Android多核心嵌入式多媒體系統設計與實作

Linux Sound Driver Architecture

賴槿峰 (Chin-Feng Lai**)**

Assistant Professor, institute of CSIE, National Ilan University

Oct 6th 2012

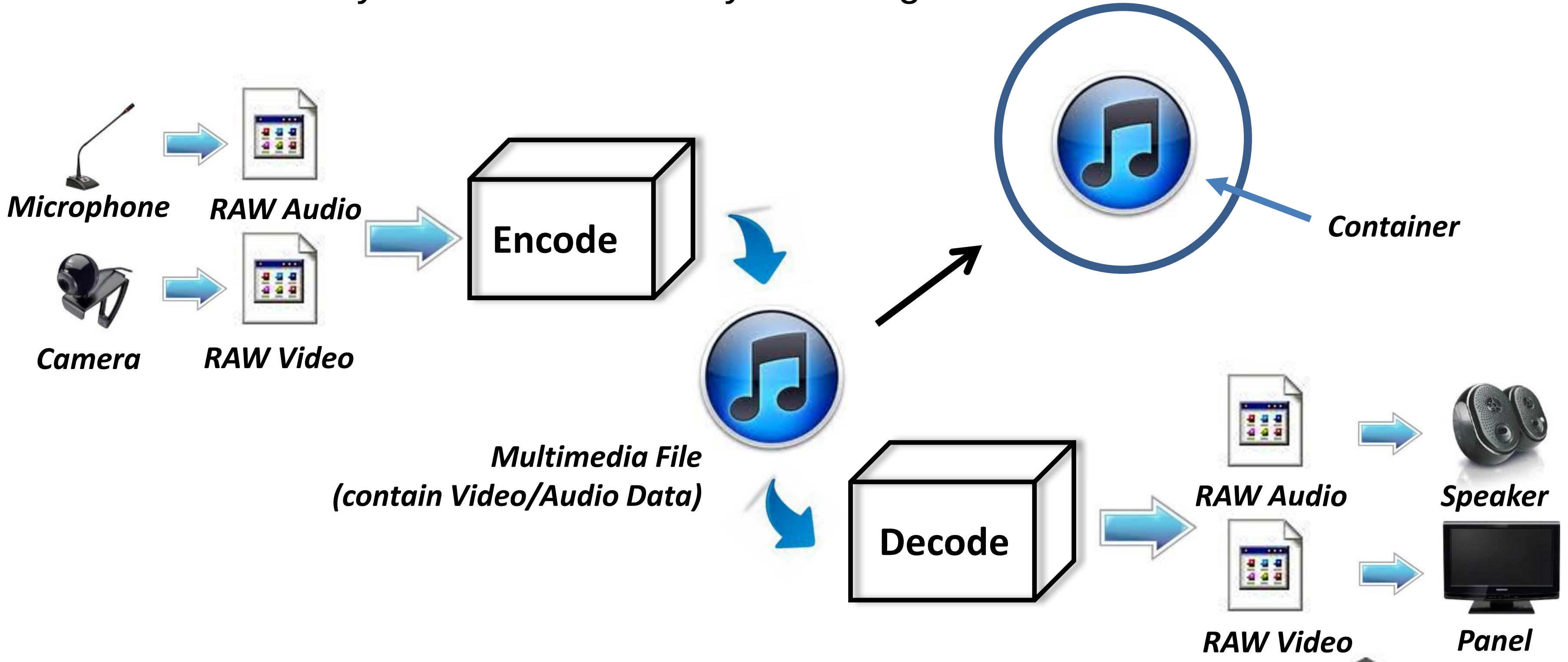
Outline

- What is Multimedia
  - Codec and Multimedia Format
  - Android Multimedia System Introduction
- Sound Subsystem in Kernel
- Alternate Sound Drivers
- Main Linux Audio Driver
  - Open Sound System, OSS
  - Advanced Linux Sound Architecture, ALSA
- Related Embedded Player
- Lab

- ***What is Multimedia***
  - ***Codec and Multimedia Format***
  - ***Android Multimedia System Introduction***
- ***Sound Subsystem in Kernel***
- ***Alternate Sound Drivers***
- ***Main Linux Audio Driver***
  - ***Open Sound System, OSS***
  - ***Advanced Linux Sound Architecture, ALSA***
- ***Related Embedded Player***
- ***Lab***

What is Multimedia

- Multimedia is media and content that uses a combination of different content forms
- Multimedia is the most important component in Android system
  - Multimedia framework is used to process video/audio input and output to satisfy certain functionality including



What is Multimedia

- Codec
  - A codec is a ***device*** or ***computer program*** capable of encoding and/or decoding a digital data stream or signal.
  - Raw multimedia data is huge, codec compress them to facilitate store and transfer
  - A codec encodes a data stream or signal for transmission, storage or encryption and decode it for playback or editing.
  - Codec[1] = ***co***mpressor ***dec***ompressor
  - Codec[2] = ***co***der ***dec***oder

