

Using an ASIO Audio Interface and Digital Audio Workstation Software with openHPSDR

Revision 3—14 Jun 2015—WU2O

Introduction

Many people are using relatively low cost, professional audio interfaces such as the Behringer UMC202, Focusrite Scarlett 2i2 and Steinberg UR22 to provide audio I/O for their Apache Labs ANAN series software defined radios (SDRs). These interfaces support balanced audio connections to microphones and to amplified studio monitor speakers. They also provide built-in -48VDC phantom power supplies for condenser microphones. More importantly, these interfaces support the use of the professional, low latency ASIO sound drivers. ASIO drivers, developed by Steinberg in the 1990's, provide near real-time latency in support of personal computer audio processing. ASIO compatible hardware and software is ideal for use with SDRs, particularly when it is desirable to provide only an Ethernet connection to the transceiver without using a microphone or speaker directly attached to the radio.

openHPSDR mRX, the SDR software application generally used with the ANAN series radios, is fully capable of interfacing with ASIO supported audio devices such as those described above. Merely select the ASIO driver and the interface in use in the "VAC1" setup window. Buffer sizes and sample rates have to be selected for glitch free audio, of course, but everything generally works in a straightforward fashion. All of the normal audio processing facilities in openHPSDR work in this configuration including COMP, VOX, DEXP, TX EQ, RX EQ, leveler, CESSB and ALC. Only MIC (gain) does not function, being handled instead by the TX Gain setting in the VAC1 setup window. PTT functionality can be accomplished using the space bar keyboard shortcut, VOX, the MOX button, or by using the serial port PTT facility found in Setup > CAT Control > CAT. My preference is to use the later facility, easily accomplished with a desktop or foot switch by wiring the switch across the serial port Pin 7 (RTS) and Pin 8 (CTS) and selecting RTS for PTT in the setup window.

A natural desire when using professional audio interfaces and ASIO drivers is to go beyond the admittedly outstanding audio processing facilities provided for in openHPSDR. By exploiting one of many different digital audio workstation (DAW) software applications to process microphone audio, greatly enhanced control of transmitted audio is possible. While this can be facilitated by using the ubiquitous Virtual Audio Cable software by Eugene Muzychenko, Muzychenko's software does not support ASIO, forcing the use of Windows sound drivers and losing the low latency benefits of ASIO.

In order to allow an all-ASIO audio routing solution for low latency connections between a professional audio interface, DAW software and openHPSDR, the audio processing and routing application produced by VB-Audio Software that goes by the rather unlikely name of Voicemeeter Banana can be used. It can be downloaded here: <http://vb-audio.pagesperso-orange.fr/Voicemeeter/banana.htm>

The following is offered as examples of how to make this all work. The exact configuration will vary depending on the audio interface and DAW software in use, but the following examples, based on a Behringer UMC202 and either Pro Tools 10 or Audiomulch, are sufficiently complete that they should act as a good guidelines for most other configurations.

Installing and Configuring an ASIO Audio Interface for use with openHPSDR

Perform the following steps first to ensure that the audio interface works well with openHPSDR and to determine and configure the optimum settings for audio sample rates and buffer sizes. Also, once this section is completed, the radio can be operated in this manner without the complexity of using a DAW.

Access the control panel for the audio interface and set the buffer size and sample rate as desired. 48KHz is probably the best choice as that is the native internal audio sample rate for openHPSDR. 24 bit sample size is also recommended for best performance with DAW software but 16 bits will work fine. In the example shown in Figure 1, I happen to use 96KHz with my UMC202 because it works well with my DAW software, and also in this example I use a buffer size of 512 on the UMC202, but your settings may be different.

