I looked at the code carefully; as well as the data sheet for ICE1712.
This data sheet gives all the codes needed for the audio card
alsa.cybermirror.org/manuals/icensemble/envy24.pdf

I think that there is a mistake in the Envy24 source code (monitoring version),
in file misc.cpp where the monitoring is set to ON.

```
    // turn on monitoring:
    // MT30-31 select digital mixer is input instead of DMA channel
10-11
    dev->ioWrite16(0x30, 0x0101, card->mtbase);
```

We should have this instead

```
    dev->ioWrite16(0x30, 0x81, card->mtbase);
```

This sends the mixer to the H/W OUT1 and H/W OUT2
- H/W OUT1 L is bound to Mixer L
- H/W OUT2 R is bound to Mixer R

See envy24.pdf p33; we want
 - bit 9:8 = 01 PSDOUT[0] Right source from digital mixer monitor Right output
 - bit 1:0 = 01 PSDOUT[0] Left source from digital mixer monitor Left output

PSDOUT[0] L = H/W out 1 (L)
PSDOUT[0] R = H/W out 2 (R)

The next instruction is not needed.

```
    // loopback psdin[0] to psdout[0]
    dev->ioWrite32(0x34, 0x00000010, card->mtbase);
```

We already have what we want:
mixer L -> H/W out 1 (L)
mixer R -> H/W out 2 (R)
PCM 3 -> H/W out 3
PCM 4 -> H/W out 4
PCM 5 -> H/W out 5
PCM 6 -> H/W out 6
PCM 7 -> H/W out 7
PCM 8 -> H/W out 8
Where PCM is the same thing as DMA (in the data sheet), which means the digital audio produced by software apps (player, DAW, etc).

PCM 1 and PCM 2 are usually where these software output the digital audio.
We need to add these PCM into the mixer, and to respect the stereo.
We have to map each stream to either L or R mixer output: setting up the level of L and muting R or vis versa:
- PCM 1 L 0db; R Muted
- PCM 2 L Muted; R 0db
- PCM 3 L 0db; R Muted
- PCM 4 L Muted; R 0db
- PCM 5 L 0db; R Muted
- PCM 6 L Muted; R 0db
- PCM 7 R 0db; R Muted
- PCM 8 LMuted; R 0db

This way each channel is not sent to both R and L (which is causing the mono sound in the current app)

This is done with the instructions p39
MT38
Default = 0x0707 which means L=-10.5dB and R=-10.5dB
For PCM 1, 3, 5, 7 we want L=0dB and R=Muted
0x7F00
For PCM 2, 4, 6, 8 we want L=Muted and R=0dB
0x007F

We need to do that to each playback stream (PCM)
MT3A (p40)

PCM1
```
	dev->ioWrite8(0x3A, 0x0, card->mtbase);
	dev->ioWrite16(0x38, 0x7F00, card->mtbase);
```
PCM2
```
	dev->ioWrite8(0x3A, 0x1, card->mtbase);
	dev->ioWrite16(0x38, 0x007F, card->mtbase);
```
PCM3
```
	dev->ioWrite8(0x3A, 0x2, card->mtbase);
	dev->ioWrite16(0x38, 0x7F00, card->mtbase);
```
PCM4
```
	dev->ioWrite8(0x3A, 0x3, card->mtbase);
	dev->ioWrite16(0x38, 0x007F, card->mtbase);
```
PCM5
```
	dev->ioWrite8(0x3A, 0x4, card->mtbase);
	dev->ioWrite16(0x38, 0x7F00, card->mtbase);
```
PCM6
```
	dev->ioWrite8(0x3A, 0x5, card->mtbase);
	dev->ioWrite16(0x38, 0x007F, card->mtbase);
```
PCM7
```
	dev->ioWrite8(0x3A, 0x6, card->mtbase);
```

```
      dev->ioWrite16(0x38, 0x7F00, card->mtbase);
```

PCM8

```
      dev->ioWrite8(0x3A, 0x7, card->mtbase);
      dev->ioWrite16(0x38, 0x007F, card->mtbase);
```

I do not know if there is a sequence to respect for each pair of instruction

Finally we must send the H/W In signals to the Mixer.
This is done with the record stream. But we prefer to balance each input to both
the L and R channels of the mixer.

H/W In 1

```
      dev->ioWrite8(0x3A, 0xA, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

H/W In 2

```
      dev->ioWrite8(0x3A, 0xB, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

H/W In 3

```
      dev->ioWrite8(0x3A, 0xC, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

H/W In 4

```
      dev->ioWrite8(0x3A, 0xD, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

H/W In 5

```
      dev->ioWrite8(0x3A, 0xE, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

H/W In 6

```
      dev->ioWrite8(0x3A, 0xF, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

H/W In 7

```
      dev->ioWrite8(0x3A, 0x10, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

H/W In 8

```
      dev->ioWrite8(0x3A, 0x11, card->mtbase);
      dev->ioWrite16(0x38, 0x0, card->mtbase);
```

By doing all this, we have the output 1/2 playing the mixer; we can hear each analog entry with zero latency mixed with each stream (PCM) played from software.

Now my problem: I have not Xcode skills, I tried to compile the current sources with no success.
Can someone help?