

Pages 2 Documentation

1.0 About Pages

1.1 Introduction

1.2 FAQ

1.3 License

2.0 Installation / Setup

2.1 Ableton Live Integration

3.0 Basic Setup

3.1 Creating Pages

3.2 MIDI Setup

3.3 Ableton Setup

3.4 Changing Pages

3.5 Pattern Recorders

4.0 Ableton Clip Launcher Page

4.1 Ableton Clip Skipper Page

4.2 Ableton Live Looper Page

4.3 Ableton Scene Launcher Page

4.4 External Application Page

4.5 Groovy Page

4.6 Machine Drum Interface Page

4.7 MIDI Keyboard Page

4.11 MIDI Sequencer Page

1.0 About Pages

Thank you for downloading Pages!

Please visit the [Pages home page](#) for the latest release.

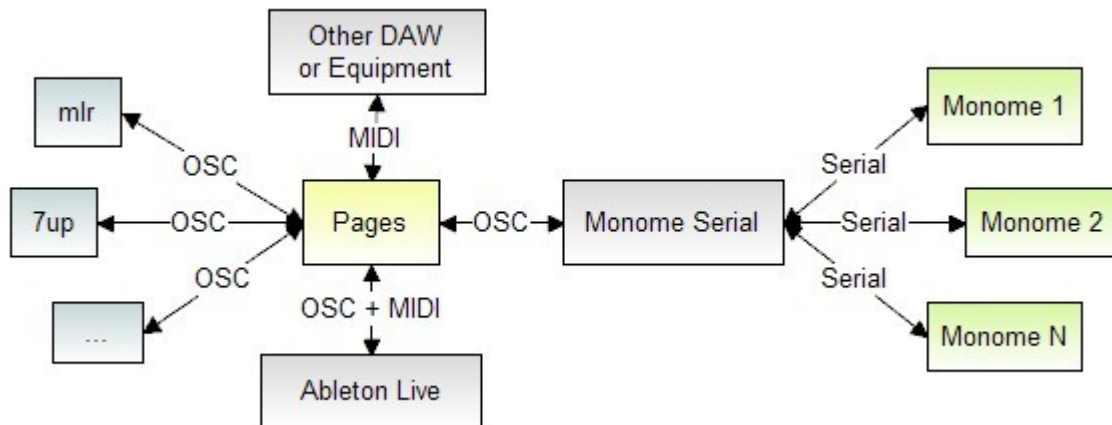
Pages is written and maintained by [Tom Dinchak](#) with contributions from members of the [monome community](#). Pages is open source software and is released under the GNU Public License.

1.1 Introduction

Pages is a Java application for controlling any number of monome-type devices. If the device can speak the monome OSC protocol then Pages will probably work with it.

Pages includes a number of MIDI-based applications, Ableton Live integrations and a way to pipe in applications external to Pages (ie. mlr, 7up, stretta suite, etc). Pages is highly configurable and is a sort of Swiss army knife for integrating monome devices with other applications and MIDI-capable equipment.

Pages allows you to run as many applications as you need simultaneously while using the monome, GUI, and/or MIDI messages to switch between applications.



This is a diagram of how Pages relates to other applications and equipment in your setup. Pages sits between Monome Serial and other monome applications to provide a paging and pattern recording overlay to applications that would otherwise be running as standalone applications. Pages also communicates to Ableton via OSC and MIDI, and any other DAW or equipment via MIDI.

1.2 FAQ

What operating systems will Pages run on?

Pages will run on OS X, Windows, and Linux.

What types of devices are supported?

Pages runs on all series of monomes and anything that can speak the MonomeSerial or SerialOSC protocols. This includes Launchpads, APC40s, Arduinomes, etc.

Pages won't load on OS X, it says the JAR file couldn't be launched, what's wrong?

You probably need to update to the latest version of Java. Pages is compiled to work on Java version 5 or higher.

How come I can't see most of my MIDI devices on OS X?

You may need to download the [mmj extension](#) for full MIDI support on OS X. Recent versions of OSX with the latest Java updates shouldn't require this library. If you are having trouble with MIDI devices and have mmj installed you might need to remove it.

How do I debug problems with Pages?

If you run Pages via the command line you will be able to see the console output from Pages. This can be very helpful in debugging issues as generally a stack trace will be generated when an issue occurs and you can use this information to diagnose the problem. To run pages from a command line, open a terminal / cmd.exe window, navigate to the folder with the Pages JAR file, and run **java -jar pages-0.2.2a7.jar**. Adjust the filename as needed for the version you're using.

Do I need Ableton Live to run Pages?

Ableton Live is not required to use the non-Ableton Live related features of Pages, such as MIDI pages or the External Application page. Pages can be made to work with any MIDI-aware software or equipment.

What version of Ableton Live is required to use the Ableton Live pages?

Version 8 is recommended and is required for OS X. On Windows, most functionality will work with version 7.

Why don't I see blinking on the Ableton Live pages when clips are playing?

You most likely need to enable a MIDI input on the page that has MIDI clock signal coming in from Ableton Live.

Why isn't MonomeSerial auto discovery finding my monomes?

Auto-discovery depends on correct implementation of the /sys/report OSC call in monome serial. The only monome serial version that I know this to work on is [the version I wrote in python for windows](#).

Does Pages support autoconfig or autofocus?

Pages currently supports autofocus through the External Application page but autoconfig is not supported at this time. This will eventually be supported.

1.3 License

Pages is open source software and is released under the GNU Public License. A copy of this license is included with the official release packages. The source code is freely available at <http://code.google.com/p/monome-pages>.

2.0 Installation / Setup

Pages is currently supported on the following operating systems:

- Windows XP or later
- OS X 10.4 or later
- Linux

Download the latest version of Pages

Get the latest version of Pages from the [Downloads page](#). Unzip the archive. You will see a JAR file, a few text files, and a number of different LiveOSC versions. The JAR file is the executable, double-clicking on that should open Pages. Verify that this is the case and check the [FAQ](#) if you are having trouble running it.

Virtual MIDI Devices

Be sure to install the appropriate package for your operating system below:

- Windows: [MIDI Yoke](#) (flakey after Windows XP but can work on Windows 7), [LoopBE](#) (works well on Windows 7)
- OS X: [IAC Driver](#)
- Linux: [virmidi-snd module](#)

Java version 6 or later

This is generally already installed but if you are having trouble launching the Pages JAR file:

- Windows: Get it from [Oracle](#)
- OS X: Use Apple's installer to get the latest version
- Linux: Use your favorite package manager, OpenJDK is fine

Python 2.5

If you are having trouble with the LiveOSC integration then you may need to install this. OSX 10.7.4 with Xcode installed appears to come with Python 2.5, Xcode may not be required however (if somebody knows for sure please let me know). As of Ableton 8.3.3 you will need to install Python 2.5 for Windows to work properly. Note that only version 2.5 will work based on some of the file paths in the LiveOSC code. Also be sure to install it to the default location (C:\Python25). This applies for Linux when running Ableton under WINE.

mmj

[mmj](#) is a drop-in MIDI library replacement for Java on OSX. I have

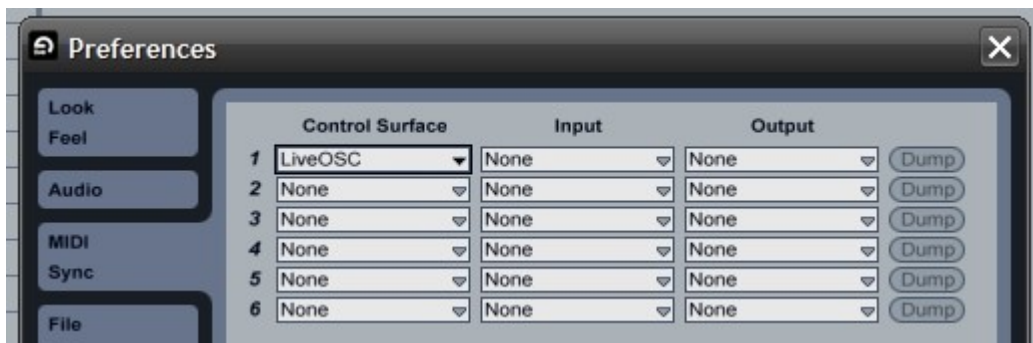
found that Ableton will stop responding to MIDI notes occasionally and installing this library can fix that problem. Download the .zip from their website and copy the included mmj.jar and libmmj.jndi to /Library/Java/Extensions to install.

2.1 Ableton Live Integration

Once you've verified that Pages opens, find the appropriate LiveOSC folder for the version of Ableton Live you're using. Copy **the LiveOSC folder inside of it** to the MIDI Remote Scripts folder of your Ableton Live installation:

- Windows: Copy the LiveOSC folder to C:\Program Files\Ableton\Live \Resources\MIDI Remote Scripts.
- OSX: Control-click on Ableton from your Applications folder and choose "Show Package Contents". Navigate to Contents/App-Resources/MIDI Remote Scripts and copy the LiveOSC folder here.

Next open Ableton Live, go to the Preferences window, and choose "MIDI / Sync". Select LiveOSC from one of the dropdowns in the Control Surfaces column:



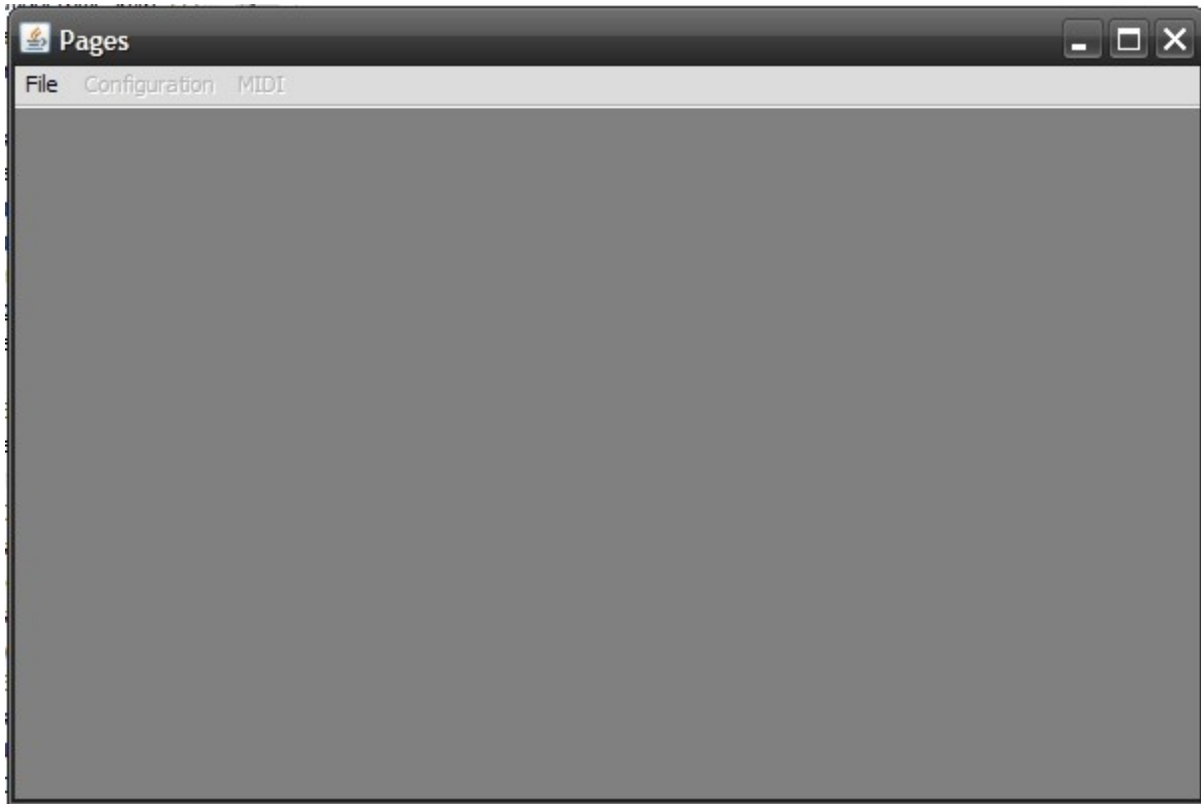
MIDI clock synchronization is set up in the same area. Choose a virtual MIDI output and click on the Sync button. Expand it with the small arrow to set the latency. This value should be negative your "Overall Latency" in the Audio tab.