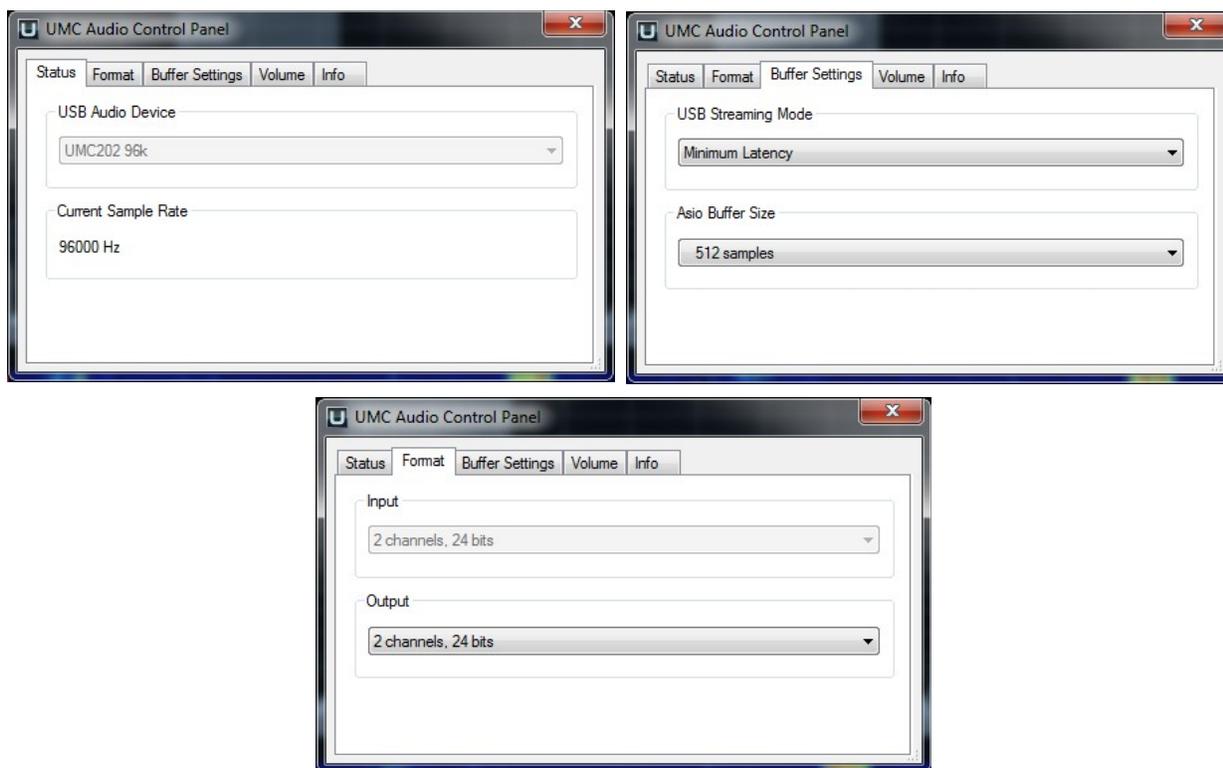


Figure 1—Example of Digital Audio Interface Configuration.

In the openHPSDR VAC setup select the ASIO driver, and the interface and the buffer size and sample rate selected in the sound interface control panel. The buffer size and sample rate in the VAC setup **must** match that selected in the interface control panel.

Enable the VAC1 interface and listen to the receiver audio using speakers or headphones attached to the audio interface. It should have no pops, crackles, glitches or other artifacts that indicate buffer overruns or underruns. Adjust the buffer size to be as small as possible to achieve the lowest latency. It is often advantageous to start with the largest buffer size and an arbitrarily large value for Buffer Latency, say 120mS. Then reduce the buffer size in steps until audio quality degrades, then go back one setting. Then work on minimizing Buffer Latency. Buffer Latency will generally wind up being a little more than twice the buffer size/sample rate. In the example shown in Figure 2, 512/96000KHz equals 5.3mS and twice that is 10.6, so it rounded up to 11mS.

Each time the buffer size or latency in the audio interface control panel or openHPSDR is adjusted it is generally necessary to disable VAC in openHPSDR otherwise errors will occur. It may be necessary to restart openHPSDR a number of times during this process in order to ensure that the sound drivers in openHPSDR are properly reset.

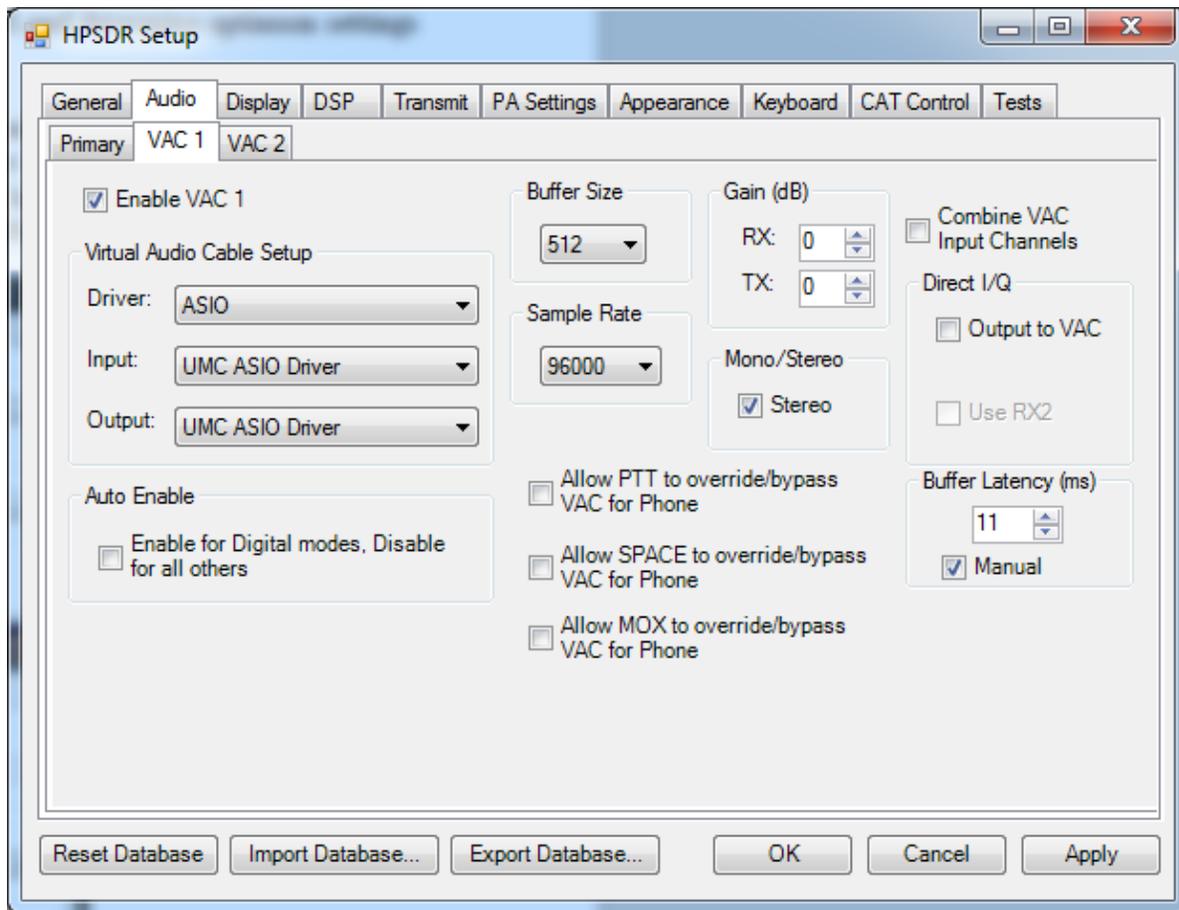


Figure 2—Example openHPSDR Configuration For Use With a Stand-alone Digital Audio Interface.

