Usage Instructions for Reaper-ATOM-Integration version 0.3.

In order to use this integration package a special, control track must be created to manage the ATOM controller. For the remainder of this document, this track will be called the “controller track”.

Create the Controller Track

1. In Reaper, Right-click in the TCP and select “Insert New Track”.

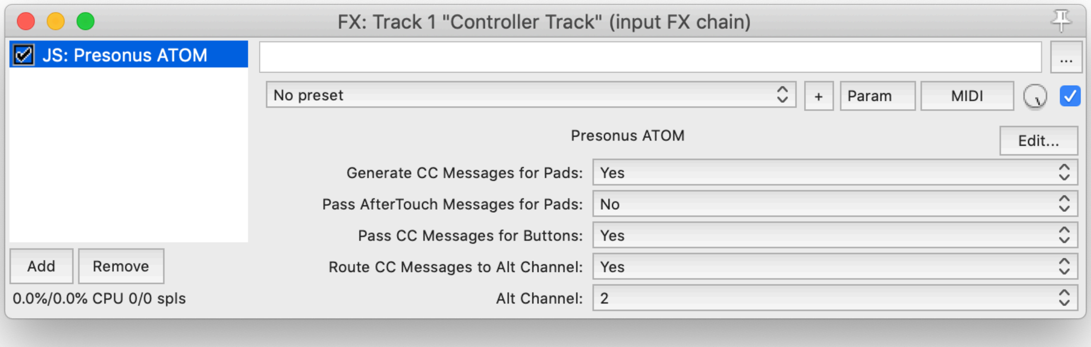
2. Set the Input for the track to “MIDI, Presonus - ATOM, Channel 1”. (In Native Control mode, the ATOM sends all messages on Channel 1)

3. Set the Record Mode for the track to “disable (input monitoring only)”.

4. Click on the Input FX button for the track to open the Input FX Window.

5. Under Filter, type “ATOM”, Select the ‘Presonus ATOM’ plugin, and click OK.

6. Leave the plugin controls set to Defaults. A detailed explanation of what each control option does can be found later in this document.



FX: Track 1 "Controller Track" (input FX chain)

☑ JS: Presonus ATOM

No preset

Presonus ATOM

Generate CC Messages for Pads: Yes

Pass AfterTouch Messages for Pads: No

Pass CC Messages for Buttons: Yes

Route CC Messages to Alt Channel: Yes

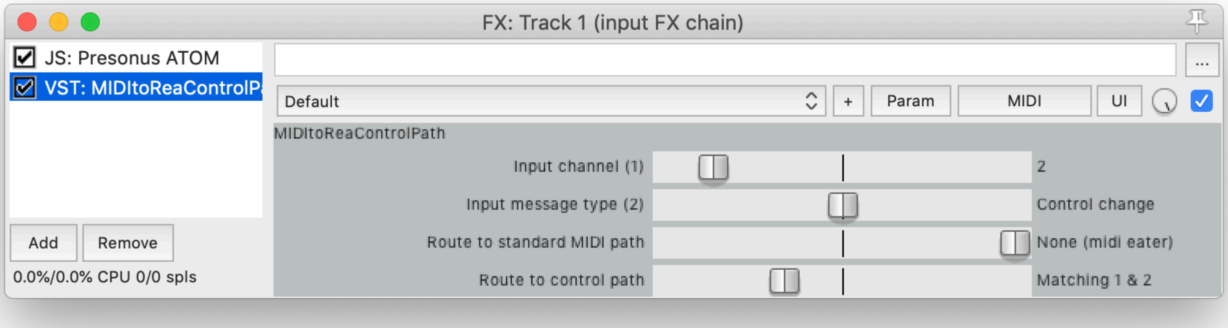
Alt Channel: 2

0.0%/0.0% CPU 0/0 spls

7. Click on “Add” to add another plugin.

8. Under Filter, type “MiditoReaControlPath”, Select the plugin, and click OK.

9. Configure MiditoReaControlPath as shown below:



MIDItoReaControlPath

Input channel (1) 2

Input message type (2) Control change

Route to standard MIDI path None (midi eater)

Route to control path Matching 1 & 2

Input Channel: 2
Input Message Type: Control Change
Route to Standard MIDI Path: None
Route to Control Path: Matching 1 & 2

10. Close the Input FX Window.

11. Arm the track. (Yes, I know Record is disabled, but we still need the MIDI input data)

Congratulations. You now have a controller track. You may want to save this track as a track template for future projects. (Note: I'll add a controller track template to the next release.)

