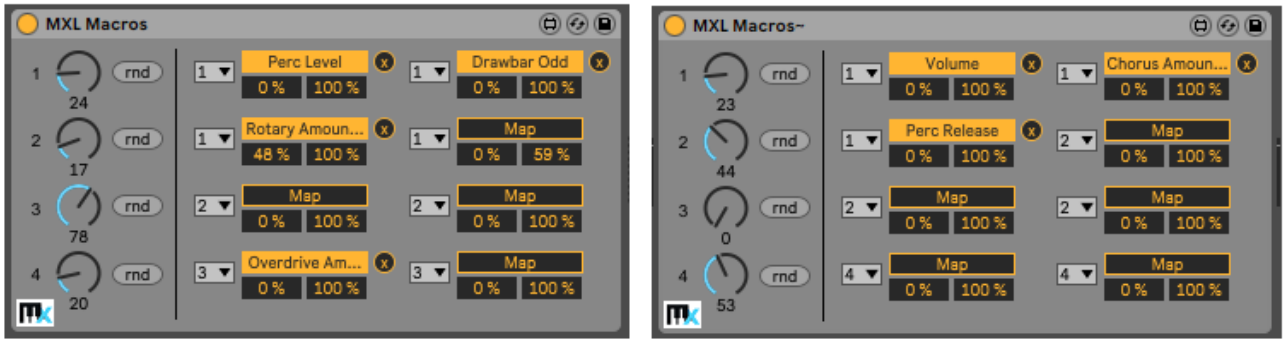
The **pressure** knob sends polyphonic MIDI pressure for all the notes that are currently playing; it currently cannot send distinct pressure values for each note. The **Prg** field allows you to send Program Change (PC) messages. The **Bank** and **Sub** fields send the Bank MSB (CC0) and LSB (CC32) and are similar to fields with same name available in Live's clips:



The most obvious use for the MXL Knobs device is adding features to MIDI keyboards that don't provide aftertouch, polyphonic pressure or don't offer enough CC knobs for your needs. However, it can be handy in many other cases, for example:

- Fields are affected by incoming MIDI messages, thus they provide an effective way to monitor only the CC messages you are interested in, more effectively than other monitor devices.
- Because you can map its knobs and buttons to CC messages, MXL Knobs provides a simple mechanism for transforming an incoming CC message into a different CC message, or into Aftertouch, Polyphonic Pressure, Pitch Bend or even Program Change messages.
- If you switch off the **fixed/spring** button the pitchbend knob does *not* automatically return to its rest position, thus allowing you to play longer musical phrases with a given pitchbend value
- The state of all knobs and fields can be stored a preset, and be automatically sent out when the preset is recalled. Presets provide a simple mechanism for setting the state of a hardware or virtual instrument during performance. This is especially effective with messages such as CC 1 (mod wheel), CC 7 (volume), CC 11 (expression) and the **aftertouch** or **pressure** values.
- The **Bank, Sub** and **Prg** fields allow you to select the Bank (MSB and LSB) and Program Change message that is emitted when you recall a preset, a feature that is extremely useful if you are controlling an external synth. For this feature to work, it is mandatory that the **send** button be enabled when you *save* the preset (not when you recall it).
- The ability to send CC 2 (breath) or CC 4 (foot pedal) messages allows you to test the behavior of hardware and software instruments that are meant to be controlled by a wind instrument or an expression pedal, without the need to connect such devices.

MXL Macros and MXL Macros~

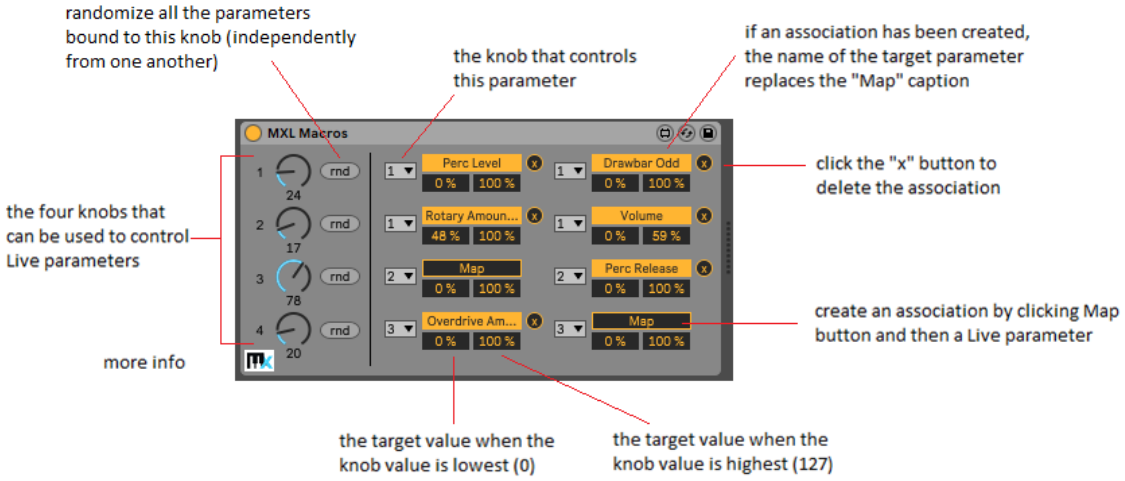


These devices allow you to associate up to eight Live parameters to a knob, which in turn can be controlled via automation or bound to a MIDI CC message and controlled from an external MIDI keyboard. They also allow you to randomize all the parameters bound to a knob with a single click.

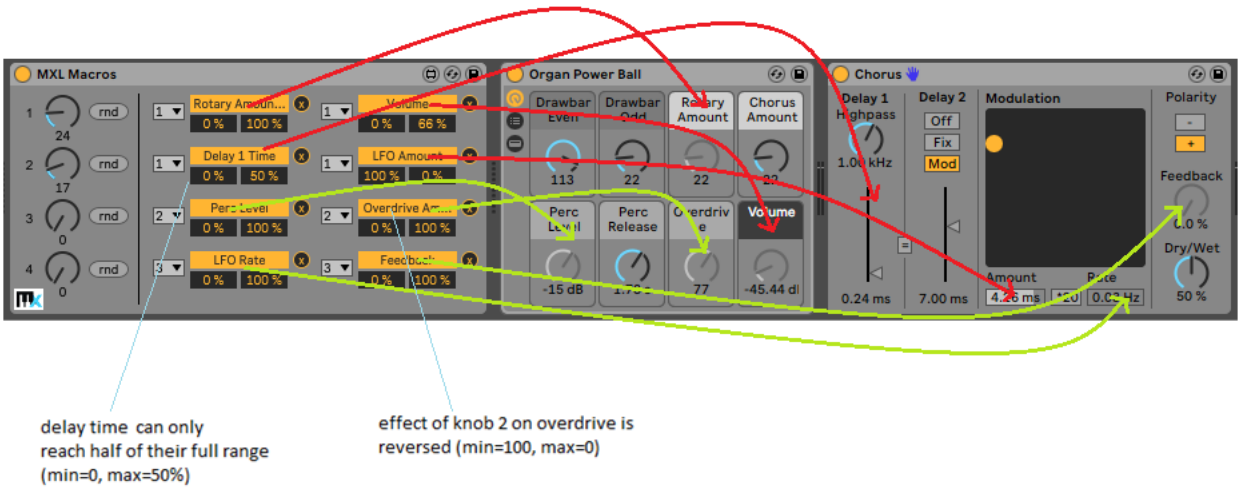
The two devices are identical, except MXL Macros can be used at the beginning of MIDI tracks (before audio instruments) and MXL Macros~ can used in audio tracks (or in MIDI tracks, but after audio

instruments). Both devices are “transparent” to MIDI or audio signals, thus they can be inserted in the middle of the device chain of a track without affecting the track in any way.

You can associate two parameters to each of the four knobs, or associate all eight parameters to the same knob, and all the intermediate configurations. For example, in previous figure, knob #1 is associated with two parameters, knob #2 can be associated with two parameters, and knob #3 can be associated with four parameters.

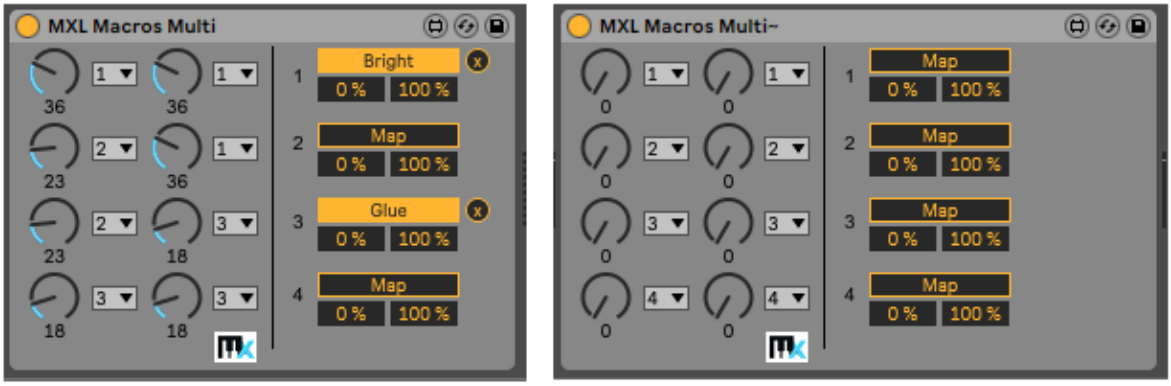


Here an example of how to use MXL Macros to control multiple Live parameters. More specifically, knobs 1 and 2 control four parameters each, two in the Organ instrument and two in the Chorus effect, whereas knobs 3 and 4 are not bound to any.



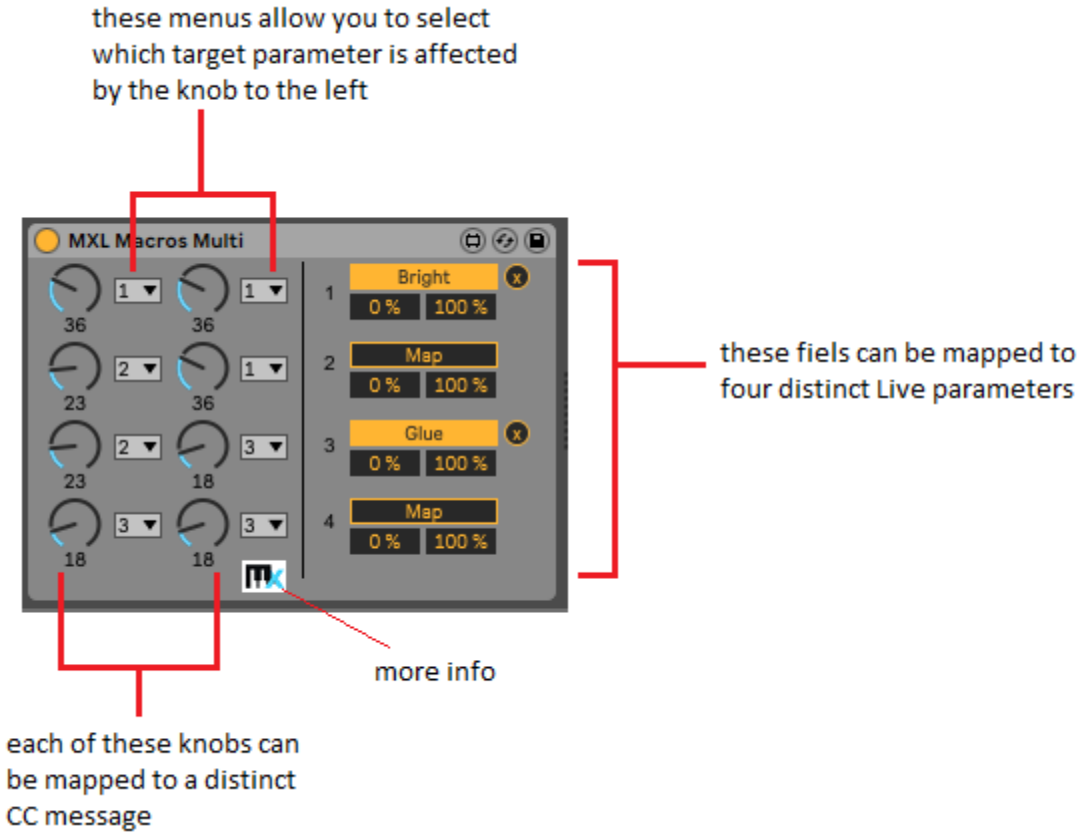
For more information, read the Parameter Mapping section, earlier in this manual.

MXL Macro Multi and MXL Macro Multi~



In some scenarios, you might need to bind a given Live parameter to two (or more) distinct MIDI messages, for example if you use both a MIDI keyboard and a MIDI pedalboard. Live doesn't permit to do that, and these two devices fill this gap.

They are identical, except MXL Macros Multi can be used at the beginning of MIDI tracks (before audio instruments) and MXL Macros Multi~ can used in audio tracks (or in MIDI tracks, but after audio instruments). Both devices are "transparent" to MIDI or audio signals, thus they can be inserted in the middle of the device chain of a track without affecting the track in any way.

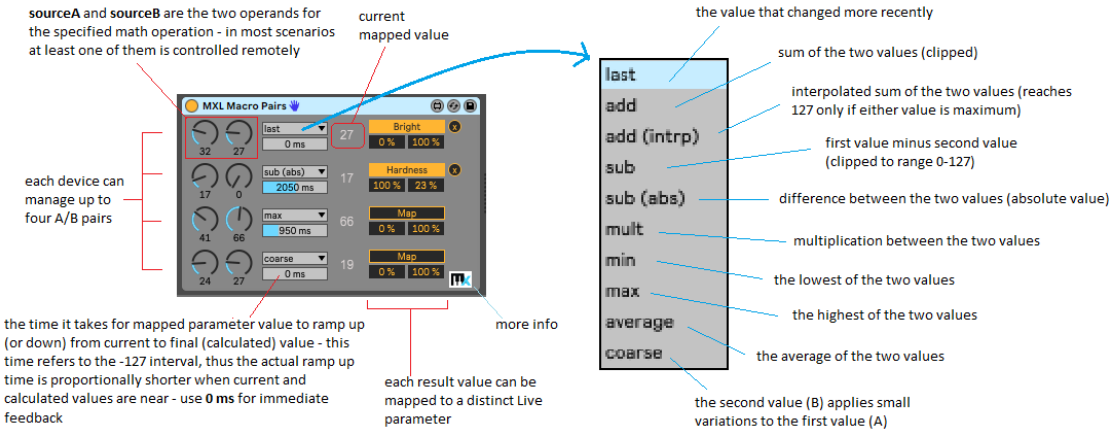


If two knobs are bound to the same target parameter – that is, if the menu to their right display the same value 1-4 – then their rotation is automatically synched.

MXL Macro Pairs and MXL Macro Pairs~

In some scenarios, you might need to combine the effect of two remote CC messages or – which is the same, given Live's ability to map MIDI messages to user interface elements – combine the effect of two knobs in your set. This is precisely what these two devices allow you to do: they use two input values and combine them in various ways by performing different math operations.

They are identical, except **MXL Macros Pairs** can be used at the beginning of MIDI tracks (before audio instruments) and **MXL Macros Pairs~** can used in audio tracks (or in MIDI tracks, but after audio instruments). Both devices are "transparent" to MIDI or audio signals, thus they can be inserted in the middle of the device chain of a track without affecting the track in any way.

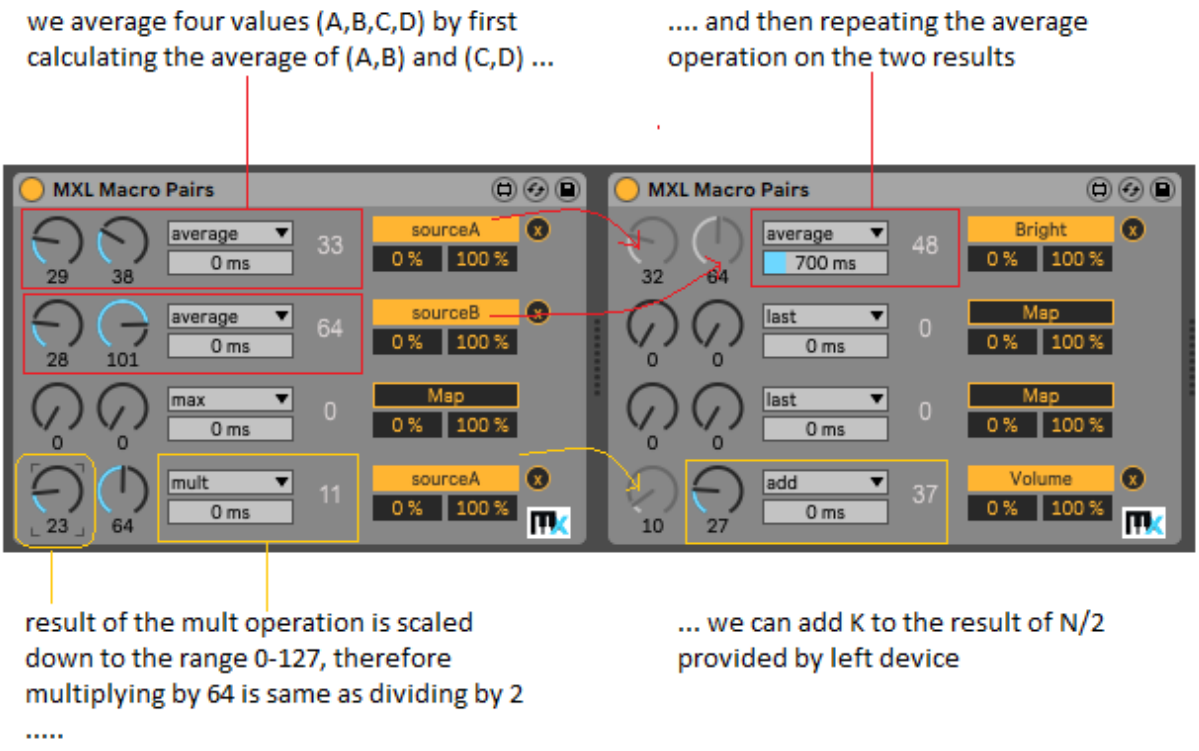


Here a more precise description of each math operation that can be applied to the two input values A and B:

last : the last value that changed
add : the sum **A+B**, clipped if it exceeds the value 127
add (intrp) : interpolated addition is equal to **A+(127-A)*B/127** and is such that it reaches the maximum value (127) only if either value is 127 (or both)
sub : the difference **A-B**, emits zero if result is negative (i.e. if A < B)
sub (abs) : the absolute value of the difference, or **abs(A-B)**
mult : the multiplication of A by B, scaled down to the interval 0-127 (i.e. **A*B/127**)
min : the lowest value, or **min(A,B)**
max : the highest value, or **max(A,B)**
average : the average value, i.e. **(A+B)/2**
coarse : the result of **A+B/16**, so that you can use A for coarse variations and B for more granular ones

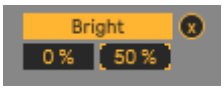
The ramp time refers to the time it takes for the mapped value to go from 0 to 127 (or back). In most cases, the current value and the result value are closer and the time is proportionally smaller than what the field indicates. Use **0 ms** for instantaneous feedback.

While MXL Macro Pairs can only combine pairs of values, you can use multiple devices for more complex operations. Here are two examples:

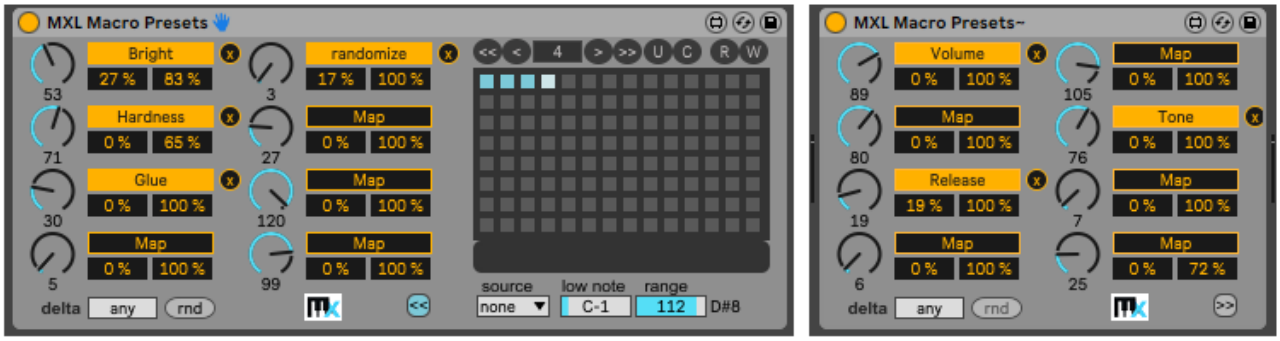


the example in the bottom part shows how you can calculate the expression N/2+K

Also, bear in mind that you can tweak the **min** and **max** percentages in the mapping controls. For example, setting the min value to 0% and the max value to 50% is the same as dividing the result by two:



MXL Macro Presets and MXL Macro Presets~



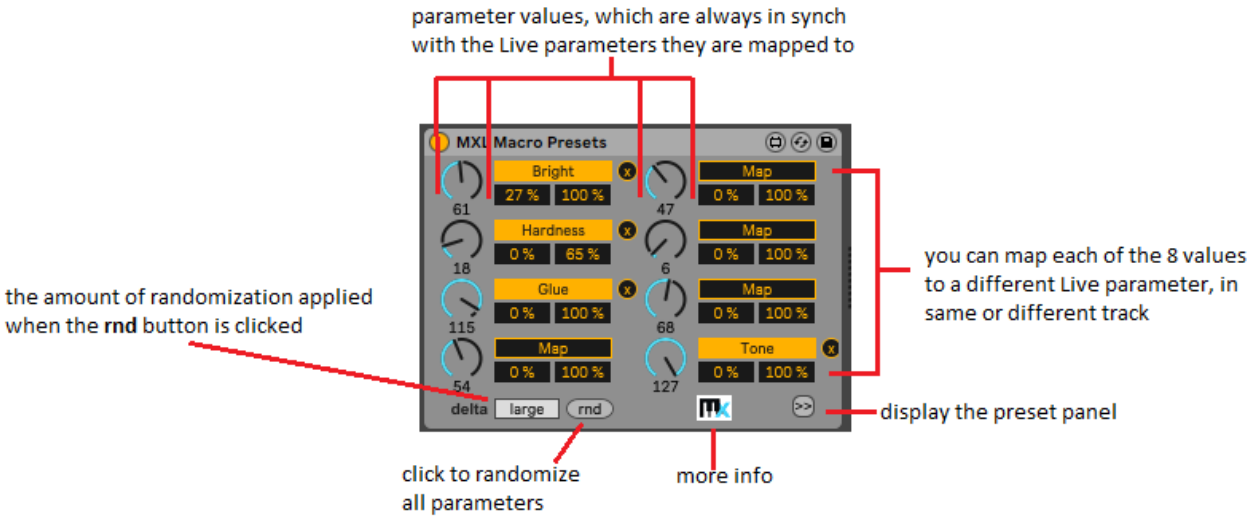
Ableton Live 11 introduced the ability to quickly randomize the value of all the macros in a rack, as well as create snapshots of the current state of such macros, that you can easily recall when necessary. However,

while this feature has been very welcome by Live users, it still has a few serious limitations: as the parameters must reside in a given rack, you can neither randomize nor create snapshots of parameters of different devices, possibly in different tracks. This is the purpose of these two devices.

They are identical, except **MXL Macro Presets** can be used at the beginning of MIDI tracks (before audio instruments) and **MXL Macro Presets~** can used in audio tracks (or in MIDI tracks, but after audio instruments). Both devices are "transparent" to MIDI or audio signals, thus they can be inserted in the middle of the device chain of a track without affecting the track in any way.

Like other MXL devices, these devices offer 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.

At the very least, even if you don't need using presets or the randomization features offered by these two devices, they can be prove to be quite useful to control up to 8 parameters in different tracks from a single devices, without having to switch to different devices/tracks each time.



Using these devices is quite simple. First, you define which Live parameters you want to control as a group: such parameters can reside in one or more devices, in the same track that hosts **MXL Macro Presets** (or **MXL Macro Presets~**) or in different tracks. Once you have defined the group (of max 8 parameters), you can:

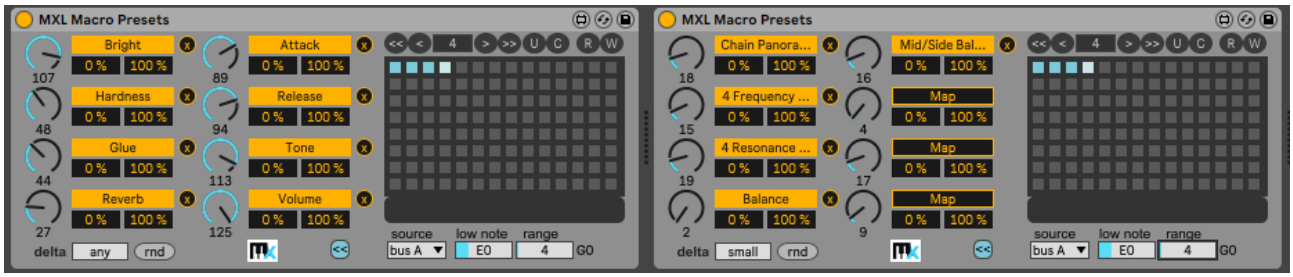
- control all parameters from a single location, which is handy if the original parameters reside in different devices or tracks that cannot be displayed simultaneously
- save them as a preset and recall them all at once, using any of the techniques discussed in the Presets section of this manual
- randomize all of them with a click on the **rnd** button.

The **delta** field allows you to control how much randomization is applied, i.e. the max distance (in either direction) between the current value and the new, randomized value. Four settings are available for this field: **any** (new value is chosen randomly in the range 0-127), **small** (new random value can be max 20 units away from current value), **medium** (new value can be max 50 units away from current value) and **large** (new value can be max 50 units away from current value).

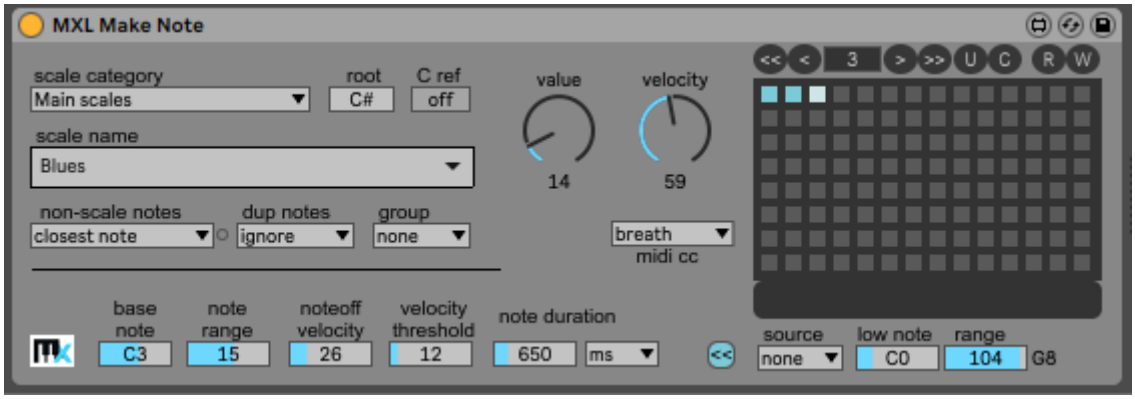
These two devices can be used in Ableton Live 10 (which does not natively offer randomization and snapshot features for macro parameters) and Live 11 (which does offer these features, but only for macros of a specific rack). The ability to map parameters that reside in devices outside a rack (possibly in a different

track) and the option of dosing the amount of randomization make these devices appealing even to users of newer Live versions.