Once the settings in openHPSDR VAC setup appear to be providing the lowest latency with good audio quality, then test transmit audio. First adjust the microphone gain on the audio interface so that the clipping indicator flashes only very rarely, if at all. This is **critically important** because clipping will exhibit rather dramatic distortion on the transmitted audio. Then adjust gain and other audio settings in openHPSDR the same as for a microphone attached to the front panel, although gain must be adjusted using the VAC TX Gain setting not the main MIC (gain) setting in the GUI. Finally, make a few QSOs to ensure that signal reports are good.

At this point the radio can be operated quite effectively using only the digital audio interface and with only an Ethernet connection to the radio. Many people will be quite happy stopping right here. However, if there is a desire to integrate digital audio workstation software processing into your configuration, read on!

Installing and Configuring Voicemeeter Banana to Route Audio Through DAW Software

This part of the example details how I setup Voicemeeter Banana to allow my DAW to route and mix both transmitted and received audio for both phone and digital modes. This may be a bit more complex than what many people wish to do, so use only the parts of the configuration that may be of interest to you.

First download and install Voicemeeter Banana. Again it can be downloaded from this location:

<http://vb-audio.pagesperso-orange.fr/Voicemeeter/banana.htm>

Most people who are reading this will already have Virtual Audio Cable or VB Cable software installed. Voicemeeter can smoothly coexist with both if necessary so there is no need to uninstall anything.

Next, configure Voicemeeter Banana as follows and as shown in Figure 3:

1. In the upper right corner, select the Hardware Out interface for A1 to be your audio interface. This will automatically select it for Hardware Input 1 as well.
2. On Hardware Input 1 select route A1. This will route audio from the DAW to the speakers or headphones attached to the audio interface.



Figure 3—Voicemeeter I/O and Routing Configuration For DAW Integration.

3. On Virtual Inputs Voicemeeter VAI0 select route B1. This will route audio from the DAW to the openHPSDR VAC transmitter input.
4. On Virtual Inputs Voicemeeter AUX select route B2. This will route audio from the DAW to Fldigi or some other digital mode program.
5. Ensure that all levels are set to 0dB.
6. Referring to the example in Figure 4, in the upper right corner, select Menu > System settings/Options and set Virtual ASIO Type to Float32LSB and Engine Mode to Swift. Then set the Preferred Main Sample Rate, which **must** match your audio interface sample rate.



Figure 4—Example Voicemeeter System Settings and Patch Insert Configuration For DAW Integration.

7. Again referring to Figure 4, at the bottom of the System settings/Options window, enable the Patch Inserts as follows:

- a. In1 Left and In1 Right. This will route microphone audio from the audio interface to the DAW.
- b. In4 Left and In4 Right. This will route receiver audio from openHPSDR to the DAW.
- c. In5 Left and In5 Right. This will route transmitted audio from Fldigi or some other digital mode program to the DAW.

9. Close the setup window, select Menu > Save Settings and save these settings. Save as type "*.xml".

Configure openHPSDR for Voicemeeter Banana

- 1. Restart openHPSDR.
- 2. Change the VAC setup to point to Voicemeeter Virtual ASIO as shown in Figure 5 and save those settings.

