Uniaud: Internals and History

David Azarewicz
www.88watts.net
david@88watts.net

Warpstock Europe 2011

About Me

- I am primarily a hardware developer. I started my career designing hardware for IBM 360 mainframe clones.
- I then joined a small company building Cray compatible computers where I had to do a little of everything including microcode and embedded systems hardware and software.
- Since then I have worked on projects from auto industry process controllers, to home automation, to radiation effects testing for satellite electronics.
- I have been using OS/2 and developing software for OS/2 since version 1.
- Being an experienced hardware developer gives me an advantage in writing device drivers.

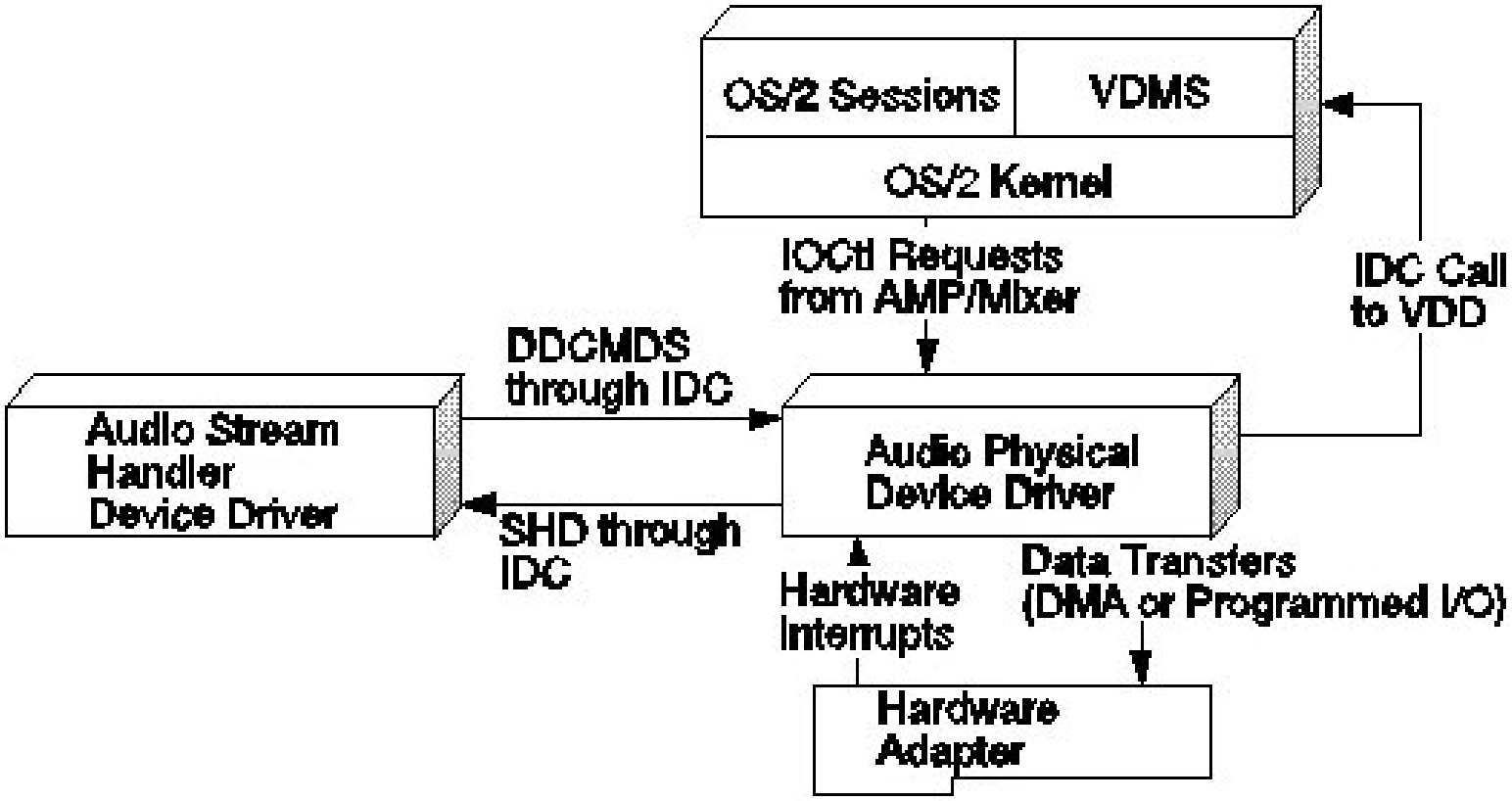
This Presentation

- What is inside Uniaud
- How Uniaud works
- What are Uniaud’s limitations
- How to work around problems
- Questions