Note: Make sure you don't enable the corresponding MIDI input in Ableton or Pages won't be able to use it.

3.0 Basic Setup

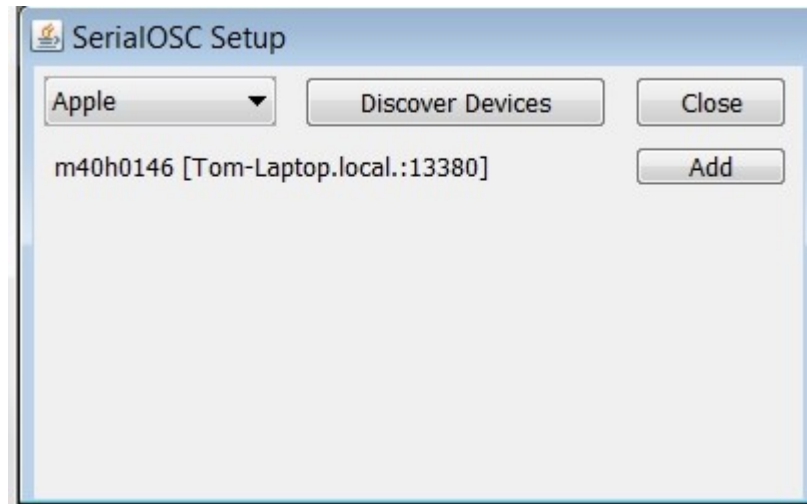
Launch the Pages JAR file. You will be presented with a window that looks like this:



This is the main Pages application window. First, create a new configuration. Do this by choosing "New Configuration..." from the File menu. Give the configuration a name and click OK.

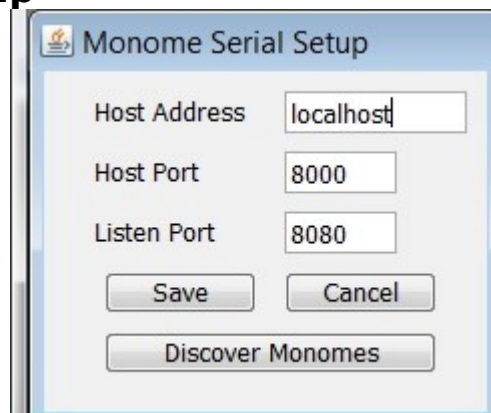
You will notice that the Configuration and MIDI menus are now active. Next we want to enable communication with the monome. Go to the Configuration menu and choose "SerialOSC Setup..." if using serialosc, otherwise choose "Monome Serial Setup..."

SerialOSC Setup



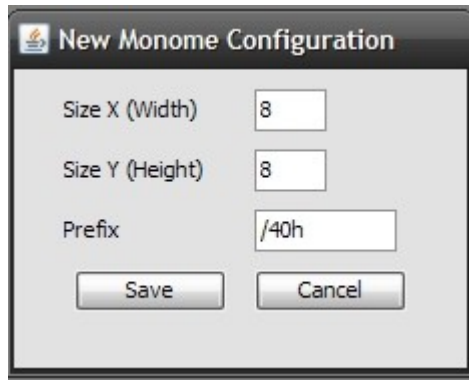
The dropdown will let you choose between Apple/JmDNS for Bonjour/Zeroconf. JmDNS is useful on Linux where the Apple library isn't as well supported. Click Discover to find your devices and click add to create a new monome configuration out of them.

Monome Serial Setup



This is the Monome Serial setup page. Generally the defaults are fine but you can adjust them as needed. You can try clicking Discover Monomes to see if the /sys/report call is supported properly by your version of Monome Serial. Click Save to start communication with Monome Serial.

Now that we have established communication with Monome Serial it's time to create a new monome configuration. Go to the Configuration menu and choose "New Monome Configuration...". You will be presented with a dialog like this:



A screenshot of a "New Monome Configuration" dialog box. The dialog has a title bar with a small icon and the text "New Monome Configuration". Inside, there are three input fields: "Size X (Width)" with the value "8", "Size Y (Height)" with the value "8", and "Prefix" with the value "/40h". At the bottom, there are two buttons: "Save" and "Cancel".

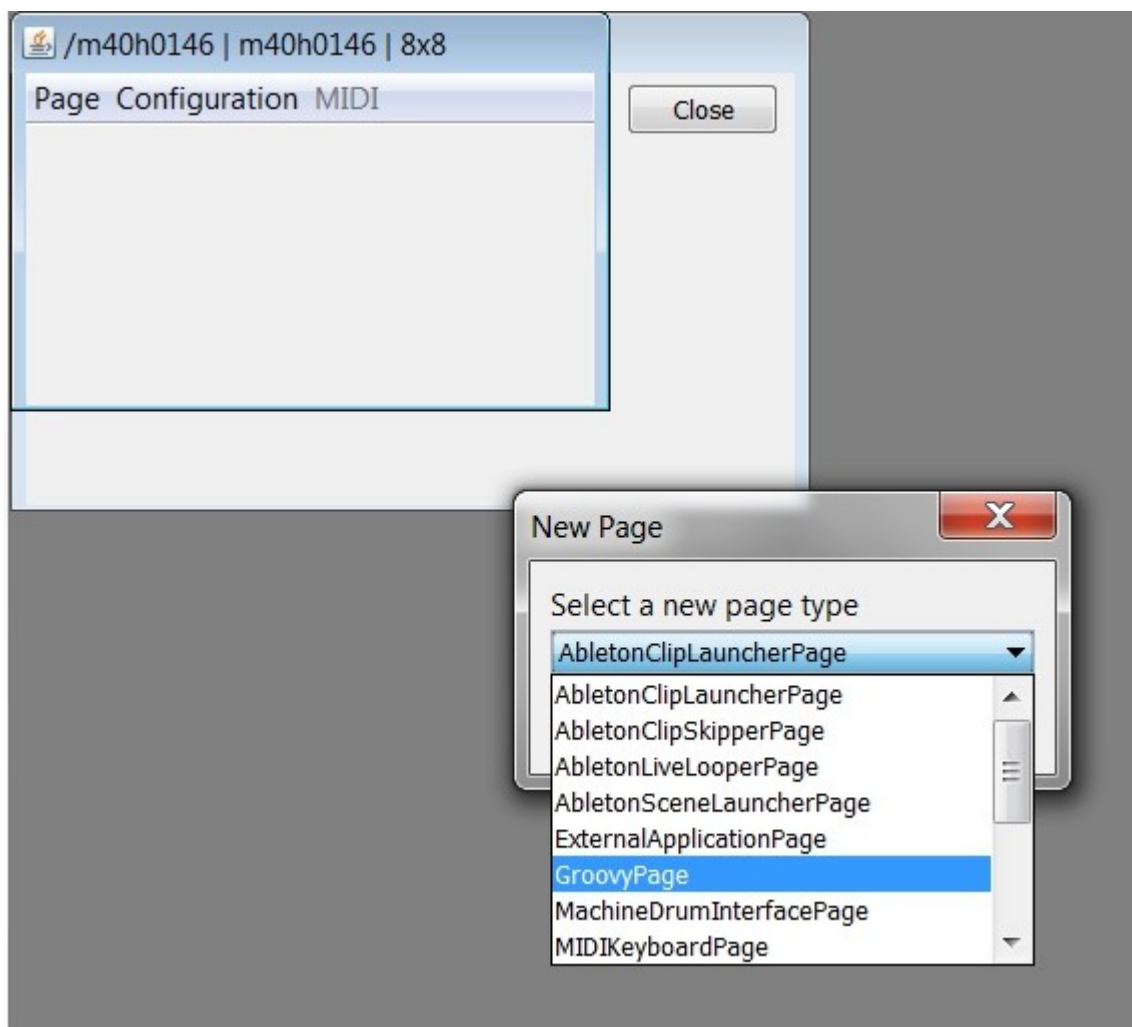
Size X (Width)	8
Size Y (Height)	8
Prefix	/40h

Save Cancel

3.1 Creating Pages

After you've created a monome configuration it's time to create a few pages to use. Each page is designed to do something different and there are a number of different types available. Some are MIDI utilities (keyboards, sequencers, etc.), some are Ableton utilities (clip launchers mostly), there's an External Application page that lets you route in other applications, say a Max/MSP application, and there's a Groovy page that lets you program your own page types in a simple scripting language.

Use the page menu to create a new page:

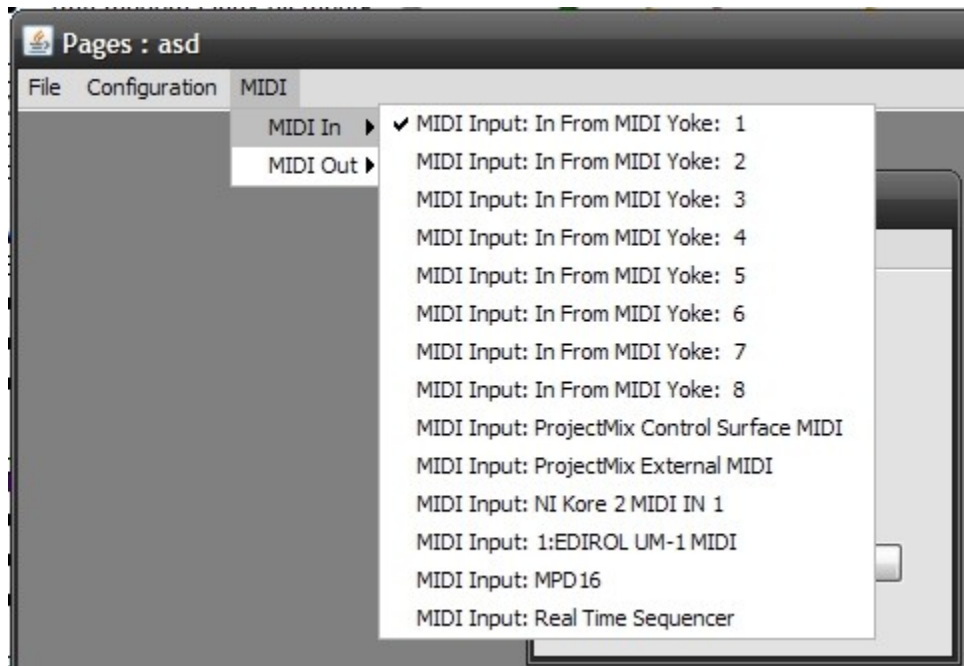


See the page reference section for more information on individual page types.

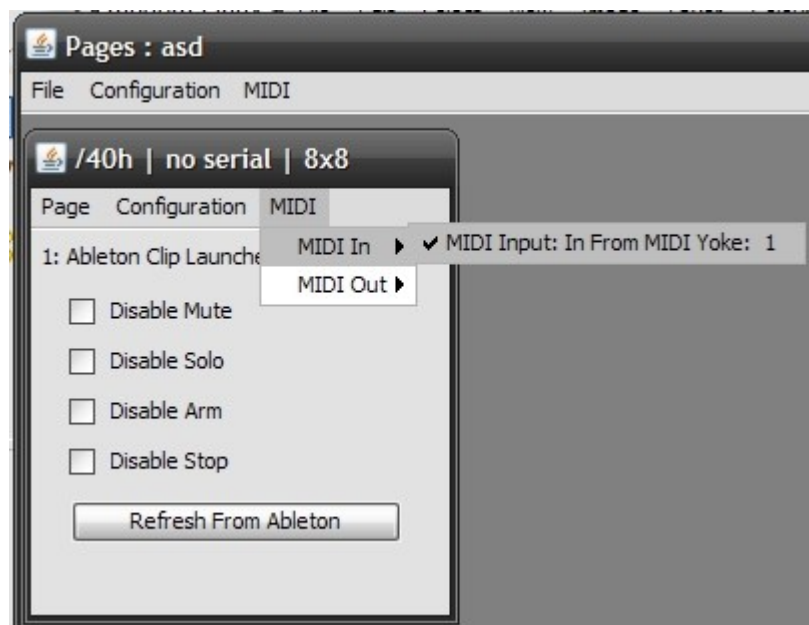
3.2 MIDI Setup

In Pages, MIDI devices are managed at two levels:

The application level:



The page level:



MIDI devices enabled at the application level are opened and available globally to all pages. However, each page needs to enable the MIDI devices that it wants to use. Many pages respond to MIDI clock messages to provide synchronized behavior. These pages include:

- All Ableton Live pages
- MachineDrum Interface
- MIDI Sequencer pages
- MIDI Generator

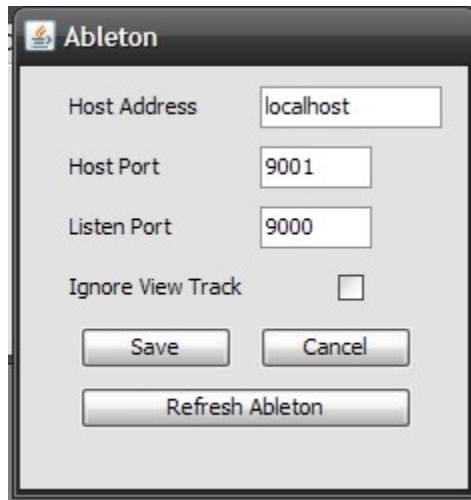
It is important to make sure that MIDI clock sync is enabled if any of these pages don't seem to be working properly.

