Support for Behringer MIDI Controllers

Chris Codella, W2PA
3 May 2017

These are modifications to the Midi2Cat package from Andrew Mansfield, M0YGG. They were first applied to OpenHPSDR/PowerSDR 3.3.16 from W5WC. The new features provide special handling for messages produced by the two popular MIDI controllers from Behringer, the CMD PL-1 and the CMD Micro. Although MIDI is a standard of sorts, the messages from so-called DJ controllers such as these can vary widely. The original code worked well with the Hercules controllers, and to a limited degree, with the Behringer controls. That function has been preserved. In fact, it's possible to use more than one of these together, connected at the same time.

Setup Process

Setting up the various controls work as they have before in OpenSDR/PowerSDR mRX PS (hereafter abbreviated Px). By that I mean that the MIDI setup dialog still works and you get a tab labeled properly as "CMD PL-1," or "CMD Micro", similar to how it works with the Hercules. The mapping procedure is mostly the same with the Behringer, with some minor exceptions.

Main Wheel

The PL-1 main wheel is big and very smooth turning, great as a main-tuning knob. You can set it to be, for example, a VFO – the most obvious mapping and probably the most useful. If you do, the code makes use of the wheel's speed sensitivity to change tuning rate in three steps – slow, medium, and fast. It transitions through the speeds gradually as you turn faster. If you give it a good spin you'll fly across the band, and if you turn it slowly you'll fine-tune the radio using the step size defined elsewhere (mine is set to 10Hz). In my setup, I've eliminated the wheel's sensitivity to touching its top surface. It may be useful for something but I haven't thought of anything yet. And besides, it's always possible to touch the top surface accidentally while tuning, causing something else to happen inadvertently – so I'm guessing it's probably not going to be useful for anything. All of this is true also for the two large wheels on the CMD Micro.

To program a main wheel, go through MIDI setup as usual, spinning it clockwise and counterclockwise. The max/min numbers won't matter because to get the speed sensitivity, the code reads the all the reported values anyway and does the right thing. If you plan to use the single large PL-1 wheel for both VFOs, first program it for VFOA, then keep reading the next section. If you're programming the two large wheels on the Micro, just follow the above instructions.

For the PL-1, there is a new CAT function I've added called "Toggle Wheel to VFOA/VFOB." You can map this function to any of the buttons. It is used to toggle the PL-1's main wheel’s function between controlling VFOA and VFOB. To set up this function properly you need to do it in this order:

1. In MIDI Setup, map VFOA to the main wheel as above.
2. Map a button of your choice to the “Toggle Wheel to VFOA/VFOB” function.
3. Save and exit MIDI Setup.
4. Press the button you just mapped in step #2 once (and **only** once) to switch to VFOB.
5. Go back into MIDI setup.
6. Map VFOB to the wheel, which behaves now as a different control from that in step #1.
7. Save and exit setup.
8. Have fun.