What is Multimedia

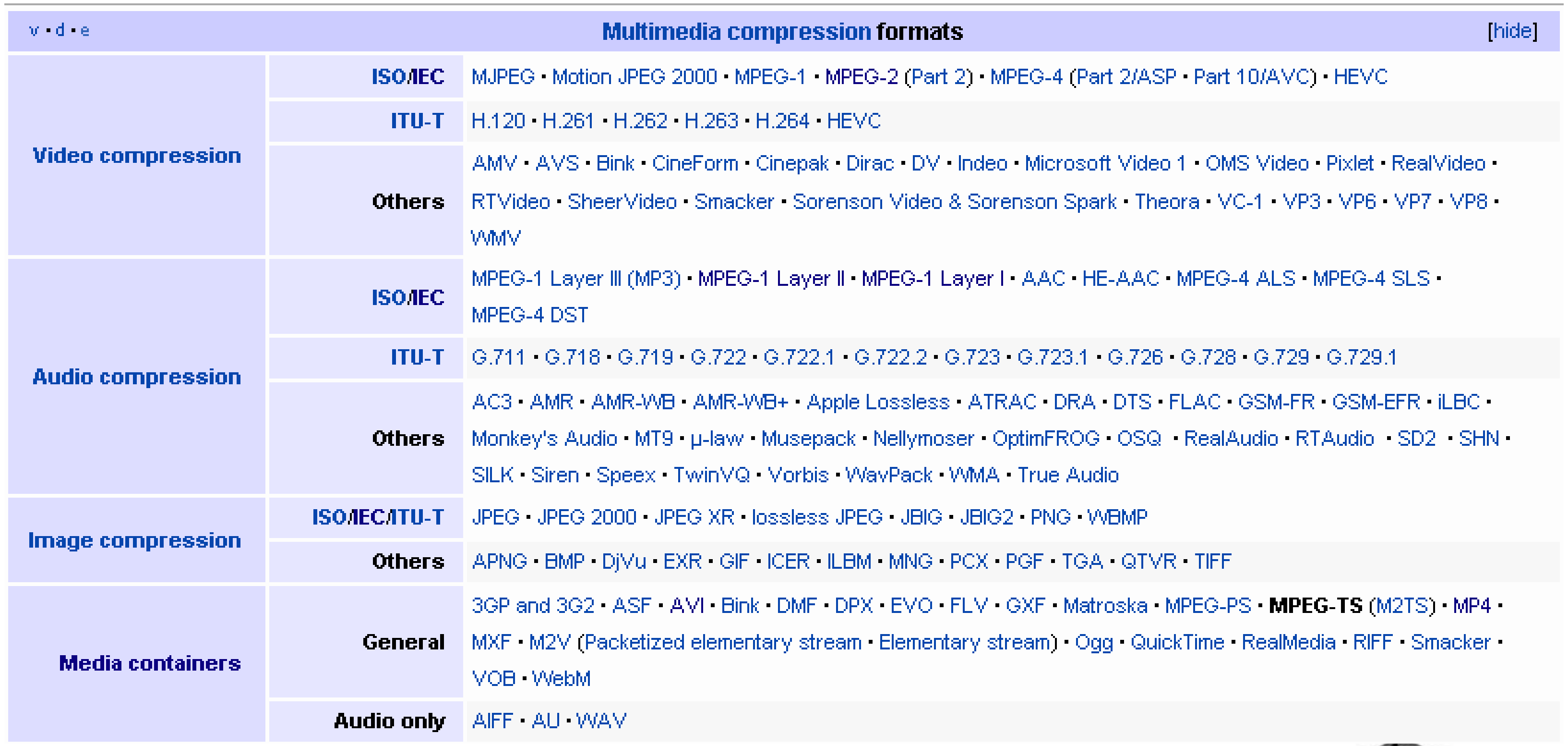
- File Format
  - Call ***Container*** or ***wrapper format***
  - The container file is used to identify and interleave different data types. Simpler container formats can contain different types of audio formats. Container does not describe how the data warped is encoded. It ***not be able to decode contained data***. You maybe were told to download right decoder.
  - Always contain ***coded video***, ***coded audio***, ***subtitles***, ***chapter-information***, maybe ***advertisement*** and ***synchronization information*** needed to playback various streams together

What is Multimedia – Container Format

- Some containers are exclusive to audio
  - AIFF (Mac OS)
  - WAV (Windows)
- Other containers are exclusive to still images
  - TIFF
- Other flexible containers can hold many types of audio and video
  - 3GP
  - ASF(container for Microsoft WMA and WMV)
  - AVI (the standard Microsoft Windows container, also based on RIFF)
  - MP4 (standard audio and video container for the MPEG-4)
  - Flash Video (FLV, F4V)

What is Multimedia

- multimedia File knowledge
  - Organization
    - ***ISO / IEC***
    - ***ITU-T***



v · d · e **Multimedia compression formats** [hide]