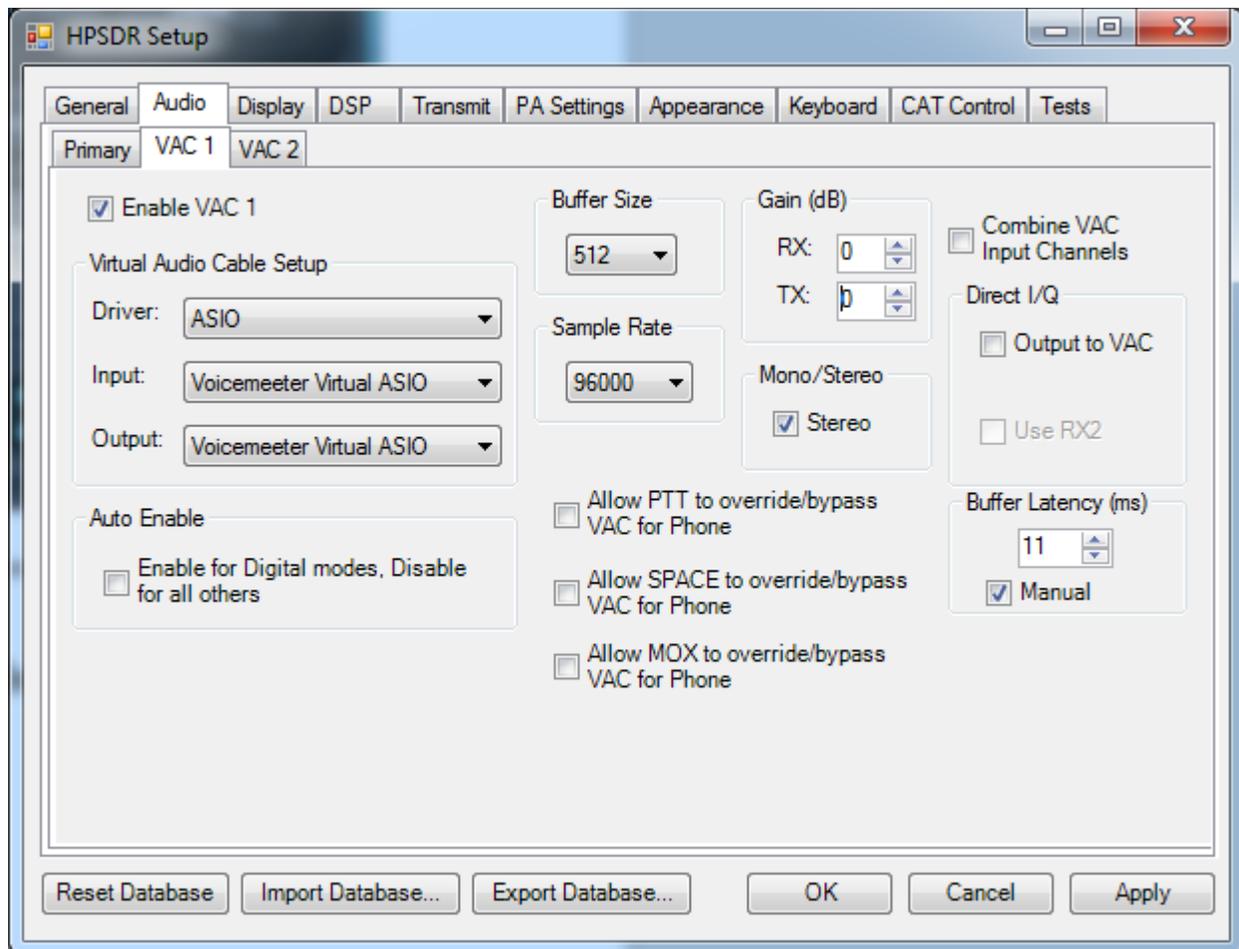


Figure 5—openHPSDR Configuration For Use With Voicemeeter.

Initial Testing

Before moving on to DAW setup, test the Voicemeeter Banana, audio interface and openHPSDR configuration as follows (this is facilitated by the fact that the Voicemeeter Patch Inputs are bypassed when the DAW is not running and connected, so ensure that the DAW software is not running):

1. Speak into the microphone and listen to the output of the audio interface. You should hear yourself. Don't worry about levels right now.
2. On Virtual Inputs Voicemeeter VAIO select route A1 as shown in Figure 6. The radio received audio should be audible. Again don't worry about levels right now.
3. On Hardware Input 1 select route B1 in Figure 6. Make a test transmission on the radio. This should work. Once again, don't worry about levels right now.



Figure 6—Voicemeeter Test Configuration.

If you ever wish to operate the radio without the DAW you can use these settings to do so (and ensure the DAW software is not running). However, with the goal being to actually integrate and use the DAW, continue with the following step:

4. **Return the settings back to their original configuration:** on Virtual Inputs Voicemeeter VAIO select route B1 and on Hardware Input 1 select route A1 as shown in Figure 7.



Figure 7—Voicemeeter DAW Configuration.

Configure the DAW Software

The following sections detail how to configure two popular DAW applications, Pro Tools and Audiomulch. Each DAW will be different, of course. And each operator will probably want to do this a bit differently as well. However it is hoped that these will serve as an adequate examples in general.

Configuring Pro Tools

The following is what I did in Pro Tools 10.

1. Set up the Playback Engine to be Voicemeeter Insert Virtual ASIO as shown in Figure 6. Hardware buffer size in the example is shown as 512. This may be different for your specific configuration but in any event **must** match that set in the audio interface. The remaining settings should be as appropriate for your PC and are probably best left the same as the default settings.