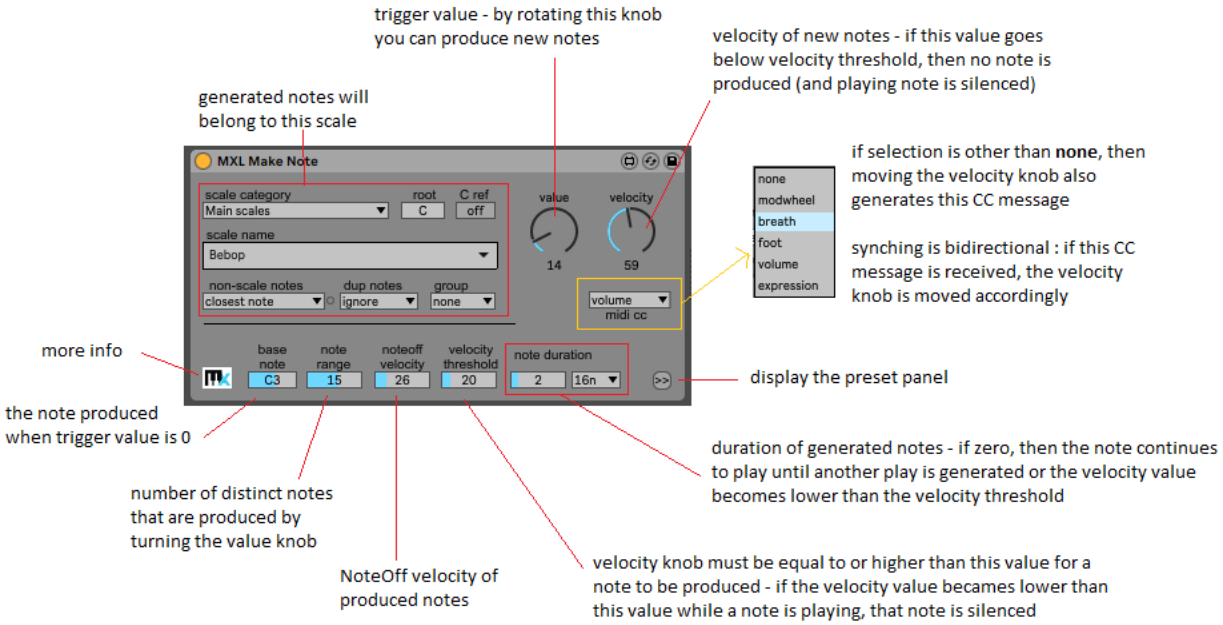
Notice that you can easily control and create presets of more than just 8 parameters: you just need two or more **MXL Macro Presets** or **MXL Macro Presets~** with the same number of stored presets and with identical **source**, **low note** and **range** settings, so that you can recall a preset with same number by sending them a single note.



MXL Make Note



This device is a simple note generator, which produces notes of given pitch, velocity and duration when you rotate the **value** knob. Because this knob can be remotely controlled by means of incoming CC messages, basically you can generate notes using knobs on your MIDI keyboard or – even better - wearable MIDI controllers such as Hot Hand 3 (Source Audio), Wave (Genki Instruments) or Wiggle (MIDIS).



While you can generate notes by rotating the two knobs using your mouse, you get more interesting results by controlling these knobs remotely via a CC message.

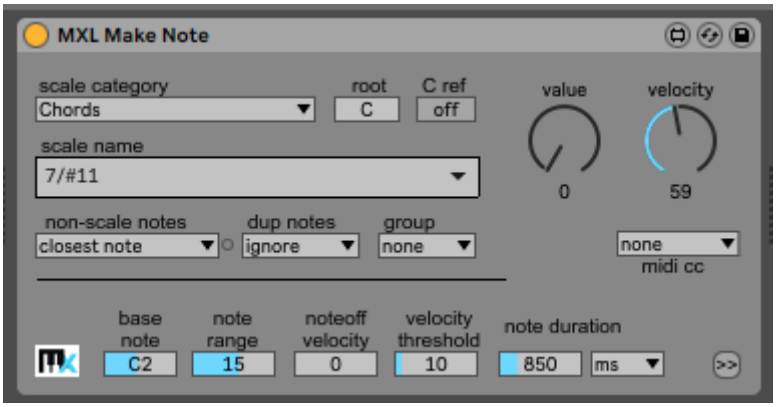
You generate notes by turning the **value** knob, whose range is from 0 to 127 and is scaled to the interval specified by **base note** and **note range** fields. For example, if current scale is C Major (7 notes) , **base note** is C2 and **note range** is 15, by turning the **value** knob you generate all notes in the interval from C2 to C4.

If **note duration** is nonzero, then it specifies for how long generated notes are sustained; if it is zero, then generated notes continue to play until another note is generated or until the **velocity** knob is turned counter-clockwise to bring its value below the current **velocity threshold**.

If the **midi cc** menu is set to any value other than **none**, turning the **velocity** knob also emits the selected CC MIDI message. For example, you can control the volume of notes by selecting **volume** (CC7) or **breath** (CC2, for instruments that are designed to work with wind instruments). The other menu options let you generate **modulation** (CC1, often associated with vibrato effect), **foot** control (CC4) or **expression** (CC11).

The association of **velocity** knob with a CC message is bidirectional: if the track receives the specified MIDI message then the knob reacts accordingly.

It might not be immediately obvious that the MXL Make Note device allows you to create custom arpeggios over chords. This is possible because the last **scale category** option allows you to select one among many different chords, from simple triads to more complex chords with added 7th, 9th, 11th and 13th extensions:

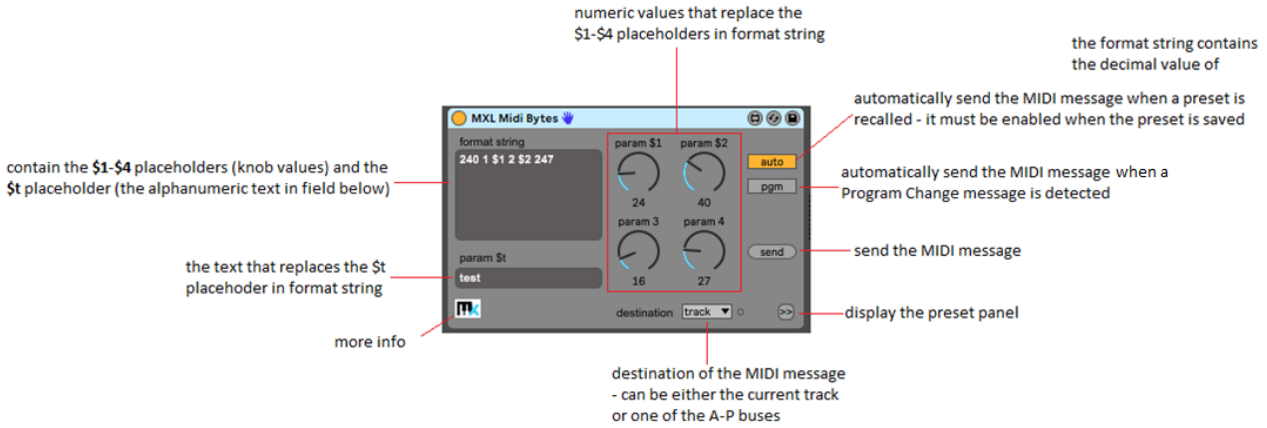


Paired with the ability to bind the **value** control to a MIDI message, this feature allows you to arpeggiate a chord by turning a knob on your MIDI keyboard while you play something else with the other hand.

MXL Midi Bytes

This device allows you to send an arbitrary sequence of MIDI bytes, including system messages. It comes with a format string that contains placeholders that are replaced by the value of four numeric knobs and one text field.

Like other MXL devices, MXL Midi Bytes offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.



The MXL Midi Bytes is for musicians with a deep knowledge of MIDI internals, because it assumes that you know how to build MIDI messages out of individual bytes. It is especially useful to control hardware and software instruments that accept SysEx messages.

A SysEx message always starts with the byte 240 and ends with the byte 247; all the bytes in between – which must be in the range 0-127 – make the body of the SysEx message. For example, a hypothetical hardware synthesizer might perform a dumpout of all its presets when it receives the following SysEx message:

240 23 67 121 247

In addition to sending MIDI messages containing only constant bytes, such as the above one, **MXL Midi Bytes** can emit messages containing variable fields, that are replaces by the value of the four knobs **param $1 … param $4**. The following example sends out a NoteOn-NoteOff message pairs:

144 $1 $2 128 $1 $3

where **$1** is the pitch of the note, **$2** is the note velocity, and **$3** is the release (NoteOff) velocity. If the **MXL Midi Bytes** device is followed by a Live instrument in the same track, you will clearly hear a very short note. Notice that you have to click the **send** button to actually emit the MIDI bytes.

In addition to the four **$1…$4** numeric parameters, the **format string** field can contain the **$t** placehoder, which is replaced by the ASCII codes of the text typed in the **param $t** field. You might use this feature to display a string on the LCD panel of an external hardware peripheral, if you know how to format the SysEx message that does the trick.

This device can be useful to emit a series of Control Change (CC) and Program Change (PC) messages in sequence. For example, say that you want to select PC 10 and reset the modulation wheel (CC1) to zero and the MIDI volume (CC7) to its max value. Here's the format string that does the trick:

192 9 176 1 0 176 7 127

where:

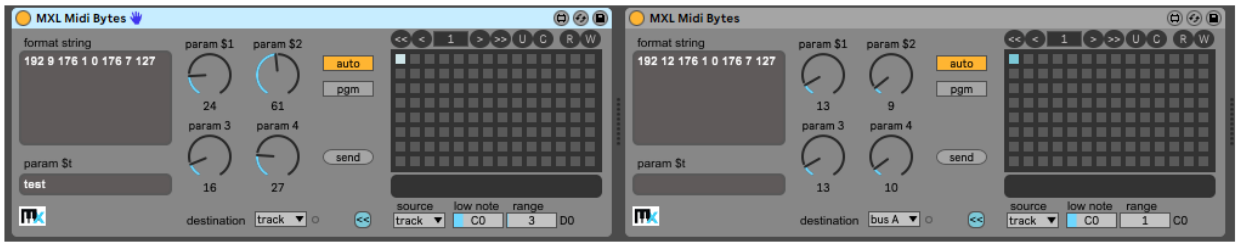
192 9 sends Program Change 10 (PC numbers are zero-based)
176 1 0 sends CC1 with value 0
176 7 127 sends CC7 with value 127

You can store the current value of all fields in a preset, and recall it with any of the available mapping mechanisms. You can also have these messages sent out *as soon as* you recall the preset. For this technique to work, it is necessary that the **auto** button be enabled when you save the preset.

Interestingly, you can emit the message whenever you send a Program Change message to the track, if the **prg** button is enabled.

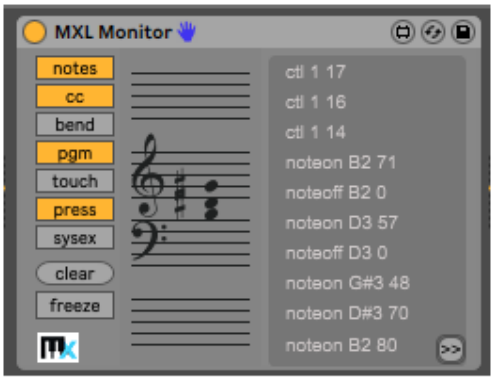
Unless you are very familiar with MIDI, you might need some guesswork to define what bytes you must emit. Here's a tip: use the MXL Monitor device to intercept a flux of MIDI messages, then possibly replace some values with the $1-$4 placeholders, and paste the result inside MXL Midi Bytes. (You don't need to delete the asterisks that MXL Monitor uses as message separators, because MXL Midi Bytes ignores them.)

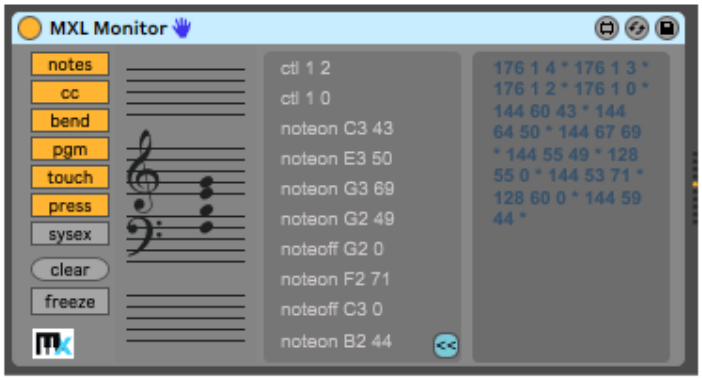
Finally, notice that you can send the message to a destination other than the current track and you can use a chain of MXL Midi Bytes devices with presets that send to different tracks, as shown in this figure:



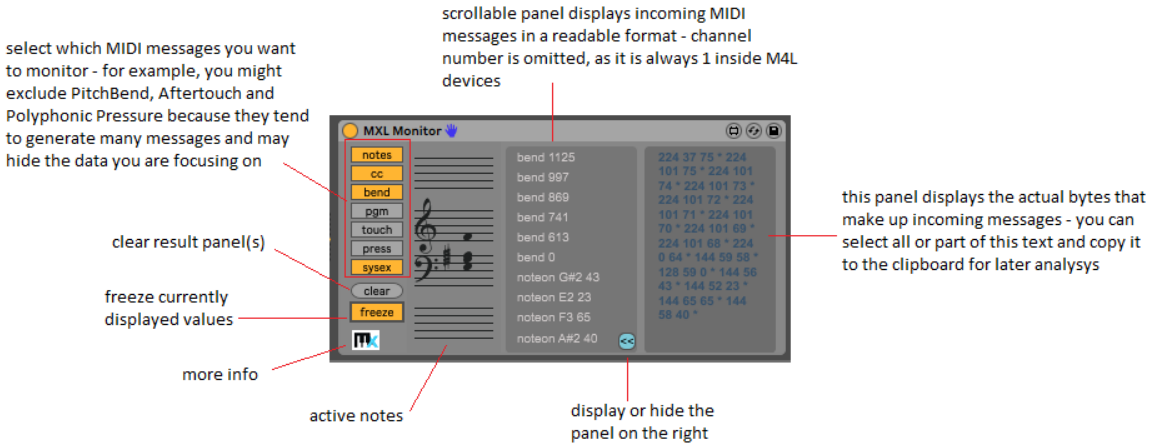
Preset #1 of the device on the left sends PC 10, CC1 with zero value and CC7 with max value to current track. Preset #1 of the device on the right sends PC 13 and same CC messages to Bus A. Because of the **auto** state and the way preset panels are configured, when the note C0 is received on current track, then both devices emit their MIDI bytes.

MXL Monitor





This device allows you to monitor incoming MIDI messages, and displays them in many formats:



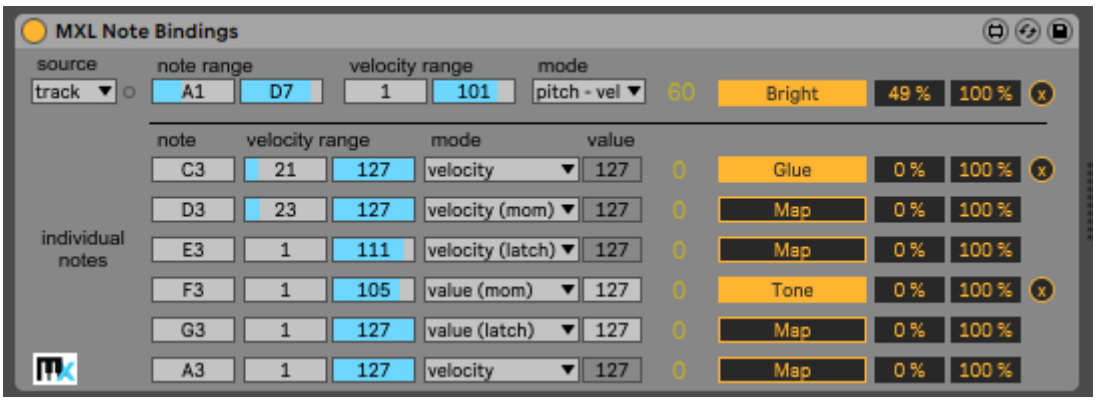
MXL Monitor has several features that are missing in other similar devices, such as Live's MIDI Monitor:

- traces only the messages you are interested in
- shows NoteOff release velocity
- displays PitchBend values with 14-bit precision – quite conveniently, value 0 corresponds to the rest position, therefore positive values mean "bending up" and negative values mean "bending down"
- displays the actual decimal value of individual MIDI bytes and lets you copy them to the clipboard and paste somewhere else
- does ***not*** display channel information, which is always 1 inside M4L devices and is therefore a waste of screen estate

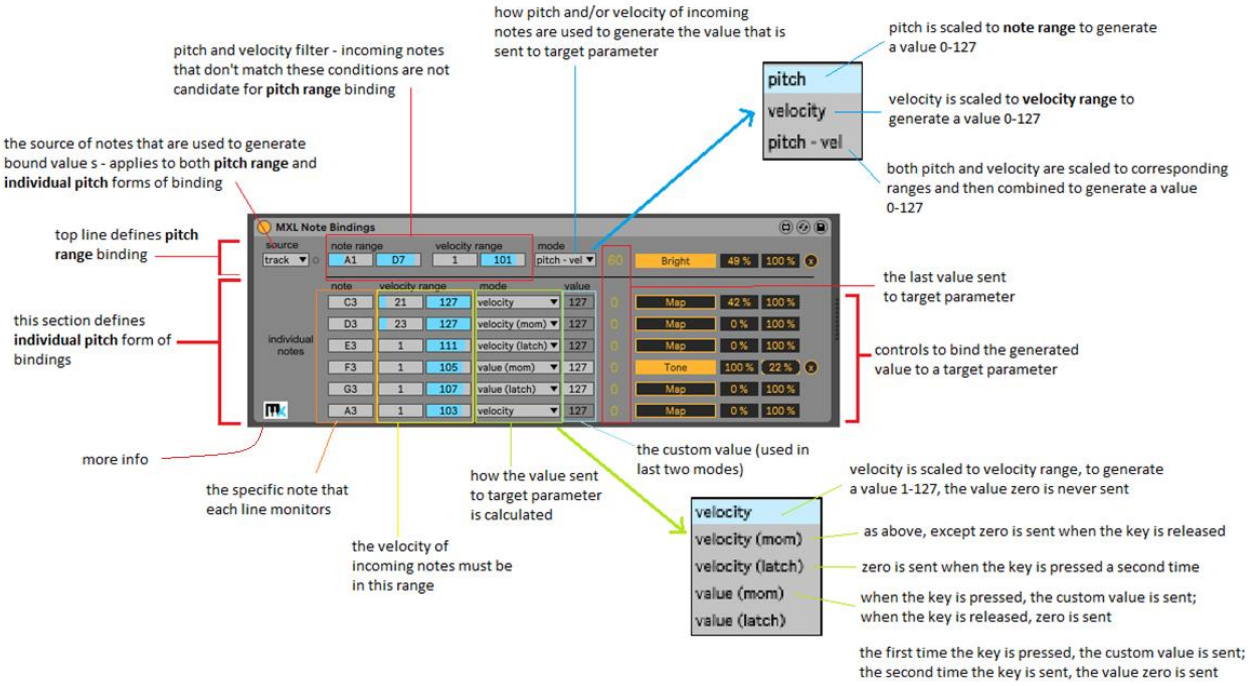
You can paste the contents of the right panel into MXL Midi Bytes, possibly replace some values with placeholders **$1-$4** and then later resend multiple messages with one click.

MXL Monitor displays notes on musical clefs but does not include a graphical representation of a MIDI keyboard. For that purpose, use the MXL Keyboard device, which can extend over 2-5 octaves, has chord recognition and can be used to play notes, too.

MXL Note Bindings



This device allows you to map the pitch and/or velocity of incoming notes to a Live parameter, in the same or a different track. The notes that control the mapping can be received from the current track or a Bus A-P. All MIDI messages received directly from the track are passed as-is to the next device in same track.



The MXL Note Bindings device supports two distinct forms of mappings:

- **Pitch range:** the top line of controls allows you to associated pitch and velocity of any note in the specified range.
- **Individual Pitch:** The bottom lines of controls below it allow you to monitor six individual notes: in this latter case, you send the note velocity (in absolute, momentary, or latch mode) or a fixed value (in momentary or latch mode) to the Live's target parameter.

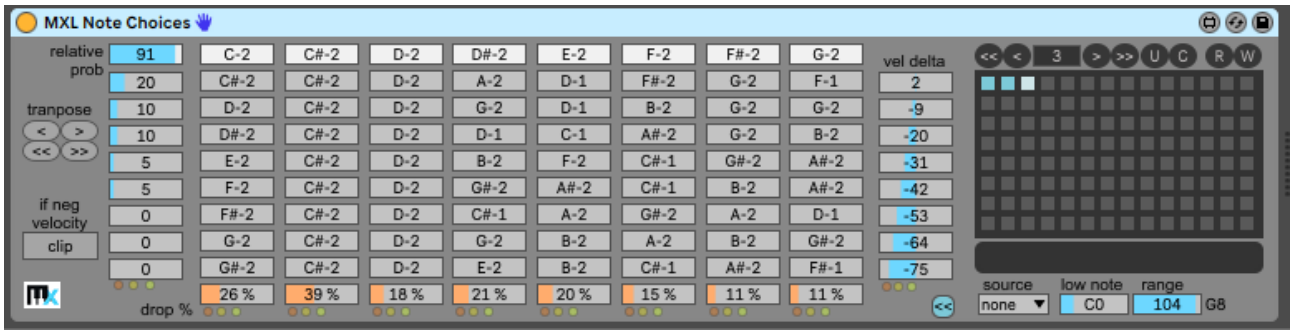
In ***absolute*** mode the value doesn't change when you release the key; in ***momentary*** mode the value goes back to zero when you release the key; in ***latch*** mode you need to hit the key a second time to bring the value back to zero.

It is crucial to understand how MXL Note Bindings evaluates the value sent to target parameter:

- when the velocity is used to generate the value sent to target parameter, the velocity is scaled to the **velocity range** interval, so that the lowest accepted velocity generates zero and the highest accepted velocity generates the value 127 (in both ***pitch range*** and ***individual pitch*** mapping)
- when the note pitch is used to generate the value sent to target parameter, the pitch is scaled to the **note range** interval, so that the lowest accepted pitch in the range generates zero and the highest accepted pitch generates the value 127 (in ***pitch range*** mapping only)
- when the **pitch – vel** mode is selected, both pitch and velocity are scaled to their corresponding intervals and the result is combined, so that the highest value (127) is generated only when the device receives the highest allowed note with the highest allowed velocity
- when the custom value is used to generate the value sent to target parameter, no scaling occurs (in ***individual pitch*** mapping only)

For more information, read the Parameter Mapping section, earlier in this manual.

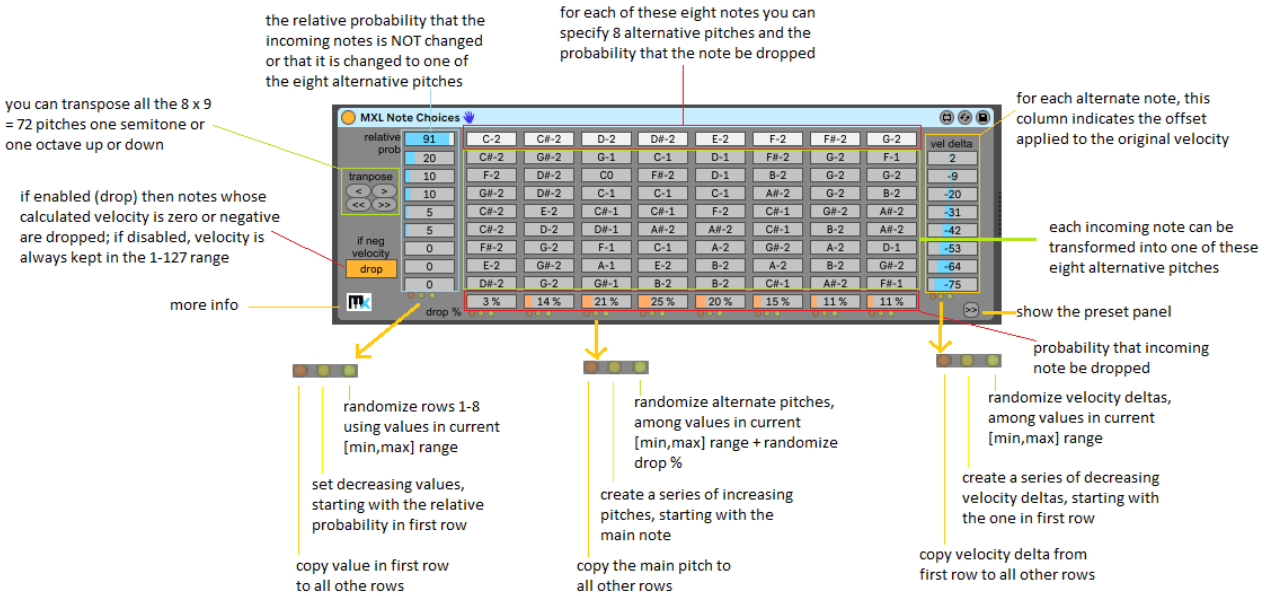
MXL Note Choices



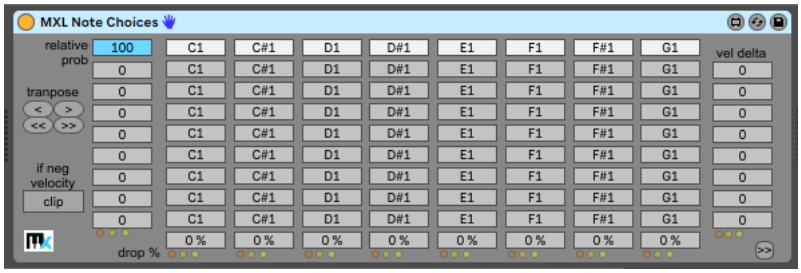
This device is the most complete note randomizer in the MXL Pack collection. It gives you a very granular control on what happens when a specific note is received. It is especially useful to randomize drum racks, but can be applied to melodic tracks as well.

When you randomize a drum track, you typically want control on how individual drum samples are played back. MXL Note Choices gives you the ability to replace any specific drum sample with another sample and/or vary its velocity, or even drop it entirely.

MXL Note Choices offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.

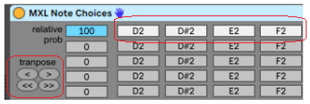


When loaded for the first time, MXL Note Choices is initialized with notes from C1to G1, no delta velocity offsets, zero probability that the note be dropped:



The following figures summarize the typical actions you perform next. It looks like you have to spend a lot of time to set all these fields, but MXL Note Choices provides several shortcuts to accelerate the process:

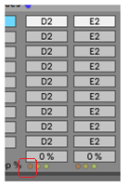
(1) The first thing to do on a freshly loaded device is using transpose buttons to bring notes (first row) in the right pitch range



(2) adjust individual pitches to match the notes in your drum set that you want to randomize - notes don't have to be sorted in any way

if the same note appears in two or more columns, the rightmost one is used to randomize that note

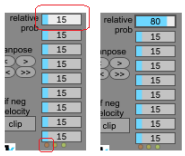
(3) it is convenient to start with a series of alternate notes that is equal to the main note - you can complete this step quickly by clicking the tiny orange button near the bottom border



The device is now ready to intercept one of the eight notes indicated by the topmost row. When one of these notes arrives, MXL Note Choices decides to leave it unchanged or replace by one of the alternate pitches in the subsequent rows. The ***relative*** probability used to take this decision is indicated in the leftmost column.

You can select the individual probability for each note variation, or you can speed up the process using one of the following procedures, which use one of the three tiny buttons at the bottom of the probability column:

(4A) you can now set the relative probability that the first note is not changed (first row) or that it is replaced by one of the other choices - if these probabilities are the same, just set the topmost slider and the click the tiny button at the bottom



TIP: if the main note has higher probability, simply reset the first slider after clicking the orange button

(4B) to set decreasing probabilities, just set the topmost slider to a convenient value, then click the tiny yellow button

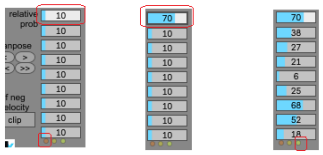
if a row has zero probability, the corresponding alternate pitches will be never selected during the randomization process

(4C) to set random probabilities in a range, perform these three steps:

select lowest value in top slider, then click the tiny orange button

next, select the highest value in topmost field

finally, click the tiny green button to generate random probabilities between the two values



You now need to decide what happens when MXL Note Choices randomly selects a row other than the topmost one.