Many pages also send MIDI output. The idea is the same -- enable the output globally and then select individual outputs per page. MIDI messages generated by a page will be echoed to all selected MIDI outputs.

OSX Notes: If you are having trouble with Ableton recognizing MIDI notes from pages then you may need to install the [mmj library](#). Unzip the file and copy mmj.jar and libmmj.jnilib to /Library/Java/Extensions and restart pages.

3.3 Ableton Setup

The Ableton setup pane is accessible via the main Configuration menu and looks like this:



The default Host Address, Host Port, and Listen Port values should work out of the box with LiveOSC, so there should be no need to adjust those values.

The "Ignore View Track" checkbox controls whether or not playing clips, muting/soloing tracks, or other track-specific Ableton Live functions should send a message to Ableton Live to select the associated track. If Ableton Live is jumping around on you as you work with the Ableton Live pages and you want that to stop, check this box.

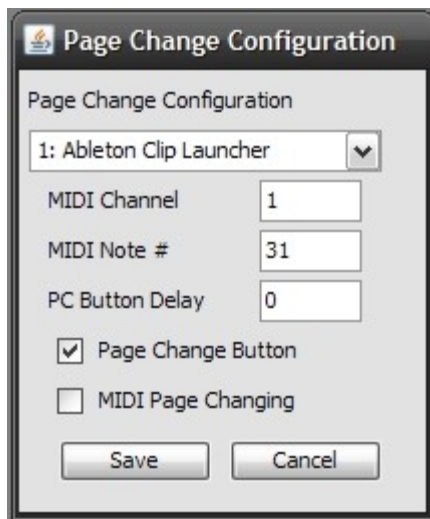
The Refresh Ableton button will force a refresh of the current Ableton Live state for all pages.

3.4 Changing Pages

There are three methods to switch between pages:

- In the GUI by using the Next / Prev Page selections from the file menu or selecting the page from the list.
- On the monome using the page change button (the button in the bottom right corner of the monome). Hold down the page change button and press another button on the bottom row to switch to the corresponding page.
- With a MIDI page change rule (explained below).

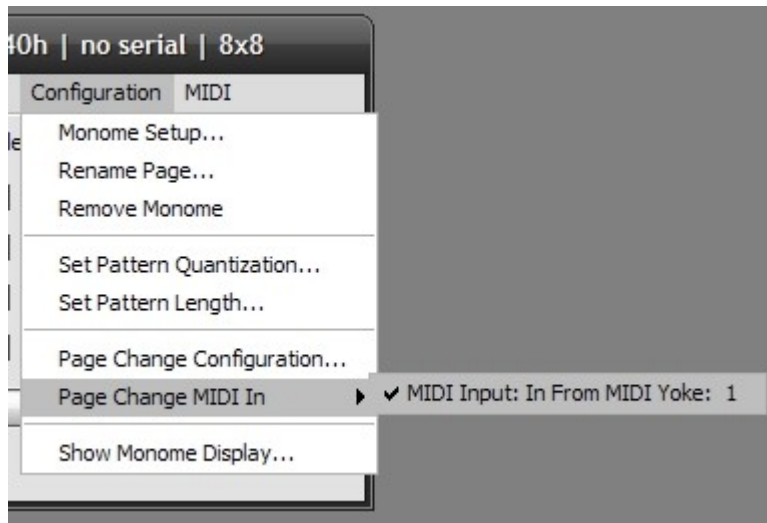
You can always use the GUI to switch between pages. The "Page Change Configuration" dialog controls which of the other two methods is active. This dialog is found in the Configuration menu of each monome configuration and looks like this:



The default is for only the page change button to be active. You can choose to have both the Page Change Button and MIDI Page Changing, either one, or no page changing at all by clicking the check boxes.

The PC Button Delay value can be set to stifle press events when the page change button is held for a duration of time. The duration is defined in milliseconds (so 1000 = 1 second). Normally when the page change button is released after no page changing has been done a quick "press/release" message is sent to the page. If the delay is defined (greater than 0) and you hold the page change button for a period greater than the duration and do not do anything else, the press event will still be ignored.

You can also use MIDI note messages to trigger page changes. These are called MIDI page change rules. To assign MIDI page change rules, select individual pages with the drop down and input a MIDI channel and MIDI note number that should trigger changing to that page. There's a "Page Change MIDI In" sub-menu in the Configuration menu to enable MIDI input devices to listen to page change messages from:



3.5 Pattern Recorders

Pattern recorders are built into each page and allow the user to store and play back a series of press events. The pattern recorders will run in the background regardless of which page you currently have selected, so they're a great way to sequence events live and keep the sequence looping while working on other pages.

The pattern recorders are accessed by holding down the page change button and pressing a button in the top row. Each row represents one pattern, and you can have multiple patterns running on the same page at the same time. Just select the next pattern and start pressing buttons. You can select a playing pattern to stop it and select it again to re-record into the slot.

The Configuration menu in the monome configuration window has some options that control how these patterns behave. You can set the pattern length (in bars) or the pattern press quantization. These settings apply to all patterns on the currently selected page, so each page can have individual pattern configuration.

Patterns run on MIDI clock messages so you need to have MIDI sync coming into Pages for them to operate properly. Enabling a device that is receiving MIDI clock messages in the application MIDI configuration menu is sufficient

4.0 Ableton Clip Launcher Page

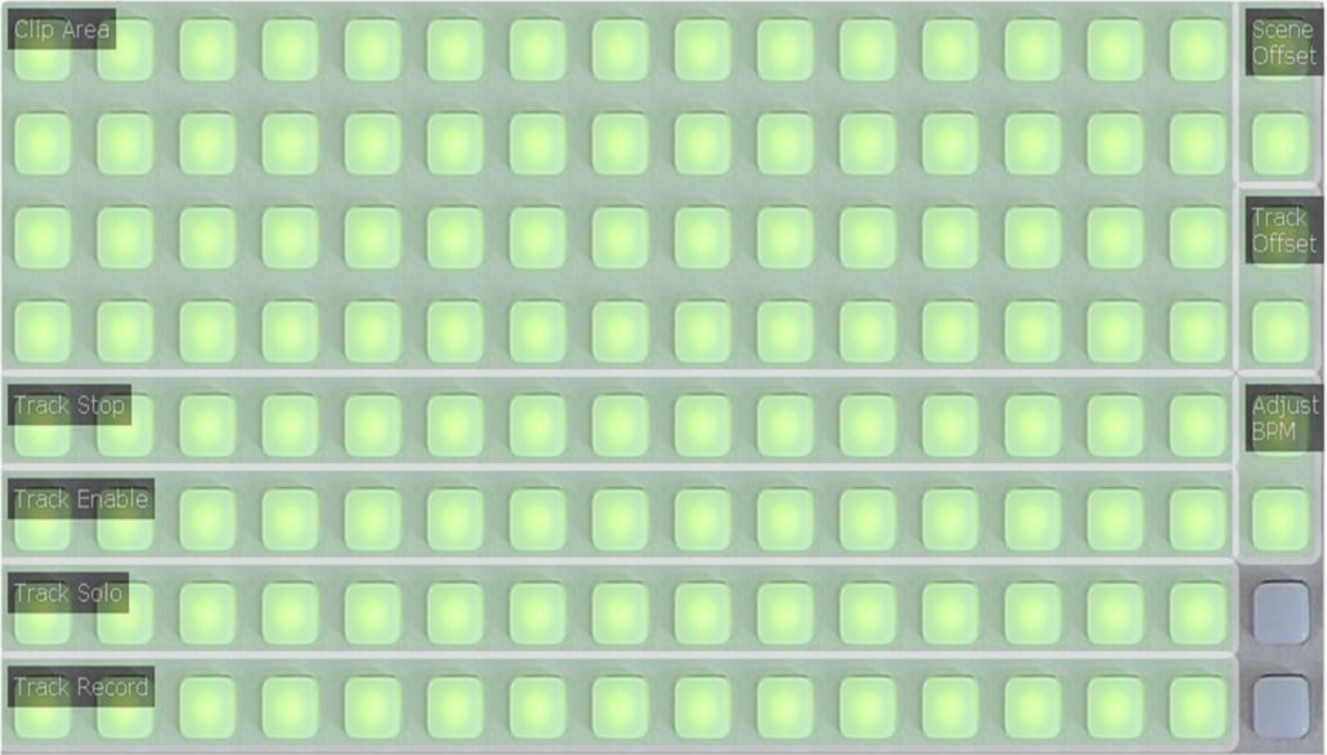
This page type is a basic clip launcher similar to APC40 behavior. The upper portion of the page represents the clip state of Ableton while the lower 4 rows show the track state (these rows can be disabled in the GUI). The rightmost column contains controls for moving the clip view, adjusting the bpm, toggling overdub mode and performing an 'undo' operation.

Make sure to enable MIDI Sync for proper clip state flashing.