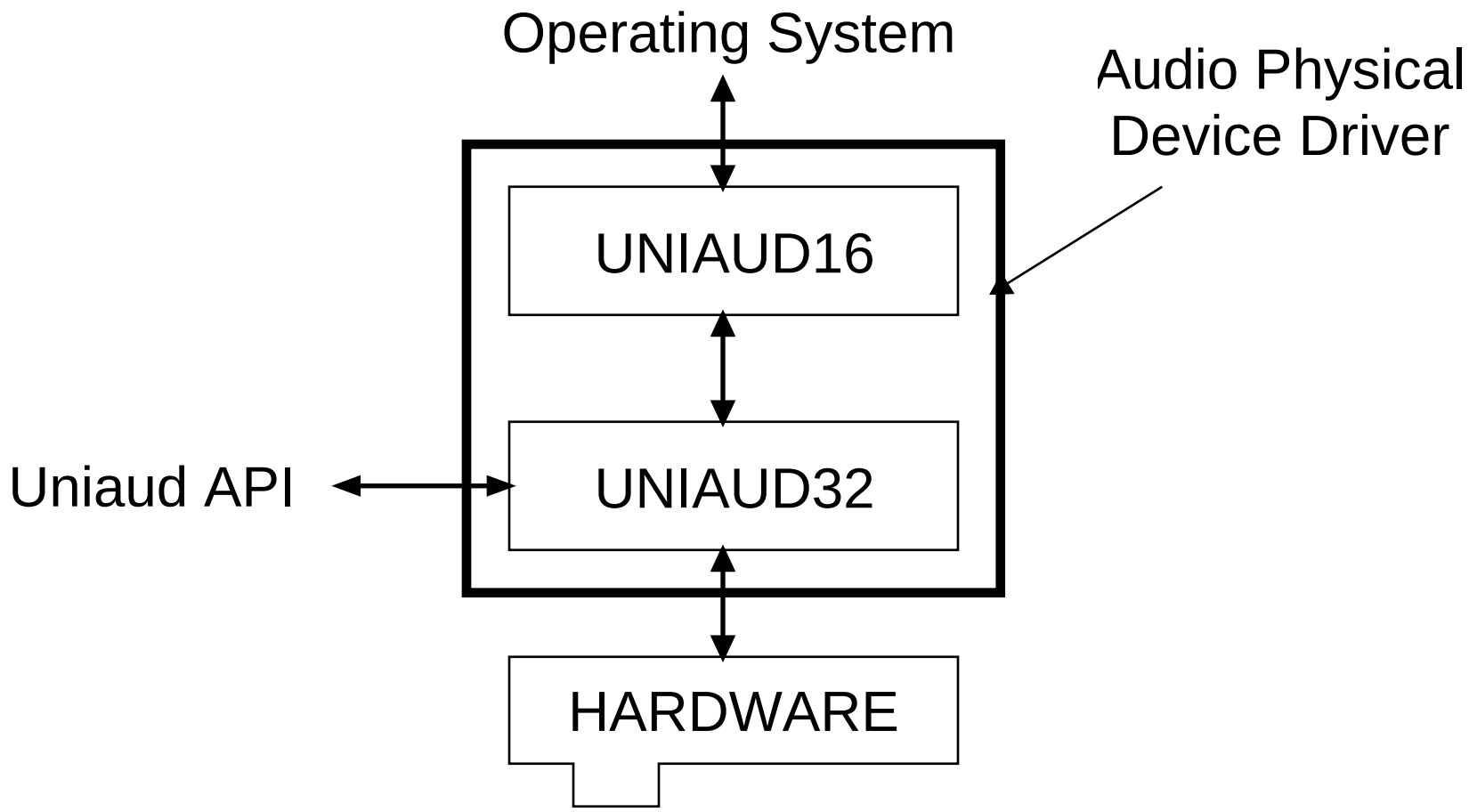
A Brief Look at Uniaud's History

- Uniaud has its origins with the SBLive driver.
- The SBLive driver was based on the Saint Tropez example from the DDK.
- There were problems with the Tropez sample driver in the DDK that.
- Now all drivers based on the Tropez sample driver in the DDK have the same problems.
- Most of these problems have now been found and fixed in Uniaud.

