Guide to using M-Audio Audiophile USB with ALSA and Jack

v1.5

Thibault Le Meur <Thibault.LeMeur@supelec.fr>

This document is a guide to using the M-Audio Audiophile USB (tm) device with ALSA and JACK.

History

- v1.4 - Thibault Le Meur (2007-07-11)
  - Added Low Endianness nature of 16bits-modes found by Hakan Lennestal <Hakan.Lennestal@brfsodrahamn.se>
  - Modifying document structure
- v1.5 - Thibault Le Meur (2007-07-12) - Added AC3/DTS passthru info

Audiophile USB Specs and correct usage

This part is a reminder of important facts about the functions and limitations of the device.

The device has 4 audio interfaces, and 2 MIDI ports:

- Analog Stereo Input (Ai)
  - This port supports 2 pairs of line-level audio inputs (1/4" TS and RCA)
  - When the 1/4" TS (jack) connectors are connected, the RCA connectors are disabled
- Analog Stereo Output (Ao)
- Digital Stereo Input (Di)
- Digital Stereo Output (Do)
- Midi In (Mi)
- Midi Out (Mo)

The internal DAC/ADC has the following characteristics:

- sample depth of 16 or 24 bits
- sample rate from 8kHz to 96kHz
- Two interfaces can't use different sample depths at the same time.

Moreover, the Audiophile USB documentation gives the following Warning:

> Please exit any audio application running before switching between bit depths

Due to the USB 1.1 bandwidth limitation, a limited number of interfaces can be activated at the same time depending on the audio mode selected:

- 16-bit/48kHz ==> 4 channels in + 4 channels out
  - Ai+Ao+Di+Do
- 24-bit/48kHz ==> 4 channels in + 2 channels out, or 2 channels in + 4 channels out
  - Ai+Ao+Do or Ai+Di+Ao or Ai+Di+Do or Di+Ao+Do
- 24-bit/96kHz ==> 2 channels in _or_ 2 channels out (half duplex only)
  - Ai or Ao or Di or Do

Important facts about the Digital interface:

- The Do port additionally supports surround-encoded AC-3 and DTS passthrough, though I haven't tested it under Linux
  - Note that in this setup only the Do interface can be enabled
- Apart from recording an audio digital stream, enabling the Di port is a way to synchronize the device to an external sample clock
  - As a consequence, the Di port must be enable only if an active Digital source is connected
  - Enabling Di when no digital source is connected can result in a synchronization error (for instance sound played at an odd sample rate)

Audiophile USB MIDI support in ALSA

The Audiophile USB MIDI ports will be automatically supported once the following modules have been loaded:

- snd-usb-audio

- snd-seq-midi

No additional setting is required.

Audiophile USB Audio support in ALSA

Audio functions of the Audiophile USB device are handled by the snd-usb-audio module. This module can work in a default mode (without any device-specific parameter), or in an "advanced" mode with the device-specific parameter called `device_setup`.

Default Alsa driver mode

The default behavior of the snd-usb-audio driver is to list the device capabilities at startup and activate the required mode when required by the applications: for instance if the user is recording in a 24bit-depth-mode and immediately after wants to switch to a 16bit-depth mode, the snd-usb-audio module will reconfigure the device on the fly.

This approach has the advantage to let the driver automatically switch from sample rates/depths automatically according to the user's needs. However, those who are using the device under windows know that this is not how the device is meant to work: under windows applications must be closed before using the m-audio control panel to switch the device working mode. Thus as we'll see in next section, this Default Alsa driver mode can lead to device misconfigurations.

Let's get back to the Default Alsa driver mode for now. In this case the Audiophile interfaces are mapped to alsa pcm devices in the following way (I suppose the device's index is 1):

- hw:1,0 is Ao in playback and Di in capture
- hw:1,1 is Do in playback and Ai in capture
- hw:1,2 is Do in AC3/DTS passthrough mode

In this mode, the device uses Big Endian byte-encoding so that supported audio format are S16_BE for 16-bit depth modes and S24_3BE for 24-bits depth mode.

One exception is the hw:1,2 port which was reported to be Little Endian compliant (supposedly supporting S16_LE) but processes in fact only S16_BE streams. This has been fixed in kernel 2.6.23 and above and now the hw:1,2 interface is reported to be big endian in this default driver mode.