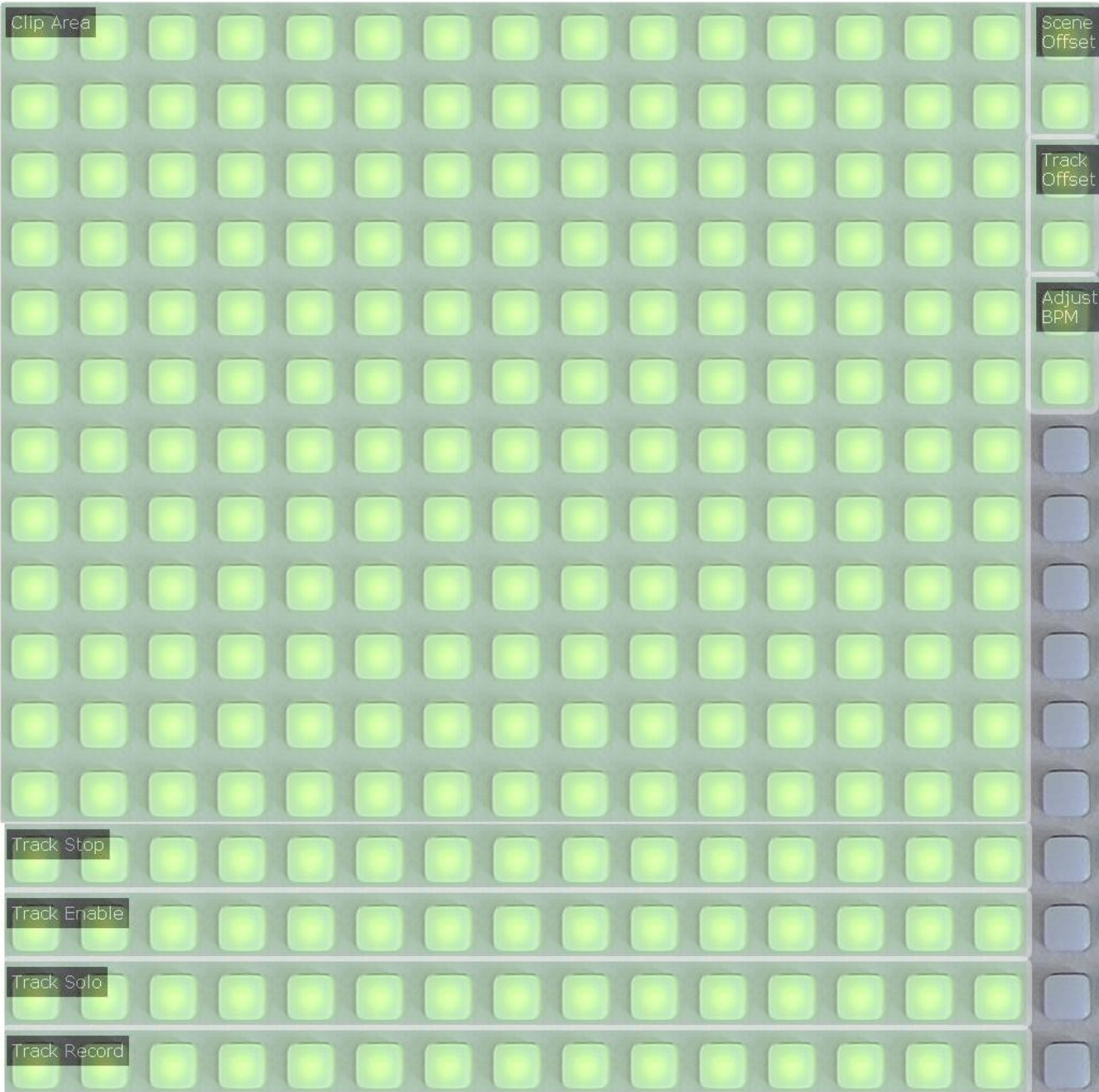
64 Mode



128 Mode



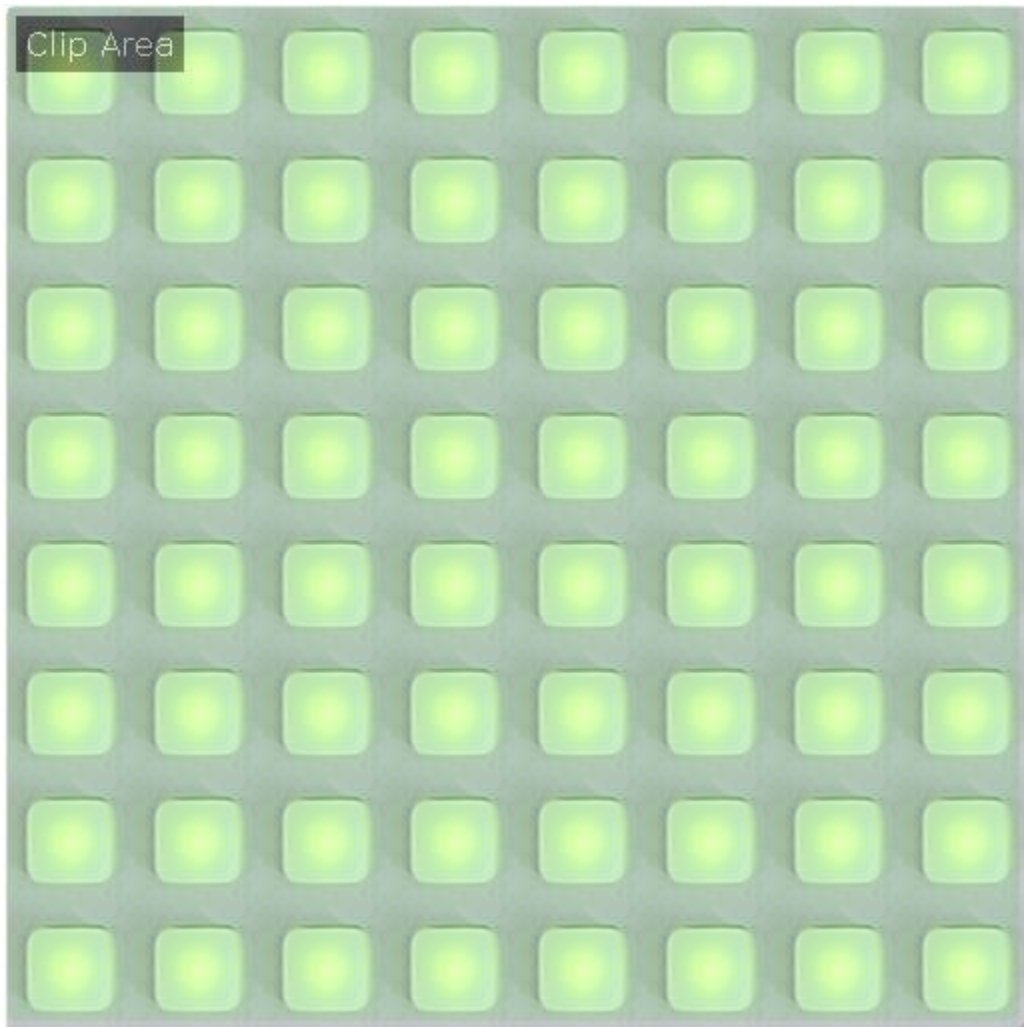
256 Mode



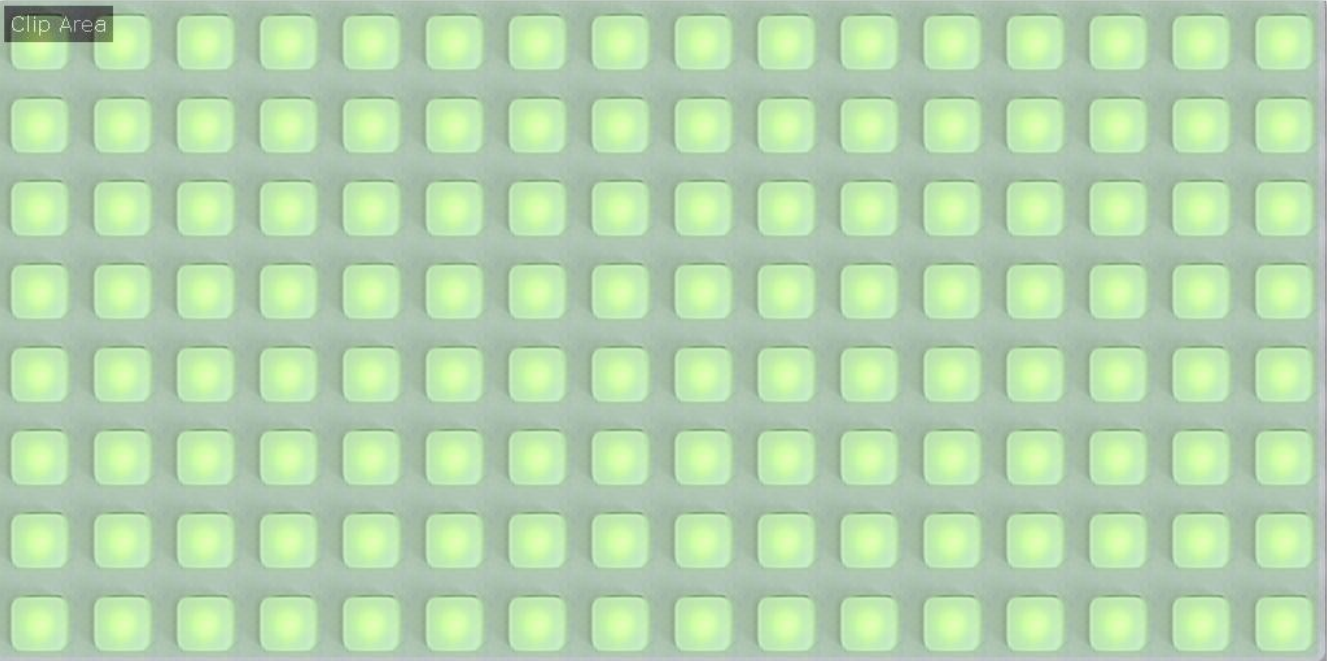
4.1 Ableton Clip Skipper Page

This page attempts to mimic MLR behavior in Ableton with the currently playing clip in each track. Unfortunately it is not very accurate and mostly unusable. Ableton version 7 will work decently with it but the timing in 8 is different and it isn't recommended.

