Embedded I/O Management

What to Expect?

- Basic I/O Management Support
- Advanced I/O Management
  - Video Subsystem
  - Audio Subsystem

I/O Management Overview

- Linux provides a uniform i/f to on-board I/O Devices
- Categorized as follows
  - Link oriented (Network Devices)
  - Block oriented (Storage Device)
  - All Other (Sequential) Devices
- Sequential Device or Character Device category is one of the largest, with majority of Devices falling under this
- So, based on the specialized functions, this have been further categorized
  - Basic I/O (GPIOs, and all that uses plain character drivers)
  - Custom I/O (tty, audio, video, ..)

Basic I/O Management

- Device Category
  - Domain-specific Electronics
  - Actuators, Sensors, ...
  - General Purpose I/O
  - A2D, D2A, ...
- Driver Type: Character

Bus I/O Management

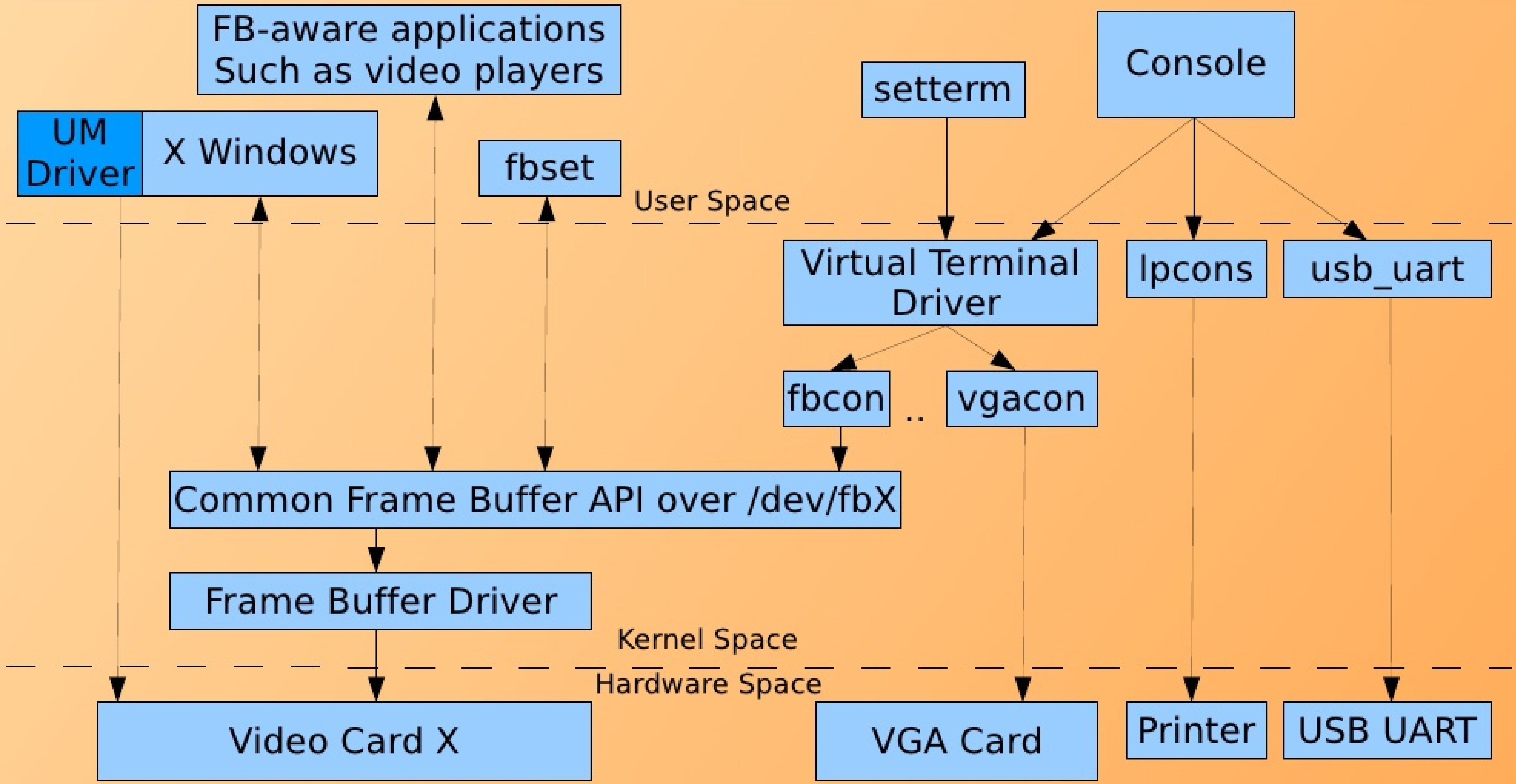
- Device Category: I^2C, SPI, ...
- Driver Category: Character with Platform
- Porting mostly involves
  - Respective bus controller code (driver) to be enabled in the kernel