Where the Audio Physical Device Driver fits in (From IBM)



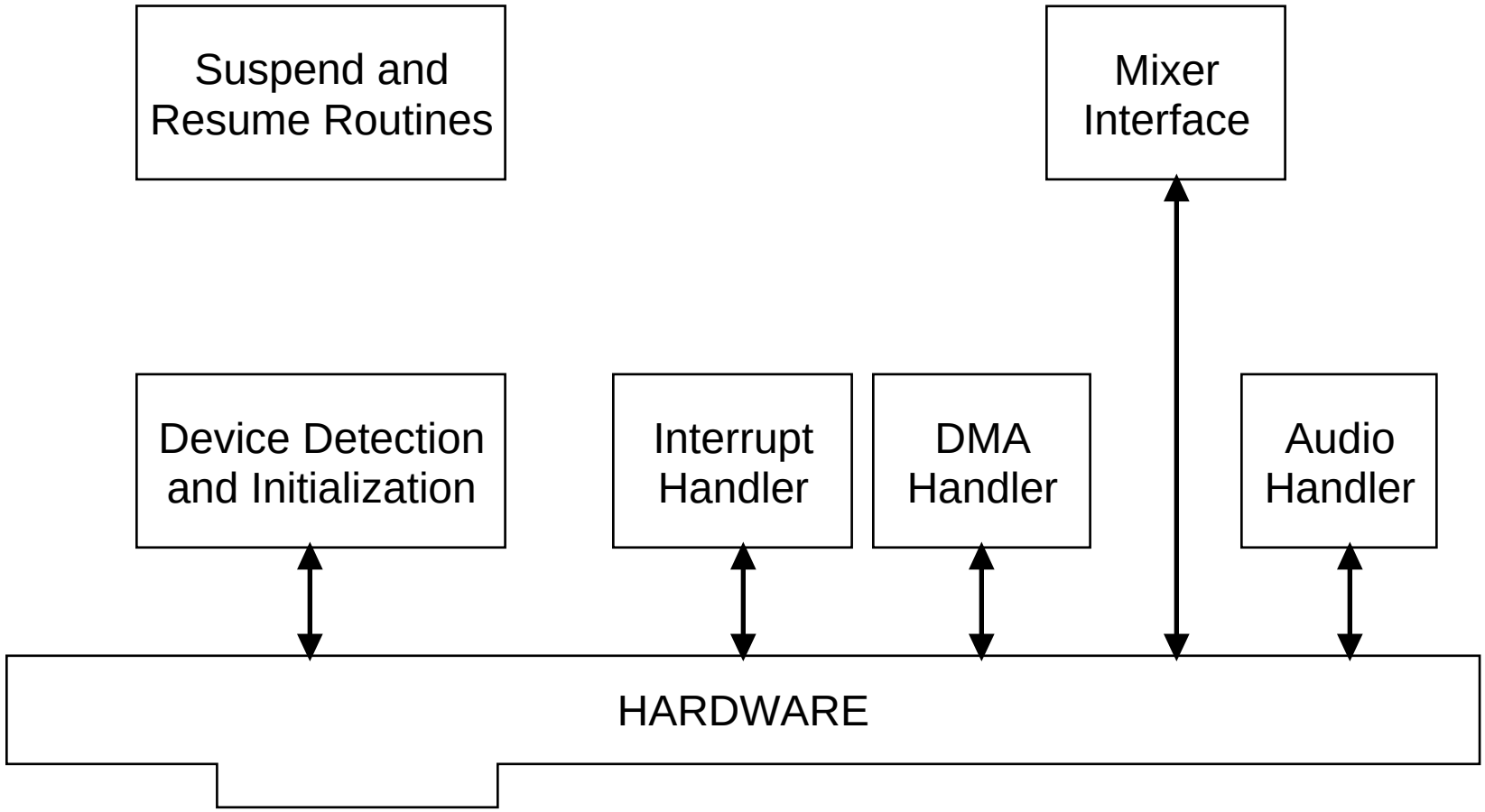
Uniaud Has 2 Separate Drivers



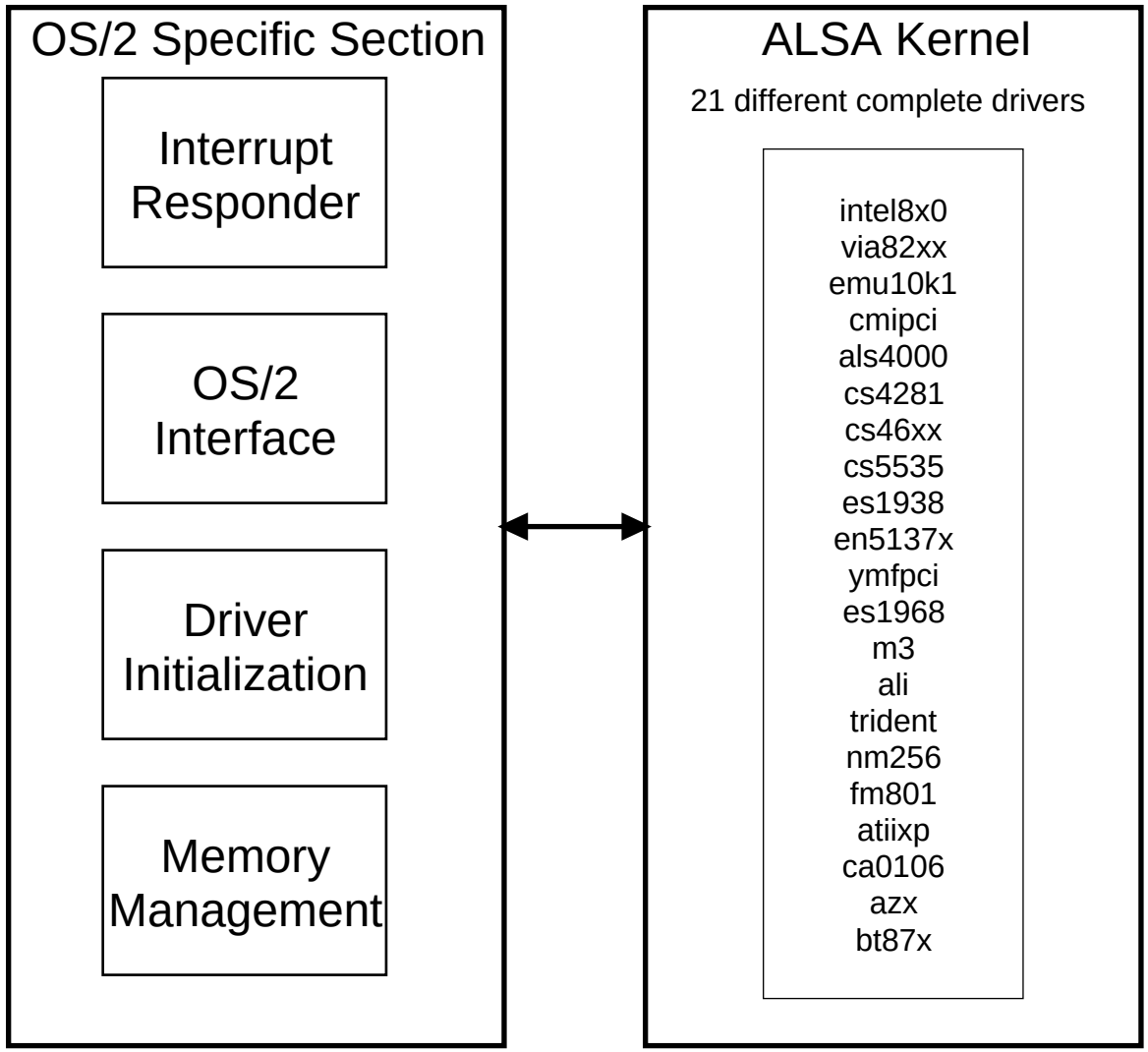
What Uniaud16 Does

- Interface between the operating system / applications and Uniaud32.
- Does not talk directly to hardware.
- Does not handle any interrupts.
- Has to appear to be a physical device driver to OS/2 even though it is not.
- Handles frequency conversions.
- Handles application buffer management.
- Has to maintain synchronization with Uniaud32.
- Once written correctly and working, this part of Uniaud should not need to change.