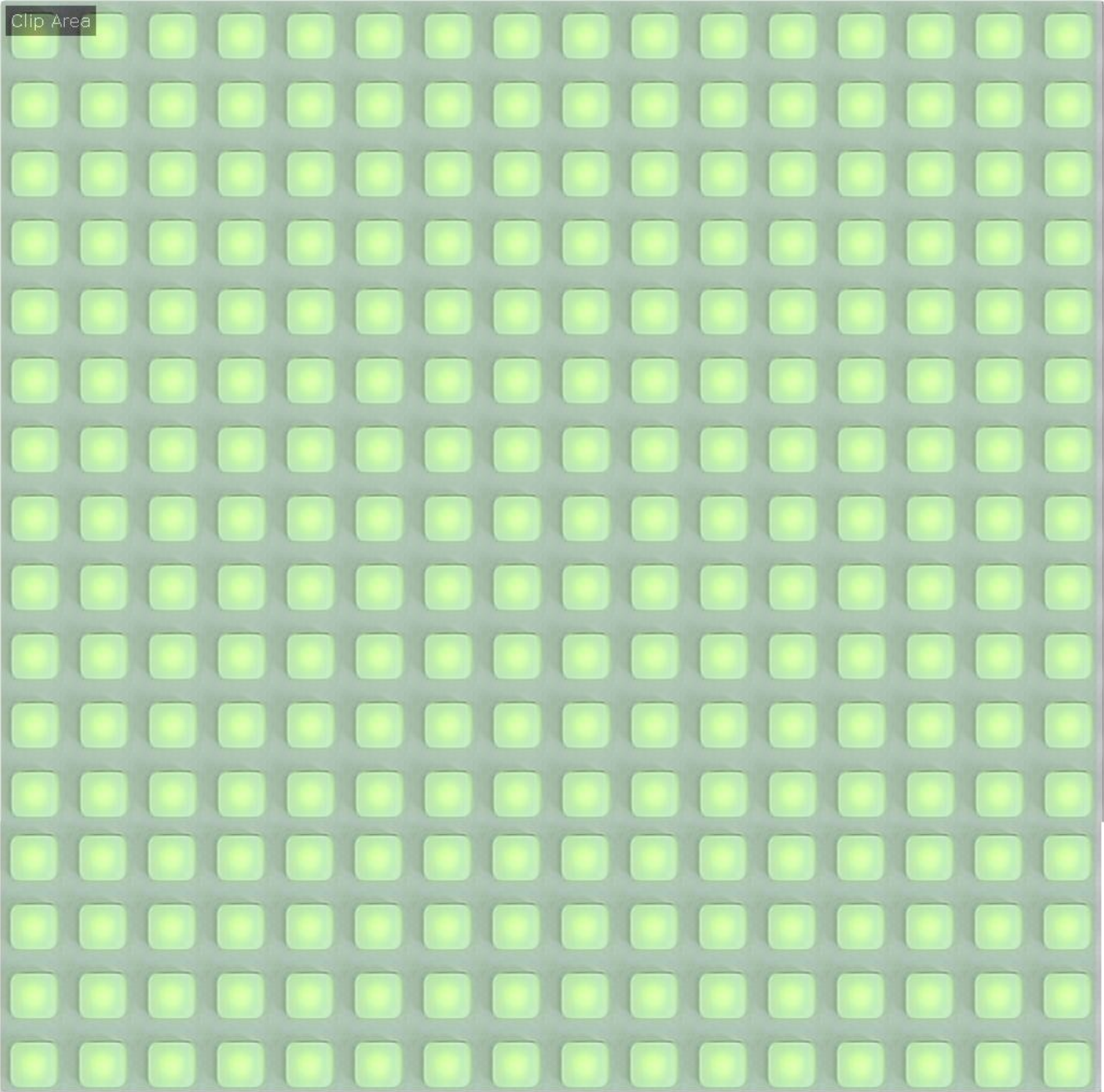
64 Mode



128 Mode



256 Mode

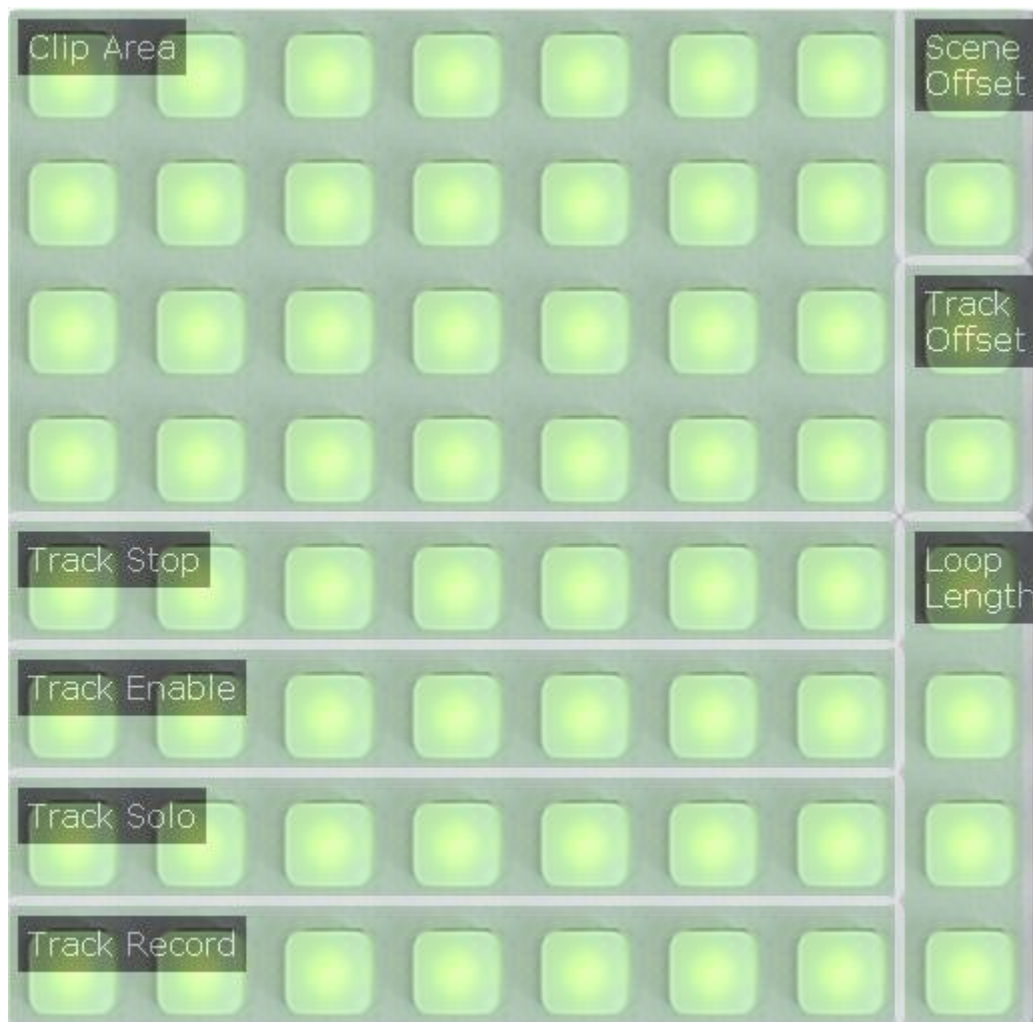


4.2 Ableton Live Looper Page

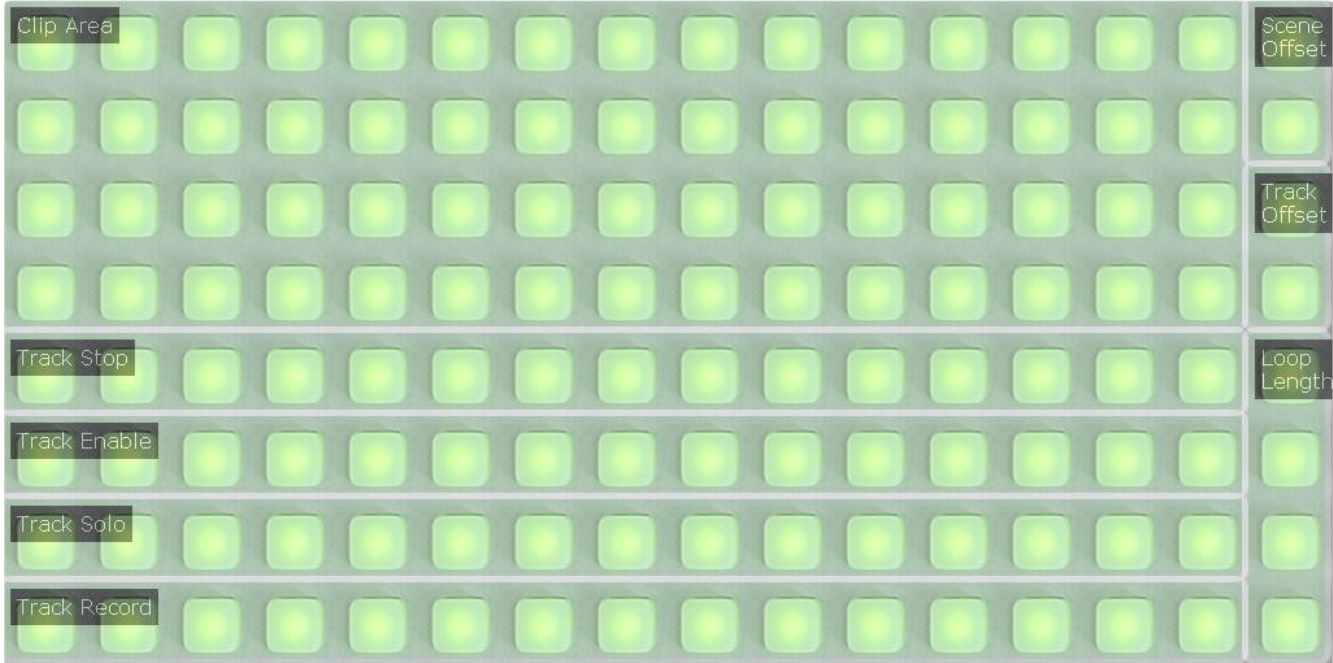
This page behaves much in the same way as the clip launcher page except that it will retrigger clips after a certain amount of time. This behavior can be used to make Ableton act like a giant looping device by setting record enable for the desired track and playing in audio or MIDI. Ableton will record X bars of the information until the retrigger event is sent, at which point it will stop recording and begin playing back the material as a loop. The button layout is the same as the clip launcher except that the bottom buttons on the rightmost column have been replaced with a loop length selector. The buttons act as a selector and the top button is 1 bar loops, followed by 2, 4, 8, and up on a 256.

Make sure to enable MIDI Sync for proper clip state flashing.

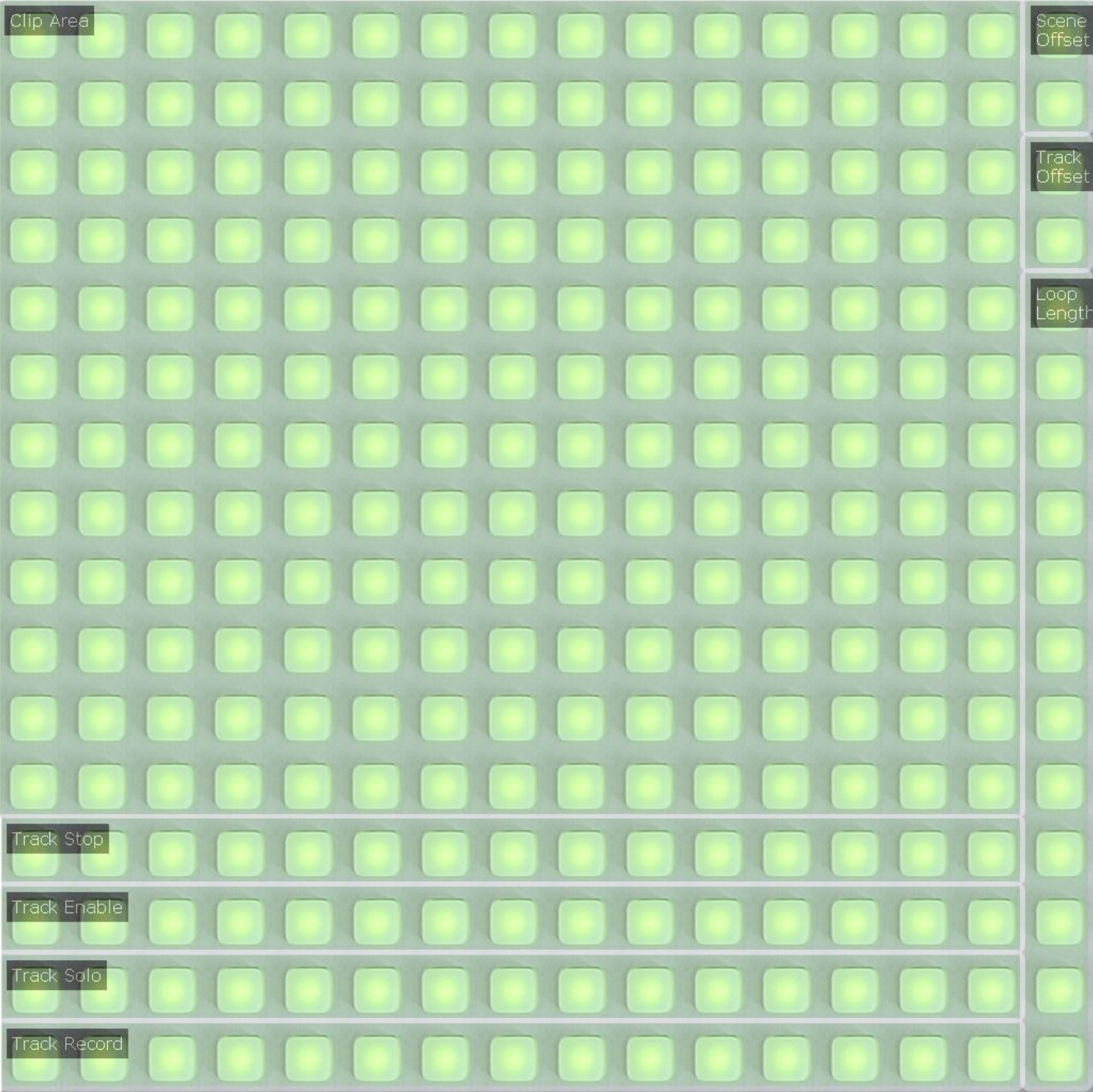
64 Mode



128 Mode



256 Mode



4.3 Ableton Scene Launcher Page

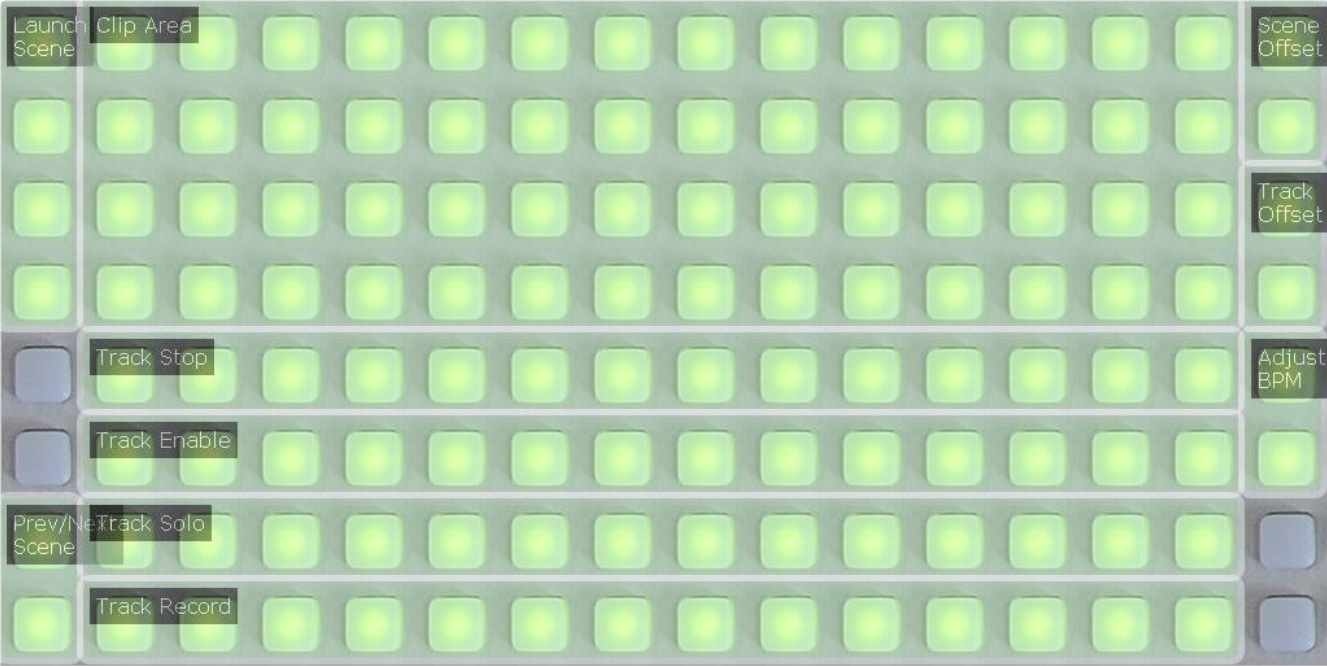
This page is identical to the clip launcher page except that the leftmost column controls the scene launch buttons in Ableton. The active scene will flash and each button will trigger the corresponding scene. The two buttons below the clip area in the leftmost column trigger the previous and next scenes.