Custom I/O Management

- Topics under Consideration
  - Video
  - Audio

Video Drivers

Video Subsystem



FB Programming Interface

- Header: <linux/fb.h>
- Data Structures
  - struct fb_info – Main data structure
  - struct fb_ops – Entry points
  - struct fb_var_screen_info – Resolution, ...
  - struct fb_fix_screen_info – FB start addr, ...
  - struct fb_cmap – RGB colour map
- APIs
  - int register_framebuffer(struct fb_info *fb_info);
  - int unregister_framebuffer(struct fb_info *fb_info);
  - struct fb_info *framebuffer_alloc(size_t size, struct device *dev);
  - void framebuffer_release(struct fb_info *info);
  - int fb_alloc_cmap(struct fb_cmap *cmap, int len, int transp);
  - void fb_dealloc_cmap(struct fb_cmap *cmap);
- Source: drivers/video/

struct fb_ops

- fb_open – Open
- fb_release – Close
- fb_check_var – Check video parameters
- fb_set_par – Set video controller registers
- fb_setcolreg – Create pseudo colour palette map
- fb_blank – Blank / Unblank display
- fb_fillrect – Fill rectangle with pixel lines
- fb_copyarea – Copy rectangular area between screens
- fb_imageblit – Draw an image to the display
- fb_rotate – Rotate the display
- fb_ioctl – Ioctl interface for device specific commands

Console Programming Interface

- Header: <linux/console.h>
- Data Structures
  - struct console – top-level console driver
  - struct consw – bottom-level console driver
- APIs
  - void register_console(struct console *);
  - int unregister_console(struct console *);
  - int register_con_driver(const struct consw *csw, int first, int last);
  - int unregister_con_driver(const struct consw *csw);

Porting a Video Driver

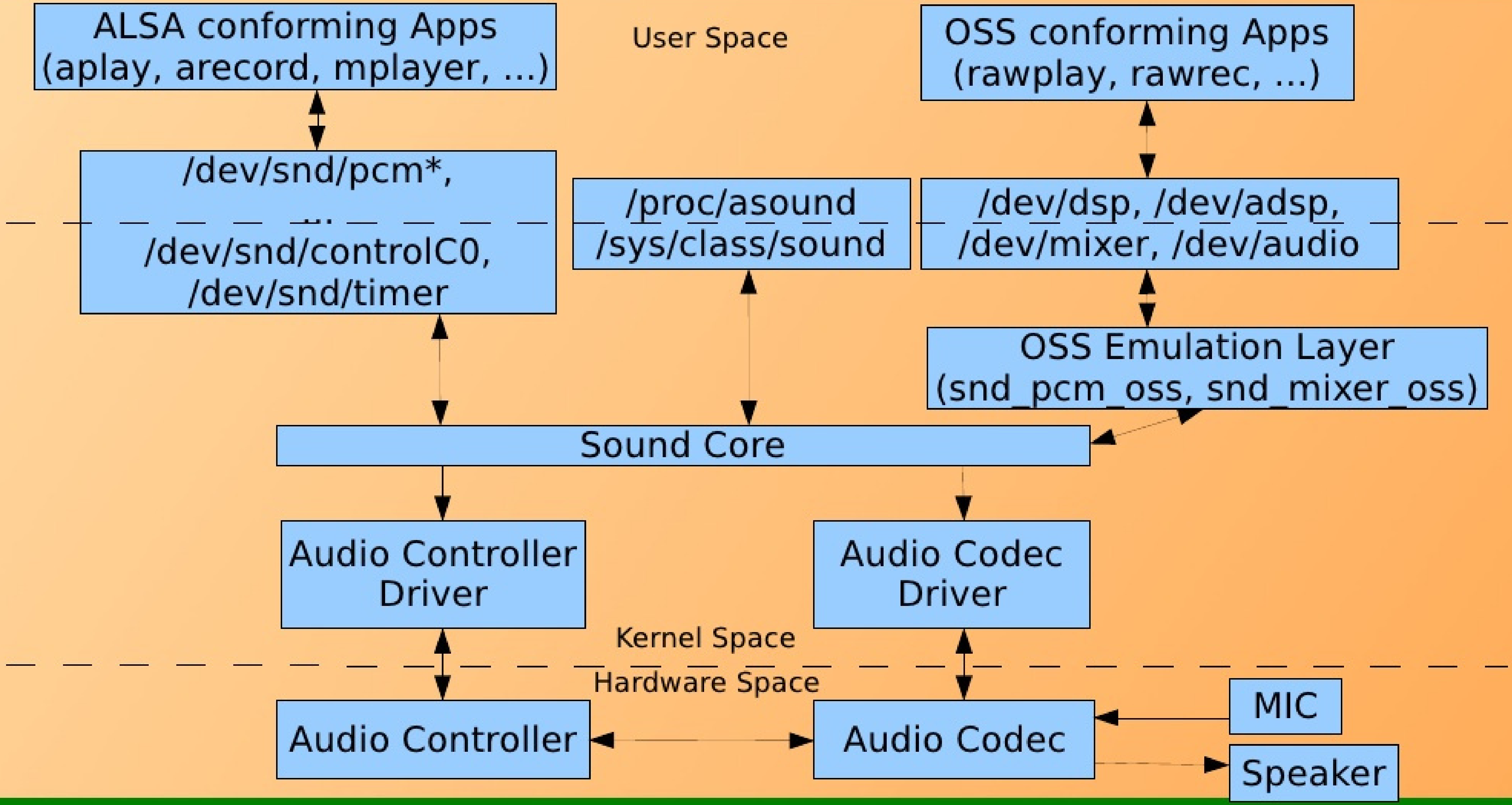
- Standard Video Chipset
  - Mostly involves changing pin assignments as per the Board Design
- New Video Chipset
  - Complete Driver as per the preceeding discussions, need to be implemented