Video compression	**ISO/IEC**	MJPEG · Motion JPEG 2000 · MPEG-1 · MPEG-2 (Part 2) · MPEG-4 (Part 2/ASP · Part 10/AVC) · HEVC
	ITU-T	H.120 · H.261 · H.262 · H.263 · H.264 · HEVC
	Others	AMV · AVS · Bink · CineForm · Cinepak · Dirac · DV · Indeo · Microsoft Video 1 · OMS Video · Pixlet · RealVideo · RTVideo · SheerVideo · Smacker · Sorenson Video & Sorenson Spark · Theora · VC-1 · VP3 · VP6 · VP7 · VP8 · WMV
Audio compression	**ISO/IEC**	MPEG-1 Layer III (MP3) · MPEG-1 Layer II · MPEG-1 Layer I · AAC · HE-AAC · MPEG-4 ALS · MPEG-4 SLS · MPEG-4 DST
	ITU-T	G.711 · G.718 · G.719 · G.722 · G.722.1 · G.722.2 · G.723 · G.723.1 · G.726 · G.728 · G.729 · G.729.1
	Others	AC3 · AMR · AMR-WB · AMR-WB+ · Apple Lossless · ATRAC · DRA · DTS · FLAC · GSM-FR · GSM-EFR · iLBC · Monkey's Audio · MT9 · μ-law · Musepack · Nellymoser · OptimFROG · OSQ · RealAudio · RTAudio · SD2 · SHN · SILK · Siren · Speex · TwinVQ · Vorbis · WavPack · WMA · True Audio
Image compression	**ISO/IEC/ITU-T**	JPEG · JPEG 2000 · JPEG XR · lossless JPEG · JBIG · JBIG2 · PNG · WBMP
	Others	APNG · BMP · DjVu · EXR · GIF · ICER · ILBM · MNG · PCX · PGF · TGA · QTVR · TIFF
Media containers	**General**	3GP and 3G2 · ASF · AVI · Bink · DMF · DPX · EVO · FLV · GXF · Matroska · MPEG-PS · **MPEG-TS** (M2TS) · MP4 · MXF · M2V (Packetized elementary stream · Elementary stream) · Ogg · QuickTime · RealMedia · RIFF · Smacker · VOB · WebM
	Audio only	AIFF · AU · WAV

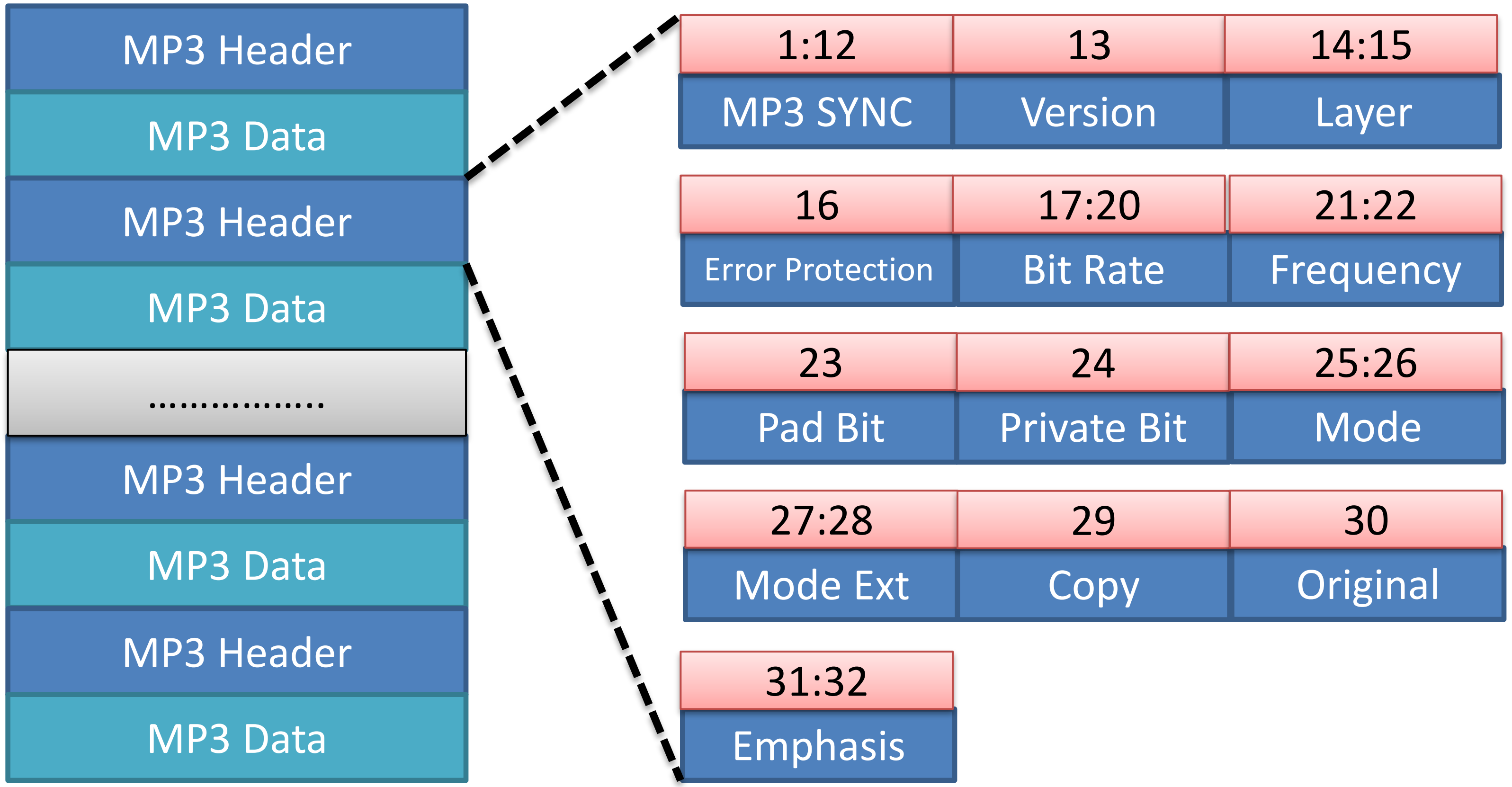
MMN Lab.