Make sure to enable MIDI Sync for proper clip state flashing.

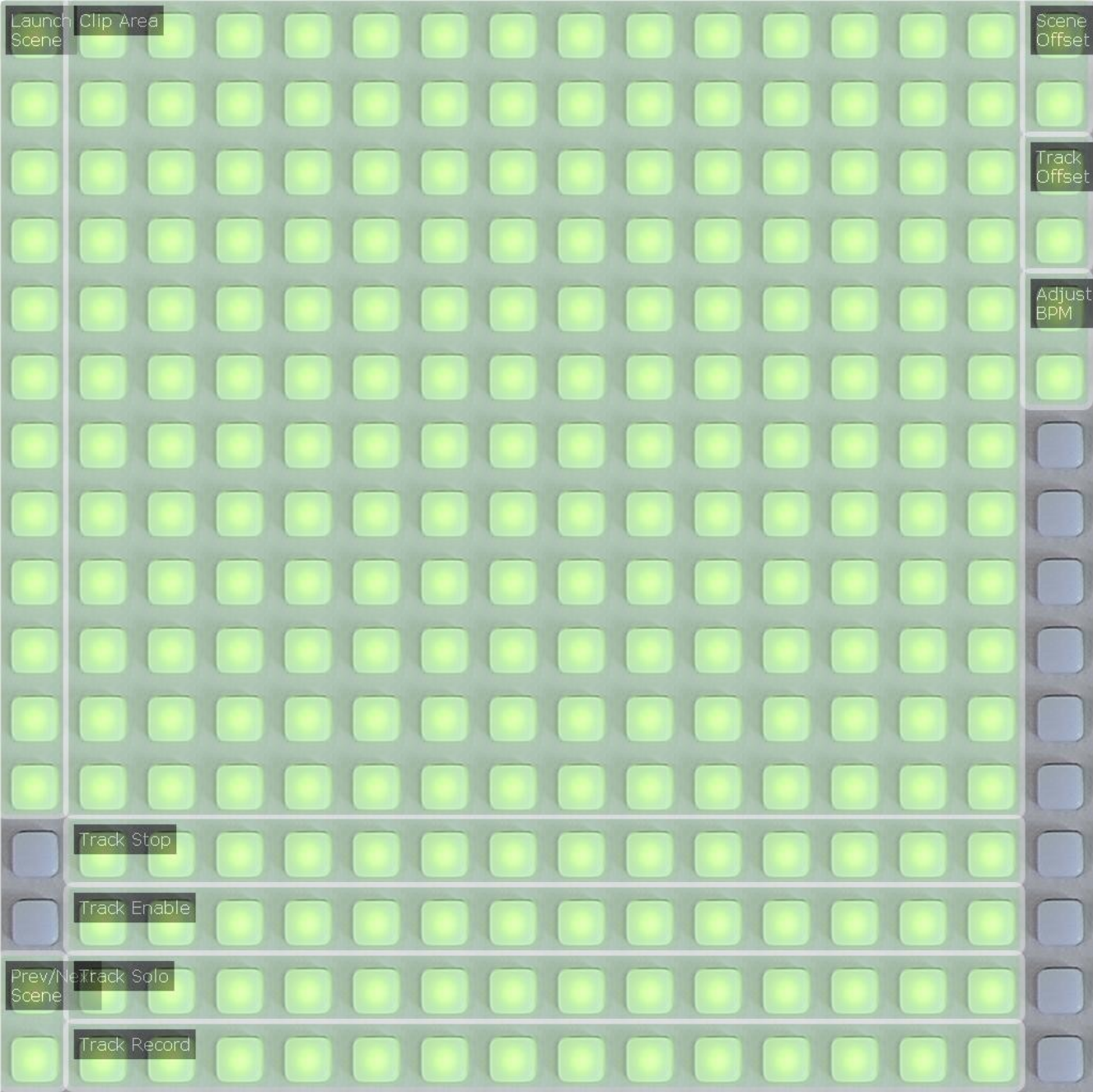
64 Mode



128 Mode



256 Mode



4.4 External Application Page

This page is used to route other monome applications through Pages and gain the benefit of the page switching / pattern recorder overlays that Pages provides. It supports both Monome Serial and SerialOSC applications and you can use any combination of applications regardless of the protocol used with your monome itself.

The OSC in and out ports default to the standard Monome Serial ports that the original monome applications used. When using an old Monome Serial application that doesn't support SerialOSC, use these ports and adjust the prefix to match what the application expects. Clicking the 'Update Preferences' button will enable the external application page and you should now be able to use the application you've routed.

Clicking 'Update Preferences' will also register a Bonjour/Zeroconf service for SerialOSC-enabled applications to connect to. This service will be named "extapp-[OSC in port]-[prefix]". Setup should be as simple as selecting the extapp device from the dropdown in the Max/MSP or M4L application and clicking 'Connect'.

A Disable LED Cache option exists if the cache is giving you trouble but I haven't heard reports of this being needed in a long time and the feature will most likely be removed.

The Ignore /sys/prefix option will cause the external application page to ignore /sys/prefix messages. This will prevent applications from changing the prefix of external application pages via OSC messages. This is useful when you have multiple applications using the same UDP ports and one application is setting the prefix of every external application page to its prefix.

4.5 Groovy Page

This page allows you to create your own custom page types that interact with the Pages subsystem. You get the benefits of Pages' built-in MIDI/OSC routing, monomeserial and serialosc protocol implementation, pattern recorders, Ableton LiveOSC integration, page changing mechanisms and multiple monome support without the hassle of coding all that junk yourself.

There are 4 buttons on the page:

- Save**: saves the current script to a file on your hard drive
- Load**: loads a script into the text box, the script is not started automatically
- Log Window**: shows a log window with any script errors
- Run**: starts the script running
- Stop**: stops the running script

Syntax errors will prevent the script from running. Check the log window if nothing is happening. If an error occurs while the application is running however, the application will continue to run.

Here is the template script that loads when you create a new Groovy Page:

```
import org.monome.pages.configuration.GroovyAPI;

class GroovyTemplatePage extends GroovyAPI {

    void init() {
        println "GroovyPage starting up";
    }

    void stop() {
        println "GroovyPage shutting down";
    }

    void press(int x, int y, int val) {
        led(x, y, val);
    }

    void redraw() {
        clear(0);
        led(0, 0, 1);
        row(1, 255, 255);
        col(2, 255, 255);
    }

    void note(int num, int velo, int chan, int on) {
        noteOut(num, velo, chan, on);
    }

    void cc(int num, int val, int chan) {
        ccOut(num, val, chan);
    }

    void clock() {
        clockOut();
    }

    void clockReset() {
        clockResetOut();
    }
}
```

This example script basically echoes back all events it receives. This is how the script works:

```
import org.monome.pages.configuration.GroovyAPI;

class GroovyTemplatePage extends GroovyAPI {
```

This is basic initialization code. It's saying we want to use the GroovyAPI class and we want to create a new class named "GroovyTemplatePage" that extends it. This API hooks in to the Pages event system and provides a number of utility functions. You will want to change GroovyTemplatePage to a name of your choosing.

```
    void init() {
        println "GroovyTemplatePage starting up";
    }
```

The init() method is called when the script is first started. This init method will print a message to the console window. You will only see the console if running Pages via command line.

```
    void stop() {
        println "GroovyTemplatePage shutting down";
    }
```

The stop() method is called when the script is stopped with the stop button. It's important to terminate any threads you may have spawned in the stop method -- see the MIDIXYPage.groovy file for an example.

```
    void press(int x, int y, int val) {
        led(x, y, val);
    }
```

This is the press event method. Whenever this page receives a press, this method will be called. In this case we're lighting up the led that is pressed.

```
    void redraw() {
        clear(0);
        led(0, 0, 1);
        row(1, 255, 255);
        col(2, 255, 255);
    }
```

The `redraw()` method is called whenever Pages needs to redraw the entire monome (ie. when a page is changed). In this case we are clearing the monome, turning on a led at 0,0, drawing a row at y=1, and drawing a column at y=2. Pages' LED-caching layer will optimize your LED messages, so keep your code simple.

```
void note(int num, int velo, int chan, int on) {
    noteOut(num, velo, chan, on);
}

void cc(int num, int val, int chan) {
    ccOut(num, val, chan);
}

void clock() {
    clockOut();
}

void clockReset() {
    clockResetOut();
}
```

These are MIDI event methods. They will be called when an enabled MIDI input receives an event of the appropriate type. They each echo the received message back to any enabled MIDI outputs.

You can also send OSC messages:

```
Object[] args = [0, 0, 1];
sendOSC("/grid/led/set", args, "localhost", 8000);
```

Ableton

Here's an example of a basic clip launcher page:

```
import org.monome.pages.configuration.GroovyAPI;
import org.monome.pages.ableton.AbletonTrack;
import org.monome.pages.ableton.AbletonClip;

class SimpleClipLauncherPage extends GroovyAPI {

    void init() {
        println "SimpleClipLauncherPage initialized";
    }

    void press(int x, int y, int val) {
        if (val == 1) {
            abletonOut().playClip(x, y);
        }
    }

    void redraw() {
        boolean drewLed;
        for (int x = 0; x < sizeX(); x++) {
            for (int y = 0; y < sizeY(); y++) {
                drewLed = false;
                AbletonTrack track = ableton().getTrack(x);
                if (track != null) {
                    AbletonClip clip = track.getClip(y);
                    if (clip != null) {
                        int state = clip.getState();
                        if (state > 0) {
                            led(x, y, 1);
                            drewLed = true;
                        }
                    }
                }
                if (!drewLed) {
                    led(x, y, 0);
                }
            }
        }
    }

    void note(int num, int velo, int chan, int on) {
    }

    void cc(int num, int val, int chan) {
    }

    void clock() {
    }

    void clockReset() {
    }

    public boolean handleAbletonEvent() {
        redraw();
    }
}
```

A few things to note:

```
import org.monome.pages.ableton.AbletonTrack;
import org.monome.pages.ableton.AbletonClip;
```

These imports will pull in Ableton objects from Pages and make them available to the script. A full list of Ableton objects and their functionality is included below.

```
abletonOut().playClip(x, y);
```

There are two exposed Ableton objects, `ableton()` and `abletonOut()`. `abletonOut()` is used to send commands to Ableton Live. In this case we're instructing it to play clip number `y` on track `x`.

```
AbletonTrack track = ableton().getTrack(x);
```

The `ableton()` object is used to query information about Ableton's state. This pulls out an `AbletonTrack` object which contains `AbletonClip` objects and track information.

```
public boolean handleAbletonEvent() {
    redraw();
}
```

The `handleAbletonEvent()` method is an interface method you can define to cause some action to happen when an Ableton event is received by Pages (usually redrawing the LED state).