Uniaud 32: What is in a Basic Audio Physical Device Driver

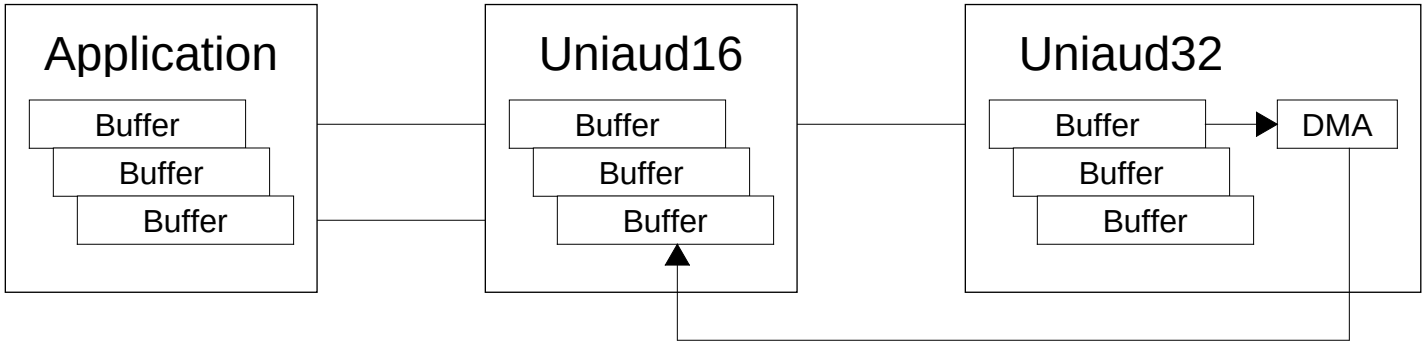


Inside Uniaud 32



My Changes - Uniaud16

- It was not multi-thread safe. There were hacks / patches in the code to work around these problems, but none of them actually addressed the reentrancy issues. I fixed the reentrancy issues and removed all the hacks.
- There were buffer management problems.



There is still a small A-V sync issue due to the double buffering done. However it is probably as good as it is going to get.

- There was a problem with the Pause/Resume logic which I fixed.

My Changes - Uniaud32

- There was a problem with the PCI bus scan where it wouldn't find some audio hardware. This was fixed.
- I made some changes to the way that the multiple drivers get initialized.
- I made some slight changes to the audio buffer management.
- Uniaud32 did not handle APIC interrupts correctly. This was fixed.
- I added some additional "quirks" for some special hardware cases.
- The Uniaud API library was converted to use the Open Watcom compiler. Some structure alignment issues that caused the Uniaud API to not work were corrected.

The Unimix Utility

- The Unimix utility was cleaned up and enhanced.
- Can be used to work around many mixer type of problems.
- Unimix talks directly to Uniaud32, so it doesn't go through Uniaud16 and does not have the limitations imposed by MMOS2.
- Unimix can be used to switch between different audio outputs.
- You have more flexible and detailed control of the different volume levels.
- If you need to control certain settings on startup, you can put the unimix command
  - In your STARTUP.CMD file
  - In your Startup folder
  - RUN= statement in CONFIG.SYS
    RUN=unimix -id5 -cnt0 -val50 -cnt1 -val50

The Unimix Command Line

unimix -id5 -cnt0 -val50 -cnt1 -val50

Usage:
 -dev - card to work
 -card - show card info
 -pcms - show PCM instances info
 -list - full (id,name,bounds,value) control(s) list
 -names - just control(s) name(s) and ID
 -id<num> - use control num (for list or set value). Begins from 1
 -val<num> - set value for control
 -cnt<num> - set value for count number <num> in control. Begins from 0
 -get - get value switch. uses with -id and/or -cnt
 -powerget - get power state
 -powerset<num> - set power state
 -save<file> - save dump of all values of all controls to file
 -load<file> - load dump from file to all values of all controls

Useful Uniaud Links

- The Wiki page at the Uniaud development home: **http://svn.netlabs.org/uniaud**
- The ftp download page for Uniaud at netlabs: **ftp://ftp.netlabs.org/pub/uniaud**

My settings in .mplayer/config:

```
quiet=yes
vo=kva:dive
ao=kai:dart
```