You can select – for each column – the alternative pitches that can replace the incoming note. You can set each pitch individually, or you can speed up the process with one of the following procedures, which use the three tiny buttons at the bottom of note columns:

(5A) if alternate notes are "close" to the original pitch, click the tiny yellow button to create a series of ascending pitches

(5B) if alternate notes are close to each other but not to the main note, just temporarily change the main note, click the tiny yellow button and then reset the topnost field to its original value

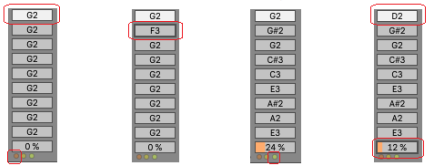
(5C) if alternate notes are to randomly chosen in a range - say G2 to F3), you can follow this 4 step procedure

select lowest note, then click the orange tiny button

select highest note in second row

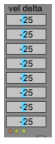
click tiny green button to randomize notes and drop %

restore main note and adjust drop % if so desired



Finally, you need to set velocity offset for each of the eight alternate pitches. Again, you can set values in the sliders in rightmost column, or you can speed up the process using the tiny buttons near the bottom border:

(6A) if you want to apply the same velocity delta to all alternate pitches, select the value in the topmost slider and click the tiny orange button



(6B) if you want to apply descending velocities, going from a max value to a min value, follow these three steps

select the max value in topmost slider and click tiny orange button

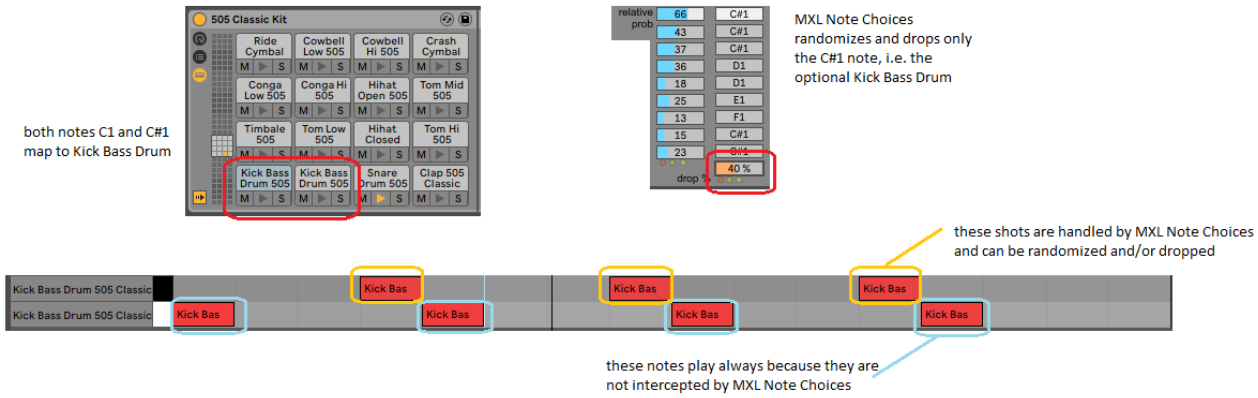
select min value in 2nd row

click the tiny yellow button to create descending values

(6C) if you want to apply random velocities, you can follow the same procedure seen in **(6B)**, except that you click the tiny green button

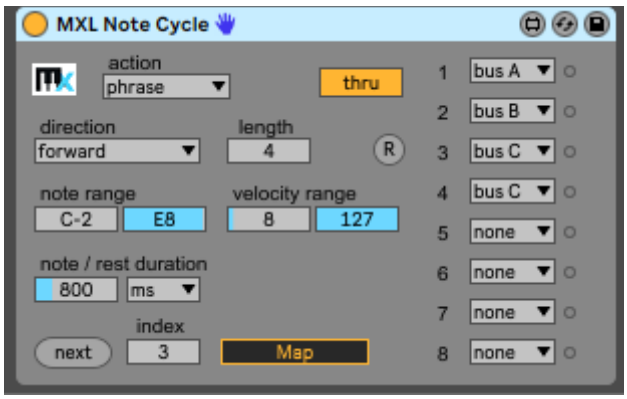
(7) finally, decide what happes when the resulting velocity would be zero or negative - you can decide to play the note anyway (with the lowest velocity = 1) or drop the note entirely

Apparently, this device only allows you to ***drop*** incoming notes randomly. However, when used with drum tracks, you can use MXL Note Choices also to ***add*** new drum shots randomly, using the following simple trick:

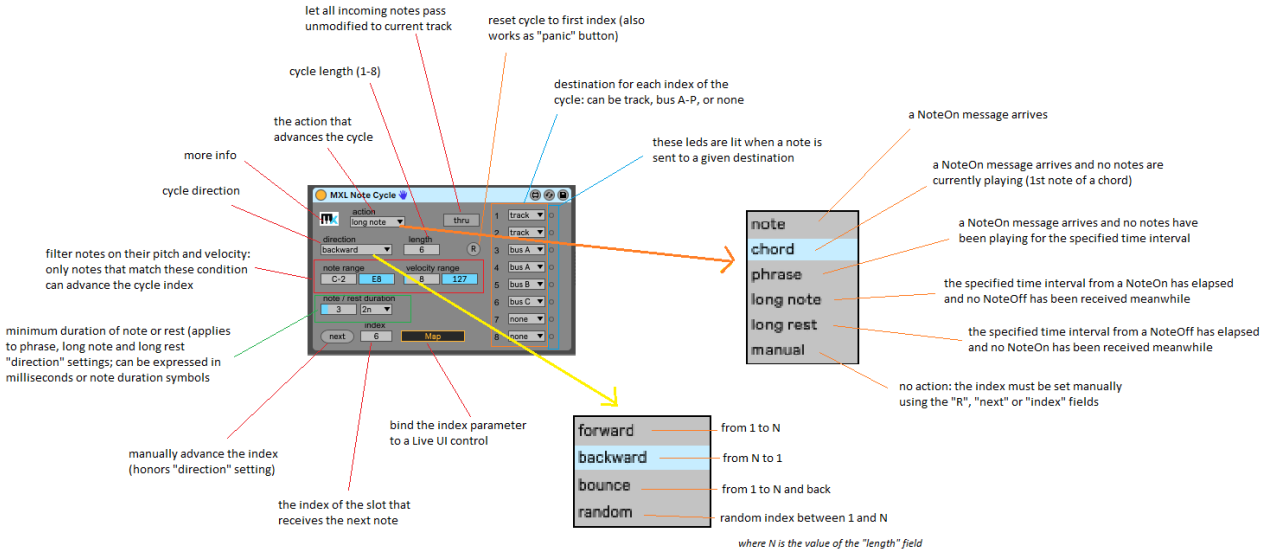


Finally, bear in mind that only the eight notes appearing in the topmost row are handled in any way, and all other notes flow through it unchanged. Thanks to this behavior, you can randomize more than just eight notes simply by using two or more MXL Note Choices devices in a serial fashion.

MXL Note Cycle



This device allows you to send incoming notes to different destinations, in a cyclic manner. You can establish which action advances the cycle index and can map the current cycle index to any Live parameter, in same or different track.



In any given moment, the **index** field points to the destination 1-8 that will receive subsequent note messages. You can drag this number up or down to manually select a slot, click the **next** button to advance the index to the "next" destination (according to the **direction** setting), or you can click the **R** button to reset the index to 1 (this button also performs a "panic" operation and flushes any pending NoteOff message).

When **action** is set to "manual", you can only change the destination slot by means of these three fields. Because these fields are mappable, in practice you can perform these operations by operating a knob or pressing a button on your MIDI keyboard. All other options in the **action** menu allow you to advance the index automatically:

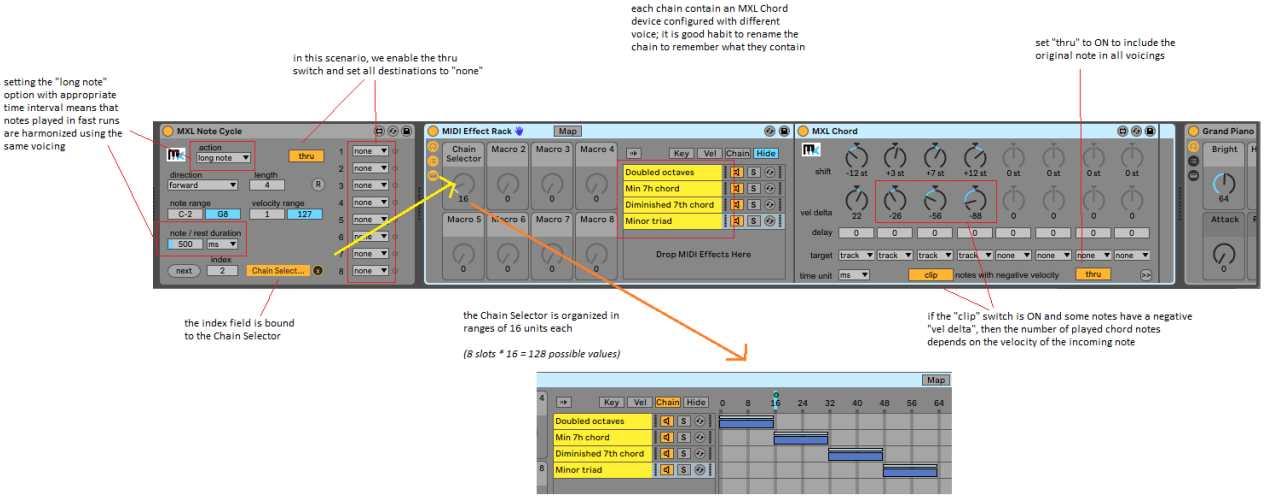
- the **note** option is the simplest and most intuitive: the index is advanced when a NoteOn message is detected (provided that the note matches the **note range** and **velocity range** filter settings)
- the **chord** option also intercepts NoteOn messages, except it advances the index only if no other note is currently playing - this ensures that all the notes of a given chord are sent to the same destination
- the **phrase** option is advances the index when a NoteOn message arrives when no other notes are playing *and* the duration of the rest after the last NoteOff message is at least equal to the value of the **note/rest duration** field – in practice, you can use this option when you want to send all the notes of a musical phrase to the same destination, where you decide the shortest rest interval for a new set of notes to qualify as a "new" phrase
- the **long note** option advances the index if a NoteOn message is detected and the time interval specified by **note/rest duration** has elapsed without receiving a NoteOff message (the NoteOn and NoteOff messages do not have to be related to the same pitch)
- the **long rest** option is the opposite of the previous one: it advances the index if a NoteOff message is detected and the time interval specified by **note/rest duration** has elapsed without receiving a NoteOn message – if you are playing a monophonic instrument, this option is the same as "phrase"

As already explained, the **note range** and **velocity range** fields dictate what notes can advance the index used for the next note. This mechanism, for example, allows you to advance the index only when a note in a given pitch range is played, and play all other notes with the current index. However, notice that the index is advanced ***after*** a note that matches the filter, whereas the note in question uses the index that was current when it was received.

You can set the **direction** in which the index advances, chosen among **forward**, **backward**, **bounce** (back and forth) and **random**. The random setting is especially effective and useful in a variety of circumstances: for example, you might set some destinations to **track** and others to **none**, to drop incoming notes randomly. For example, if **length** is set to 5 and two destinations are set to **none**, the probability of dropping notes is exactly 40 percent.

As if all these options weren't enough, the ability to map the **index** field to any Live parameter allows you to implement a large number of techniques that would not be possible otherwise. For more information, read the Parameter Mapping section, earlier in this manual.

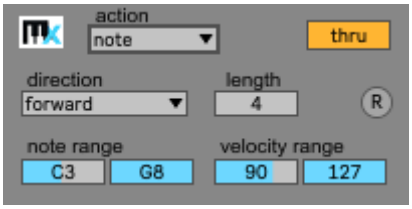
In the next example, we use a MXL Note Cycle together with a MIDI Effect Rack that contain multiple instances of MXL Chord to create a "rotating harmonizer", an effect made popular by saxophone and EWI master Michael Brecker in the late 80s: each time you play a note a different harmonization is applied to the note:



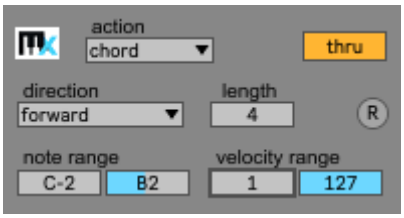
The many options offered by MXL Note Cycle allow you to customize this effect in many ways that are not available in other, similar devices. For example, by setting the **action** field to "long note" the harmonization doesn't change when playing short notes – as you would do in a fast run – therefore these notes are rendered using "parallel" harmonies. Alternatively, you can change the harmonization only at the beginning of each musical sentence (**action** = phrase) or only after a longer sustained note (**action** = long note).

For even greater flexibility, bear in mind that incoming note messages must pass through the **note range** and **velocity range** filter to be considered as candidates for triggering the action. This holds true for both NoteOn and NoteOff messages: in the latter case, a NoteOff message is analyzed only if its companion NoteOn message was accepted by the filter.

For example, If you configure MXL Note Cycle as follows then you can advance the **index** value when emphasizing notes in the melody from C3 upward, but not if you play chords in the lower portion of the keyboard:

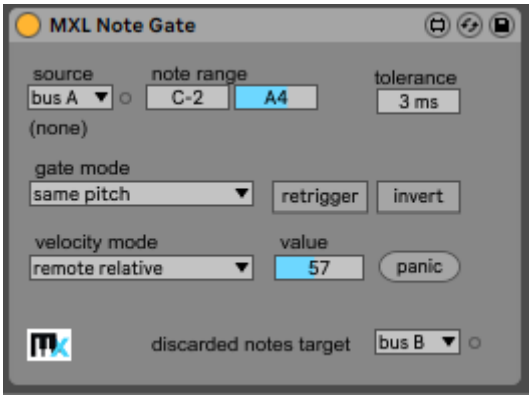


In the following example you advance the index only playing a chord in the lower part of the MIDI keyboard; all notes from C3 onward use the index value as set by the last chord:

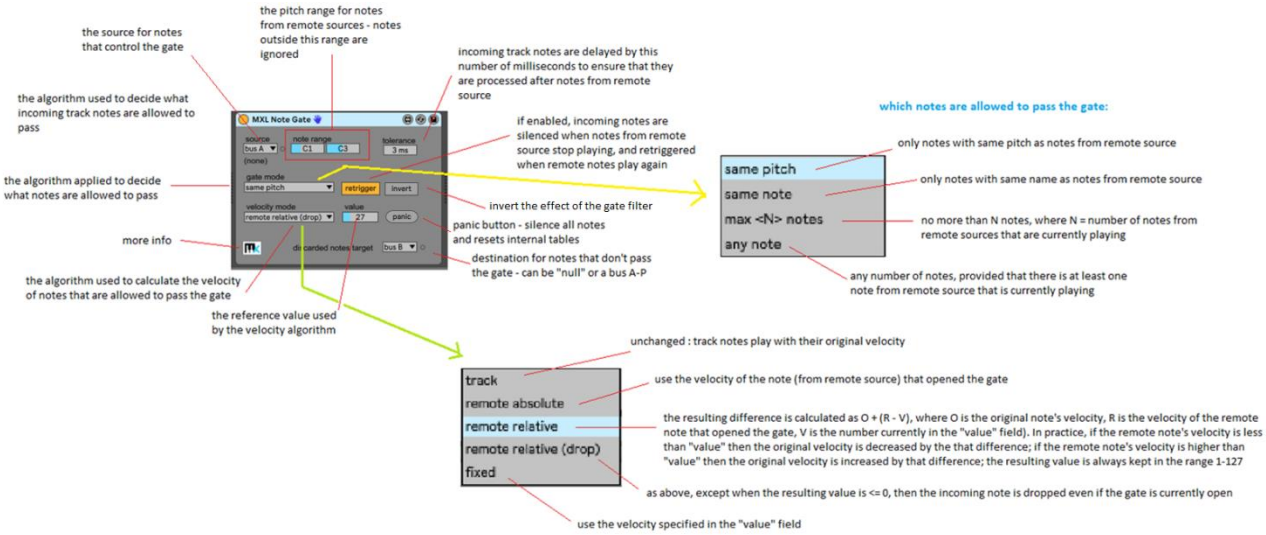


Finally, remember that you can remotely control the **index** field using the many techniques that Live gives you: a knob on your keyboard, an LFO device, etc.

MXL Note Gate



This devices works as a “gate” that lets only some notes pass and blocks all others (or sends them to an alternate target). Unlike other similar devices that are based on a gate controlled by an LFO or similar mechanism, MXL Note Gate uses note messages from a remote source to decide which notes can pass and how their resulting velocity is calculated.



***Note 1**: in following descriptions, we call* incoming notes *or* track notes *the notes arriving from the current track, and* remote notes *those arriving from the origin indicated by the **source** field. In most live performance scenarios, a MIDI instrument emits notes of one of these kinds and a clip/track emits notes of the other kind. However, only for clarity’s sake, in the following figures we will use clips from two tracks to show the effect of remote notes over track notes.*

***Note 2**: in the following examples we assume that remote notes are processed* before *track notes; this assumption can be always taken as valid in live performances, in that the performer*

*can play slightly "ahead of time" as necessary. This assumption can be considered valid also in playback scenarios, if the **tolerance** value compensates for very short delay that Live may introduce when emitting notes from different tracks.*

The most important fields are the **gate mode** menu (which determines which track notes find the gate open) and the **retrigger** switch (which determines what happens when a closed gate reopens). Of course, you should also consider the **invert** button, which reverses the effect of the gate mode selection.

The most simple setting for gate mode is **any note**: if there is at least one remote note currently playing then the gate is open, else the gate is closed; when the gate is open, all incoming notes pass, else none of them does. In this mode, the pitch of the remote note is irrelevant.

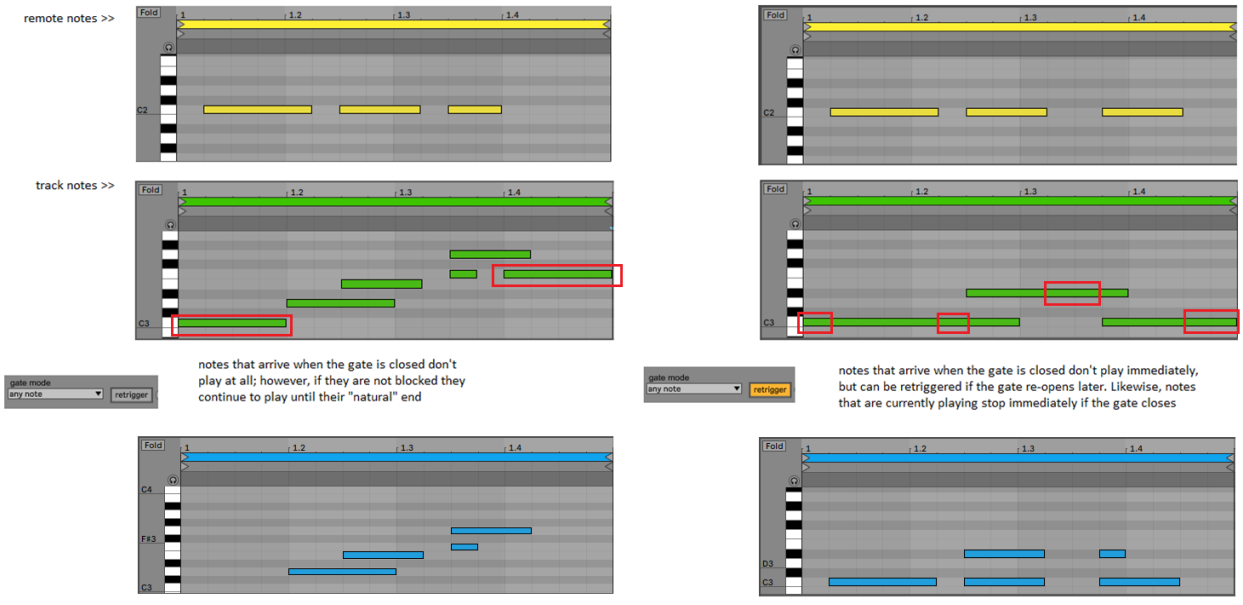
In live performances, you may use a MIDI keyword as the remote source that allows or blocks notes coming from a pre-recorded clip, to add variations etc. Because one single note from the keyboard is enough to open the gate, you can narrow the **note range** field to that single note. In addition, by setting different note ranges for different MXL Note Gate devices in different tracks, you can freely play or silence multiple clips at the same time.

Notice that this is roughly equivalent to controlling the mute switch of a track, except that the result is more graceful because notes always reach their natural end – i.e., until the NoteOff message is received - and tail reverb can be heard, rather than being abruptly silenced when the track is muted.

If the **invert** button is enabled, the **any note** filter is reversed: if no remote notes are playing, the gate is open and accepts all incoming notes; if one or more remote notes are playing then the gate is closed.

If **retrigger** mode is enabled, the device provides an effective way to repeat notes quickly. In live performances, you may use a clip as a remote source that opens or closes the gate for notes that you play on your MIDI keyboard. If the clip contains many short notes then you can press one or more keys on your keyboard and let MXL Note Gate play them in very fast staccato style.

In non-live scenarios, you can use one pre-recorded clip to decide when the gate opens and closes. For example, the clip in the main track can contain chords made of long notes, whereas the remote clip can work the "rhythm track" that plays and stops chord notes in the desired comping style.



The **max <N> notes** setting is less permissive than “any note”: if the remote source is playing one note, only one track note at a time is allowed to pass and the remaining ones are blocked; if two notes are currently playing on the remote source, then max two track notes are allowed to pass, and so on. Again, if the **invert** button is enabled, the filter works in reversed fashion.

The **same note** setting let track notes pass if and only if a remote note with same name is currently playing, even if the two notes are in different octaves. For example, if the remote source is currently playing C2 and E2, then all C and E track notes (e.g. C3, E4, etc.) pass the gate, but F and G notes do not. If the **invert** button is enabled, then all track notes pass the gate *except* C and E notes.

Finally, the **same pitch** setting is the least permissive of the group: a track note passes the gate if and only if a remote note with same pitch is currently playing.

The **velocity mode** adds more variety to the output from the MXL Note Gate device. If **track** mode is enabled, remote notes only decide whether a track note actually plays (as seen above) but do not affect the velocity of the output notes.

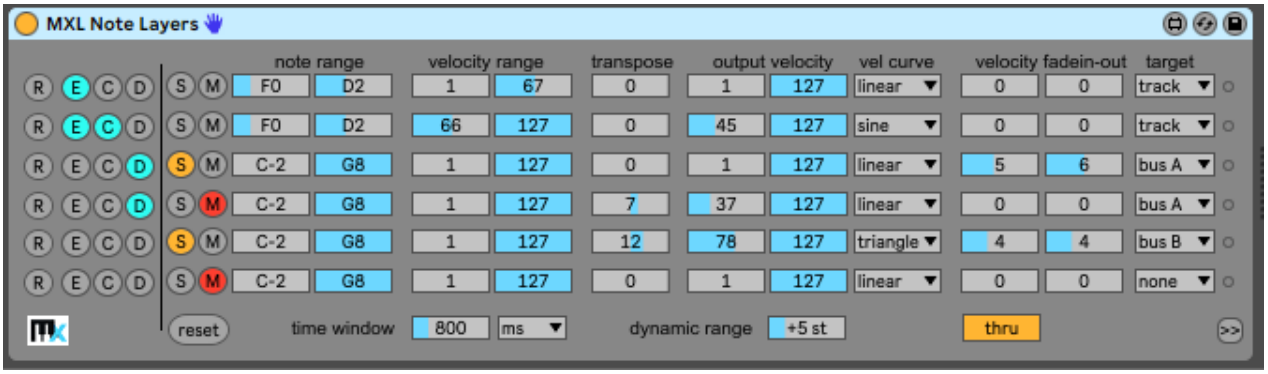
If **remote absolute** mode is selected, then output notes inherit the velocity of remote notes. If the remote clip has a duration that is not an integer number of measures, you can create interesting variations by repeating the same sequence (from the track clip) with accents that fall on different notes at each repetition.

If **remote relative** mode is selected, the neat effect is that output notes preserve the accents they have in the track clip – i.e. notes with original high velocity are still accented – yet the actual velocity of all notes is increased or decreased according to the **value** setting. For the sake of illustration, if **value** is set to 80 and that a remote note with velocity 95 arrives: the difference is +15 and this number is added to the velocity of all incoming notes as long as the remote note doesn’t change. However, if another remote note with velocity 55 arrives, the difference is -25 and this number is subtracted from the velocity of all incoming notes. If the incoming note has velocity 25 or lower, the resulting velocity is 1.

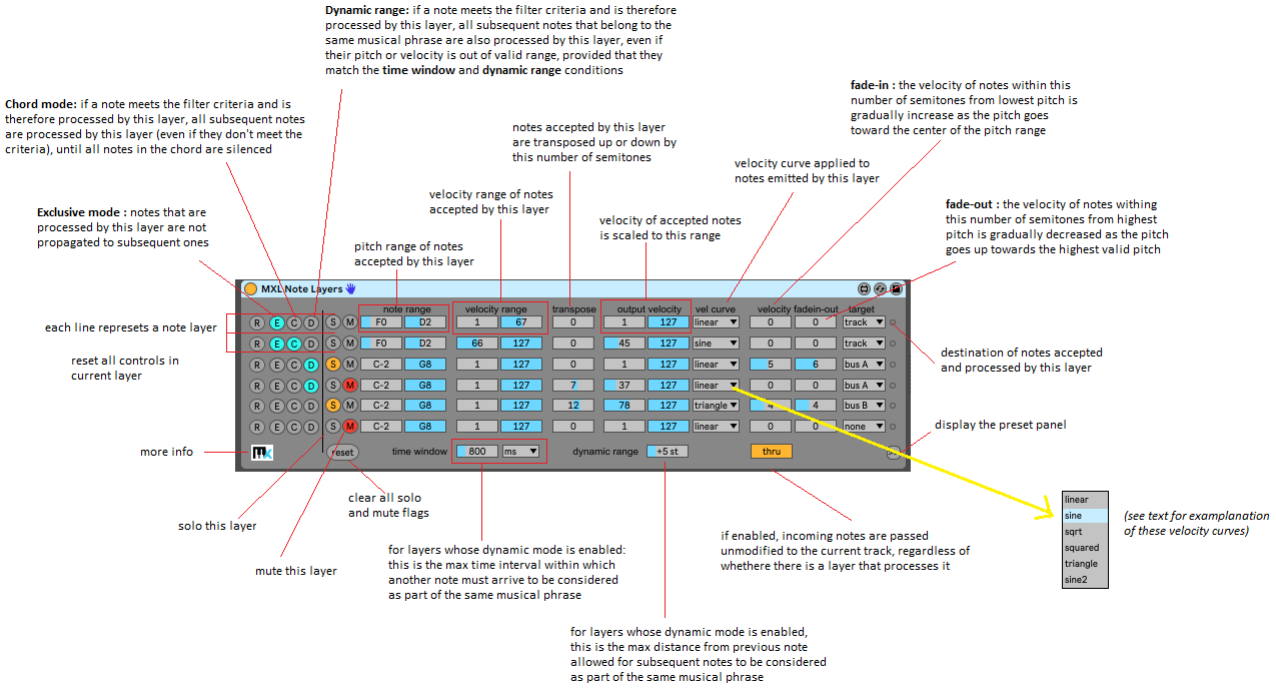
The **remote relative (drop)** option works in the same way, except that notes whose calculated velocity is zero or negative are dropped and don’t play at all. The **fixed** option is self-explanatory.

This deceptively simple device has few options, yet it is highly versatile and can be used for a variety of musical effects, both when performing live and when playing clips. As mentioned before, even when you use pre-recorded clips for both track and remote notes, you can get very interesting (and unexpected) results if the duration of either clip is *not* an exact multiple of the duration of the other clip.

MXL Note Layers



This device allows you to define up to six “layers of notes”: you decide which notes a layer can process and how those notes must be processed. It is similar to the key/vel filters you find in Live’s MIDI Racks and Instrument Racks, but offer many more options.