What is Multimedia – MP3

- Audio Related Format – MP3
  - MP3 , which means MPEG-I or MPEG-2 Layer 3
  - MPEG-1 Audio (MPEG-1 Part 3) ,Which included MPEG-1 Audio layer I 、II and III , was published in 1993 (ISO/IEC 11172-3)
  - MPEG-2 Audio (MPEG-2 Part3) with additional bit rate and sample rate was published in 1995
  - MP3 format is an lossy compression algorithm to reduce the amount of data size , which reduce the file size up to 1/12 from original
  - MP3 format provide different bit rate(usually between 128 ~ 320kbps) , moreover , the CD bit rate is 1411.2 kbps
  - MP3 support different sampling rate , include MPEG-1 (32,44.1,48KHZ) 、MPEG-2(16,22,24KHZ)

What is Multimedia – MP3

- MP3 File Structure



What is Multimedia – AAC

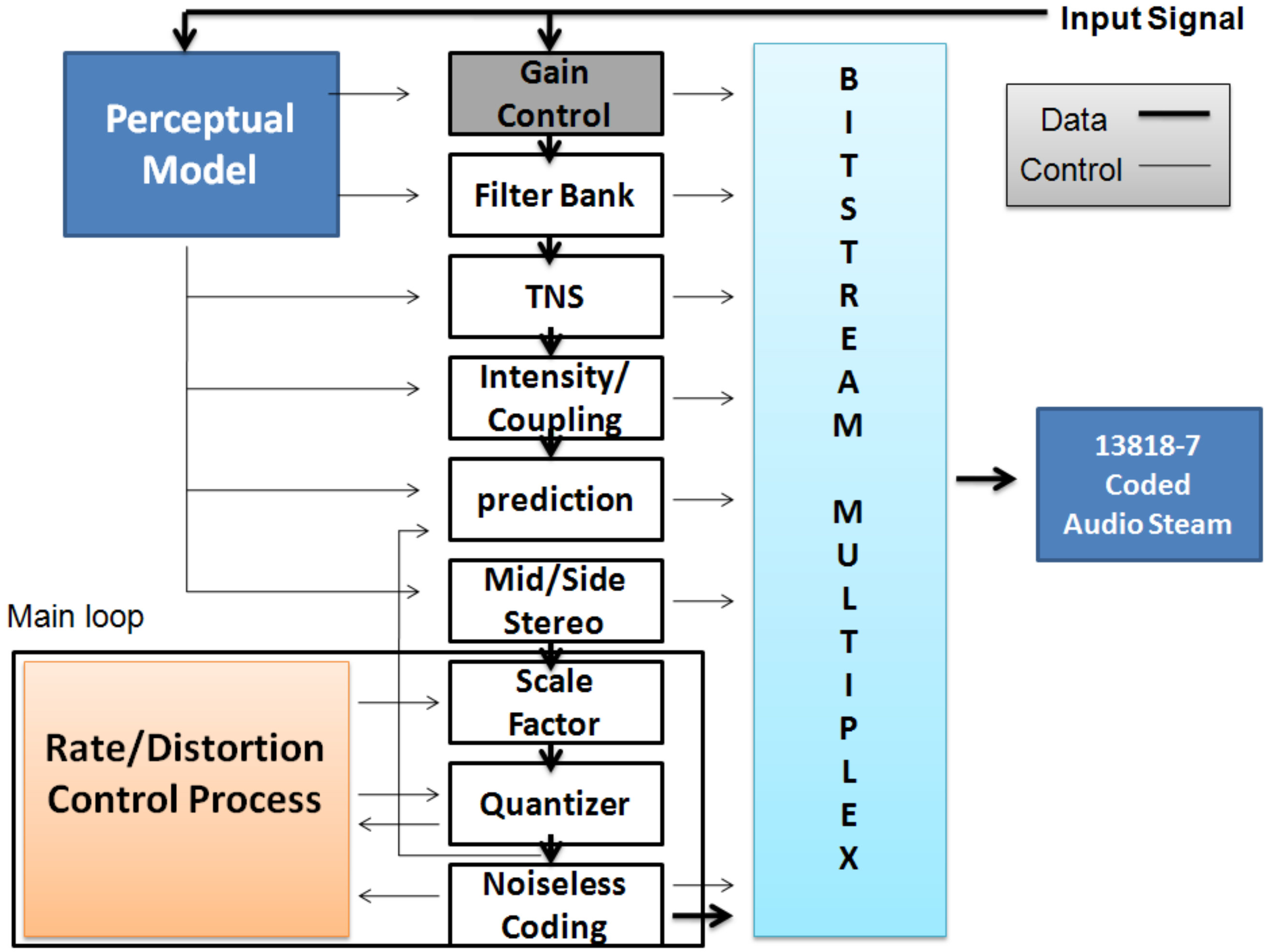
- Audio Related Format – AAC
  - AAC (Advance Audio Coding) , which defined in MPEG-2 Audio(MPEG-2 Part 7) and MPEG-4 Audio(MPEG-4 Part 3)
  - AAC is a wideband audio encoding algorithm , which use lossy compression
  - Signal components that are perceptually irrelevant are discarded
  - Redundancies in the coded audio signal are eliminated
  - More sample frequencies (from 8 to 96 kHz) than MP3 (16 to 48 kHz)
  - Up to 48 channels
  - Arbitrary bit-rates and variable frame length
  - Much better handling of audio frequencies above 16 kHz