Here's a full list of the Ableton objects and methods available:

ableton()

ableton() will return an AbletonState object that can be used to check the current state of Ableton. It has the following methods:

```
AbletonTrack getTrack(int x)
-- returns an AbletonTrack object

HashMap getTracks()
-- returns a the current AbletonTracks in a HashMap

int getOverdub()
-- returns the state of the overdub button

float getTempo()
-- returns the current project tempo

int getSelectedScene()
-- returns the currently selected scene number
```

abletonOut()

abletonOut() returns an AbletonControl object that can be used to send commands to Ableton. It has the following methods:

```
void playClip(int track, int clip)
-- plays the clip on the given track number, clip/track numbers start at 0

void stopClip(int track, int clip)
-- stops the clip on the given track number

void armTrack(int track)
-- arms track

void disarmTrack(int track)
-- disarms track

void muteTrack(int track)
-- 'mutes' a track (disables it)

void unmuteTrack(int track)
-- 'unmutes' a track (enables it)

void stopTrack(int track)
-- stops the currently playing clip on track

void viewTrack(int track)
-- forces the view to select track

void trackJump(int track, float amount)
-- makes the currently playing clip in track jump by amount bars

void undo()
-- forces an undo in Ableton (ctrl-Z)

void redo()
-- forces a redo in Ableton (ctrl-Y)

void setOverdub(int state)
-- sets overdub mode off/on (0/1)

void setTempo(float tempo)
-- sets the Ableton project tempo

void launchScene(int scene)
-- launches a scene

void soloTrack(int track)
-- solos a track

void unsoloTrack(int track)
-- unsolos a track

void setSelection(int offsetX, int offsetY, int width, int height)
-- sets the red ring
```

AbletonTrack

An AbletonTrack object is returned from the `ableton().getTrack()` method. These objects contain information about the track and clips contained within the track.

You will need to import AbletonTrack to use it:

```
import org.monome.pages.ableton.AbletonTrack;
```

The methods of an AbletonTrack object are:

```
AbletonClip getClip(int clip)
-- returns the AbletonClip in slot clip
```

```
HashMap getClips()
-- returns a HashMap of AbletonClips in this track
```

```
AbletonLooper getLooper(int looper)
-- returns the AbletonLooper device number looper on the track
```

```
HashMap getLoopers()
-- returns a HashMap of AbletonLooper devices on the track
```

```
int getSolo()
-- gets the solo state of the track (0=off, 1=on)
```

```
int getArm()
-- gets the arm state of the track (0=off, 1=on)
```

```
int getMute()
-- gets the mute state of the track (0=off, 1=on)
```

AbletonClip

An AbletonClip object is returned from the AbletonTrack getClip() method. These objects contain information about a particular clip.

You will need to import AbletonClip to use it:

```
import org.monome.pages.ableton.AbletonClip;
```

The methods of an AbletonClip object are:

```
int getState()
-- gets the state of the clipslot: 0=empty, 1=stopped, 2=playing, 3=triggered

float getLength()
-- gets the length of the clip in bars

float getPosition()
-- gets the current playhead position of the clip (UNRELIABLE)
```

AbletonLooper

An AbletonLooper object is returned from the AbletonTrack getLooper() method. These objects contain information about a particular Looper device.

You will need to import AbletonLooper to use it:

```
import org.monome.pages.ableton.AbletonLooper;
```

The methods of an AbletonLooper object are:

```
int getState()
-- gets the state of the Looper: 0=stopped, 1=recording, 2=playing, 3=overdub
```

4.6 Machine Drum Interface Page

This page is intended to control an Elektron MachineDrum drum machine. It would probably do interesting things to a MonoMachine as well. The page is divided into 4 modules:

- Pattern Manager
- Kit Randomizer
- Kit Editor
- LFO Manager

The modules are currently hard-coded to specific quadrants but this should eventually be customizable.

You will want to enable a MIDI input that's receiving clock sync from another source (a DAW generally) and enable the MIDI output that's connected to your MachineDrum's MIDI input.

Pattern Manager - Upper Left Quadrant

This is pattern sequencer / selector with mute buttons for each track. You can also use this page to select a song or toggle other global settings.

Pattern Bank Select (A - H)

Pattern Select (1 - 16)

Pattern Sequencer

Track Mutes (1-16)

Song Select (1-8)

Song Offset (x8)

Extended Mode

Song/Pattern mode

Global Settings 0/1

Run Sequence

Extended/Classic Mode

song/Pattern mode

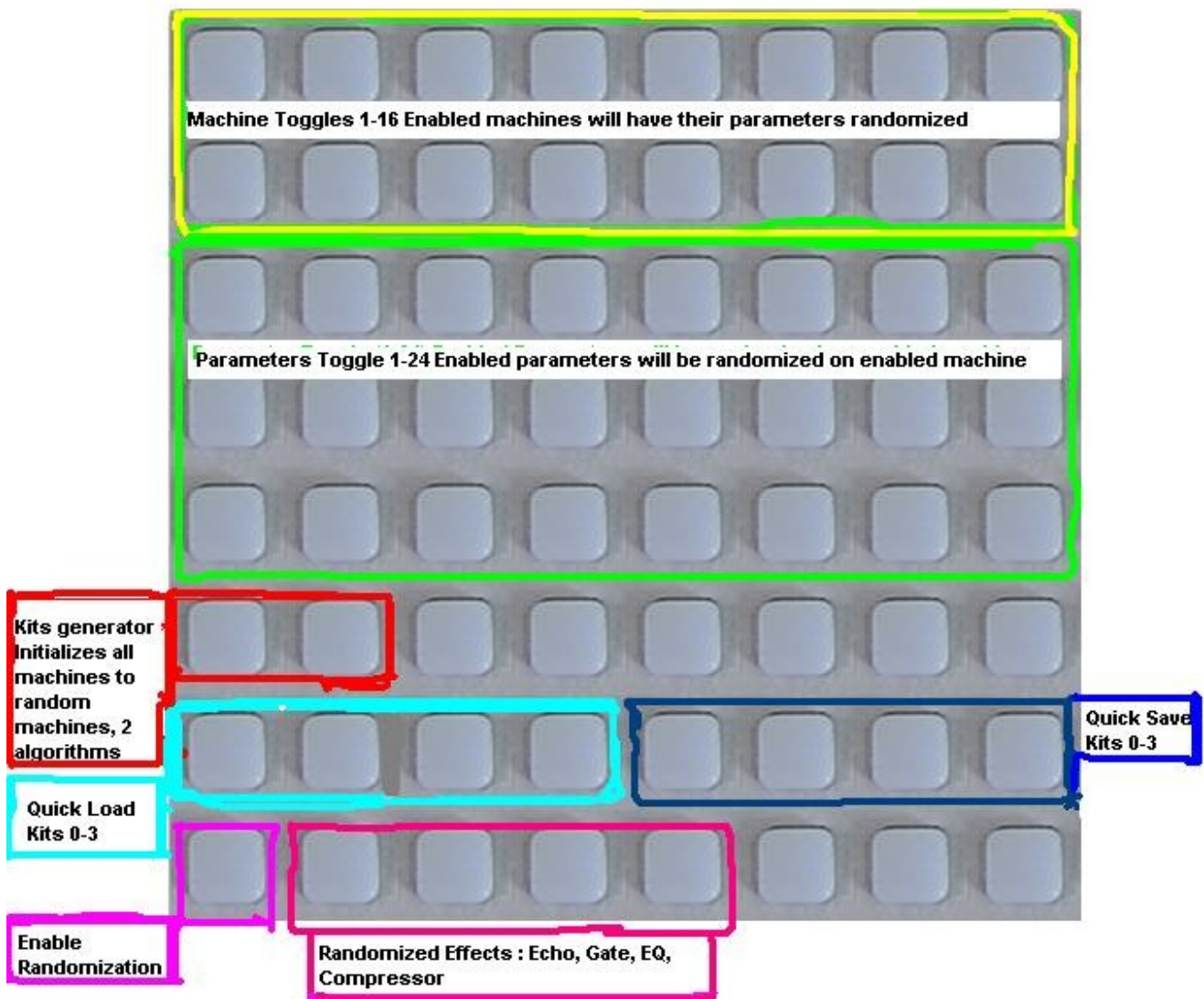
Global Settings 0/1

Run Sequence

Kit Randomizer - Upper Right Quadrant

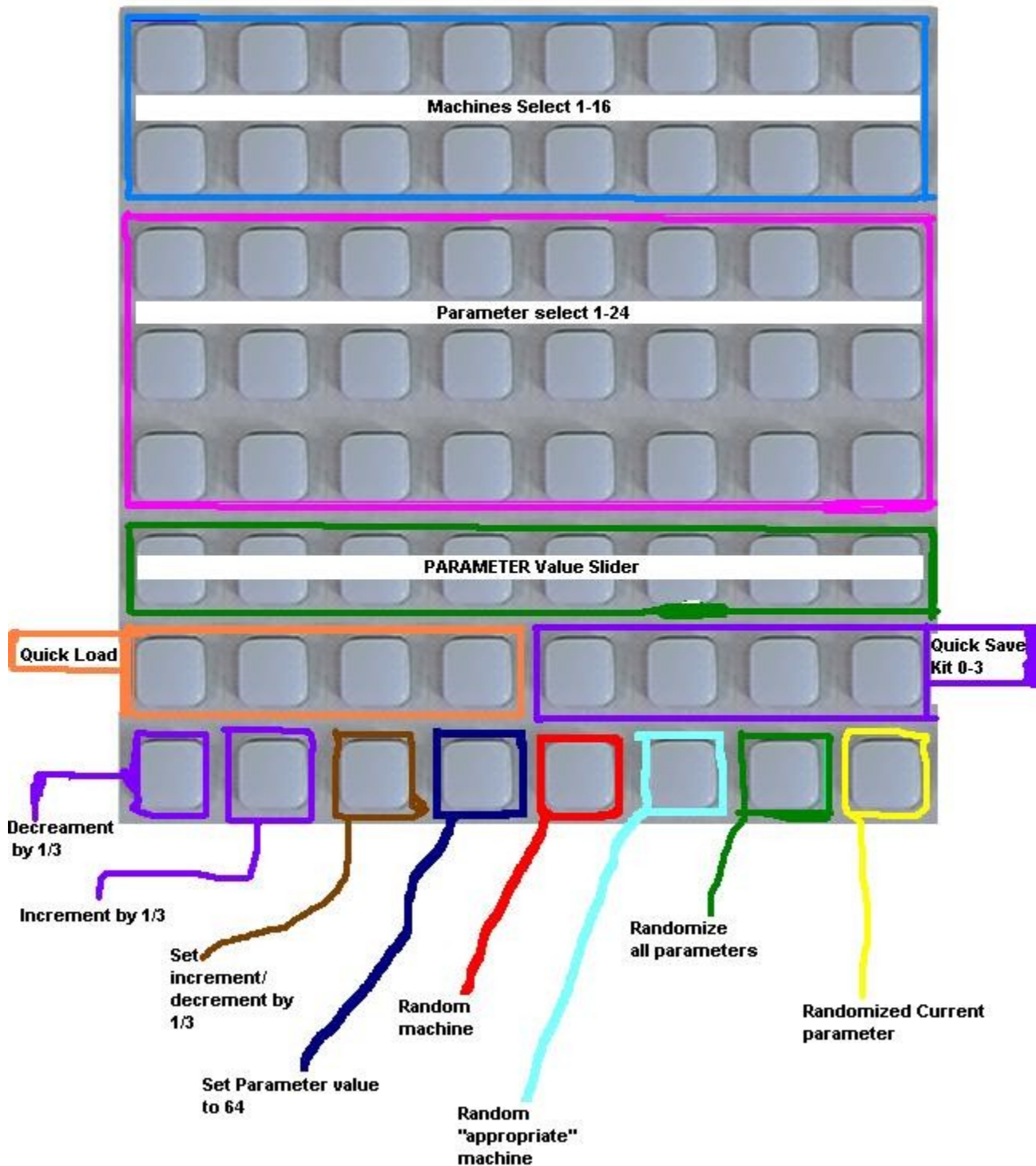
This module is made for obliterating drum kits. It morphs the kit over time based on the Speed setting in the GUI (less ms is faster, don't go below 20). You can choose which machines and which parameters get affected. There are also 4 quick save/quick load slots that save in kit slots 0-3 on the MachineDrum. You can use these buttons to instantly save or recall the state of randomized kits. You can also generate new kits and randomize the effects settings.