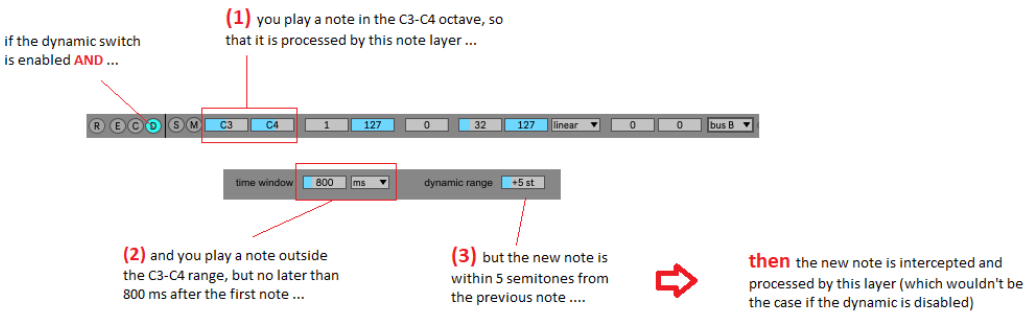
Like many others MXL devices, MXL Note Layers offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.

Let’s examine the controls that affect the criteria a layer uses to decide or reject a note. The **note range** and **velocity range** fields to filter out notes are quite intuitive, as are the **solo** and **mute** switches. (Tip: use the reset button near the bottom edge to reset all solo/mute switches.)

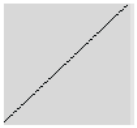
By default, all layers process incoming notes and then pass them along to the subsequent layer. However, if the **exclusive** mode is enabled, accepted notes are *not* passed to other layers. You can use this feature in many creative ways; for example, you can set the target menu to “none” to ensure that notes with high velocity are not played at all:

If enabled, the **chord** feature ensures that all the notes of a chord are sent to the same destination, even if their velocity is outside the allowed range (provided that their pitch is in **note range**). For example, the following setup allows you to play chords with two different timbres using the same area of your MIDI keyboard; if the first note of the chord has low velocity then that note and all other notes of the same chord will be sent to Bus A, even if the other notes’ velocity is higher than 32. Notice that the **exclusive** switch (“E”) ensures that notes intercepted by the first layer are not propagated to the second layer:

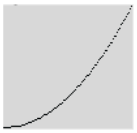
The **dynamic** switch (“D”) allows you to create melodies with a given instrument – that is, a specific **target** - in one area of your MIDI keyboard and ensure that you continue to use that timbre even if subsequent notes go outside the **note range** setting, as the following figure illustrates:



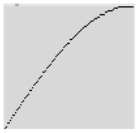
The **vel curve** field lets you select a velocity curve for all notes intercepted by the current layer. The usual curve shapes - **linear**, **sine**, **sqrt** (square root) and **square** – specify how fast or slow the output note's velocity grows when the original note's velocity grows. The remaining two – **triangle** and **sine2** – are peculiar in that the output velocity grows only in the first half of velocity valid range, and decreases after that point:



linear : the output velocity is equal to input velocity (if velocity range and output velocity are identical)

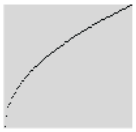


square : low velocities are de-emphasized, meaning that you have to play harder to get a specific output volume

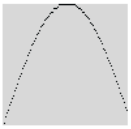


sine : low input velocities are slightly emphasized

triangle : output velocity grows faster than usual until it reaches the midpoint of range, than it decreases and becomes zero when pressing the key with max strenght - this setting is useful if you have 2+ layers that morph from one timbre to another depending on how strongly you press the key on your keyboard

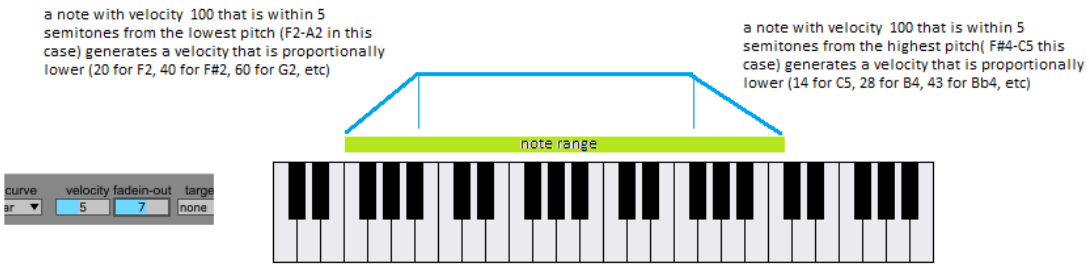


sqrt : low input velocity are strongly emphasized - meaning that you can play softly and get higher volume than usual

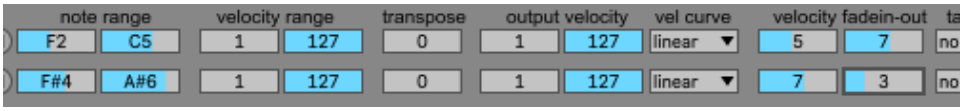


sine2 : similar to triangle, except the velocity increases and decreases more quickly near the range extremes

The **velocity fadein-out** fields offer a way to "blend" different timbres assigned to different areas of your MIDI keyboard, so that you can have a smooth transition from one timbre to the next one in the pitch regions where two instruments overlap. Let's see how fade-in and fade-out settings work:

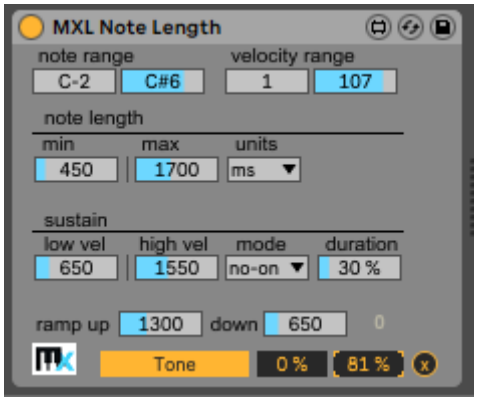


Now, consider these two layers:

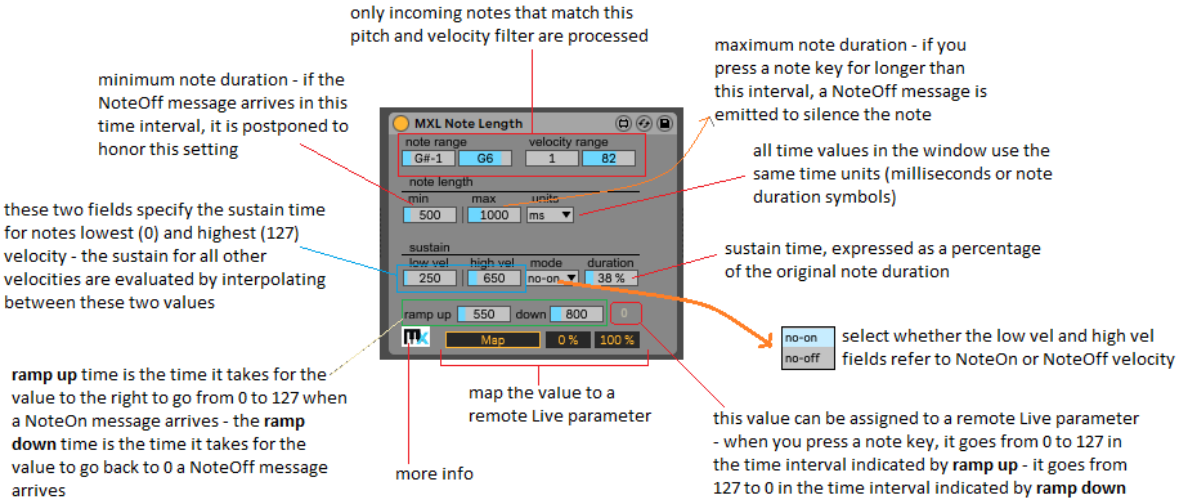


As you see, they overlap in the seven notes from F#4 to C5: by setting the fade-out velocity for the first layer equal to the fade-in velocity of the second layer, you can achieve the desired smooth transition when you go from the former to the latter range.

MXL Note Length



This device allows you to control minimum and maximum note duration, to sustain a note after you release the key on your keyboard, and to remotely change a Live parameter when keys are pressed and released.



To leverage the MXL Note Length device it is essential to understand what happens when it receives a NoteOn and NoteOff message.

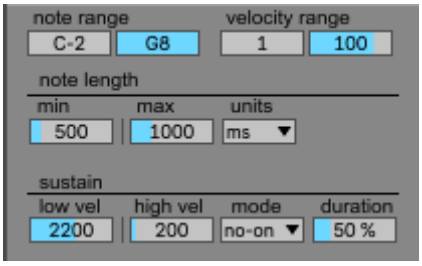
1) If the incoming note doesn't match the **note range** or **velocity range** filter, it is ignored and passed as-is to the next device or instrument in current track.
2) If nonzero, the **note length: min** value is the minimum note duration: if a NoteOff message arrives in this interval it is held and emitted only when the interval expires (or if another NoteOn message for the same note arrives in the meantime). Notice that the **units** field specifies whether this time interval is expressed in milliseconds or a note duration symbol.
3) If nonzero, the **note length: max** value is the maximum note duration: even if you continue to press the key on the MIDI keyboard after this interval, a NoteOff message is emitted anyway to silence the note.

At this point, sustain time comes into play, which postpones the moment when sound actually terminates. The sustain time depends on two factors: the note velocity and the original duration of the note (that is,

the time while you pressed the key on the keyboard). The actual sustain time is given by the sum of the sustain time calculated from the note velocity and the sustain time calculated from note duration. The **mode** menu specifies whether the device uses the NoteOn or NoteOff (release) velocity in its calculations.

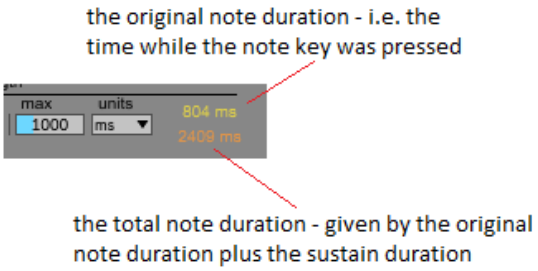
4) The **low vel** value is the sustain time for notes with the lowest accepted velocity, as specified in the velocity range fields; the **high vel** value is the sustain time for notes with the highest accepted velocity. If at least one of these two fields are nonzero, then the sustain time is calculated using the actual note velocity to interpolate between these two values. Notice that **low vel** can be higher than **high vel**, in which case louder notes are sustained for a shorter time interval.
5) If nonzero, the **duration** percentage specifies how the sustain time depends on the original note duration.
6) The total sustain time is evaluated as the sum of the two durations evaluated in the two previous points.

To see how all these values concur to sustain the note, say that you hit the C3 key on your keyboard, with velocity equal to 50 and for 800 milliseconds, with the following settings:



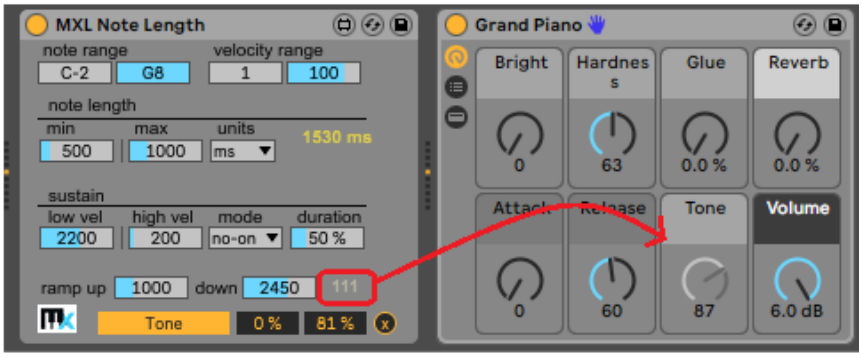
The value 800 included in the **min-max** range, therefore MXL Note Length does not artificially shorten or lengthen the note duration. The **mode** menu is set to ***note-on***, therefore the value 50 will be used to evaluate an additional sustain time. Being 50 is the middle point in the velocity range interval, the sustain time is the middle point in the **low vel** – **high vel** interval, which is equal to 1200 milliseconds. The duration field says that sustain time is half of the original note duration (800 / 2 = 400 milliseconds), therefore the resulting sustain time is 1200 + 400 = 1600 milliseconds.

This sustain time brings the total note duration to 800 + 1600 = 2400 milliseconds. This is confirmed by the values that appear to the right of the **units** menu:



Notice that the two durations showed can – and usually do – differ from the theoretical values. The discrepancy is usually under 10 milliseconds and doesn't really affect the overall effect.

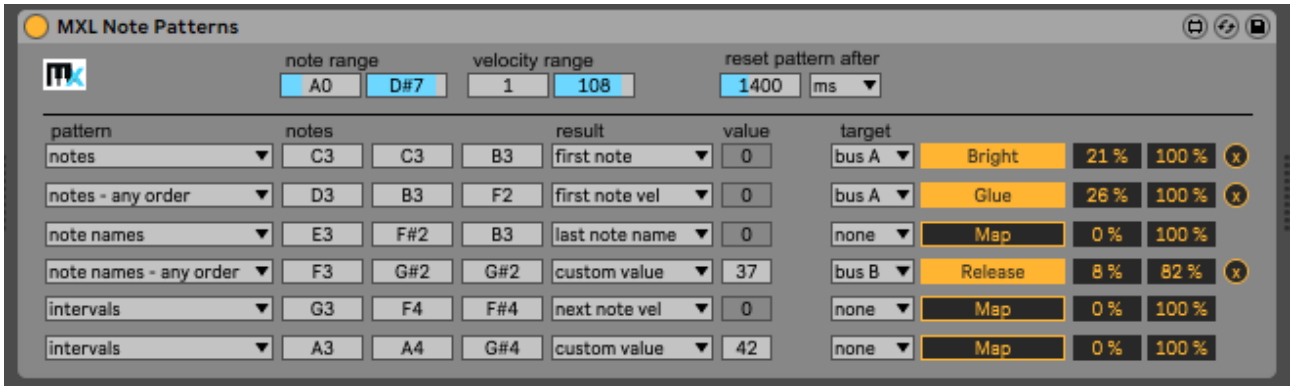
The MXL Note Length device allows you to map a ***ramp value*** to a remote Live parameter. The ramp value starts growing from 0 to 127 when you press a key on the keyboard, and then goes back to zero when you release the key. The **ramp up** field specifies how long it takes to reach the value 127 when you hit the key, while the ramp down field specifies how long it takes to go back to 0 when you release the key. The actual value is shown to the right of the **ramp down** field:



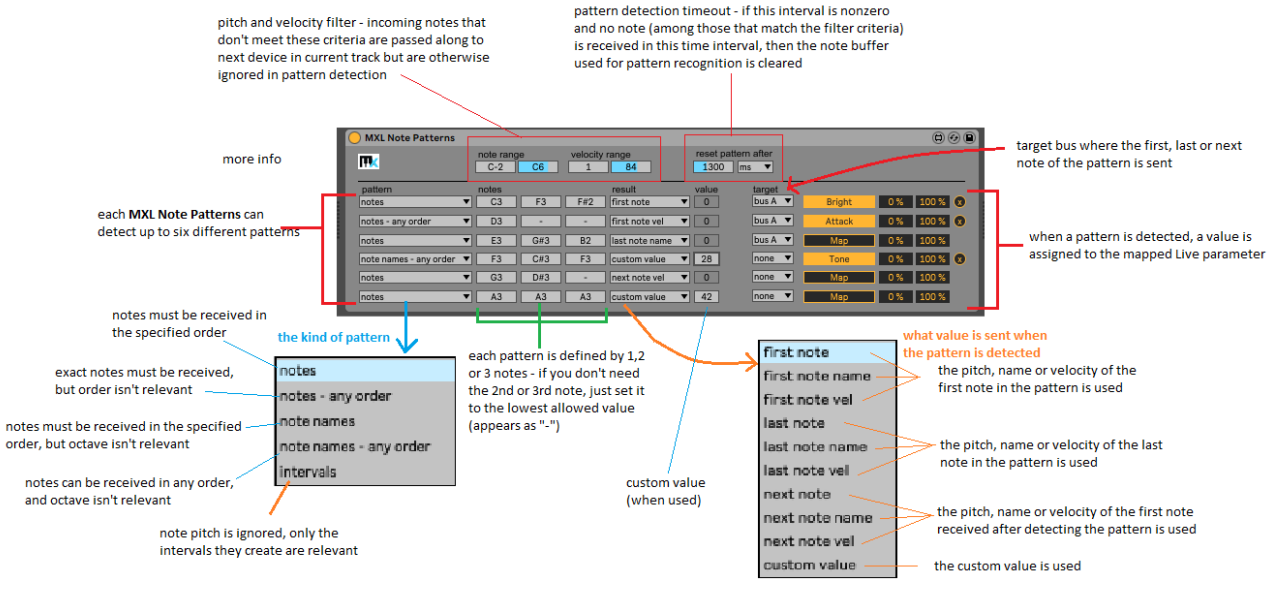
Notice that the ramp up sequence occurs only for the first note after a rest: if you play a chord, only the first key you hit triggers the ramp up sequence. Likewise, the ramp down sequence occurs only when no more keys are pressed.

To see a variation of this behavior that uses a richer ADSR or DAHDSR curve, please check the MXL Envelope device.

MXL Note Patterns



This device allows you to detect patterns in incoming notes and – when a pattern is detected - to use pitch, name, or velocity of such notes to control other parameters in Live, in same or different track. Each instance of MXL Note Patterns can detect up to six different patterns: if you need to handle more patterns, just add more instances of this device.



MXL Note Patterns is “transparent” to MIDI messages, in the sense that all incoming messages are passed along to the next device in the same track. However, if incoming notes satisfy the **note range** and **velocity range** filter, they are analyzed to check whether they match one of the six defined pattern.

Each note that passes the pitch/velocity filter is appended to the ***note buffer*** and such buffer is then compared to the defined patterns. The buffer always contains the three most recent notes (or fewer): when the fourth note arrives, the oldest note in the buffer is discarded. Moreover, if time interval indicated by the **reset pattern after** field elapses without receiving any note (that passes the filter), then the buffer is cleared. (If the timeout is zero, this step never occurs.)

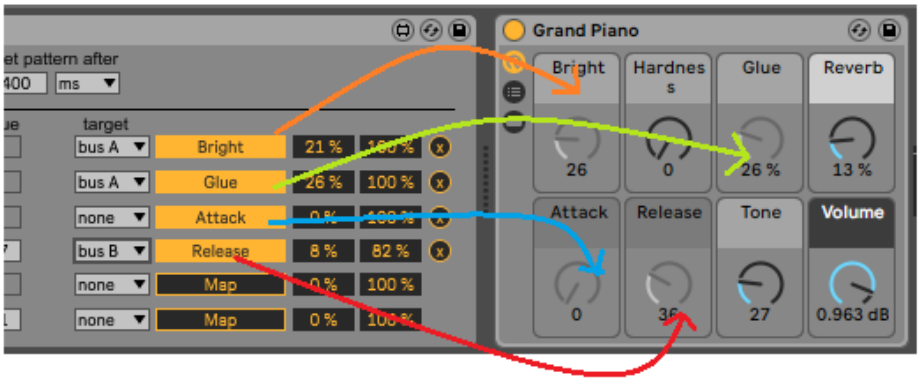
Each pattern consists of one, two or three notes. In the examples that follow, we will assume the pattern is defined as C3-E3-G3. These notes are constantly compared with note buffer, but the algorithm used for comparison depends on the option that you selected in the **pattern** menu:

notes: the pattern is matched if the buffer contains the three notes, in specified order (in our example, if the C3-E3-G3 notes are received in this order).
notes – any order: as above, but order is not significant; in our example, the C3-G3-E3 and G3-E3-C3 sequences would both match the pattern.
note names: the pattern is matched if the buffer contains three notes with the specified name, in that order; in our example, the C2-E2-G3 and C4-E3-G4 sequences would both match the pattern.
note names – any order: as above, but order is not significant; in our example, the C3-G2-E2 and G2-E3-C4 sequences would both match the pattern.
intervals: notes pitches are not significant, only the intervals the form are; in our example, the C3-E3-G3 form two intervals (4 semitones + 3 semitones), therefore any set of three notes that creates these intervals – in other words, any major triad arpeggio that starts on root note – matches the pattern. (Notice that when the **intervals** option is selected, the pattern should include at least two notes.)

When a pattern is matched, **MXL Note Patterns** briefly display the value assigned to the remote Live parameter, if a mapping has been defined. Moreover, unless the target menu is set to **none**, a note whose pitch corresponds to that value is sent to the Bus A-P. (For example, if value is 62 then the note D3 is sent to the specified bus.)

> ***Note**: if the current set of incoming notes matches two or more pattern, each of the matching pattern sends the value to remote Live parameter and a note to the target bus. However, the order in which these values or notes are sent is undefined, therefore it is recommended that you don’t define “overlapping” patterns or – if you do – at least that they send notes to different buses.*

The main purpose of the MXL Note Patterns is controlling a remote Live parameter. This device is very flexible and powerful: as usual, with power comes a certain degree of complexity, it is crucial to understand how to get the best out of this device.



The **result** menu dictates how the value is evaluated. This menu contains ten options, some of which require an in-depth explanation. The last option, named **custom value,** is the simplest one: the value sent is taken from the **value** field in the pattern row.

With the custom value option off the table, let's focus on the first nine options, which can be subdivided in three groups of three options each.

The first group consists of the **first note**, **first note name** and **first note vel** options: when one of these options is selected, the first note of the matching pattern is used to generate the result, which is – respectively – the note pitch, the note name (e.g. 0 for C, 1 for C#, etc.) or the note velocity. This value is also the note sent to the target bus, if one is selected in the **target** menu.

Notice that the first note of the matching pattern does not necessarily corresponds to the note that appears in the pattern definition. In fact, assuming that the pattern is defined as C3-E3-G3 and the pattern type is **notes – any order**, then the first note can be E3 or G3. Likewise, if pattern type is **note names – any order**, then any C, E or G note can be the first note. Finally, if pattern type is set to **intervals**, then ***any*** note can be the first note of the detected pattern.

The second group of options in the **result** menu is made of **last note**, **last note name** and **last note vel**. This group is similar to the first group, except it uses the last note in the pattern. Notice that this is not necessarily the third note currently in the buffer that holds incoming notes, because a pattern can consists of two notes, or even just a single note.

The third group of options in the result menu contains **next note**, **next note name** and **next note vel**. In this case, the result is derived from the ***first*** note that matches the note/velocity filter and that arrives immediately after a given pattern has been detected. In this case, the result may have no relationship with the notes that define the pattern.

However, knowing what note is used to generate the value assigned to the remote Live parameter is only part of the story, because the actual value being assigned depends on other factors too:

- **first note**, **last note** and **next note** options: the pitch of the note – which is a number in the range 0-127 - is used to generate the result. If the **note range** filter is set to C-2 to G8 (i.e. it accepts all notes), then this number is assigned as-is to the target Live parameter. In all other cases, the pitch number is scaled so that the first note in the interval generates zero and the last note in the interval generates 127, the median value generates 64, and so forth.

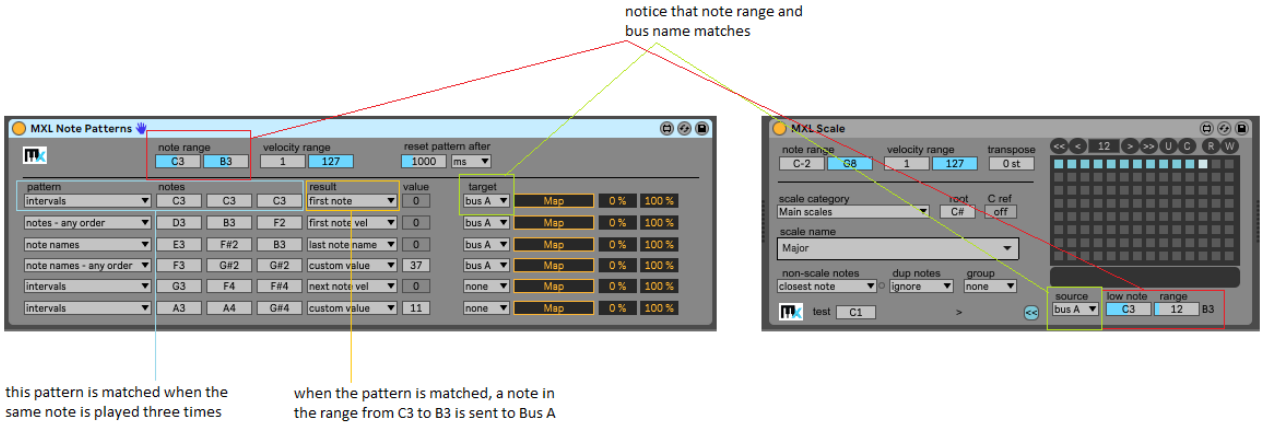
  For example, if the filter is set to C1-D#2 (16 semitones) and the selected note is C1, the value being generated is zero; the note C#1 generates the value 8; the note D1 generates the value 16, and so forth, until the note D#2 that generates the value 127. Thanks to this scaling mechanism, you can assign the full range of any Live parameter.

- **first note name**, **last note name** and **next note name** options: in these cases the note generates a value in the interval 0-11 (corresponding to notes C to B); this value is scaled to the interval 0-127, therefore the note C# corresponds to 10, the note D corresponds to 20, and so forth. Again, this mechanism allows covering the full range of any Live parameter.

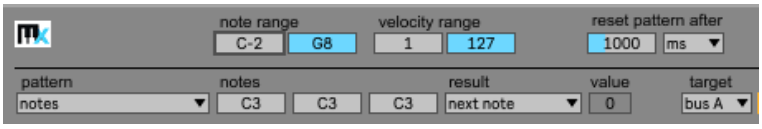
- **first note vel**, **last note vel** and **next note vel** options: the velocity of the selected note – which is a value in the range 1-127 - is used to generate the result. If the **velocity filter** is set to 1-127, then the velocity is assigned as-is to the target Live parameter. In all other cases, the velocity value is scaled to the velocity range filter, so that the lowest accepted velocity generates the value 1 and the highest accepted velocity generates the value 127. As in previous cases, this scaling mechanism allows covering the full range of any Live parameter.

For more information, read the Parameter Mapping section, earlier in this manual.

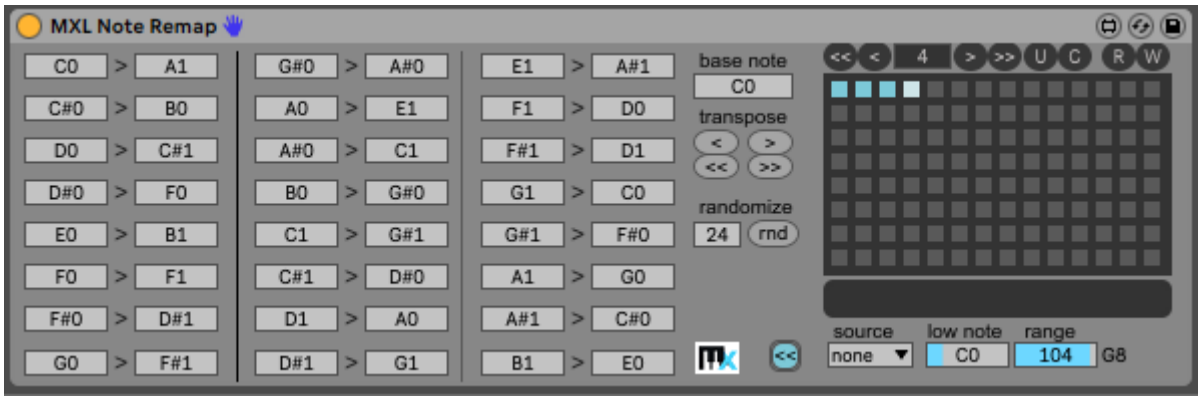
In addition to controlling a remote Live parameter, the MXL Note Patterns device can be used to select a preset in MXL devices that support them. In the following example, we have defined a very simple pattern – the same note played three times – that can be used to select one of the 12 presets defined in a MXL Scale device, in same or different track:



Here's another example: if you play the C3 note three times, then the ***next*** note is used to select a preset. In this case, there is virtually no limit to the number of presets you can select:

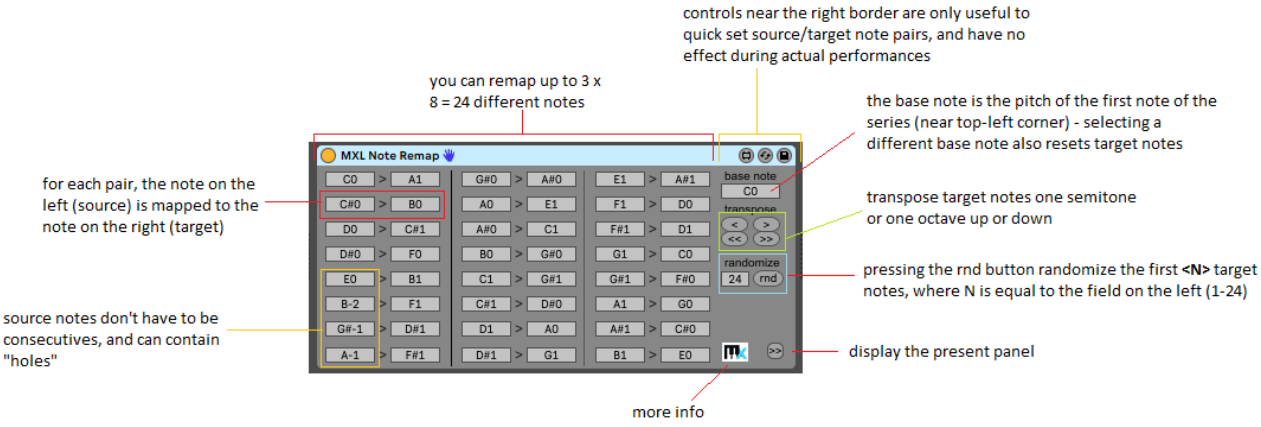


MXL Note Remap



This device allows you to remap up to 24 notes to other notes of your choice. It is intended to retune notes coming from MIDI drum-like controllers – e.g. Roland Handsonic – that emit a fixed set of notes.