Examples:

- playing a S24_3BE encoded raw file to the Ao port:

```
% aplay -D hw:1,0 -c2 -t raw -r48000 -fS24_3BE test.raw
```

- recording a S24_3BE encoded raw file from the Ai port:

```
% arecord -D hw:1,1 -c2  -t raw -r48000 -fS24_3BE test.raw
```

- playing a S16_BE encoded raw file to the Do port:

```
% aplay -D hw:1,1 -c2 -t raw -r48000 -fS16_BE test.raw
```

- playing an ac3 sample file to the Do port:

```
% aplay -D hw:1,2 --channels=6 ac3_S16_BE_encoded_file.raw
```

If you're happy with the default Alsa driver mode and don't experience any issue with this mode, then you can skip the following chapter.

Advanced module setup

Due to the hardware constraints described above, the device initialization made by the Alsa driver in default mode may result in a corrupted state of the device. For instance, a particularly annoying issue is that the sound captured from the Ai interface sounds distorted (as if boosted with an excessive high volume gain).

For people having this problem, the snd-usb-audio module has a new module parameter called `device_setup` (this parameter was introduced in kernel release 2.6.17)

Initializing the working mode of the Audiophile USB

As far as the Audiophile USB device is concerned, this value let the user specify:

- the sample depth
- the sample rate
- whether the Di port is used or not

When initialized with `device_setup=0x00`, the snd-usb-audio module has the same behaviour as when the parameter is omitted (see paragraph "Default Alsa driver mode" above)

Others modes are described in the following subsections.

16-bit modes

The two supported modes are:

- `device_setup=0x01`
  - 16bits 48kHz mode with Di disabled
  - Ai,Ao,Do can be used at the same time
  - hw:1,0 is not available in capture mode
  - hw:1,2 is not available
- `device_setup=0x11`
  - 16bits 48kHz mode with Di enabled
  - Ai,Ao,Di,Do can be used at the same time
  - hw:1,0 is available in capture mode
  - hw:1,2 is not available

In this modes the device operates only at 16bits-modes. Before kernel 2.6.23, the devices where reported to be Big-Endian when in fact they were Little-Endian so that playing a file was a matter of using:

```
% aplay -D hw:1,1 -c2 -t raw -r48000 -fS16_BE test_S16_LE.raw
```

where "test_S16_LE.raw" was in fact a little-endian sample file.

Thanks to Hakan Lennestal (who discovered the Little-Endiannes of the device in these modes) a fix has been committed (expected in kernel 2.6.23) and Alsa now reports Little-Endian interfaces. Thus playing a file now is as simple as using:

```
% aplay -D hw:1,1 -c2 -t raw -r48000 -fS16_LE test_S16_LE.raw
```

24-bit modes

The three supported modes are:

- `device_setup=0x09`
  - 24bits 48kHz mode with Di disabled
  - Ai,Ao,Do can be used at the same time
  - hw:1,0 is not available in capture mode
  - hw:1,2 is not available
- `device_setup=0x19`
  - 24bits 48kHz mode with Di enabled
  - 3 ports from {Ai,Ao,Di,Do} can be used at the same time
  - hw:1,0 is available in capture mode and an active digital source must be connected to Di
  - hw:1,2 is not available
- `device_setup=0x0D` or `0x10`
  - 24bits 96kHz mode
  - Di is enabled by default for this mode but does not need to be connected to an active source
  - Only 1 port from {Ai,Ao,Di,Do} can be used at the same time
  - hw:1,0 is available in captured mode
  - hw:1,2 is not available

In these modes the device is only Big-Endian compliant (see "Default Alsa driver mode" above for an aplay command example)

AC3 w/ DTS passthru mode

Thanks to Hakan Lennestal, I now have a report saying that this mode works.