What is Multimedia – AAC

- AAC Encode Flow & AAC Profile
    - AAC use Filter Bank 、Temporal Noise Shaping、Prediction and Quantizer method to generate high quality audio stream
    - There are three main AAC profile defined in MPEG-2 Part7
    - MPEG-4 part 3 merge MPEG-2 Part7 basic profile and extend some useful profiles

What is Multimedia – AAC

- AAC Encode Flow & AAC Profile



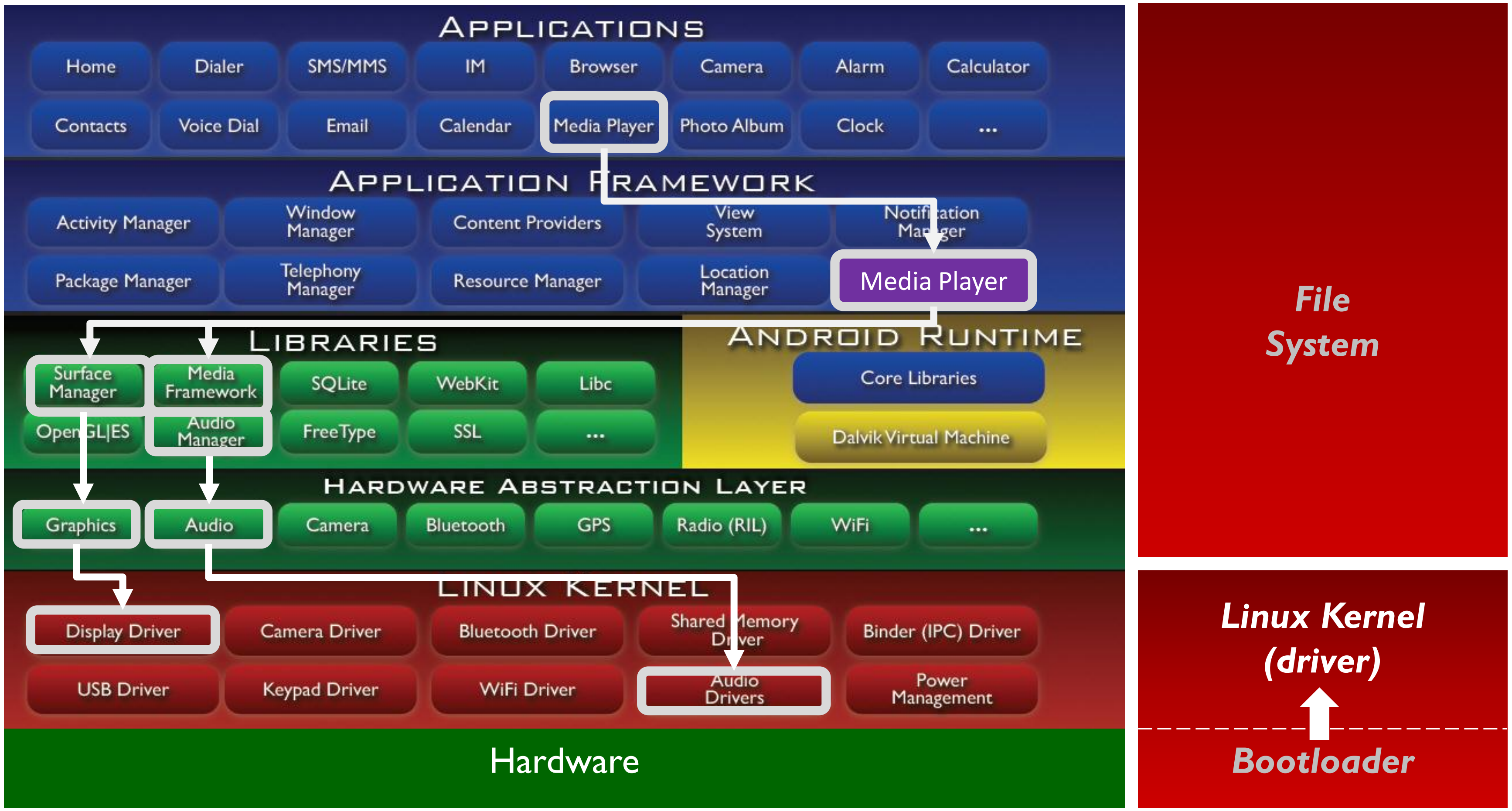
What is Multimedia