Small Knobs

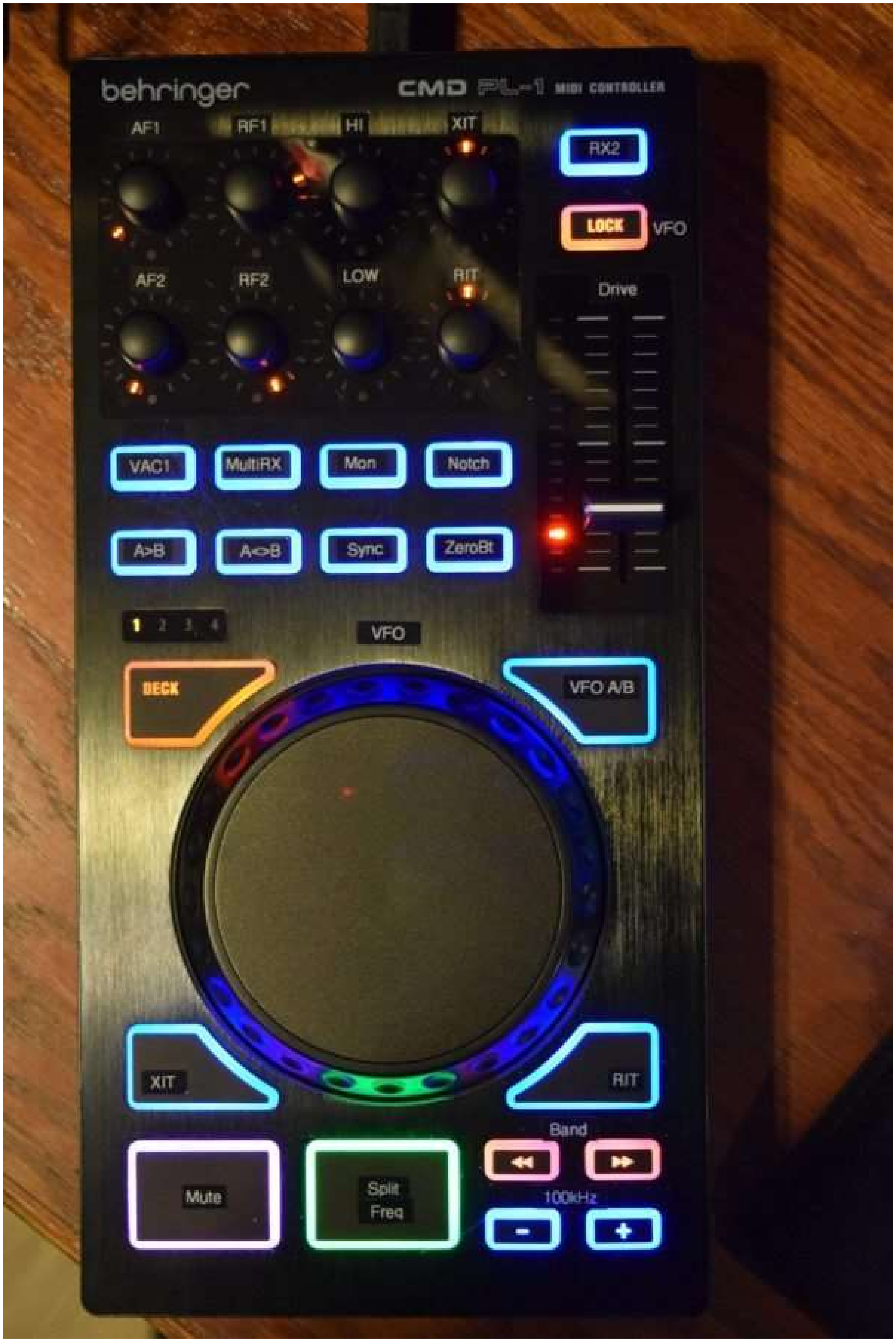
The small knobs on the Behringer CMD PL-1, and the center knob on the Micro, all behave like mini-wheels (they keep turning without limit) instead of knobs as Px defines them or is expecting knobs to behave, i.e. with limits and values, like a slider or potentiometer. That’s why Px lumps knobs and sliders together. So when you program a PL-1 knob, you have to choose “Wheel” as the control type in the MIDI setup. You can map any knob (mini-wheel) to anything in the UI that makes sense. That is, it can be used to control anything that has a value that changes continuously, whether or not that value has limits in the Px window, such as on sliders.

In addition to handling messages from the Behringer controllers sent to the computer, I’ve written code that activates the LEDs on the PL-1, such as those around each knob and along side the single slider. As you rotate the knobs or slide the slider, the LEDs light up roughly corresponding to a percentage of the specific function’s setting, somewhere between its minimum and maximum values. At the time I’m writing now, this feature works for AF gains, AGC gains, RIT/XIT, drive level, CW speed and some others. I can add more if needed.