MXL Note Remap offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.



You can check how incoming notes are remapped by looking at the area near the bottom right corner; notes that are remapped appear in orange:

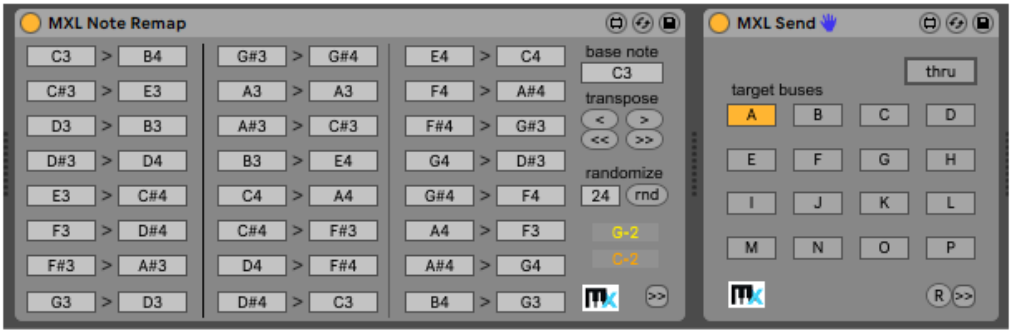
Notice that notes that are not remapped are not played at all.

If you need to remap more than 24 notes, you can use two MXL Note Remap devices in parallel, in two distinct tracks that send their output to the same instrument:

the MXL Note Remap in main track
remaps notes from C1 to B1

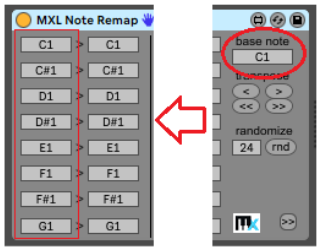
... then sends the result to the **MXL Receive** device in main track

the MXL Note Remap in the secondary
track remaps notes from C3 to B4

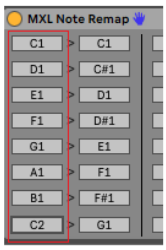


Even if the main purpose of MXL Note Remap is customizing the behavior of MIDI controllers, its randomization feature offers you a simple way to generate new melodies from existing ones. For example, say that you want to randomize the notes in C Major scale, in the octave between C1 and C2, so that they map to another note in the (larger) interval from G0 to F2. Here's how to proceed:

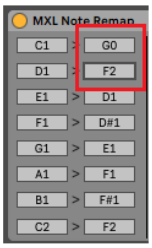
(1) use the base note field to reset source/target pairs



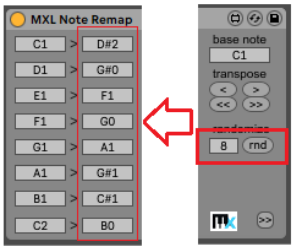
(2) change source notes to match the C Major scale in the octave of interest



(3) set any two target notes equal to lowest and highest notes you want to obtain after randomization

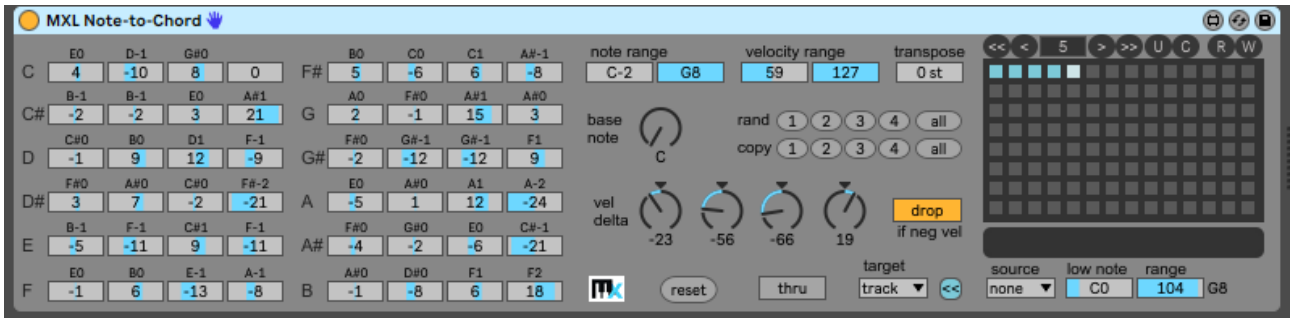


(4) set the number of targets to randomize and click on the rnd button



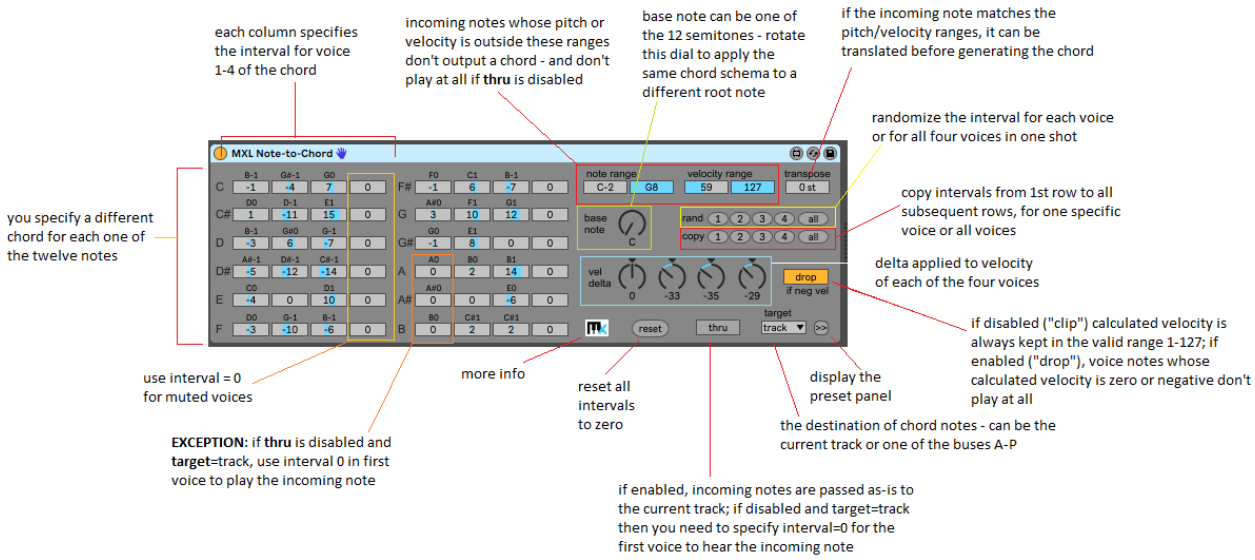
Notice that you can now randomize repeatedly by simply clicking on the **rnd** button. This button can be operated using standard Live automation.

MXL Note-to-Chord



This device is the ultimate harmonizer, in that it allows you to specify up to four intervals (or voices) for each of the 12 semitones, and optionally apply a velocity offset for each of the four voices.

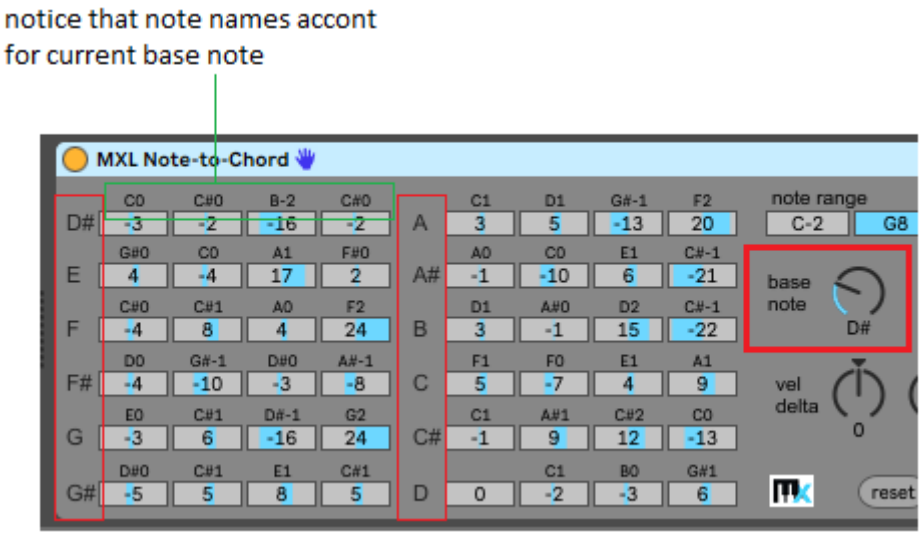
MXL Note-to-Chord offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.



MXL Note-to-Chord allows you do define a 4-voice chord for each of the twelve semitones, or 5-voice chords if you include the incoming note itself (by enabling the **thru** option). This flexibility comes with a price, though, because you have to set as many as 48 controls to configure this device.

To help you in this task you can use the **rand** row of buttons (which randomize the intervals in a specific voice or all voices at once) or the **copy** row of buttons (which copy the intervals in first row to all subsequent rows). The **reset** button sets all 48 intervals to zero.

The **base note** knob is a great time-saver when you want to implement the same harmonization to a different root note. For example, say that you have carefully configured all the chords that sound nicely on a C Major scale. If you rotate this knob, you can transpose the same intervals to another root note. As you see in the figure below, the intervals actually don't change, only note names do:



Use the **vel delta** knobs to control the relative velocity of the notes that make the chord. The velocity delta is applied to the velocity of the incoming note; if the evaluated velocity is zero or negative, it either is kept in the correct range 1-127 or doesn't play at all, depending on whether the **clip/drop** button is enabled or disabled.

Keep in mind that you can store the current configuration in one of the 100+ presets and recall it later using a variety of ways, such as sending a note to one of the buses A-P.

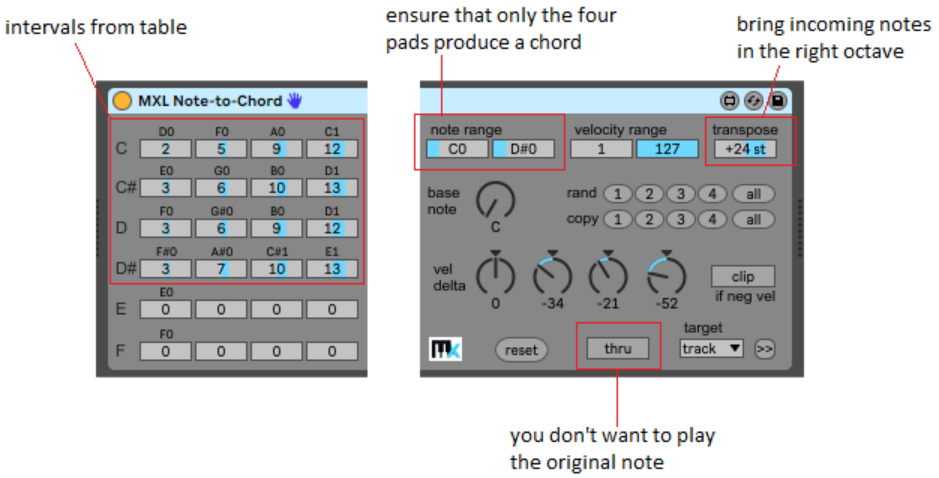
The MXL Note-to-Chord device is useful also to program the pads found on MIDI keyboards or controllers. In those cases, usually the note received from the controller has little to do with the chord you want to play. In this scenario, you may want to disable the **thru** option to omit the incoming note, and you need to do some math to calculate the actual intervals. Let's see how.

For simplicity's sake, say your MIDI controller has four pads that emit notes C1 C#1 D1 and D#1, and that you want to use them to play the following chords: Dmaj7, Emin7, Ddim7 and F#7 in the octave that starts at C3. The interval between the incoming and desired note is often more than 24 semitones and is therefore larger than the intervals that can be expressed for the four voices, but you can use the **transpose** field to zero this gap.

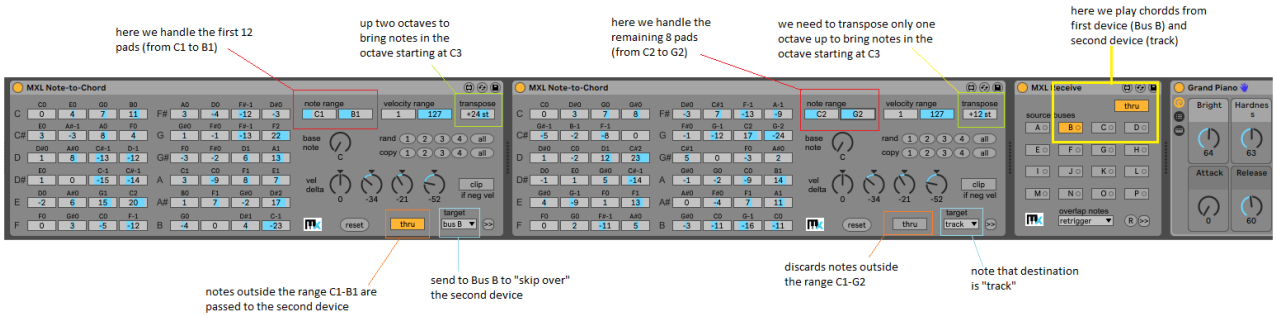
Once you have moved the incoming note in the right octave, you calculate the distance in semitones of each desired note from C3 and you correct these values to account for the pad number. The following table shows all the values involved:

pad #	original note	transposed note	chord	chord notes	semitones from C3	correction for Nth pad	intervals to use
1	C1	C3	Dmaj7	D3 F3 A3 C4	2 5 9 12	0	2 5 9 12
2	C#1	C#3	Emin7	E3 G3 B3 D4	4 7 11 14	-1	3 6 10 13
3	D1	D3	Fdim7	F3 G#3 B3 D4	5 8 11 14	-2	3 6 9 12
4	D#1	D#3	F#7	F#3 A#3 C#4 E4	6 10 13 16	-3	3 7 10 13

These are the settings for the device that correspond to the above table:

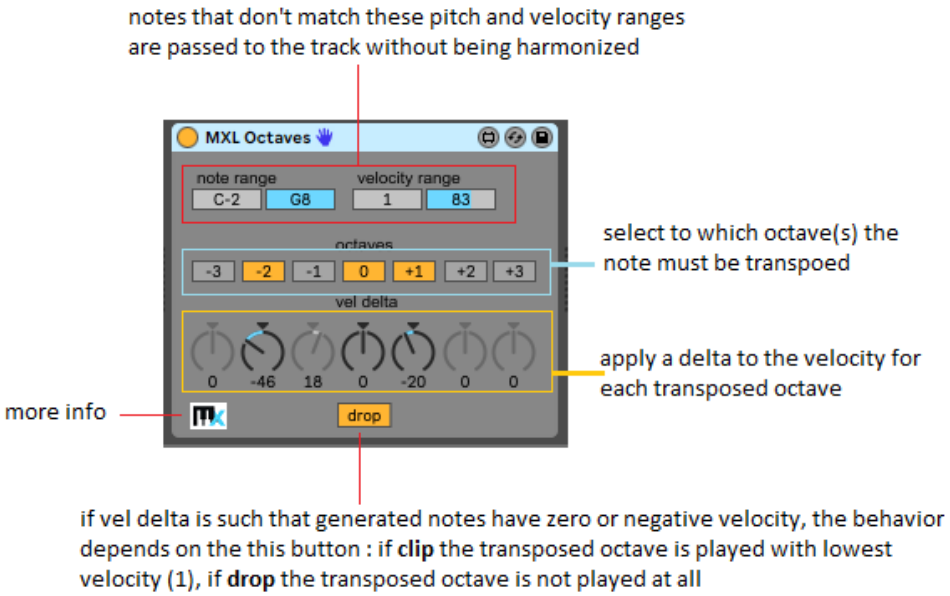


You can extend this mechanism to all 12 notes. If your MIDI controller has more than 12 pads, you can still map them to chords by using two or more **MXL Note-to-Chord** devices, as shown in the following figure:

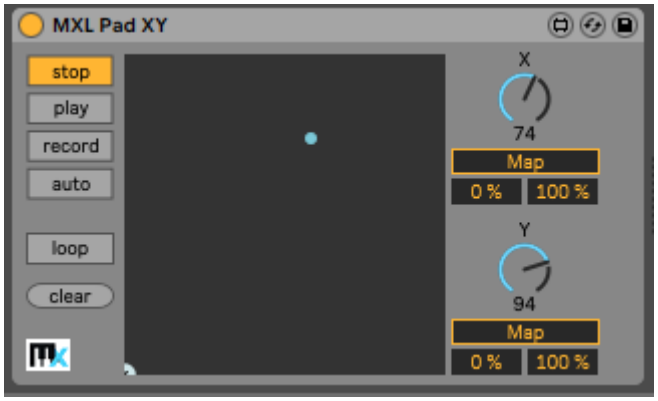


MXL Octaves

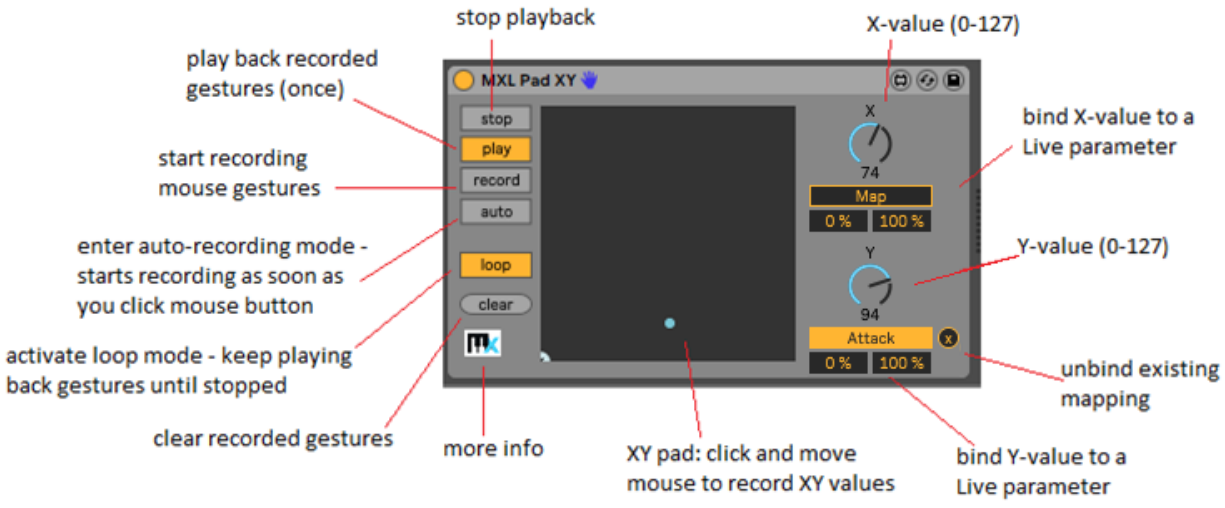
This device allows you to transpose notes to different octaves and control the velocity of each transposed octave. You can achieve the same effect using other and more powerful MXL devices, for example MXL Chord, however in many scenarios MXL Octaves is preferable because it can be configured more quickly and easily.



MXL Pad XY

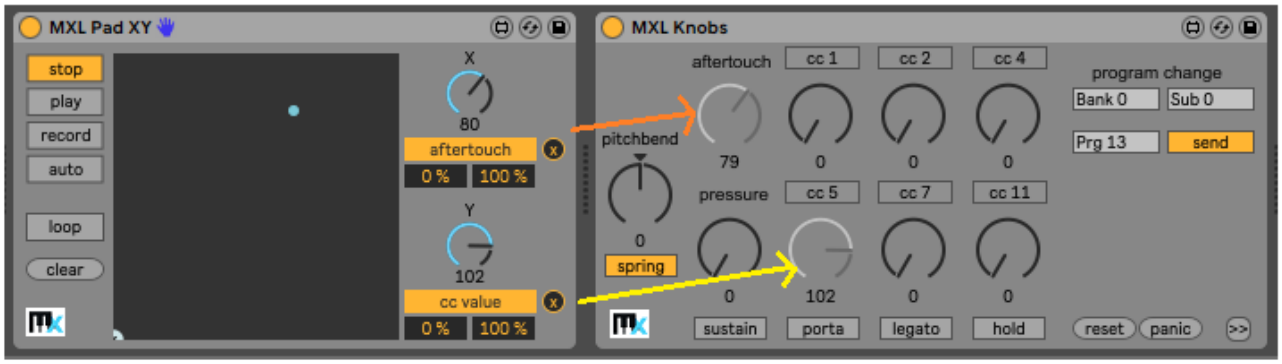


This device allows you to control two Live parameters using an XY pad. You can also record mouse movements and playback them in one-shot or loop mode.



Notice the minor difference between **record** and **auto** button: the former account for time elapsed between the click on the button and the first click on the pad; the latter starts counting when you click the pad, therefore playback starts immediately.

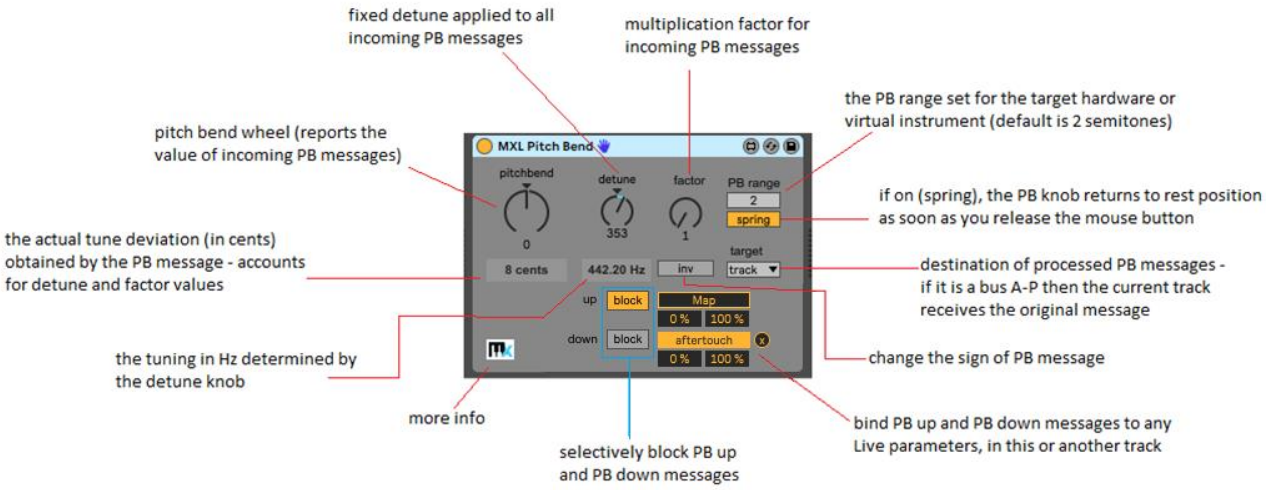
The main purpose of MXL Pad XY is controlling two Live parameters with the mouse, however you can pair it with another MXL device to send MIDI message. For example, you can control two dials in the MXL Knobs device to send Aftertouch and Control Change (CC) messages:



For more information, read the Parameter Mapping section, earlier in this manual.

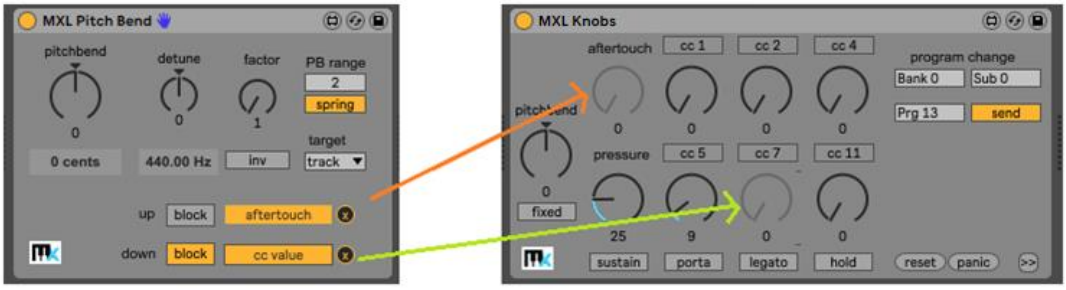
MXL Pitch Bend

This device provides several functions related to PitchBend (PB) messages, such as selectively blocking PB up and PB down messages, scaling their value, and inverting the wheel direction. It also allows mapping the value of PB messages to a Live parameter and send the PB message to an alternate destination.



The MXL Pitch Bend device behaves differently, depending on what you select in the **target** menu: if destination is the current track (default) the incoming PitchBend message is processed according to the state of the various fields and the passed along to the Live device or instrument to its right in the current track. If destination is a bus A-P then the incoming PB message is sent as-is to the track and a new PB message goes to the specified bus.

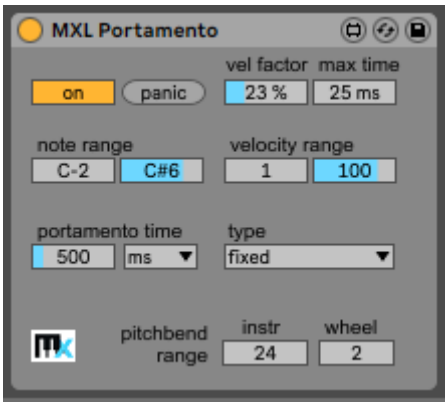
You can use the two **Map** fields to map either the PB up or PB down messages to a Live parameter. When used in this fashion, MXL Pitch Bend provides a simple mechanism to use the pitchbend wheel as a pair of additional knobs on your MIDI keyboard. Because you can map it to a knob in a MXL Knobs device, this mechanism also allows you to convert PB messages into Control Change (CC), Aftertouch or Polyphonic Pressure messages:



For more information, read the Parameter Mapping section, earlier in this manual.