- Multimedia Module function
  - Container format parser to recognize and unwrap file
  - Codec to encode/decode data.
  - Synchronization among various stream
  - Memory/Buffer management
  - Stream track control, playback, backwards play, forward play
  - Integrated into video/audio output system
  - Take advantage of hardware accleration
    - Hardware codec
    - Hardware overlay
    - Hardware audio flinger

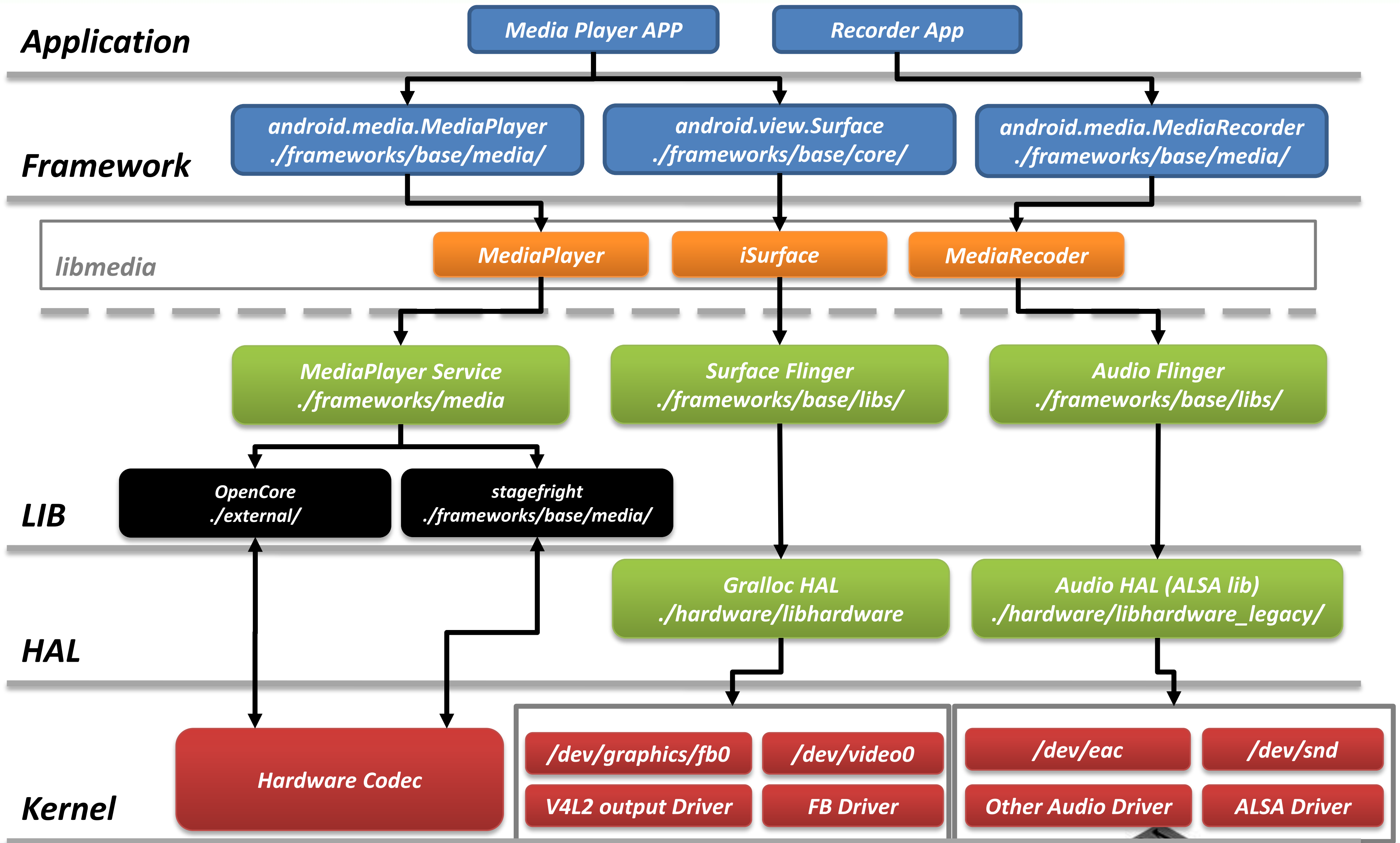
MMN Lab.