For RIT/XIT there are additional behaviors. First, the center (midpoint) LED lights only when the RIT/XIT is exactly zero. Then, the counterclockwise and clockwise LEDs light depending on how much you turn the knob. Despite the wide range of settings in PowerSDR, I chose to limit the LEDs to only indicate a maximum of plus or minus 2 kHz. I found that if I took in a wider range, they changed too slowly to have any meaning. Note: you can still keep cranking the knob and go well beyond 2 kHz all you like – but the LEDs will hit their maximum and minimum at 2 kHz or slightly less.

For most of the mappings of knobs to on screen (UI) slider controls, the LEDs also work in the reverse direction. For example, if you map a PL-1 knob to the RX1 AF Gain, then slide the on-screen RX1 AF slider in the user interface, the LEDs on the PL-1 will change just like you were turning the PL-1 knob.

The second feature of these knobs is that they act as buttons when you push down on them. So you can map them as buttons. But I’ve implemented code that detects when you push them to zero the RIT/XIT. This seems more useful than on/off, which you can map to any other button you like.


behringer
CMD PL-1 MIDI CONTROLLER
AF1
RF1
HI
XIT
RX2
LOCK
VFO
AF2
RF2
LOW
RIT
Drive
VAC1
MultiRX
Mon
Notch
A>B
A<>B
Sync
ZeroBt
1 2 3 4
VFO
DECK
VFO A/B
XIT
RIT
Band
Mute
Split Freq
100kHz

Buttons

The PL-1 and Micro buttons have fairly standard behavior—they produce messages pressing down and springing back up. You map them just like with any other controller.

Some buttons are capable of changing to one other color from their default orange. So, you can add messages to M0YGG's already existing MIDI setup dialog to make them do what you want. For the PL-1, you just need to send back the message as received, but change the *value* part of the message to a number, depending on what you want the button's LED to do. For default orange, send 0 (0x00 – 00 hexadecimal); for the alternate color, send 1 (0x01), and to have it alternate or flash between the two colors, send 2 (0x02).

The buttons that can change color have only one alternate color: e.g. PL-1 CUE can be pink/purple, Play/Pause can be green, and all the others can be blue; and on the Micro, only the Play/Pause and CUE buttons have alternate colors). I added a section in the code to change some of them to their alternate color on startup for some variety. I think it looks better than all orange. I may make this customizable in Setup some time later. For now, you can use the built-in messaging (as above) to change color however you like.

As an example, I have my PL-1's SCRATCH button mapped to toggle between VFOs. In MIDI setup, I entered a message to turn it blue for VFOA and orange for VFOB.

See the photo above for examples of mappings and colors, and some labels appropriate for SDR use.

PL-1 Slider

The PL-1's slider can be mapped in setup as a "knob or slider." I had to do some interception of its messages because its control ID changes (something none of the other controls do). Now it works fine and the change in control ID is transparent to the user (although you might notice I chose to use "73" – 0x49 – as the code). When you map it to an existing slider in the UI, the LEDs follow the slider's movement. The one bit of quirkiness comes about because there is, of course, no way to move that slider under program control. Thus, if you change the slider in the UI, the LEDs will change and become out-of-sync with the actual slider, until you move it yourself. When you do, its value will jump instantly to its actual physical setting.

Notice that the knobs don't have this problem. Since the knobs have no absolute position (they just freely turn), there is no direct correspondence between knob position and value or UI slider position, and so the quirkiness described above for the PL-1 slider's LEDs isn't an issue with the knobs.

On the Micro, the sliders behave normally but require a similar code mapping, transparent to the user, because their default behavior is to duplicate codes used by some buttons, which would preclude mapping them to specific functions without clashing with those buttons.

Multiple Controllers

M0YGG's original code nicely handles multiple controllers. The changes I made to handle the PL-1 have their effect only for the PL-1. So programming the Hercules, for example, should be unaffected. In fact, you can use both connected at the same time. You can also use both Behringer controllers at the same time.