The bottom left button turns on randomization. Make sure MIDI clock is being received by the page.



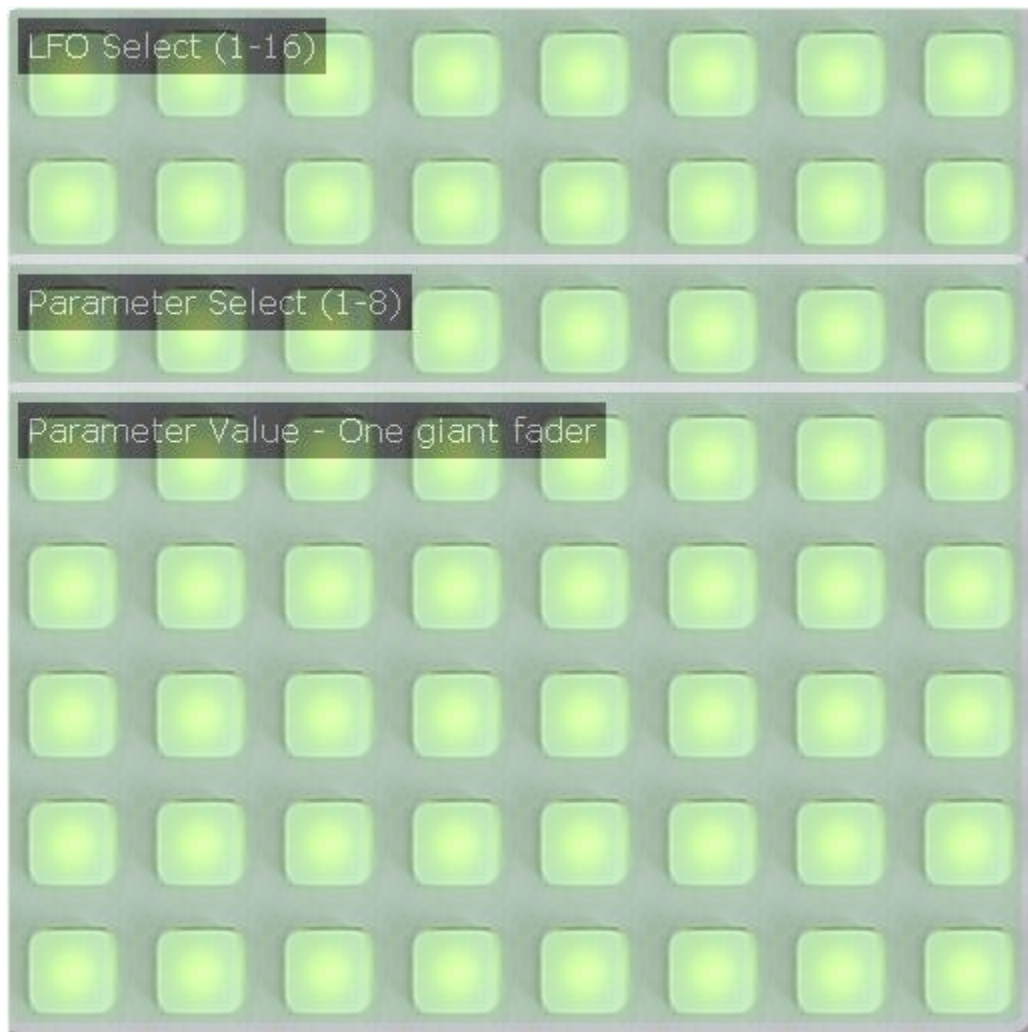
Kit Editor - Bottom Left Quadrant

This module is for more fine grained control of a kit. You select a machine to edit and can set each parameter individually. You can also randomize individual machines or randomly select new machine types.

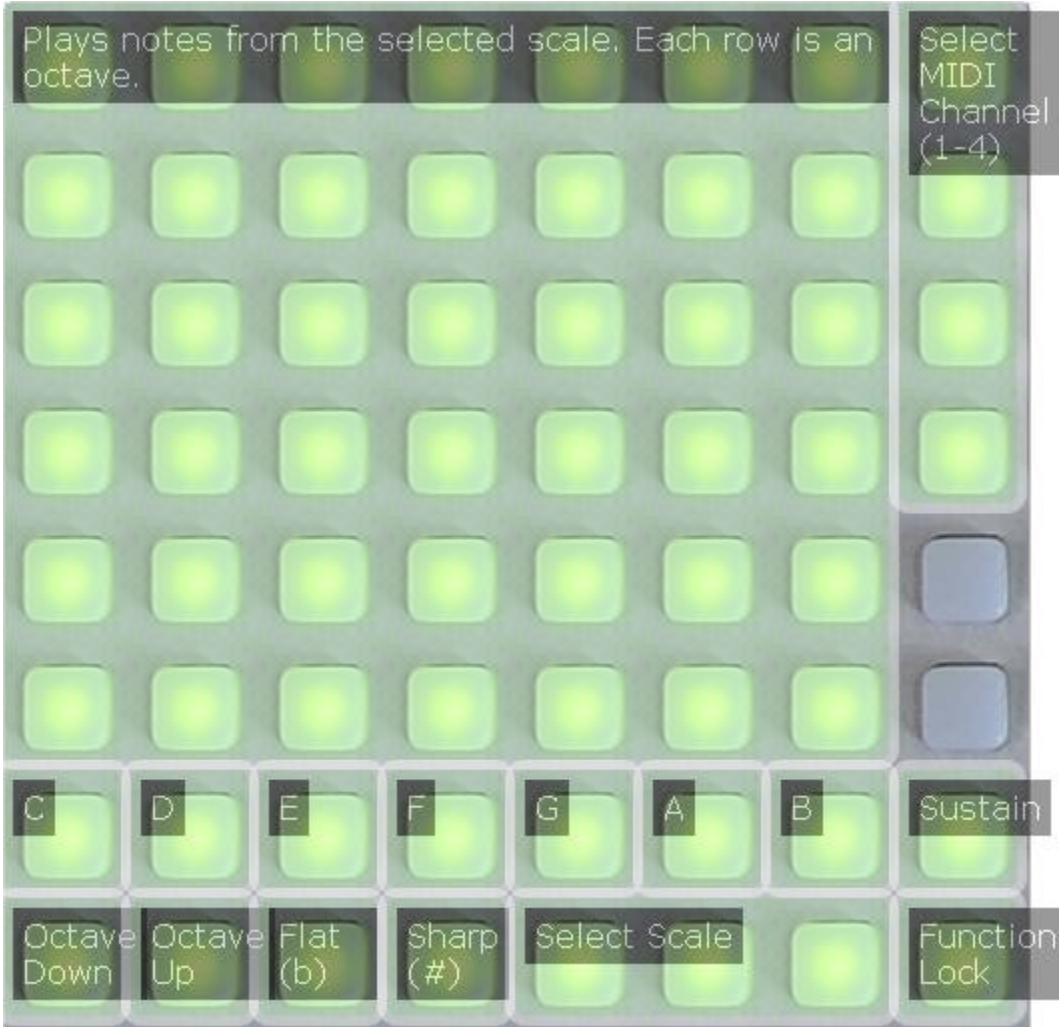


LFO Manager - Bottom Right Quadrant

This module lets you set the 8 parameters of each of the 16 LFOs. The top 2 rows select the LFO, the next row selects the pattern, and the rest of the quadrant is a giant multi-row fader for setting the value.



4.7 MIDI Keyboard Page



Plays notes from the selected scale. Every three rows is on a different MIDI channel, starting with 1.

C

D

E

F

G

A

B

Flat (b)

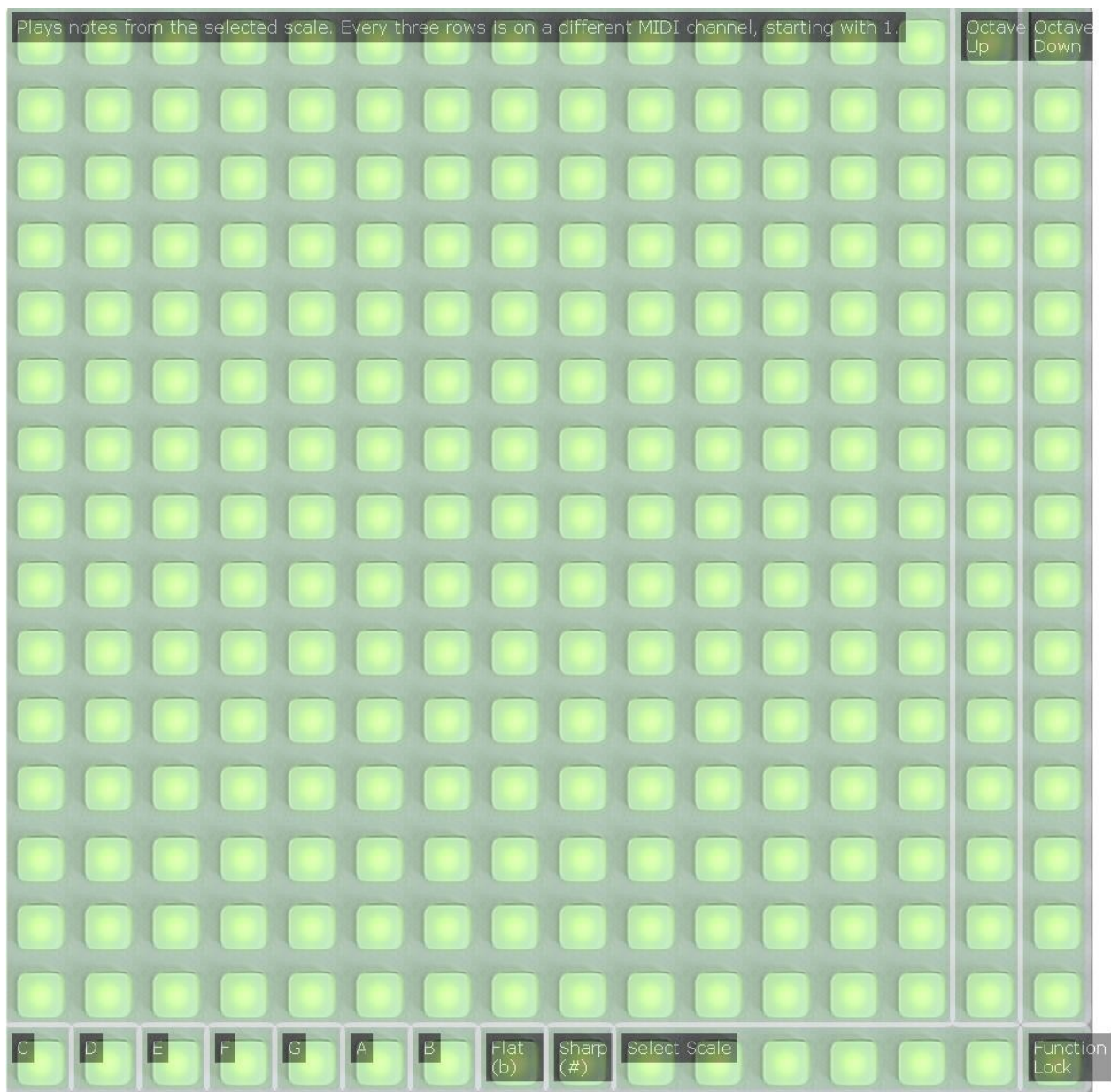
Sharp (#)

Select Scale

Octave Up

Octave Down

Function Lock



4.11 MIDI Sequencer Page