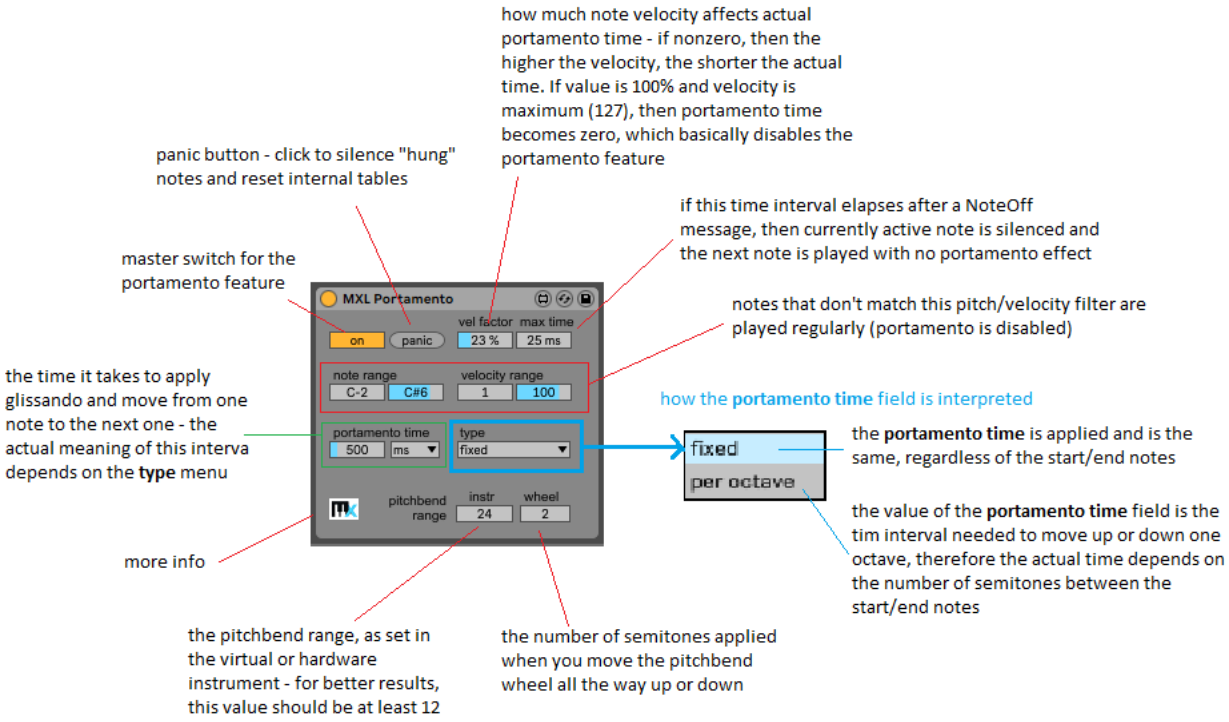
The ability to scale the incoming PB value and/or invert its signs allows you to implement a few interesting tricks by sending the processed PB message to a different track. If you have two instruments and you set the **factor** knob to any value other than 1, then the two tracks will receive different and yet "proportionally scaled" PB values. For example, if the **PB range** value is 12 semitones for both instruments and you set the **factor** field to **1/2**, then turning the pitchbend wheel half way up generates an interval of 6 semitones on current track and 3 semitones on the "remote" track, thus forming a minor 3rd interval. If you use the **inv** switch to change the sign of the PB message, then the remote track would play 3 semitones *below* the current note and the resulting interval would be 9 semitones (or a major 6th). Obviously, while the wheel is moved many dissonant intervals would be generated, but this might be exactly the effect you are looking for in some scenarios.

MXL Portamento



This device adds portamento playing style to any Live instrument, including those that lack this feature. More precisely, MXL Portamento uses pitch bending to move from one pitch to another, without emitting new NoteOn messages. For this device to work properly, it is essential that the target instrument – which

can be a Live virtual instrument or an external hardware instrument – support wide ranges for the Pitch Bend feature, and that such range matches the value you enter in the **pitchbend range – target** field.



MXL Portamento assumes that note source is monophonic: if a note arrives while the previous note is still playing, then the previous note is silenced and a glissando effect is applied to move from the previous note's pitch to the new note's pitch.

The **type** menu dictates how the **portamento time** fields are interpreted. If type is **fixed**, then the time to move from one note to the next one is fixed and is equal to the value specified in the two fields. If **type** is **per octave**, then the time specified in the two fields is the interval needed to move up or down of one octave, therefore the actual portamento time depends on the interval between the two notes. For example, if **portamento time** is set to 1200 milliseconds and type is set to **per octave**, then it takes 300 millisecond to move from C to D# (3 semitones, or 1/4 of an octave), or 400 milliseconds to move from E to G# (4 semitones, or 1/3 of an octave).

Regardless of the **type** setting, the result of the above calculation should be considered as the "theoretical" portamento time, because if the **vel factor** field is nonzero, then the "actual" time depends also on the note velocity.

In general, the **vel factor** field dictates how much shorter the portamento time is when velocity is maximum. Assuming that theoretical portamento time is 600 milliseconds and **vel factor** is set to 50%, if velocity is maximum (127) then actual portamento time is reduced by 50% (= 300 ms); if velocity is 96 (75% of its maximum value), then portamento time is reduced by 50%*75% = 37.5% and becomes 195 milliseconds.

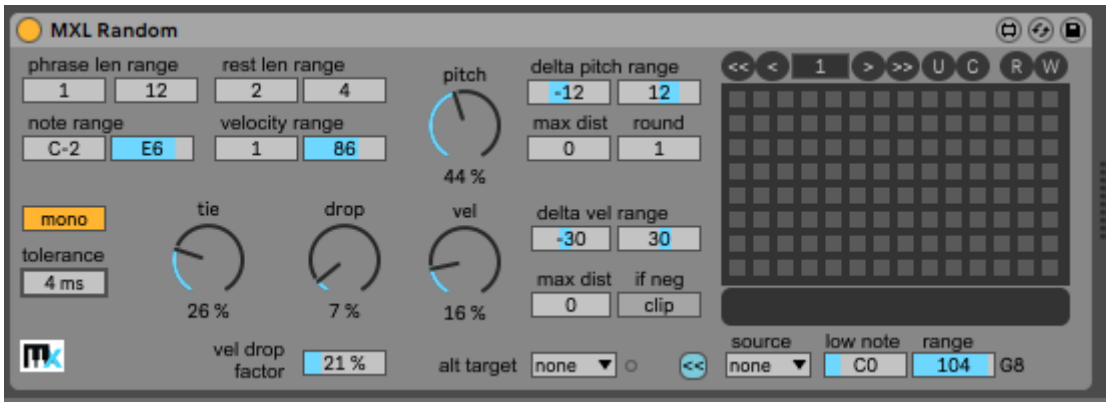
If **vel factor** is 100% and velocity is maximum (127), then portamento time becomes zero and portamento effect is disabled. In addition to this scenario, there are three more cases when portamento is disabled and a new note is played immediately at its right pitch, without any glissando:

- when the incoming note doesn't match the **note range** and **velocity range** filter
- when the current musical phrase extends to a number of semitones larger than the **pitchbend range – target** value

- when the time interval between the incoming note and the previous NoteOff message is larger than the **max time** value

As you see, the **max time** value is critical. If you are using a polyphonic MIDI controller – such as a keyboard – then you can keep this value low and still achieve the portamento effect if you hit the second note's key while the first note's key is still pressed. However, if you play a monophonic MIDI instrument – for example, a wind controller – then you might need to calibrate this value until you feel comfortable with your playing style and skills.

MXL Random

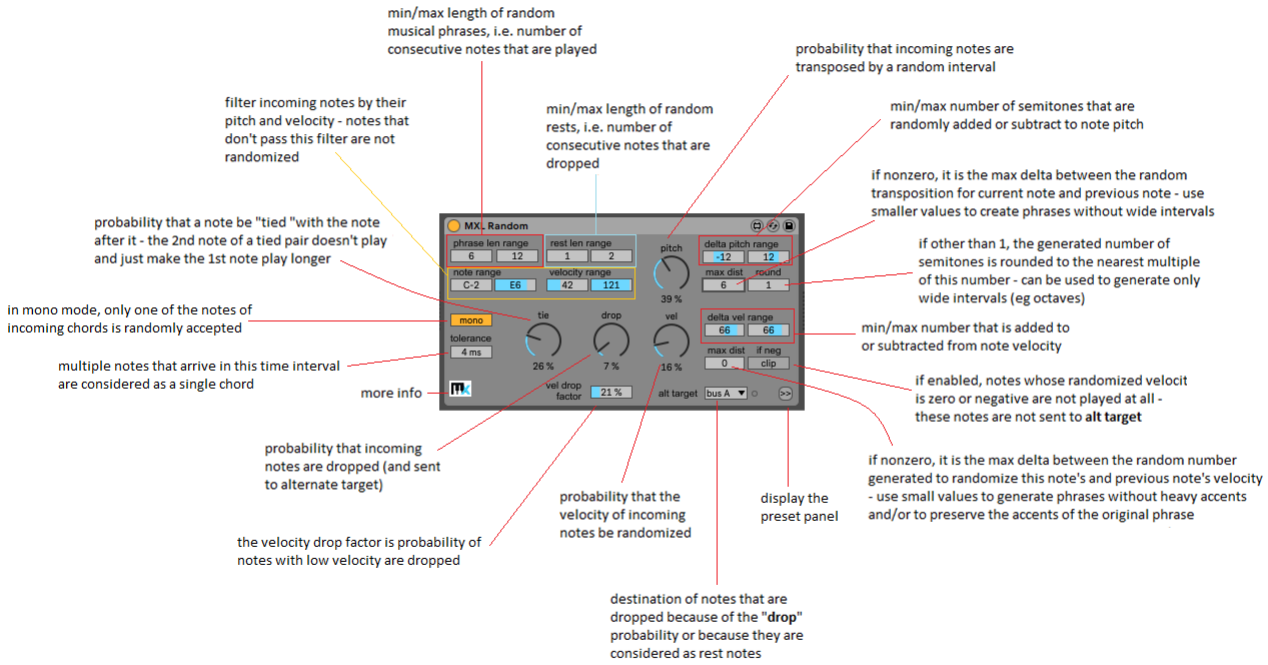


This device offers many ways to randomly drop notes and/or randomize their pitch or velocity. Unlike many other similar devices, it is designed to control "randomness" and create musical results rather than just a series of random notes. It works best when it processes an uninterrupted stream of notes: by selectively dropping notes, it creates musical phrases separated by rests.

MXL Random allows you to define what a *musical phrase* is, or how many notes must be played without any interruption (that is, without dropping any note in-between). Two consecutive musical phrases are kept apart by a ***rest***, which is defined as one or more incoming notes that ***are*** dropped.

Assuming that you use a source that emits a continuous stream of notes – typically, a clip in current track - you can define the minimum and maximum length of phrases in terms of how many incoming notes are played, using the **phrase len range** fields. Likewise, you can define the duration of rests in terms of how many incoming notes are dropped, by means of the **rest len range** field pair: if these fields are both zero, the output is just a long series of random notes with no interruptions, unless you set the **drop %** dial to a nonzero value.

Mono mode provides even more control on what notes can be played in any given moment: if this mode is enabled, MXL Random selects only one note of incoming chords, which is great to create random arpeggios and base lines.



To leverage the potential of MXL Random it is useful to understand what happens internally when an incoming note is detected:

1) The device decides whether the incoming note is part of the current musical phrase or is part of the current rest. In the latter case, the note is sent to the target indicated by the **alt target** menu. If this menu is set to **none**, this note is not played at all.
2) If the note belongs to the current phrase, it is passed through the filter established by the **note range** and **velocity range** fields: if the note doesn't pass this filter, it is not randomized and is emitted as-is.
3) if **mono** mode is enabled, the device uses the **tolerance** field to check whether the note belongs to the current chord; if this is the case, only one note in the chord is selected and becomes a candidate for randomization. All other, unselected chord notes are discarded and not sent anywhere.

4) If the note matches the filter then the **drop %** value is accounted for and the note might be randomly discarded. These discarded notes go to the **alt target** destination.
5) Furthermore, if the **drop vel factor** field is nonzero, then notes with lower velocity are candidate for being dropped (read below). These notes, too, go to the **alt target** destination.

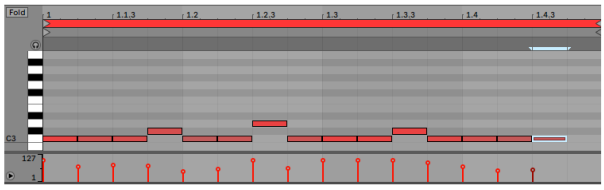
6) If the note is not discarded, then it becomes a candidate for pitch randomization, according to the **pitch %** dial. If this value is nonzero, then the note might be transposed by a random number of semitones in the range indicated by **delta pitch range** fields.
7) If the **max dist** field is nonzero, then the randomly generated interval is forced to be within that distance from the value generated for the previous note. Notice that this constraint is applied to the randomly generated number added to (or subtracted from) the pitch of the incoming note, and ***not*** to the pitch of consecutive notes. This means that two consecutive notes might form an interval wider than this value, if their original pitches are different.
8) If the **round** value is not 1, then the randomly generated number of semitones is forced to be a multiple of that value. In practice, you can use this field to generate wide intervals, for example an integer number of octaves.

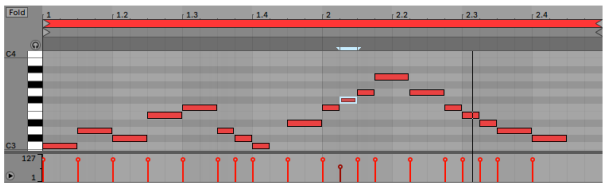
9) The result of all these calculations is the new pitch of the note. If this pitch is invalid – i.e. outside the range from C0 to G8, then the transposed note is not played at all (and is not sent to the **alt target** destination).

10) If the **vel %** knob is nonzero, then the velocity of the note might be changed by adding or subtracting a number to its original velocity. The range of this random number is indicated by the **delta vel range** fields.
11) If the **max dist** field is nonzero, then the randomly generated number is forced to be close to the number generated for the previous note. You can use this field to avoid strong unexpected accents in the generated phrase or to "humanize" a melody by slightly modify note velocity yet still preserving accents in the original melody.
12) If the **clip/drop** button is disabled then the resulting random velocity is automatically kept in the valid range 1-127. If the button is enabled and the resulting velocity is zero or negative, then the note is discarded (and not sent to **alt target** destination).

13) If the note has "survived" all the previous steps, the **tie %** knob is accounted for. If this value is nonzero then the note becomes a candidate for being "tied" with the subsequent note. When this happens, the subsequent note will not play and its effect will be to make the current note longer.

This device is typically used with notes stored in a Live clip. You can use a clip containing notes with mostly same pitch or clips containing musical phrases with a recognized shape.





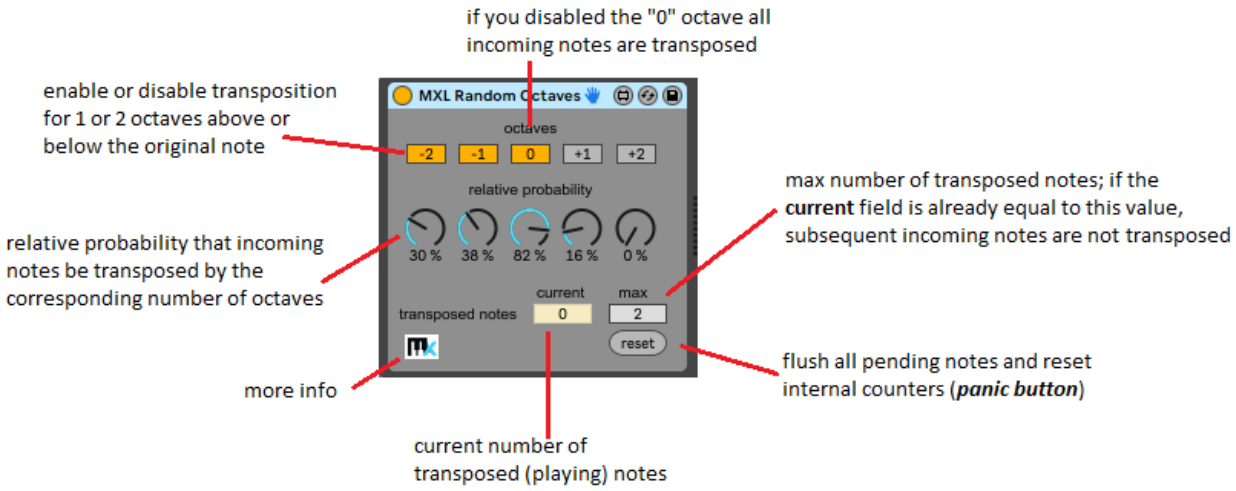
In the former case, you would probably use MXL Random to generate melodies with a high degree of randomness, for example with high values for the **delta pitch range** and **delta vel range** fields, and with appropriate settings for creating rests of suitable duration.

In the latter case, you may want to preserve the contour of the phrase, therefore you will select smaller values for those fields and you might decide not to create new phrases because you are satisfied with the rests already in the clip. In this scenario, you may want to set both values for the **rest len range** to zero and maybe silence some notes with the **drop %** knob, and tie notes together with the **tie %** knob.

Using this device with notes coming from a MIDI instrument is less common, yet it can deliver some interesting results. For example, you might use an arpeggiator to split longer (sustained) notes into many shorter ones that can be then randomized with MXL Random.

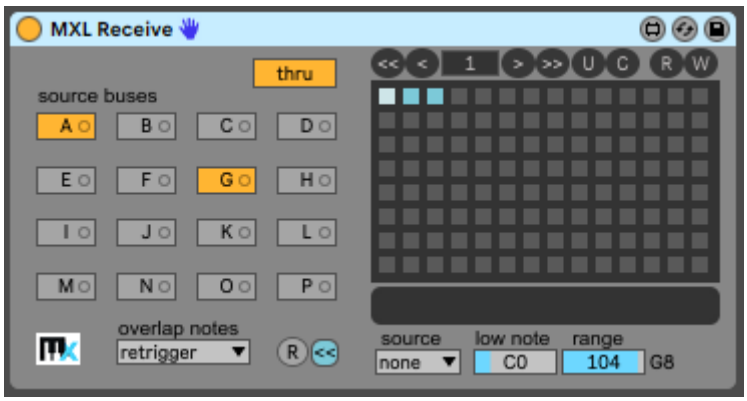
MXL Random Octaves

This simple device randomly transposes incoming notes by 1 or 2 octaves up or down, and allows you to specify the relative probability of each of transposition. Its main purpose is creating random inversions of incoming chords, so that a new yet harmonically equivalent chord is randomly created each time. Additionally, you can control randomness by setting an upper limit to how many transposed notes can play in any given moment.



Notice that setting the relative probability to zero is same as disabling the corresponding octave transposition.

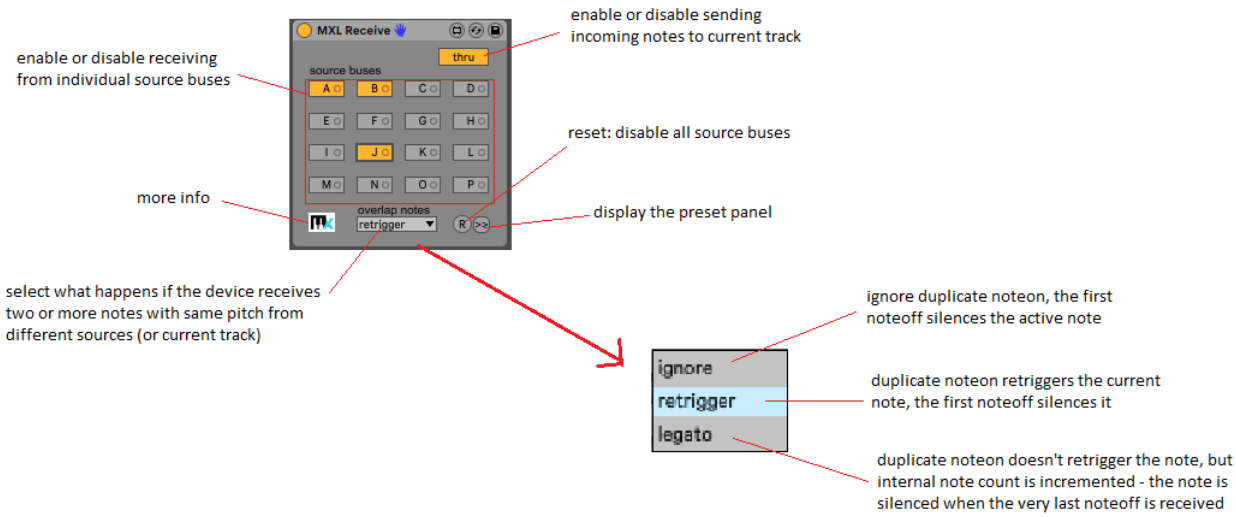
MXL Receive



This device allows you to receive MIDI data from other tracks, where you have placed an MXL Send device that is properly configured. It is conceptually similar to but offers many more features than the Max MIDI Receiver device that is included in the "Max 7 Pitch and Time Machine" pack. Much of the functionality of MXL Pack is based on this device and its MXL Send companion.

This device offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.

The first, immediately apparent difference from other similar devices is that you can receive from multiple sources, which are named Bus A, Bus B … Bus P. If the **thru** button is enabled (default), MIDI bytes received from the current track are passed unmodified to the next device. The **R** button resets the device by switching off all the A-P buttons.



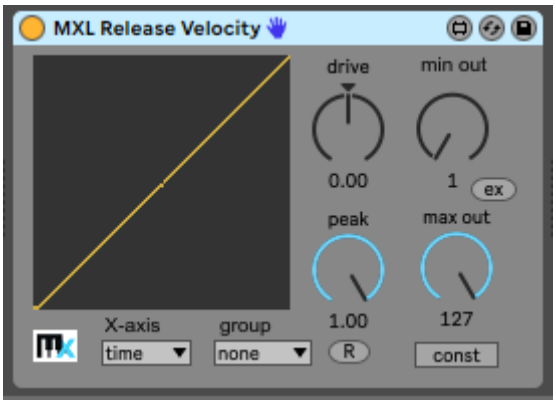
The **overlap notes** menu lets you decide what happens when the device receives a NoteOn MIDI message – from a bus or the current track - and a note with same pitch is already playing. Three settings are available:

- **ignore**: the incoming NoteOn message is ignored so that the note continues to play; the first NoteOff message silences the active note.
- **retrigger**: a NoteOff message is emitted before the incoming NoteOn, so that the note is retriggered; the first NoteOff message silences the active note.
- **legato**: the incoming NoteOn message is not emitted and the note continues to play; unlike the "ignore" setting, MXL Receive keeps track of how many NoteOn messages are received and silences the note when the very last NoteOff message is detected. In most cases, this setting provides the most "musical" results.

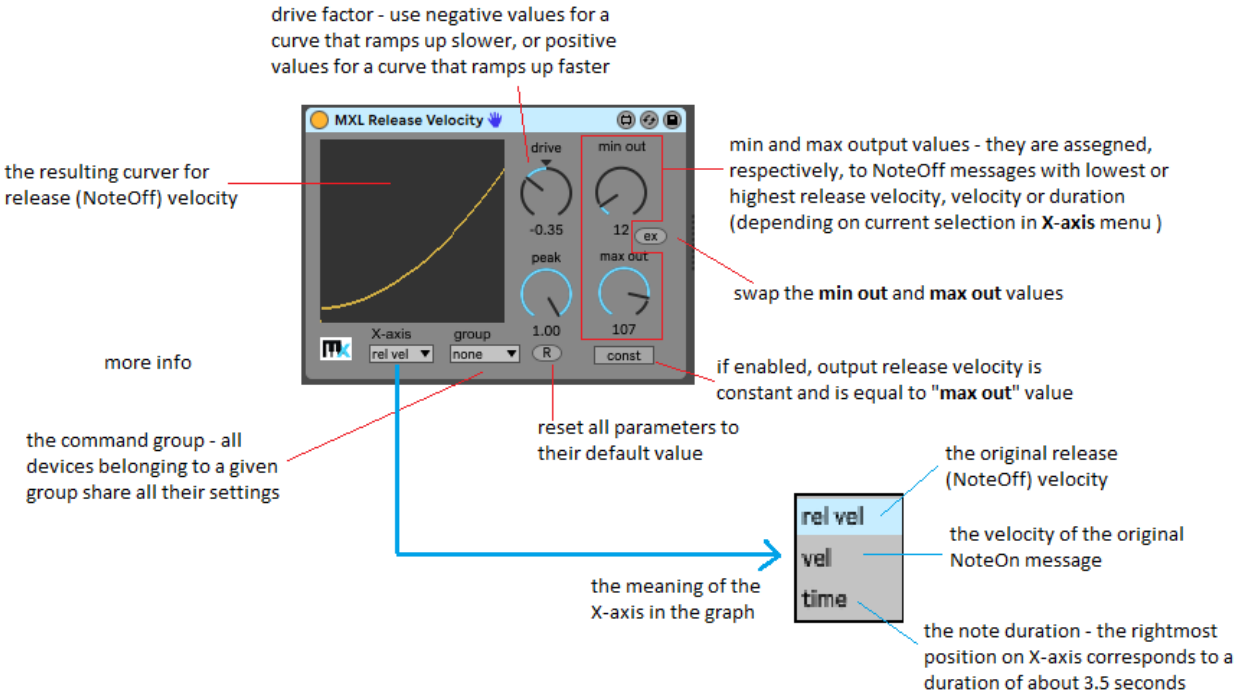
The ability to store current configuration in a preset and quickly recall it later provides a great degree of flexibility, especially if you consider that multiple MXL Send and MXL Receive devices can react in different ways to the same remote command.

There are many ways you can effectively combine MXL Send and MXL Receive devices. A very common one is having multiple clips sending their note to the same track (or virtual instrument), as if Live were supporting two or more clips per track.

MXL Release Velocity

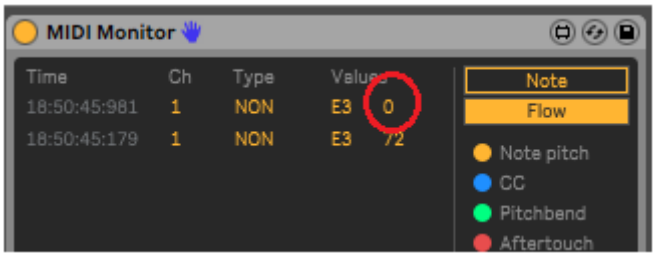


This device allows you to set a curve for release (NoteOff) velocity. It can generate a value that is based on the original release velocity (if your MIDI keyboard emits such information) or the attack velocity of the original NoteOn message, or it can compute a value based on note duration, i.e. how long you pressed the key on the MIDI keyboard.

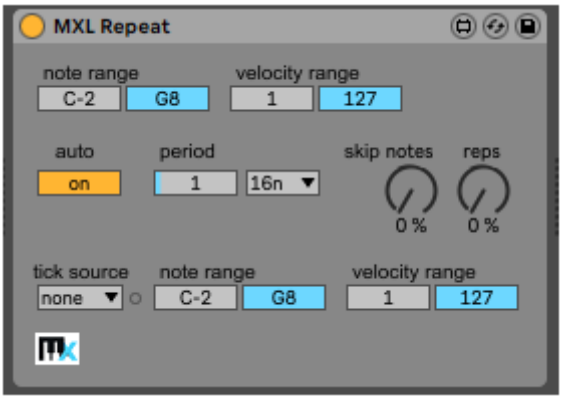


See the documentation of MXL Velocity for more information about how the **drive**, **peak**, **min out** and **max out** parameters, and how you can use the **group** menu to synchronize multiple instances of the MXL Release Velocity devices.