Browse some Video Drivers

- For Frame Buffer drivers
  - Browse the drivers/video/ folder
- For Console drivers
  - Browse the drivers/video/console/ folder

Audio Drivers

Audio Subsystem


ALSA conforming Apps (aplay, arecord, mplayer, ...)
User Space
OSS conforming Apps (rawplay, rawrec, ...)
/dev/snd/pcm*, ... /dev/snd/controlC0, /dev/snd/timer
/proc/asound /sys/class/sound
/dev/dsp, /dev/adsp, /dev/mixer, /dev/audio
OSS Emulation Layer (snd_pcm_oss, snd_mixer_oss)
Sound Core
Audio Controller Driver
Audio Codec Driver
Kernel Space
Hardware Space
Audio Controller
Audio Codec
MIC
Speaker

ALSA Sound Card Interface

- Header: <linux/sound/core.h>
- Data Structure: struct snd_card
- APIs
  - int snd_card_create(int idx, const char *id, struct module *module, int extra_size, struct snd_card **card_ret);
  - int snd_card_free(struct snd_card *card);
  - int snd_card_register(struct snd_card *card);

ALSA PCM Interface

- Header: <linux/sound/pcm.h>
- Data Structure
  - struct snd_pcm
  - struct snd_pcm_ops
- APIs
  - int snd_pcm_lib_malloc_pages(struct snd_pcm_substream *substream, size_t size);
  - int snd_pcm_lib_free_pages(struct snd_pcm_substream *substream);
  - int snd_pcm_new(struct snd_card *card, const char *id, int device, int playback_count, int capture_count, struct snd_pcm **rpcm);
  - void snd_pcm_set_ops(struct snd_pcm * pcm, int direction, struct snd_pcm_ops *ops);

ALSA Sound Card Interface

- Header: <linux/sound/control.h>
- Data Structure: struct snd_kcontrol_new
- APIs
  - int snd_ctl_add(struct snd_card * card, struct snd_kcontrol * kcontrol);
  - int snd_ctl_remove(struct snd_card * card, struct snd_kcontrol * kcontrol);

Porting a Audio Driver

- Standard Audio Codec
  - Mostly involves changing pin assignments as per the Board Design
- New Audio Codec
  - Complete Driver as per the preceeding discussions, need to be implemented

Browse some Audio Drivers

- For ALSA drivers
  - Browse the sound/ folder
  - Say sound/arm/aaci.*

What all have we learnt?

- Basic I/O Management Support
- Advanced I/O Management
  - Video Subsystem
    - Frame Buffer Programming Interface
    - Console Programming Interface
    - Porting
  - Audio Subsystem
    - ALSA Programming Interface
    - Porting

Any Queries?