Understanding What The ATOM JSFX Plugin Does

The JSFX plugin acts as a MIDI message converter, router, and filter for messages coming from the ATOM.

Generate CC Messages for Pads (default: Yes) - The ATOM sends Note On and Note Off messages for pads. This tells the plugin to generate a matching CC message for each Note On and Note Off message coming from the ATOM. Data1 of the CC message is the Note number. Data2 of the CC message has a value of 127 for “pressed” and 0 for “released”. These CC messages can then be passed to the Reaper Control Path to control the pad LEDs. This option should be “Yes” on the controller track and “No” on any other track using the ATOM as an input device.

Pass Aftertouch Messages for Pads (default: No) - This tells the plugin to either allow pad aftertouch messages through or to filter them out. The ATOM generates Aftertouch messages that cannot be turned off. Sometimes Aftertouch messages can cause unwanted behavior, particularly when making beats. This is “No” for the controller track and could be either “Yes” or “No” when the ATOM is used as an instrument track input. This is user preference.

Pass CC Messages for Buttons (default: Yes) - This tells the plugin to either allow CC messages from button presses and releases or to filter them out. This should be “Yes” on the controller track and “No” for any other track using the ATOM as an input device.

Route CC Messages to Alt Channel (default: Yes) - This tells the plugin to “move” all CC messages to a channel other than Channel 1. This option must be set to “Yes” for the controller track, otherwise, the ATOM script actions will not be triggered. Setting this to “No” would cause (and allow) all of the incoming CC messages to be recorded along with the Notes. Generally, we wouldn’t want to record the CC messages along with the Note data, but this could have some interesting applications when combined with MiditoReaControlPath.

Alt Channel (default: 2) - This is the destination channel for “Route CC Messages to Alt Channel”. Leave this at “2” otherwise all of the ATOM script actions will fail to trigger. If this must to be changed due to a conflict, all of the ATOM script actions would need to be re-mapped as well.

Pad Colors

From a MIDI perspective, it takes four MIDI messages to set a pad color. A message is sent on Channel 1 to Turn the pad LED On or Off, and then three messages are sent on Channels 2, 3, and 4 to set the Red, Green, and Blue color intensity.

For this integration package, a library function is provided to send the MIDI messages to the ATOM.

ATOM.setPadColor(*string* color) where "color" is one of eight pre-defined colors or one of four user defined colors. The pre-defined and user defined colors can be found in COLORS.lua in the ATOM scripts directory.

The pre-defined colors are red, green, blue, yellow, orange, purple, cyan, and white. The four user defined colors are called user1, user2, user3, and user4. There's actually a ninth pre-defined color, "black", which simply makes the pad LEDs go dark.

Each color definition consists of six values in a Lua table. For example:

```
RED = [[ padColor = {50,0,0,127,0,0} ]]
```

The format is super critical. Do not change anything but the values in the table.

The first three values are the RGB values for the "released" state of the pad and the second three values are for the "pressed" state of the pad. The above example color definition sets a dim red color for the "released" state and a bright red for the "pressed" state. Note the invRED definition in the COLORS.lua file. This can be selected simply by putting a "-" in front of the pre-defined color name in the function call. For example, "-red" would set the pad color to bright when released and dim when pressed. If you don't like the RGB values I picked for the colors, feel free to "tweak' them in COLORS.lua. Just don't change the color names unless you want to go into the library code and change them there as well.