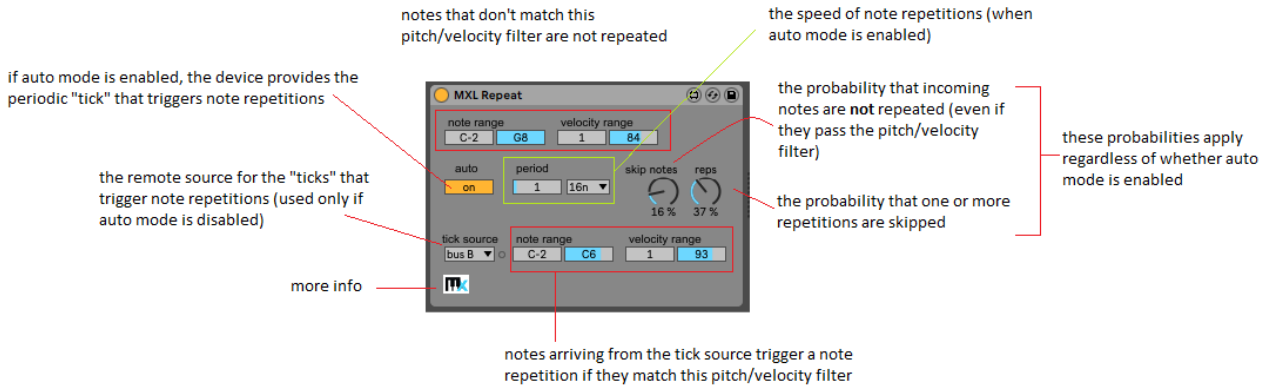
> ***Note**: a few MIDI Monitor devices – such as Live's built-in devices - display NoteOff messages as if they were NoteOn messages with zero velocity. You cannot use these devices to see the outcome from MXL Release Velocity. Instead, you should use the MXL Monitor device.*



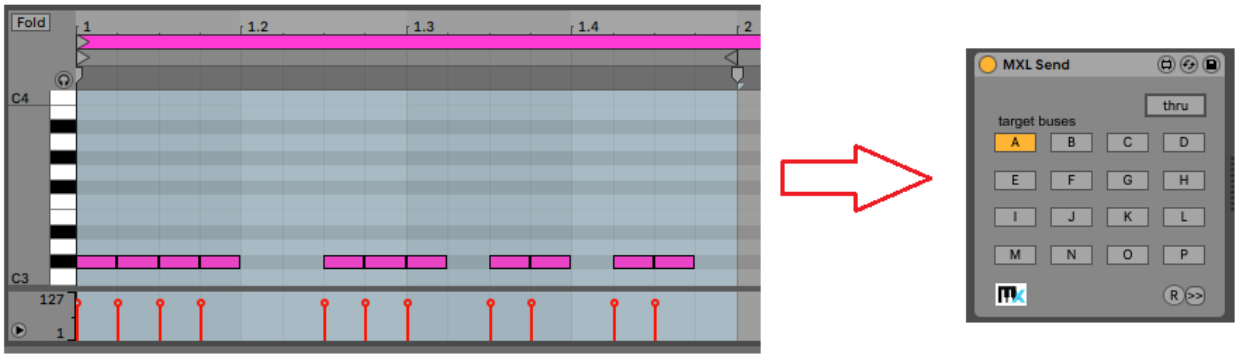
MXL Repeat



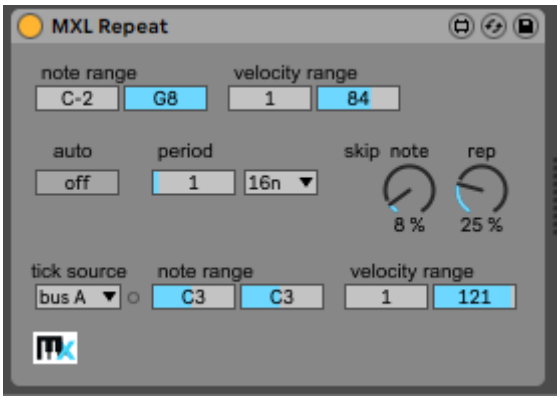
This device allows you to repeat active notes as long as you keep the key down on your MIDI keyboard. The “tick” for repeating active notes can be periodic or you can specify a source that provides them. The device also allow you to omit to apply repetition to randomly selected notes, or to skip “ticks”.



The MXL Repeat device can be used creatively to “superimpose” a rhythm to notes coming either from a clip or from a MIDI keyboard. It is especially effective if you store a series of notes inside a clip and send these notes to the remote tick source. For example, you might have a clip that sends notes to the Bus A:

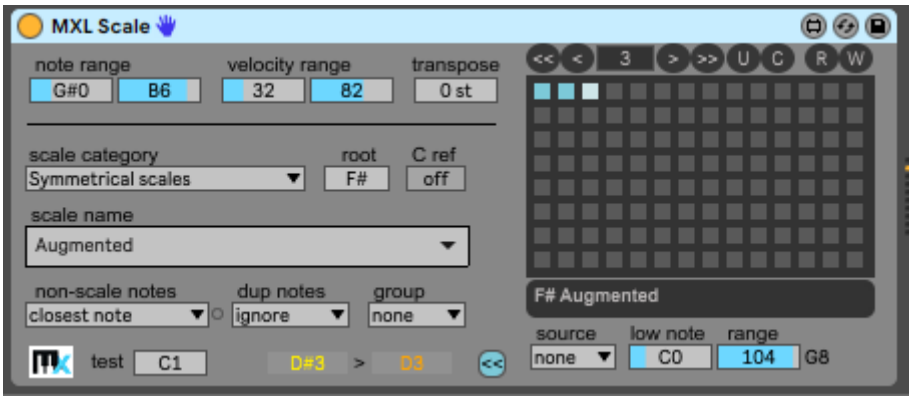


The above clip needs to be stored in a separate track and its notes must not be sent to an instrument: in fact, you don't what to actually hear these notes, because they are only useful to trigger note repetitions in another track, where the MXL Repeat device resides:



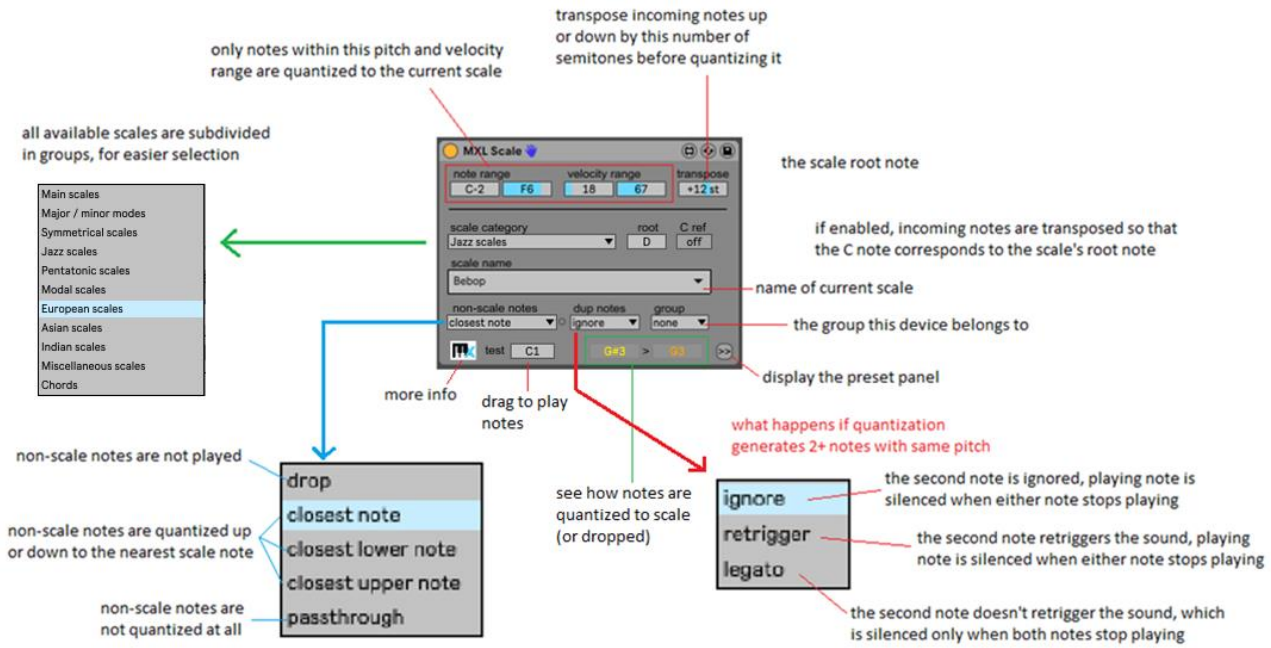
Notice that the pitch and velocity of notes in the clip is not significant, only their timing is, as long as the notes match the **note range** and **velocity range** filter in the bottom part of the MXL Repeat device. The **auto** mode must be disabled when you use a remote source to trigger repetitions, and that the **skip note** and **skip rep** probabilities apply also in this case.

MXL Scale



This device allows you to "quantize" incoming notes to a musical scale. It is therefore conceptually similar to Live's Scale native device, except it offers ***four hundred scales*** to choose from, plus many improvements and options you cannot find in other similar M4L devices.

MXL Scale offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.



When MXL Scale receives a note, it is first compared to the **note range** and **velocity range** values: notes that do not match these filter criteria are never quantized and are played as-is. You can use this feature to preserve the ability to play some notes out-of-scale for dissonant effects, for example in-passing notes with lower velocity.

If the note matches the filter, it is transposed up or down by the number of semitones selected in the **transpose** field; in most cases, you will leave this field to zero or set a whole number of octaves. This feature can be used in other creative ways, for example combined with the **group** feature (see below).

The two most important fields in this device are **scale name** and **root**. To make scale selection simpler, the 400 available scales have been grouped in categories, so that you do not have to scroll a very long list to select common scales such as Major or Blues.

The **non-scale notes** menu lets you decide what happens when the device receives a note that doesn't belong to the current scale. The most common choice is **closest note**, which quantizes the note up or down to the closest scale note, but you can also decide to always quantize up or down, or select **drop** to reject the note. The last option (**passthrough**) disables quantization and can be useful if used together with the C Ref option (see below).

The **dup notes** menu gives you control on what happens if an incoming note would be quantized to the same pitch as another note that is already playing. You can **ignore** the second NoteOn message (in which case the sound stops when the first NoteOff message is detected); you can **retrigger** the note; or you can play the second note in **legato** mode (in which case the sound stops only when both NoteOff messages are received). In most scenarios, the last mode delivers the most musical results.

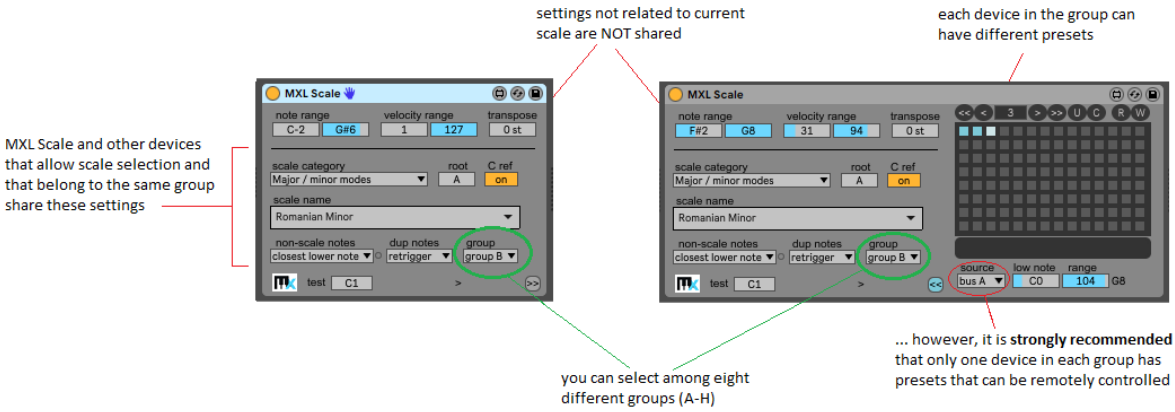
The **C ref** option provides a simplified way to play scales regardless of the current scale root. If this option is enabled, incoming notes are transposed by the number of semitones implied by the **root** field. For example, if current scale is E Major, incoming notes are transposed 4 semitones up. This means that if you press the C key on your MIDI keyboard, it will be always mapped to the scale root; if you press the D key it will be always mapped to the major 2nd note of the scale, and so forth.

The **C ref** option allows you to play any seven-note scale in any key by hitting only the white keys of your keyboard: C key is always the scale root, E key is always the 3rd (major or minor, depending on scale), etc. If

you press the C-E-G keys you obtain “tonic triad” in the key of the current scale: for example, if the current scale is Eb Major you obtain the Eb major triad (Eb G Bb); if the scale is F# Minor Melodic you obtain the F# minor triad (F# A C#), and so on. To avoid unexpected wide intervals in the melody, **C Ref** transposes up if root is in the range C# to F#, and transposes down if root is in the range G to B.

Any time the device receives a NoteOn message, MXL Scale displays both the incoming pitch and the resulting (quantized) pitch. If the note doesn’t belong to scale, the quantized pitch the right appears in a different color, or doesn’t appear at all if **non-scale notes** is set to **drop**. If you don’t have a MIDI keyboard available, drag the mouse on the **test** field to hear how the scale sounds like over the C1-C4 range (three octaves):

The **group** menu offers the ability to synchronize MXL Scale devices located in different tracks, a feature that is missing in Live’s native Scale device and that can be crucial in live performances.

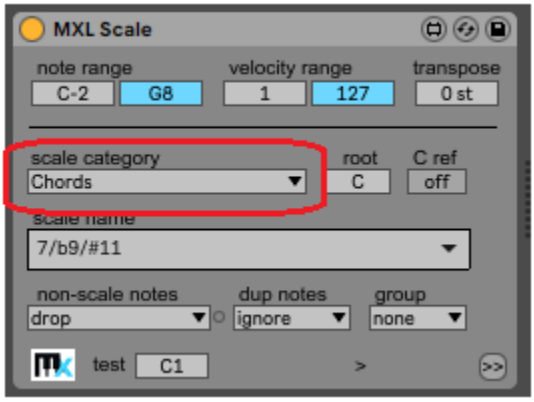


The **group** feature allows having an MXL Scale device share scale settings with any other MXL device that offers scale-related fields, for example MXL Scale Chord and MXL Scale Random devices.

The **group** feature is especially useful if one of the involved devices has presets that can be remotely controlled (see “Remote preset recall” section). You can prepare all the scales you plan to use in your performance, store them as presets in one of the MXL Scale devices in the group, and recall those presets using a pedalboard or a key on your MIDI keyboard. When this happens, all the devices in the same group will be loaded with same scale settings. (It is recommended that only one device in each given group be controlled remotely, else infinite feedback loops might ensue.)

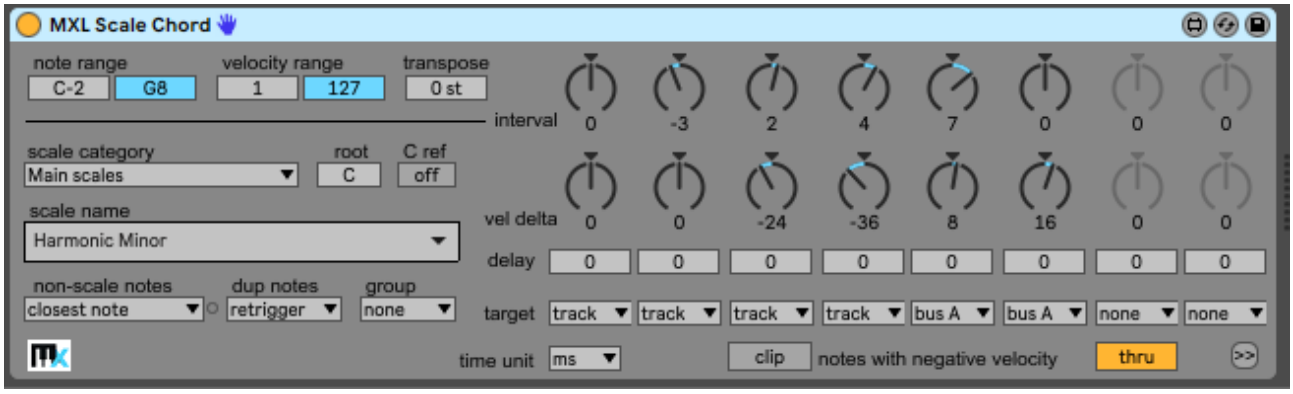
Only the fields related to scale selection and behavior are shared in the group. The **note range**, **velocity range** and **transpose** values are not shared. This detail allows you to implement a few interesting techniques: for example, an instrument in a given track might ***not*** quantize notes that are played harder (i.e. with higher velocity) and create dissonances, or it might transpose incoming notes up or down by an octave, a fifth or any other interval.

A final note: while the main purpose of the MXL Scale device is quantizing incoming notes to scales, it does support a fair number of common of ***chords***, which is the last entry in the **scale category** menu:



Quantizing incoming notes to a chord might seem less useful than quantizing to scales, but we added this feature because it makes a few tricks possible. For example, you can create arpeggios by turning a knob on your MIDI keyboard, ad explained in the MXL Make Note section.

MXL Scale Chord



This device combines the features of the MXL Scale and MXL Chord devices and allows you to implement very sophisticated *diatonic harmonizers* that can delay some notes and/or send them to different tracks.

Like MXL Scale and other MXL devices, MXL Scale Chord offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.

To appreciate the full potential of this device it is crucial to understand how incoming notes are processed:

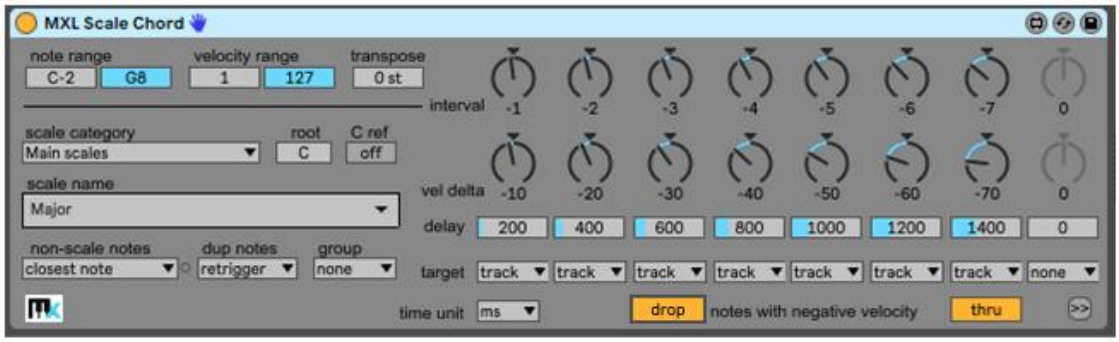
1) If the **thru** option is enabled, the note is passed as-is to the track.
2) The note is transposed up or down by the number of semitones specified in the **transpose** field.
3) The transposed note is quantized to the current scale, according to fields in the left portion of the device, most notably **scale name**, **root** and **non-scale notes**. If a previous note had generated the same quantized pitch, the conflict is resolved using the rule implied by the **dup notes** field.
4) The pitch and velocity of the original note are compared with the **note range** and **velocity range** fields; if the note doesn't match these criteria it doesn't trigger the harmonization process and the procedure stops here.

5) The note is harmonized and generates up to eight different notes, according to the settings in each vertical strip; the **interval** knob specifies the transposition in terms of ***scale steps*** (unlike the MXL Chord device, which uses semitones).
6) Note velocity is increased or decreased by the value specified by the **vel delta** knob; if the resulting velocity is negative and the **clip/drop** button is set to "drop" then the note is discarded.
7) If not discarded, the harmonized note is delayed according to the value of the **delay** field (found in each strip) and the **time unit** field (shared by all strips), and then sent to the destination indicated by the **target** menu in each strip.

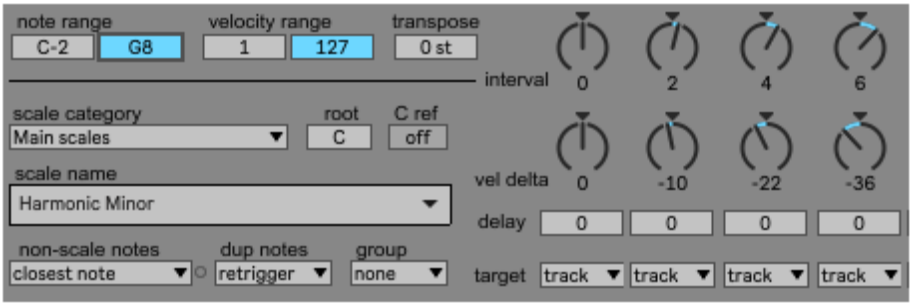
For more details, read the documentation of either MXL Scale or MXL Chord devices. Virtually all the techniques described in those sections – for example, how to combine the MXL Chord and MXL Note Cycle devices to create "rotating harmonizers" – can be applied to the MXL Scale Chord device.

The ability to harmonize notes according to scale intervals rather than plain semitones makes MXL Scale Chord stand out over other similar devices that transpose by a fixed number of semitones and *only later* apply scale quantization.

For example, because each of the eight strips has dedicated **interval** and **delay** setting, you can configure the device to play a fragment of an ascending or descending scale immediately after incoming notes. If you set the **vel delta** fields opportunely and enable the **clip/drop** function, scale notes are played with decreasing velocities and the number of played depend on how hard you hit the key on your MIDI keyboard:



You might assume that MXL Scale Chord is roughly equivalent to combining Live's Scale and Chord devices, but this assumption is incorrect. To understand why, we need to revise music harmony notions. Consider the following configuration:



This is how scale notes are harmonized:

note	harmonized notes	result chord
C	C Eb G B	C min/maj7
D	D F A C	D min7

Eb	Eb G B D	Eb aug/maj7
F	F A C Eb	F 7
G	G B D F	G 7
A	A C Eb G	A min7/b5
B	B D F A	B min7/b5

The notes that MXL Scale Chords emits are those you want to produce when harmonizing a Minor Melodic scale; however, you cannot achieve this "exact" harmonization using Live's built-in devices.

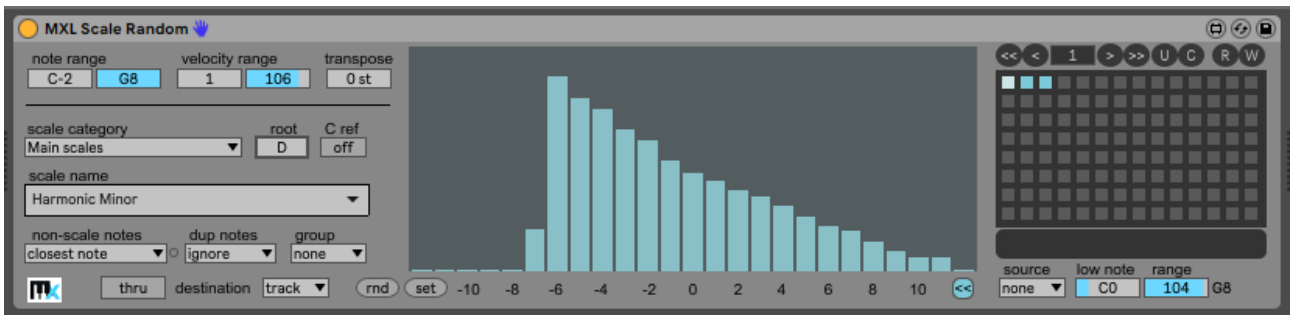
To prove this concept, let's try to achieve this "exact" behavior by combining Live's standard Chord and Scale devices. We will consider four different settings for the Chord device, to explore all possible combinations of the 3rd and 7th note of the chord in their minor and major variants:

The table below shows the result: cells in color highlight the chords that – as interesting as they can sound – do ***not*** abide by the rules of triadic harmony:

note	maj 7th	dom 7th	min7	min/maj7
C	C Eb G B	C F G B	C Eb G B	C Eb G B
D	D F A C	D G A C	D F A C	D F A C
Eb	Eb G B D	Eb G B C	Eb F B C	Eb F B D
F	F A C Eb	F B C Eb	F G C Eb	F G C Eb
G	G B D F	G A C D	G B D F	G B D F
A	A C Eb G	A D Eb G	A C Eb G	A C Eb G
B	B D F B	F A B Eb	B D F A	B D F B

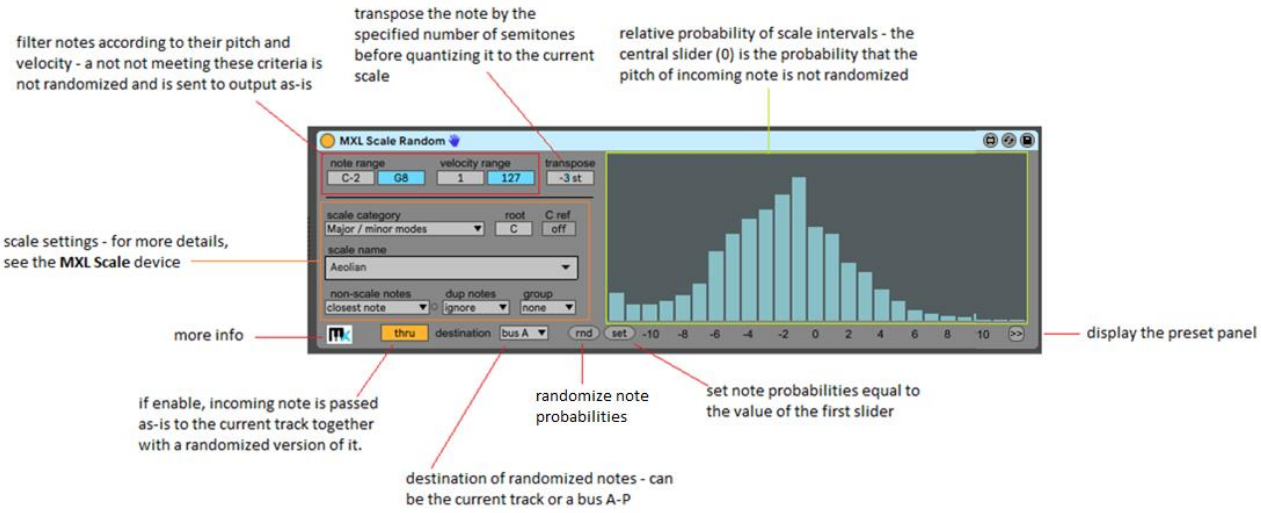
If you try different scales, you might obtain better results, but in general, if you reason "chromatically" and adjust to fit the scale only afterwards, then the harmonies you obtain are rarely "correct" according to music theory. The MXL Scale Chord device doesn't suffer from this limitation.

MXL Scale Random



This device quantizes incoming notes to the scale of your choice and then transposes the note by a random scale interval, where you can set the probability that each interval be chosen.

MXL Scale Random offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.



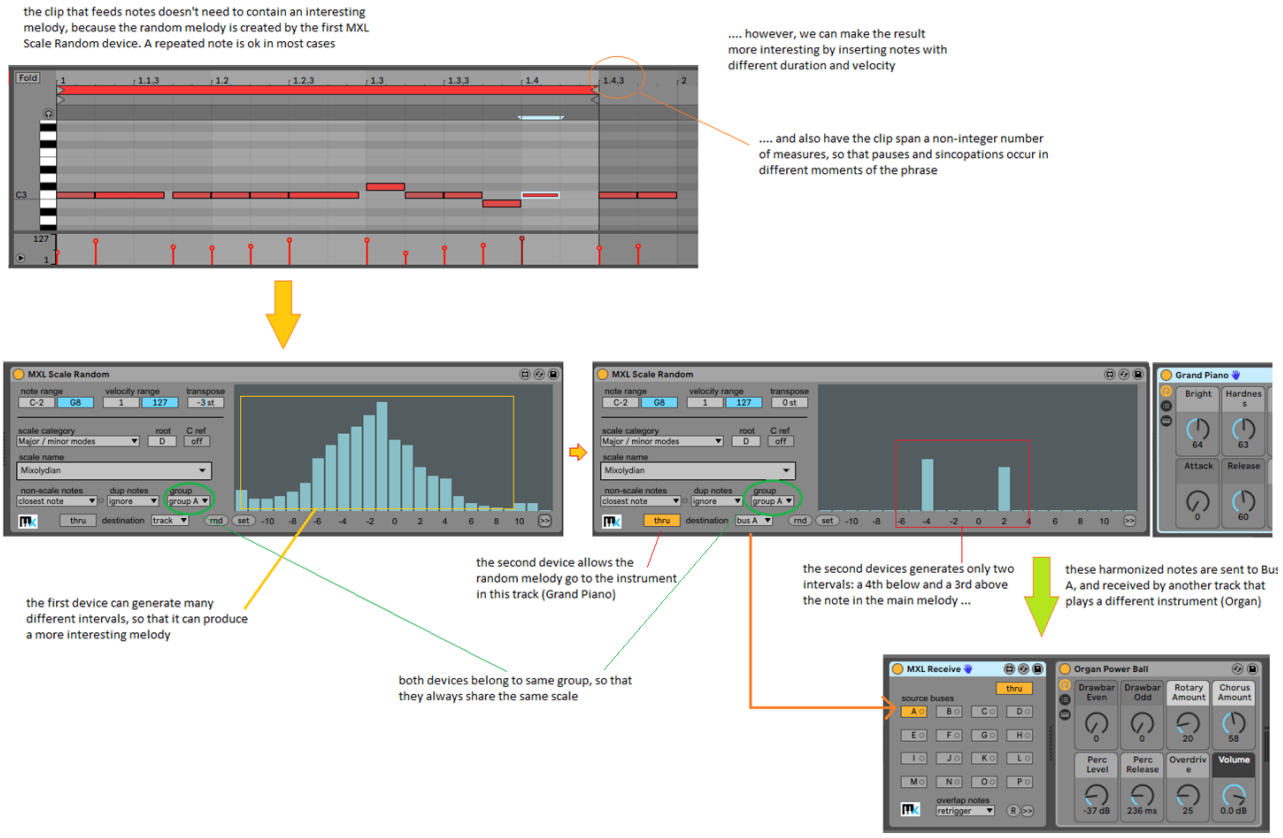
MXL Scale Random is similar to the MXL Random device, in that can generate random melodies from scratch or add variation of an existing melody, depending on whether you process a clip with a repeated note with same pitch or a clip that already contains a sketched melody. While it doesn't offer all the options you can find in MXL Random – for example, the ability to randomize note velocity, drop or tie notes – in many scenarios MXL Scale Random can produce better musical results because it allows you to reason in terms of scale intervals.