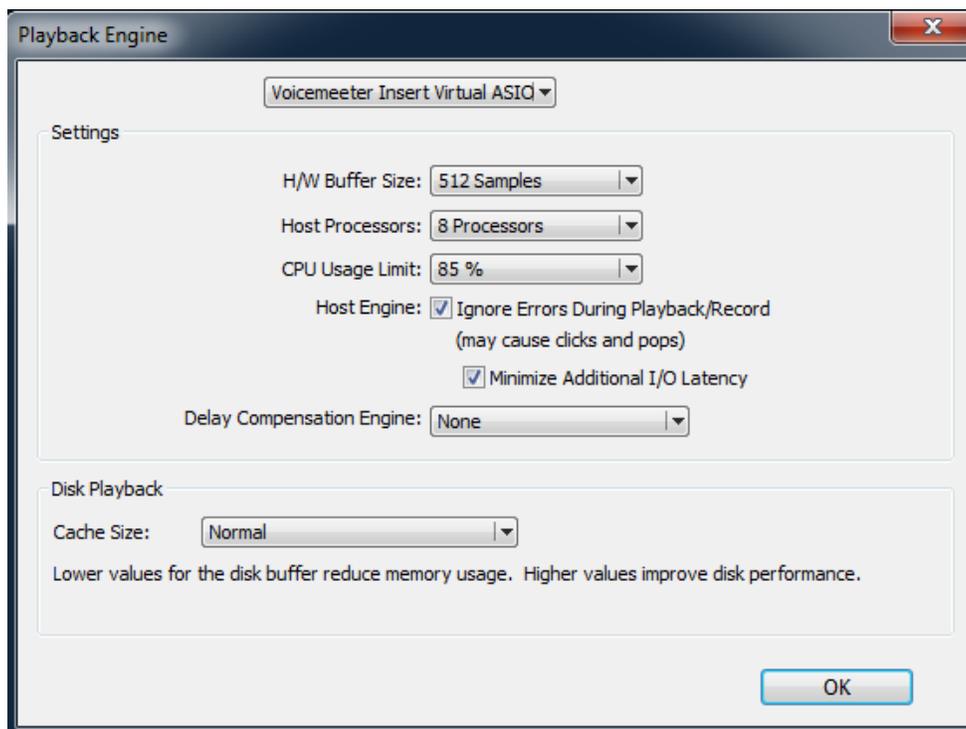


Figure 6—Example Pro Tools Playback Engine Configuration.

2. Set up the Input I/O as shown in Figure 7. You can certainly name these however you prefer.

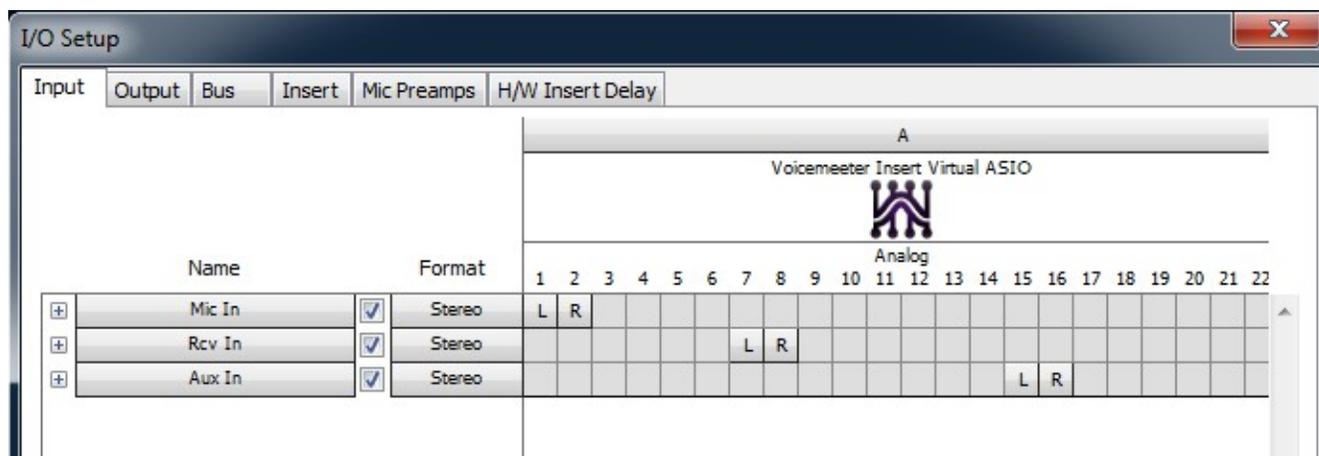


Figure 7—Pro Tools Input I/O Configuration.

3. Set up the Output I/O as shown in Figure 8. Again these can be named however you like.

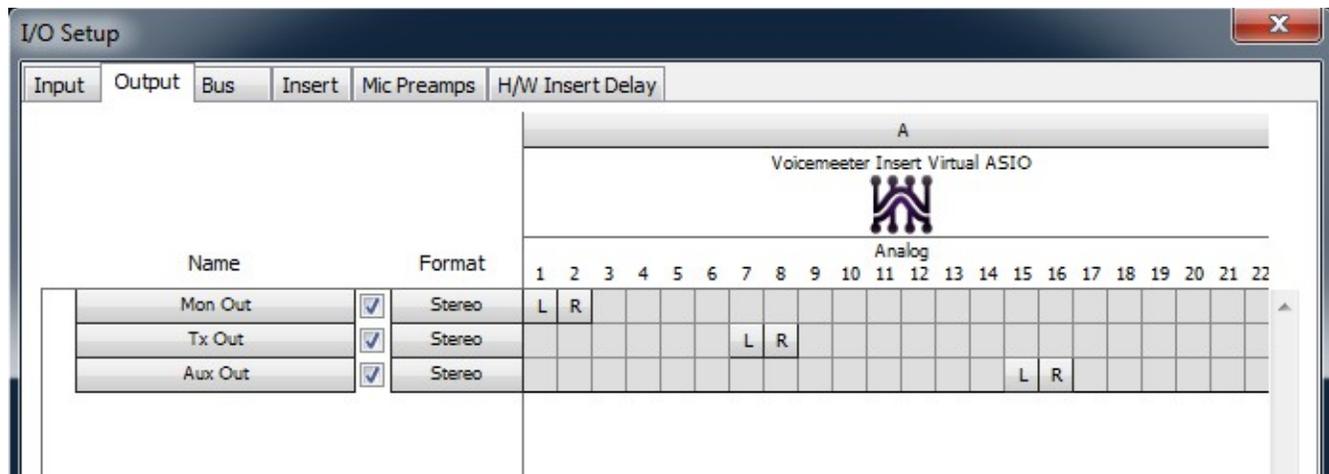


Figure 8—Pro Tools Output I/O Configuration.