Android Multimedia System Introduction

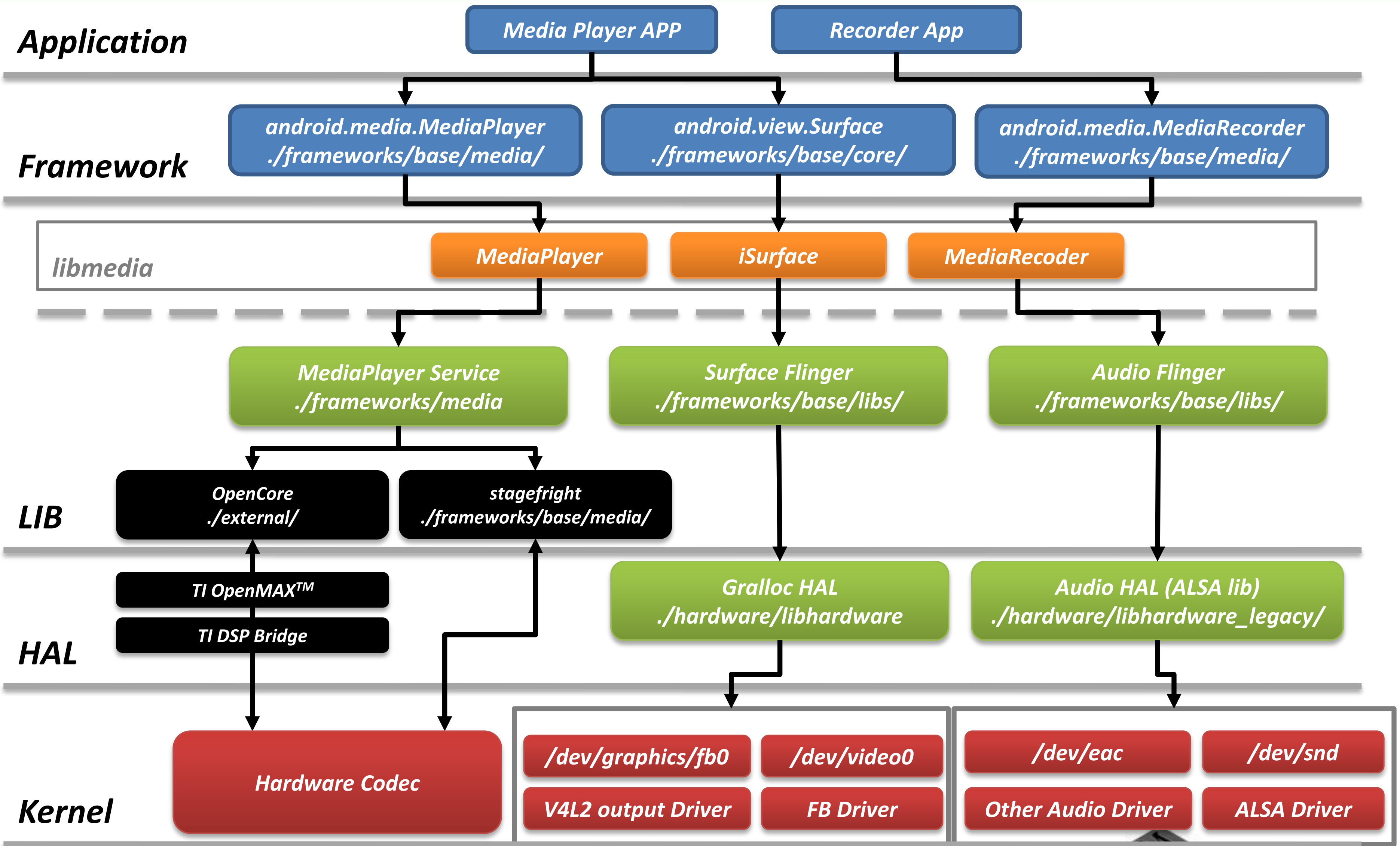
- Android system and Embedded system



Android Multimedia System Introduction


Application
Media Player APP
Recorder App
Framework
android.media.MediaPlayer ./frameworks/base/media/
android.view.Surface ./frameworks/base/core/
android.media.MediaRecorder ./frameworks/base/media/
libmedia
MediaPlayer
iSurface
MediaRecoder
MediaPlayer Service ./frameworks/media
Surface Flinger ./frameworks/base/libs/
Audio Flinger ./frameworks/base/libs/
LIB
OpenCore ./external/
stagefright ./frameworks/base/media/
HAL
Gralloc HAL ./hardware/libhardware
Audio HAL (ALSA lib) ./hardware/libhardware_legacy/
Kernel
Hardware Codec
/dev/graphics/fb0
/dev/video0
V4L2 output Driver
FB Driver
/dev/eac
/dev/snd
Other Audio Driver
ALSA Driver