> ***Tip**: If you need those missing features, just add an MXL Random device immediately before or after MXL Scale Random.*

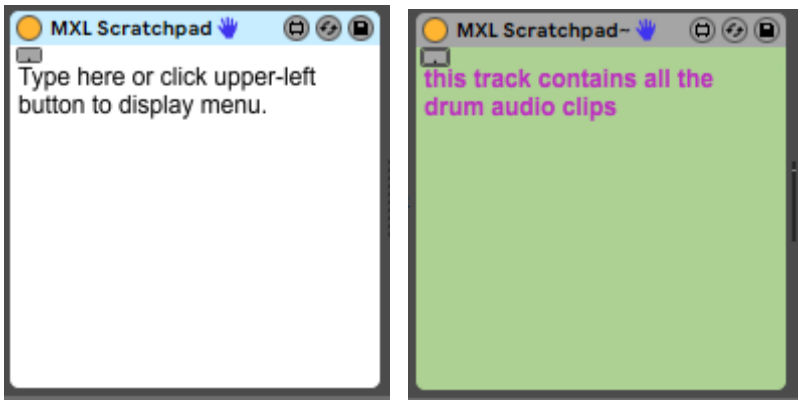
Each vertical bar represents the relative probability that a given interval be chosen for transposition. For example, the slider labeled +2 corresponds to two *scale steps* above the incoming note, or an interval of third. The actual number of semitones depends on the current scale and the received note. For example, assuming that current scale is C Major, it will be a major 3rd (4 semitones) if the incoming note is C or F or G, and a minor 3rd (3 semitones) in all other cases. Likewise, slider +4 corresponds to *four scale steps*, or an interval of fifth: it will be a diminished 5th (6 semitones) if the incoming note is B, or a perfect 5th (7 semitones) in all other cases.

Notice that sliders +7 and -7 corresponds to one octave below and above the incoming note, but *only if the scale contains seven notes*. If you select a pentatonic scale, you have to select the sliders -5 and +5 to assign the probability of transposing notes exactly one octave below or above. Quite conveniently, you can store both the current scale and all interval probabilities in one of the 100+ presets that MXL Scale Random gives you.

The ability to send transposed notes to a destination other than the current tracks adds flexibility. For example, say that you want to generate a random melody and you also want to (1) harmonize these random notes with a second note either one 3rd above or a 4th below and (2) send this additional note to a different track. Here's how you can combine two MXL Scale Random devices to perform both tasks:



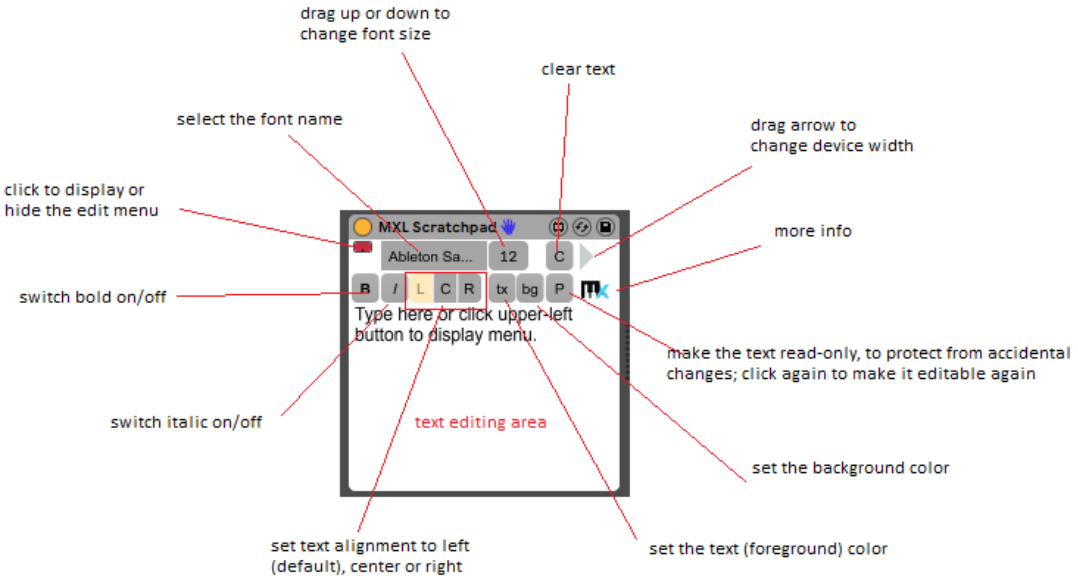
MXL Scratchpad and MXL Scratchpad~



These devices offer the ability to add notes and comments to a track. They are identical, except MXL Scratchpad can be used at the beginning of MIDI tracks (before audio instruments) and MXL Scratchpad~ can used in audio tracks (or in MIDI tracks, but after audio instruments). Both devices are “transparent” to MIDI or audio signals, thus they can be inserted in the middle of the device chain of a track without affecting the track in any way.

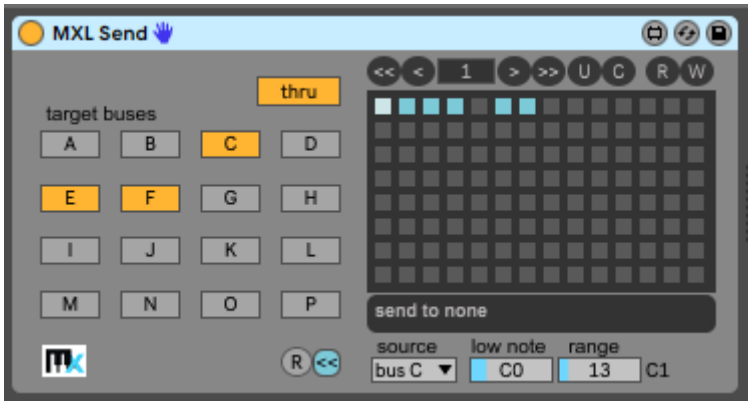
Ableton Live provides a way to associate text to a specific MIDI or audio clip, by using the Edit > Edit Info Text menu command. However, there is a simple way neither to associate a comment to the entire track nor to keep the comment always visible.

Unlike Live built-in note mechanism, these two devices allow you to spice-up your notes in a variety of ways, using the edit menu that appears when you click on the small button in the top-left corner:



You can use multiple instances of these two devices in the same track, for example to explain what each device does. You can make the scratchpad wider by dragging the arrow in the toolbar, and you can collapse the device if you do not need to read your comments:

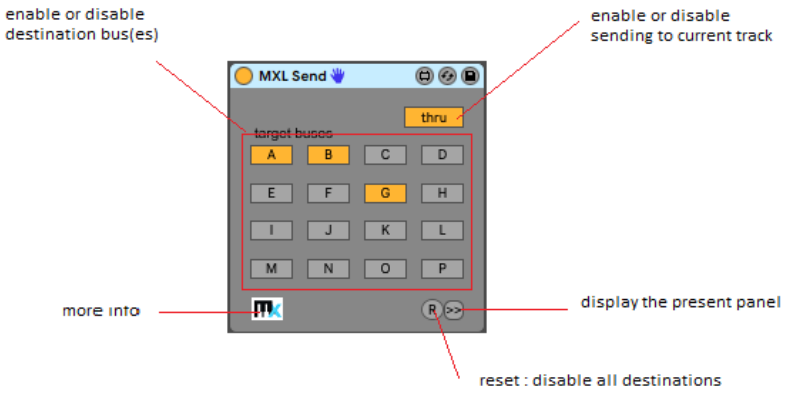
MXL Send



This device allows you to send MIDI data to other tracks, assuming that those tracks contain an MXL Receive device that is properly configured. It is conceptually similar to but offers many more features than the Max MIDI Sender device that is included in the “Max 7 Pitch and Time Machine” pack. Much of the functionality of MXL Pack is based on this device and its MXL Receive companion.

The MXL Send device offers 100+ presets that can be recalled manually, by mapping them to a MIDI controller, or by hitting a key on your MIDI keyboard. For more information, see the Presets section earlier in this manual.

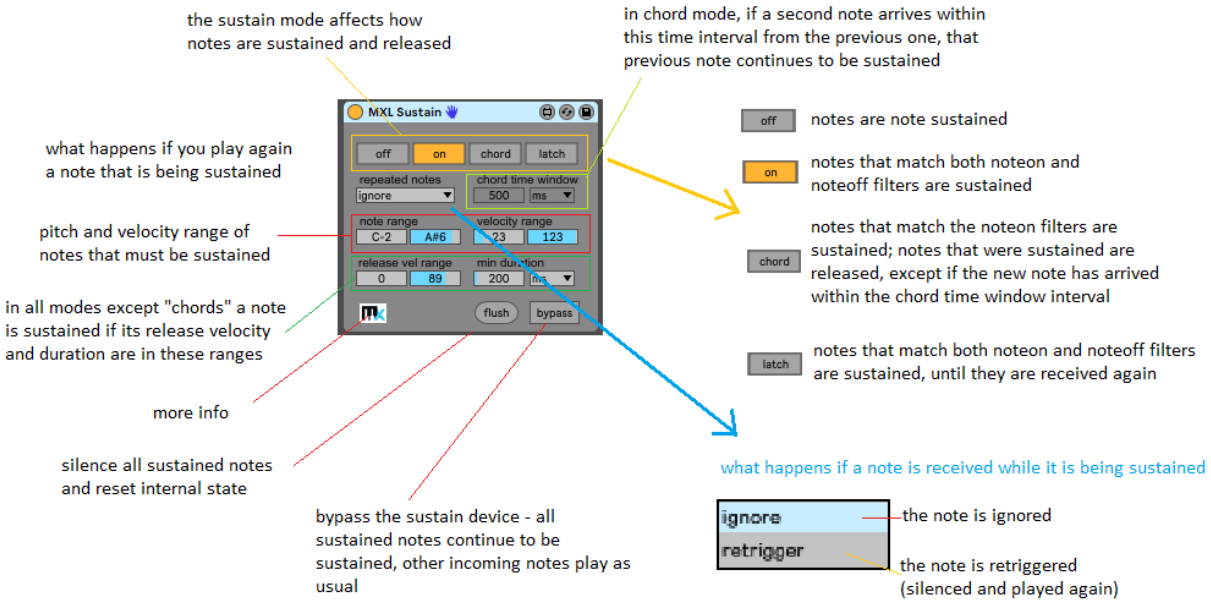
The first, immediately apparent difference from other similar devices is that you can send to multiple destinations, which are named Bus A, Bus B … Bus P. If the **thru** button is enabled (default), MIDI bytes received from the current track are passed along unmodified to the next device in addition to being sent to target buses. The **R** button resets the device by switching off all the A-P buttons:



The ability to store current configuration in a preset and quickly recall it later provides a great degree of flexibility, especially if you consider that multiple MXL Send and MXL Receive devices can react in different ways to the same remote command.

MXL Sustain

This device provides a variety of way to sustain incoming notes, until they are “flushed” (that is, silenced) by emitting the corresponding NoteOff messages. It is similar to the sustain pedal you use with MIDI keyboards or the “held” button found in most arpeggiators, except it offers more options and ways to select which notes must be sustained. The MXL Sustain device is especially useful in live performances and less useful when working with pre-recorded clips.



Note: for the sake of conciseness, in following paragraphs we refer to the value of the ***note range*** *and* ***velocity range*** *fields as the* noteon filter*, and to the value of* ***release vel range*** *and* ***min duration fields*** *as the* noteoff filter*.*

Not counting the **off** setting, which disables this device, the simplest **sustain mode** is **on**: if incoming notes match both the noteon filter and the noteoff filter, then the note is sustained. The decision of whether the note must be sustained is taken when the key on the MIDI keyboard is released. For example, you may decide to sustain only keys that you press for more than 1 second and/or that you release slowly (assuming your keyboard supports NoteOff release velocity), so that you are allowed to play quick phrases on top of sustained notes.

When **sustain mode** is set to **on,** the **repeated notes** menu becomes available and specifies what happens if you play again a note that is already suspended: you can either ignore the note or retrigger it (i.e. play the NoteOff and then NoteOn message in quick sequence).

When **sustain mode** is set to **chord** you can play chords that are automatically sustained until the next note or chord is played on the MIDI keyboard. The MXL Sustain device consider a group of notes as a “chord” if you play them in the interval defined by **chord time window**: if set appropriately, this value lets you sustain chord whose notes are not played exactly in the same instant, for example when you arpeggiate them. If a note arrives after this time interval, it is considered part of a new chord and all sustained notes are silenced. For this to happen the new note must match the *noteon filter*; if the new note matches also the *noteoff filter*, then the note is sustained.

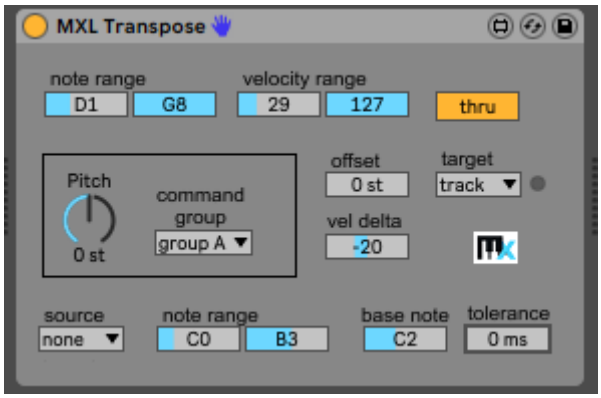
When **sustain mode** is set to **latch**, the device behaves similarly to the “hold” feature that many arpeggiators support. In this case, incoming notes are sustained until you press and release the key again. The first time you press a key it must match both the *noteon* and *noteoff filters*, whereas the second time you press the key only its pitch is significant and the latched note is silenced regardless of its attack and release velocity, or its duration.

You can silence all sustained note by clicking the **flush** button. Each time you select a different **sustain mode** an implicit flush is performed.

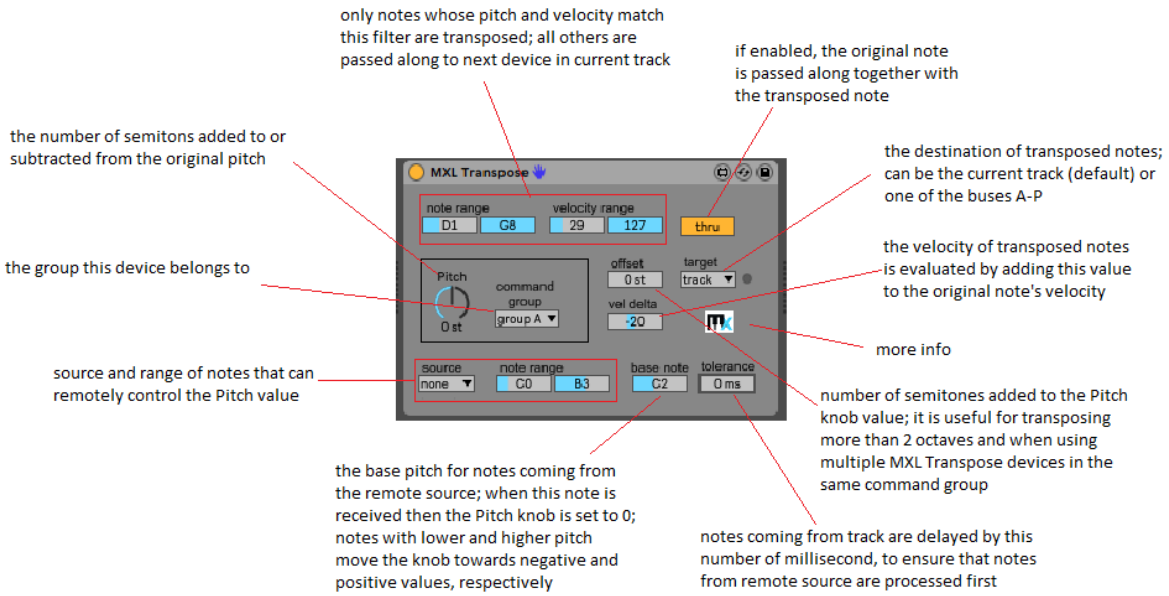
There are times, however, when you would like to sustain a set of notes – using any of the available modes – and then play regular melodies “over” those sustained notes. You can do so by enabling the **bypass**

switch, which behaves as if the MXL Sustain device were disabled except all sustained notes continue to play. (Note: if you disable a M4L device all the notes it generated are automatically silenced).

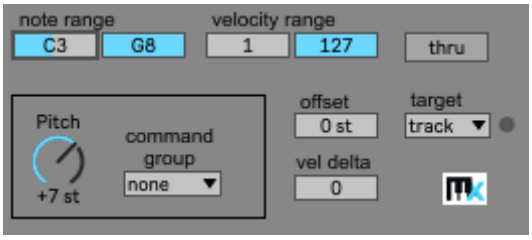
MXL Transpose



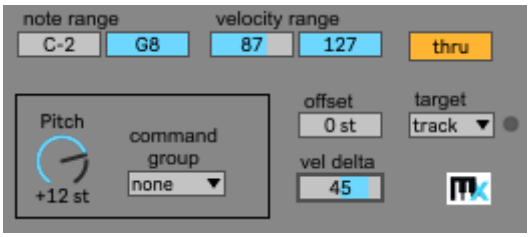
This device transposes incoming notes up or down by the specified number of semitones. It is similar to Live's Pitch device, except it offers many additional features such as the ability to: apply a filter to decide whether the note must be transposed; change the velocity of transposed notes; remotely control the amount of transposition; send transposed notes to a bus; synchronize multiple XML Transpose devices in different tracks:



You can use MXL Transpose to alter the pitch of all incoming notes or all the notes that match the filter. In this case, you should disable the **thru** switch to silence the original note and ensure that the **target** menu is set to "track":



Another common use for this device is *adding* a transposed note to the original note, either always or only if the incoming note matches the filter: in this case you should enable the **thru** switch and possibly use the **vel delta** field to emphasize (or de-emphasize) the transposed note. The following example adds one note in the upper octave, but only if the original note is played with velocity higher than 86:

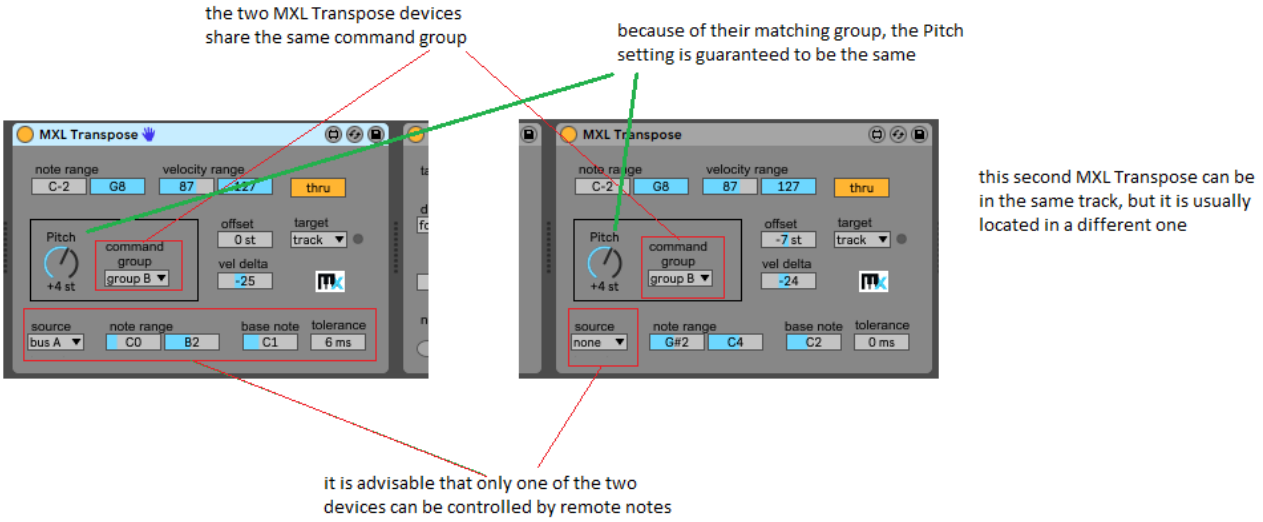


The **Pitch** field is automatable and can be bound to a MIDI parameter, for example a knob on your MIDI keyboard that emits a CC MIDI message. However, precisely controlling the pitch using a knob can be tricky, thus MXL Transpose offers an alternative mechanism for remote controlling, based on notes received from one of the buses A-P:

The **source** and **note range** fields indicate which notes are intercepted and used for remodeling controlling the amount of transposition. The **base note** is the note that corresponds to "no transposition": in the example above, the C2 note received from Bus A resets the Pitch knob to **+0 st**, the note C#2 would set the knob to **+1 st**; the D2 note would set the knob to **+2 st**, and so forth. Notes below the base note correspond to negative values: the note B1 would set the **-1 st** value, the note Bb1 would set the **-2 st** value, and so on.

For more information on the remote control mechanism and how to use the **tolerance** field, please refer to the "Remote Control" section, earlier in this manual.

Two or more MXL Translate devices belonging to different tracks can share the same command group. In this case, the devices are synchronized, as described in the "Command Groups" section. If you rotate the **Pitch** dial in one device, the new setting is immediately applied to all other devices in the same group:



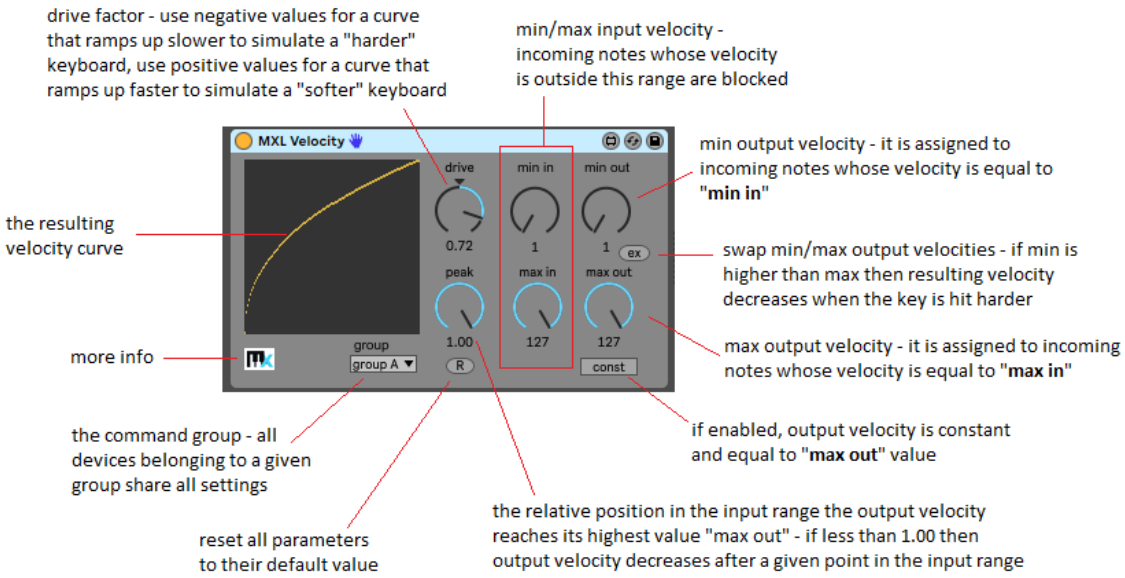
If you create a group of MXL Transpose devices, you might want to use the **offset** field to establish a fixed, additional number of semitones for a subset of the devices in the group. In the figure above, even if the **Pitch** dial shows the same value (+4), the device on the left transposes by 4 semitones up and the device on the right transposes 3 semitones down. The two resulting intervals vary when you rotate the dial, yet they

move in parallel. In most cases, you use the **offset** field to transpose by a whole number of octaves, for example **-24 st** (two octaves down) or **+12 st** (one octave up).

Notice that belonging to a group establishes a relationship “among peers”: there is no master device and there are no slave devices, and you can change the **Pitch** setting from any device in the group. If you plan to remotely control the amount of transposition using the **source** field should set this field to “none” for all but one device, which effectively becomes the “master” of the group (see figure above). Doing so reduces CPU usage and – above all – prevents infinite command loops that might even crash Live in some extreme cases.

MXL Velocity

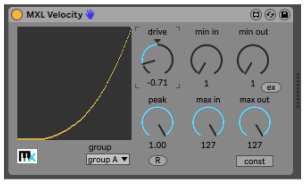
This device allows you to set a velocity curve for incoming notes using several intuitive parameters, and to synchronize two or MXL Velocity devices in the Live project, so that they always behave in the same way.



MXL Velocity exposes many of the features of Live’s Velocity device, plus a few others. Here are a few of the curves that you can defined with this device:

positive drive makes the curve ramp up faster, to simulate an "easier to play" keyboard

negative drive makes the curve ramp up slower, to simulate a "harder" keyboard



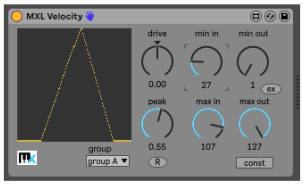
you can select minimum and maximum output values

if you reverse output range, velocity decreases if you hit they key harder

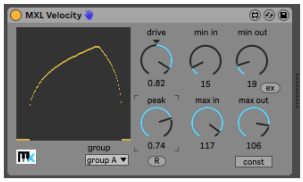
you can flatten the curve to obtain a constant output velocity (equal to "**max out**")

you can have output velocity decrease after a "peak" point in the input range

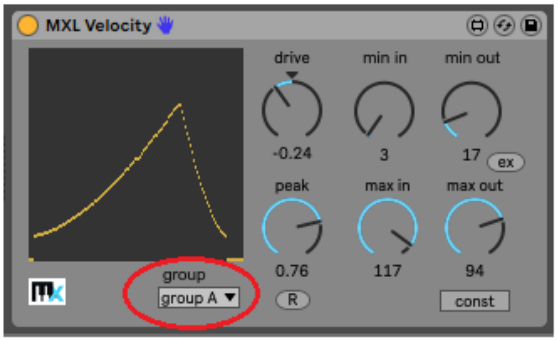
notes whose velocity is outside the input range are not played at all

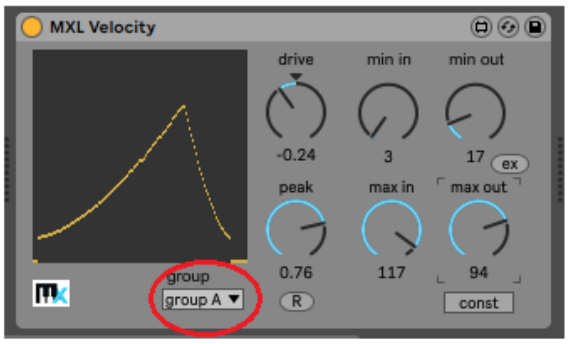


all parameters can be mixed to achieve the behavior you need



While Live's Velocity device is fine to process notes recorded in clips and tracks, it isn't really useful to implement ***velocity curves***, such as those offered by more advanced MIDI keyboard. For example, if you are sending notes from the keyboard to different Live tracks you should add a Velocity device at the start the device chain in each track and – above all - ensure that all these devices are synchronized to behave in the same way. This task is trivial with MXL Velocity, if multiple devices share the same command group:





The **peak** parameter allows you to implement some interesting behaviors when you are using a single MIDI keyboard to send notes to different tracks. As you can see in the example below, you can have different instruments play, depending on how hard you hit note keys. Or you can use constant velocity if you want an instrument to play with same intensity regardless of the input velocity.