- `device_setup=0x03`
  - 16bits 48kHz mode with only the Do port enabled
  - AC3 with DTS passthru
  - Caution with this setup the Do port is mapped to the pcm device hw:1,0

The command line used to playback the AC3/DTS encoded .wav-files in this mode:

```
% aplay -D hw:1,0 --channels=6 ac3_S16_LE_encoded_file.raw
```

How to use the `device_setup` parameter

The parameter can be given:

- By manually probing the device (as root)::

  ```
  # modprobe -r snd-usb-audio
  ```

```
# modprobe snd-usb-audio index=1 device_setup=0x09
```

- Or while configuring the modules options in your modules configuration file (typically a .conf file in /etc/modprobe.d/ directory::

```
alias snd-card-1 snd-usb-audio
options snd-usb-audio index=1 device_setup=0x09
```

CAUTION when initializing the device

- Correct initialization on the device requires that device_setup is given to the module BEFORE the device is turned on. So, if you use the "manual probing" method described above, take care to power-on the device AFTER this initialization.
- Failing to respect this will lead to a misconfiguration of the device. In this case turn off the device, unprobe the snd-usb-audio module, then probe it again with correct device_setup parameter and then (and only then) turn on the device again.
- If you've correctly initialized the device in a valid mode and then want to switch to another mode (possibly with another sample-depth), please use also the following procedure:
  - first turn off the device
  - de-register the snd-usb-audio module (modprobe -r)
  - change the device_setup parameter by changing the device_setup option in `/etc/modprobe.d/*.conf`
  - turn on the device
- A workaround for this last issue has been applied to kernel 2.6.23, but it may not be enough to ensure the 'stability' of the device initialization.

Technical details for hackers

This section is for hackers, wanting to understand details about the device internals and how Alsa supports it.

Audiophile USB's `device_setup` structure

If you want to understand the device_setup magic numbers for the Audiophile USB, you need some very basic understanding of binary computation. However, this is not required to use the parameter and you may skip this section.

The device_setup is one byte long and its structure is the following:

```
+---+---+---+---+---+---+---+---+
| b7| b6| b5| b4| b3| b2| b1| b0|
+---+---+---+---+---+---+---+---+
| 0 | 0 | 0 | Di|24B|96K|DTS|SET|
+---+---+---+---+---+---+---+---+
```

Where:

- b0 is the `SET` bit
  - it MUST be set if device_setup is initialized
- b1 is the `DTS` bit
  - it is set only for Digital output with DTS/AC3
  - this setup is not tested
- b2 is the Rate selection flag
  - When set to `1` the rate range is 48.1-96kHz
  - Otherwise the sample rate range is 8-48kHz
- b3 is the bit depth selection flag
  - When set to `1` samples are 24bits long
  - Otherwise they are 16bits long
  - Note that b2 implies b3 as the 96kHz mode is only supported for 24 bits samples
- b4 is the Digital input flag
  - When set to `1` the device assumes that an active digital source is connected
  - You shouldn't enable Di if no source is seen on the port (this leads to synchronization issues)
  - b4 is implied by b2 (since only one port is enabled at a time no synch error can occur)
- b5 to b7 are reserved for future uses, and must be set to `0`
  - might become Ao, Do, Ai, for b7, b6, b4 respectively

Caution:

- there is no check on the value you will give to device_setup
  - for instance choosing 0x05 (16bits 96kHz) will fail back to 0x09 since b2 implies b3. But _there_will_be_no_warning_ in /var/log/messages
- Hardware constraints due to the USB bus limitation aren't checked
  - choosing b2 will prepare all interfaces for 24bits/96kHz but you'll only be able to use one at the same time

USB implementation details for this device

You may safely skip this section if you're not interested in driver hacking.

This section describes some internal aspects of the device and summarizes the data I got by usb-snooping the windows and Linux drivers.

The M-Audio Audiophile USB has 7 USB Interfaces: a "USB interface":

- USB Interface nb.0
- USB Interface nb.1
  - Audio Control function
- USB Interface nb.2
  - Analog Output
- USB Interface nb.3
  - Digital Output
- USB Interface nb.4
  - Analog Input
- USB Interface nb.5
  - Digital Input
- USB Interface nb.6
  - MIDI interface compliant with the MIDIMAN quirk

Each interface has 5 altsettings (AltSet 1,2,3,4,5) except:

- Interface 3 (Digital Out) has an extra Alset nb.6
- Interface 5 (Digital In) does not have Alset nb.3 and 5

Here is a short description of the AltSettings capabilities:

- AltSettings 1 corresponds to
  - 24-bit depth, 48.1-96kHz sample mode
  - Adaptive playback (Ao and Do), Synch capture (Ai), or Asynch capture (Di)
- AltSettings 2 corresponds to
  - 24-bit depth, 8-48kHz sample mode
  - Asynch capture and playback (Ao,Ai,Do,Di)
- AltSettings 3 corresponds to
  - 24-bit depth, 8-48kHz sample mode
  - Synch capture (Ai) and Adaptive playback (Ao,Do)
- AltSettings 4 corresponds to
  - 16-bit depth, 8-48kHz sample mode
  - Asynch capture and playback (Ao,Ai,Do,Di)
- AltSettings 5 corresponds to
  - 16-bit depth, 8-48kHz sample mode
  - Synch capture (Ai) and Adaptive playback (Ao,Do)
- AltSettings 6 corresponds to
  - 16-bit depth, 8-48kHz sample mode
  - Synch playback (Do), audio format type III IEC1937_AC-3

In order to ensure a correct initialization of the device, the driver *must know* how the device will be used:

- if DTS is chosen, only Interface 2 with AltSet nb.6 must be registered
- if 96KHz only AltSets nb.1 of each interface must be selected
- if samples are using 24bits/48KHz then AltSet 2 must me used if Digital input is connected, and only AltSet nb.3 if Digital input is not connected
- if samples are using 16bits/48KHz then AltSet 4 must me used if Digital input is connected, and only AltSet nb.5 if Digital input is not connected

When device_setup is given as a parameter to the snd-usb-audio module, the parse_audio_endpoints function uses a quirk called `audiophile_skip_setting_quirk` in order to prevent AltSettings not corresponding to device_setup from being registered in the driver.

Audiophile USB and Jack support

This section deals with support of the Audiophile USB device in Jack.

There are 2 main potential issues when using Jackd with the device:

- support for Big-Endian devices in 24-bit modes
- support for 4-in / 4-out channels

Direct support in Jackd

Jack supports big endian devices only in recent versions (thanks to Andreas Steinmetz for his first big-endian patch). I can't remember exactly when this support was released into jackd, let's just say that with jackd version 0.103.0 it's almost ok (just a small bug is affecting 16bits Big-Endian devices, but since you've read carefully the above paragraphs, you're now using kernel >= 2.6.23 and your 16bits devices are now Little Endians ;-)).

You can run jackd with the following command for playback with Ao and record with Ai:

```
% jackd -R -dalsa -Phw:1,0 -r48000 -p128 -n2 -D -Chw:1,1
```

Using Alsa plughw

If you don't have a recent Jackd installed, you can downgrade to using the Alsa `plug` converter.

For instance here is one way to run Jack with 2 playback channels on Ao and 2 capture channels from Ai:

```
% jackd -R -dalsa -dplughw:1 -r48000 -p256 -n2 -D -Cplughw:1,1
```

However you may see the following warning message:

> You appear to be using the ALSA software "plug" layer, probably a result of using the "default" ALSA device. This is less efficient than it could be. Consider using a hardware device instead rather than using the plug layer.

Getting 2 input and/or output interfaces in Jack

As you can see, starting the Jack server this way will only enable 1 stereo input (Di or Ai) and 1 stereo output (Ao or Do).

This is due to the following restrictions:

- Jack can only open one capture device and one playback device at a time
- The Audiophile USB is seen as 2 (or three) Alsa devices: hw:1,0, hw:1,1 (and optionally hw:1,2)

If you want to get Ai+Di and/or Ao+Do support with Jack, you would need to combine the Alsa devices into one logical "complex" device.

If you want to give it a try, I recommend reading the information from this page: http://www.sound-man.co.uk/linuxaudio/ice1712multi.html It is related to another device (ice1712) but can be adapted to suit the Audiophile USB.

Enabling multiple Audiophile USB interfaces for Jackd will certainly require:

- Making sure your Jackd version has the MMAP_COMPLEX patch (see the ice1712 page)
- (maybe) patching the alsa-lib/src/pcm/pcm_multi.c file (see the ice1712 page)
- define a multi device (combination of hw:1,0 and hw:1,1) in your .asoundrc file
- start jackd with this device

I had no success in testing this for now, if you have any success with this kind of setup, please drop me an email.