Now you should be able to route, mix and set levels on audio completely within the DAW. Voicemeeter Banana can just run in the background (it can be minimized to the system tray).

Setting up how you want your DAW to work is well beyond the scope of this document. However, with the above configuration, here are a few ideas:

- Obviously receiver audio on Rcv In will need to be mixed to Mon Out (the audio interface speaker/headphone output). But you can also mix Mic In to Mon Out for sidetone and monitoring purposes.
- Similarly, Mic In will need to be mixed to Tx Out for transmitting, but again you can also mix Tx Out to Mon Out for sidetone and monitoring purposes.
- Rcv In can be mixed to Aux Out to feed Fldigi, WSJT-X or other digital mode program.
- Similarly Aux In will need to be mixed to Tx Out for digital mode program transmissions. You might want to mute the Mic In to Tx Out mix when doing that!

Figure 9 illustrates what my mixer setup currently looks like as an example. The I/O and bus assignments are shown above each channel strip. Not shown is the audio processing that occurs between Bus 1 and Bus 11.

Obviously the I/O setup provides tremendous flexibility and the sky is the limit in terms of the available configurations. For example, in addition to the above suggestions, I have also implemented a separate mix for Aux In called "Remote Mic" that allows me to send Aux In through the entire audio processing chain. This way it can be used with Skype or the like for remote operations. Or I can mute Remote Mic and instead use "Aux TX" for digital mode software.



Figure 9—Example Pro Tools Mixer, I/O and Bus Assignment Configuration.

Finally, note that Tx Out is handled as a mono bus using the left channel. You can run this as a stereo pair, however levels will be off by 3dB. I prefer to set it up as a mono send, that way everything is "perfect", i.e. 0dB on the DAW metering will correspond to 0dB in openHPSDR.

Configuring Audiomulch

The following is what I did in Audiomulch.

1. Set up the Audio Driver to be ASIO and the Device to be Voicemeeter Insert Virtual ASIO as shown in Figure 10. Audiomulch will set the buffer size automatically to match your sound hardware.

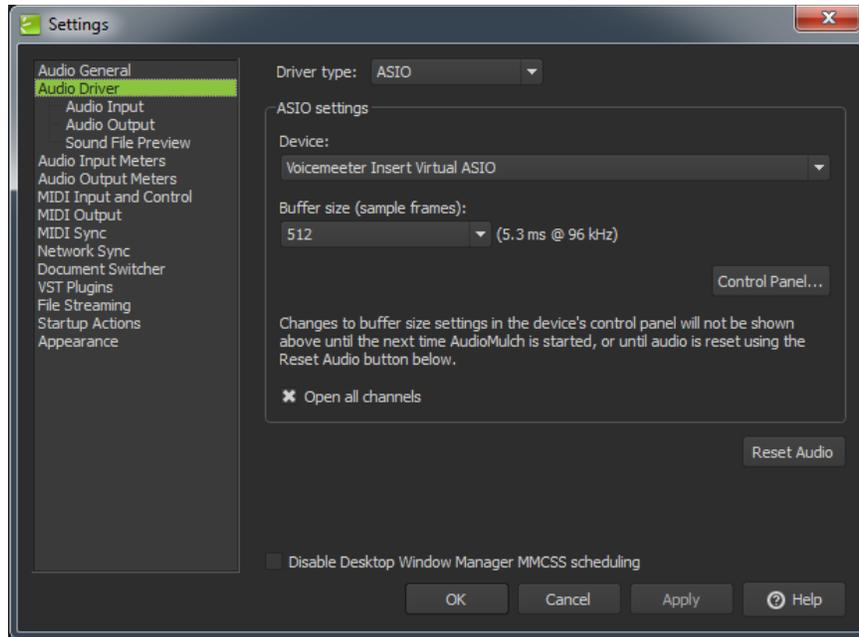


Figure 10—Audiomulch Driver Type and Device Configuration.

2. Turn on input metering as shown in Figure 11. SoundIn corresponds to Voicemeeter insert channels 1 and 2, AuxIn3 to Voicemeeter insert channels 7 and 8, and AuxIn7 to Voicemeeter Insert channels 15 and 16.

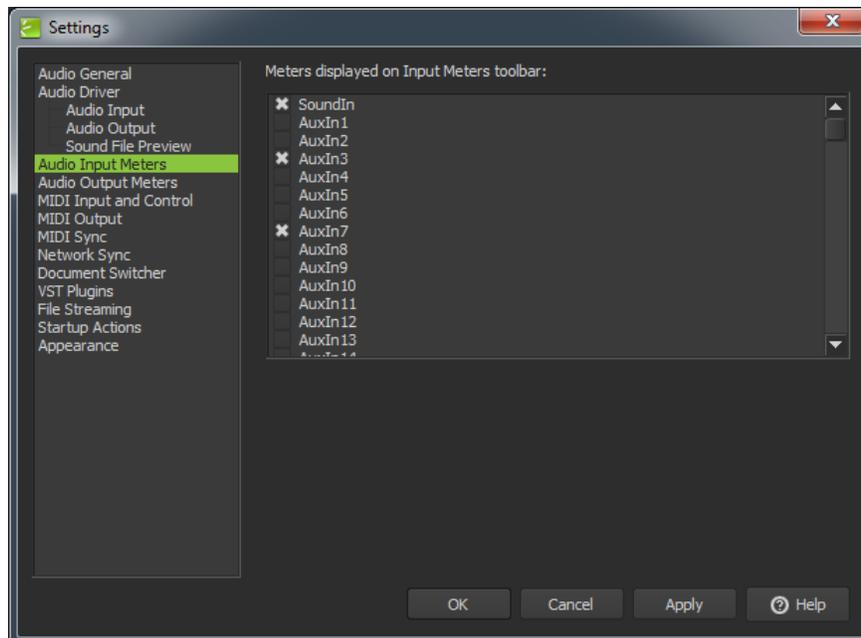


Figure 11—Audiomulch Input Metering Configuration.