In addition, color definitions don't have to be just dim/bright complimentary values of the same color. Here's an example of my Halloween user defined color:

```
USER4 = [[ padColor = {127,32,0,0,127,0} ]]-- Bright Orange / Bright Green
```

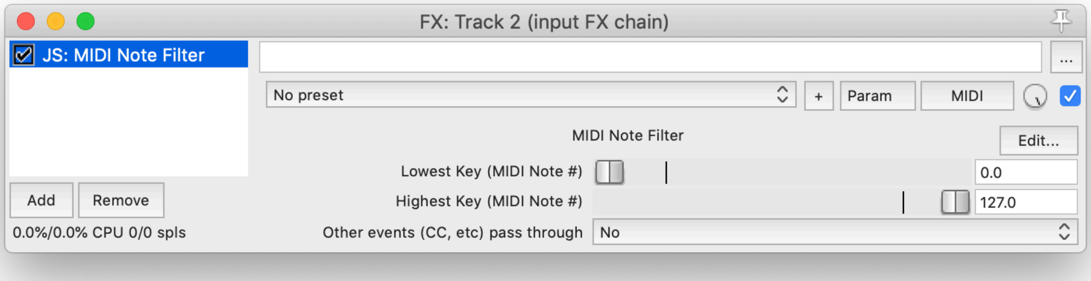
Using the ATOM on an Instrument Track

In order to use the ATOM as an input device for a MIDI track, we need to filter out the CC messages generated by the buttons and, possibly the Aftertouch messages from the pads. This can be done with a MIDI filter plugin or it can be done with another instance of the ATOM JSFX plugin. First, I'll explain how to do it with a Reaper JS MIDI plugin called "MIDI Note Filter".

1. On your instrument track, click the Input FX button to open the Input FX Window.

2. Under Filter, type "midi", Select the 'MIDI Note Filter' plugin, and click OK.

3. Configure the plugin as shown below:



FX: Track 2 (input FX chain)

☑ JS: MIDI Note Filter

No preset

MIDI Note Filter

Lowest Key (MIDI Note #) 0.0

Highest Key (MIDI Note #) 127.0

Other events (CC, etc) pass through No

0.0%/0.0% CPU 0/0 spls

Lowest Key (MIDI Note #): 0
Highest Key (MIDI Note #): 127
Other events (CC, etc) pass through: No

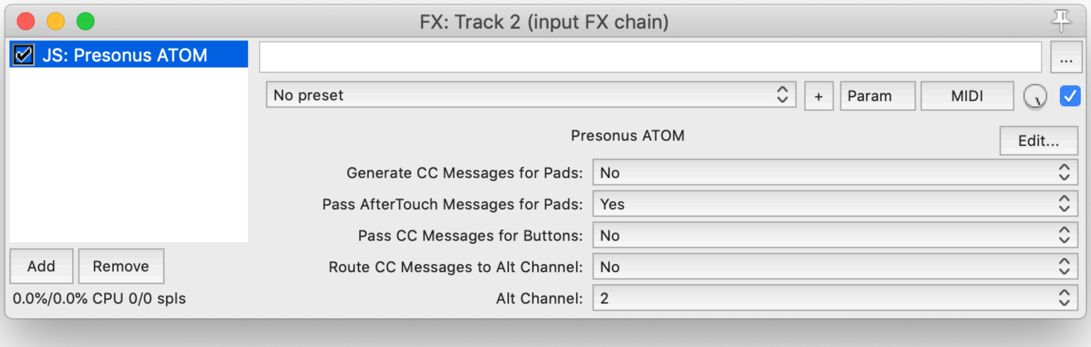
These settings allow ALL Note messages but
filter all other events (like CCs and Aftertouch)

This will work fine… unless we don't want to filter Aftertouch messages. That's where the ATOM JSFX plugin comes in.

1. Following the steps above, add the ATOM JSFX plugin to the Input FX chain.

2. Delete (or Disable) the MIDI Note Filter plugin from the Input FX chain.

3. Configure the ATOM plugin as shown below:



FX: Track 2 (input FX chain)

☑ JS: Presonus ATOM

No preset

Presonus ATOM

Generate CC Messages for Pads: No

Pass AfterTouch Messages for Pads: Yes

Pass CC Messages for Buttons: No

Route CC Messages to Alt Channel: No

Alt Channel: 2

0.0%/0.0% CPU 0/0 spls

Generate CC Messages for Pads: No
Pass AfterTouch Messages for Pads: Yes
Pass CC Messages for Buttons: No
Route CC Messages to Alt Channel: No
Alt Channel: 2

The last two options don’t really matter since the other CC options are set to No.

There are numerous other filter plugins that could be used so this is just two examples.

Enjoy