3. Turn on output metering as shown in Figure 12. SoundOut corresponds to Voicemeeter insert channels 1 and 2, AuxOut3 to Voicemeeter insert channels 7 and 8, and AuxOut7 to Voicemeeter Insert channels 15 and 16.

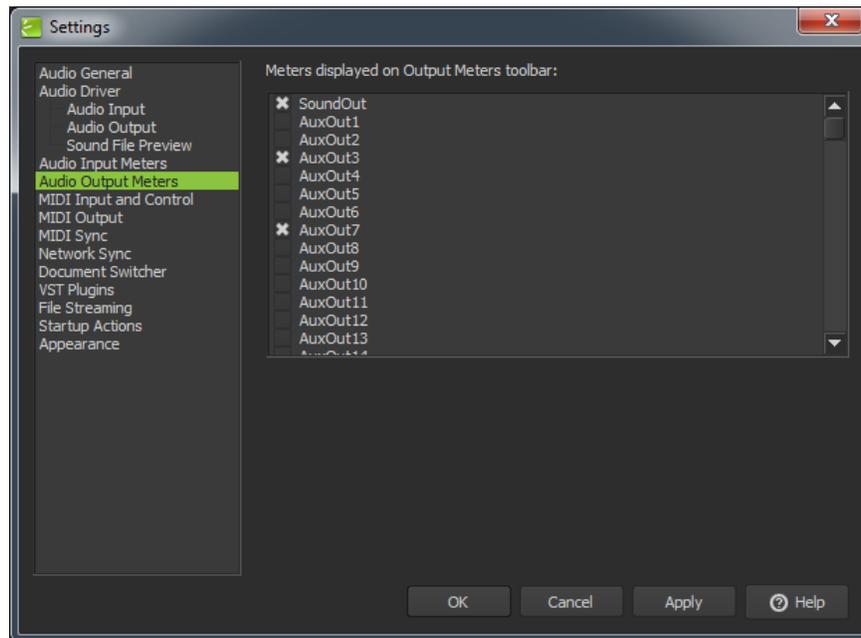


Figure 12—Audiomulch Output Metering Configuration.

4. Set up the Audiomulch patch as shown in Figure 13. SoundIn from Voicemeeter Insert channels 1 and 2 is the microphone audio from the sound interface hardware. AuxIn3 carries the receiver audio from openHPSDR on the Voicemeeter VAIO input via Voicemeeter Insert channels 7 and 8. Similarly AuxIn7 is from any digital mode or similar software on the Voicemeeter AUX Virtual Input via Voicemeeter Insert channels 15 and 16.



Figure 13—Audiomulch Patch Configuration.

5. Of course this is all much easier to understand if the I/O and mixer virtual devices are all renamed as shown in Figure 14. Indeed, this makes it almost self-explanatory. The Mon_Mix mixer adjusts the receiver audio and the microphone monitor (sidetone) audio into the audio interface speaker channel. The TX_Mix adjusts both the microphone audio and audio from any digital mode software out to the openHPSDR transmit audio input. And the Digi_RX_Lvl mixer adjust the receiver audio level into the digital mode software or anything else attached to the Voicemeeter AUX virtual channel.

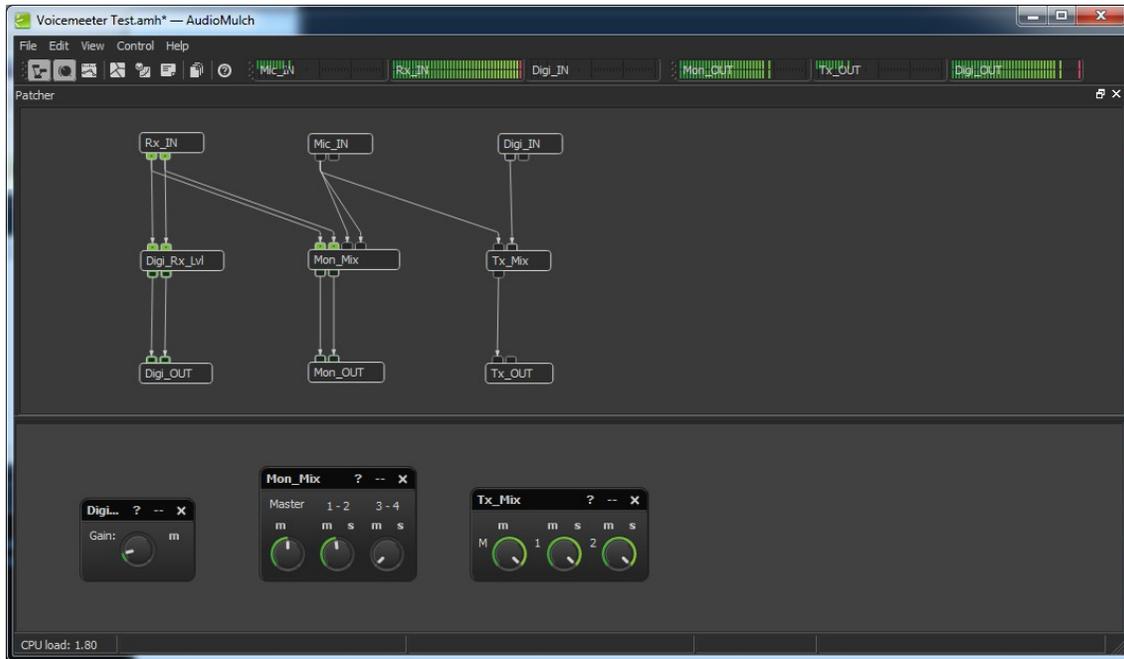


Figure 14—Audiomulch Patch Configuration With Renaming.

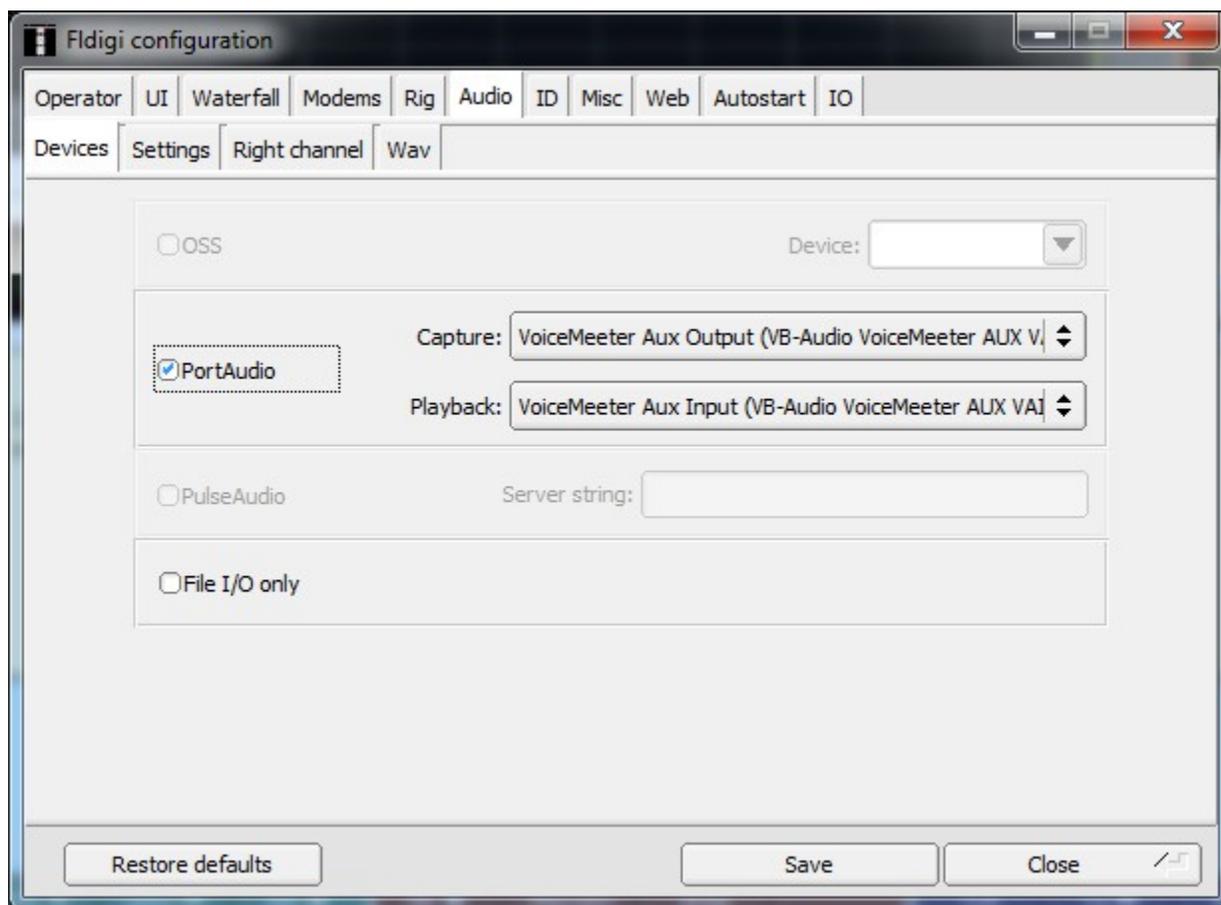
6. Finally, in Figure 15 is an example of how any type of audio processing plug-in can be placed into any of the various audio paths. In this case Ozone 5 is processing the microphone audio before it is mixed out to openHPSDR.



Figure 15—Audiomulch Patch Configuration With Renaming.

Configuring Digital Mode or Other Software

If you want to use Fldigi, WSJT-X, Skype or the like with this setup it is actually pretty easy. Voicemeeter Banana will automatically entertain either ASIO or Windows driver connections on any input and will perform any translations necessary. So although Fldigi, for example, only supports Windows drivers, it can easily be attached to the Voicemeeter AUX virtual input that was already set up in the instructions above.



Just remember to set up appropriate mix paths and levels in your DAW.

Obviously, since Fldigi and similar programs use Windows sound drivers, the latency will not be as good as it is with the microphone and speaker audio. However this is not an issue for any non-ARQ digital mode. Where ARQ modes are concerned I have successfully used this configuration (barely) with RMS Express by reducing the transmit and receive DSP filters to 256 (in openHPSDR Setup > DSP > Options).