Android Multimedia System Introduction


Application
Media Player APP
Recorder App
Framework
android.media.MediaPlayer ./frameworks/base/media/
android.view.Surface ./frameworks/base/core/
android.media.MediaRecorder ./frameworks/base/media/
libmedia
MediaPlayer
iSurface
MediaRecoder
MediaPlayer Service ./frameworks/media
Surface Flinger ./frameworks/base/libs/
Audio Flinger ./frameworks/base/libs/
LIB
OpenCore ./external/
stagefright ./frameworks/base/media/
HAL
TI OpenMAX™
TI DSP Bridge
Gralloc HAL ./hardware/libhardware
Audio HAL (ALSA lib) ./hardware/libhardware_legacy/
Kernel
Hardware Codec
/dev/graphics/fb0
/dev/video0
V4L2 output Driver
FB Driver
/dev/eac
/dev/snd
Other Audio Driver
ALSA Driver

Sound Subsystem in Kernel

- The sound subsystem manage all the sound feature in linux kernel
- There are different type of hardware specification support in kernel source tree , which locate in kernel/sound
- The sound module are divided into many parts to satisfy many unix platform

MMN Lab.

Sound Subsystem in Kernel

- Sound source layout in kernel

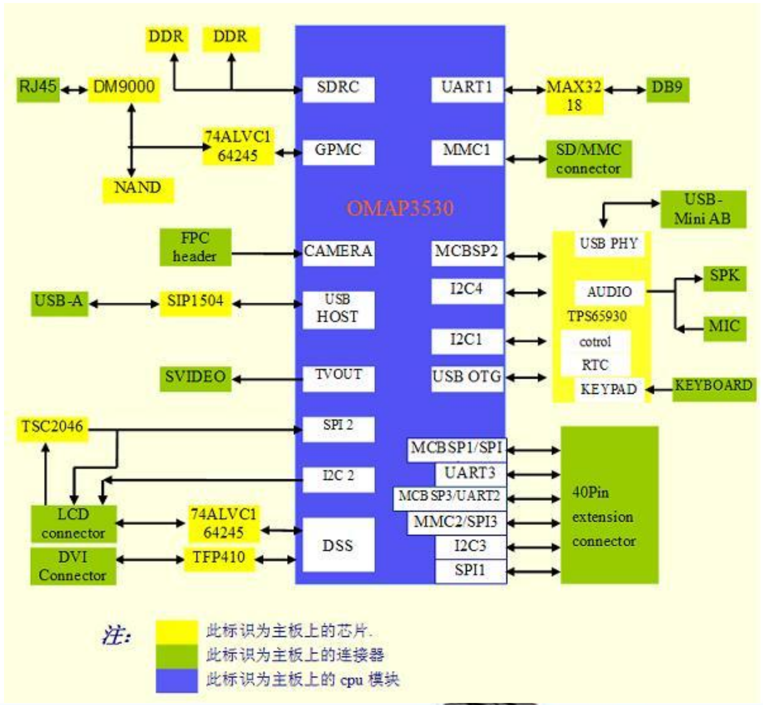
Directory	Description
aoa	Apple Onboard Audio driver
arm	Support for sound devices specific to ARM architectures
core	OSS API and ALSA API
i2c	some i2c driver for ALSA
driver	Some driver for sound card or chip
mips	MIPS sound driver
oss	oss architecture
pci	Sound subsystem for pci device
soc	ALSA for SoC audio support
usb	Sound subsystem for usb device

Sound Subsystem in Kernel

- Linux Sound Card Type
  - ISA bus
    - The oldest sound cards using the original (non Plug and Play) ISA bus
  - ISA Plug and Play
    - The extended version of the ISA bus that supports software identification and configuration of card settings
  - PCI bus
    - The higher bandwidth PCI bus which provides identification and configuration of cards in software
  - USB
    - The newer bus architecture for external hot-pluggable devices

Sound Subsystem in Kernel

- Linux Sound Card Type
  - I2S
    - The electrical serial bus interface standard used for connecting digital audio devices together
  - I2C
    - I2C is a two-wire interface , which provide master/slave architecture to provide faster data transfer



- *What is Multimedia*
  - *Codec and Multimedia Format*
  - *Android Multimedia System Introduction*
- *Sound Subsystem in Kernel*
- ***Alternate Sound Drivers***
- *Main Linux Audio Driver*
  - *Open Sound System, OSS*
  - *Advanced Linux Sound Architecture, ALSA*
- *Related Embedded Player*
- *Lab*

Alternate Sound Drivers

- Linux Sound Card Type
  - OSS/4Front
    - Designed by 4Front Technologies that is supported on a number of sound card , and also compatible with applications written for the standard kernel sound drivers
  - ALSA
    - Advanced Linux Sound Architecture (ALSA) , are full duplex, fully modularized, and compatible with the sound architecture in the kernel
  - Turtle Beach
    - Which is designed for high quality hard disk recording/playback in time

Alternate Sound Drivers

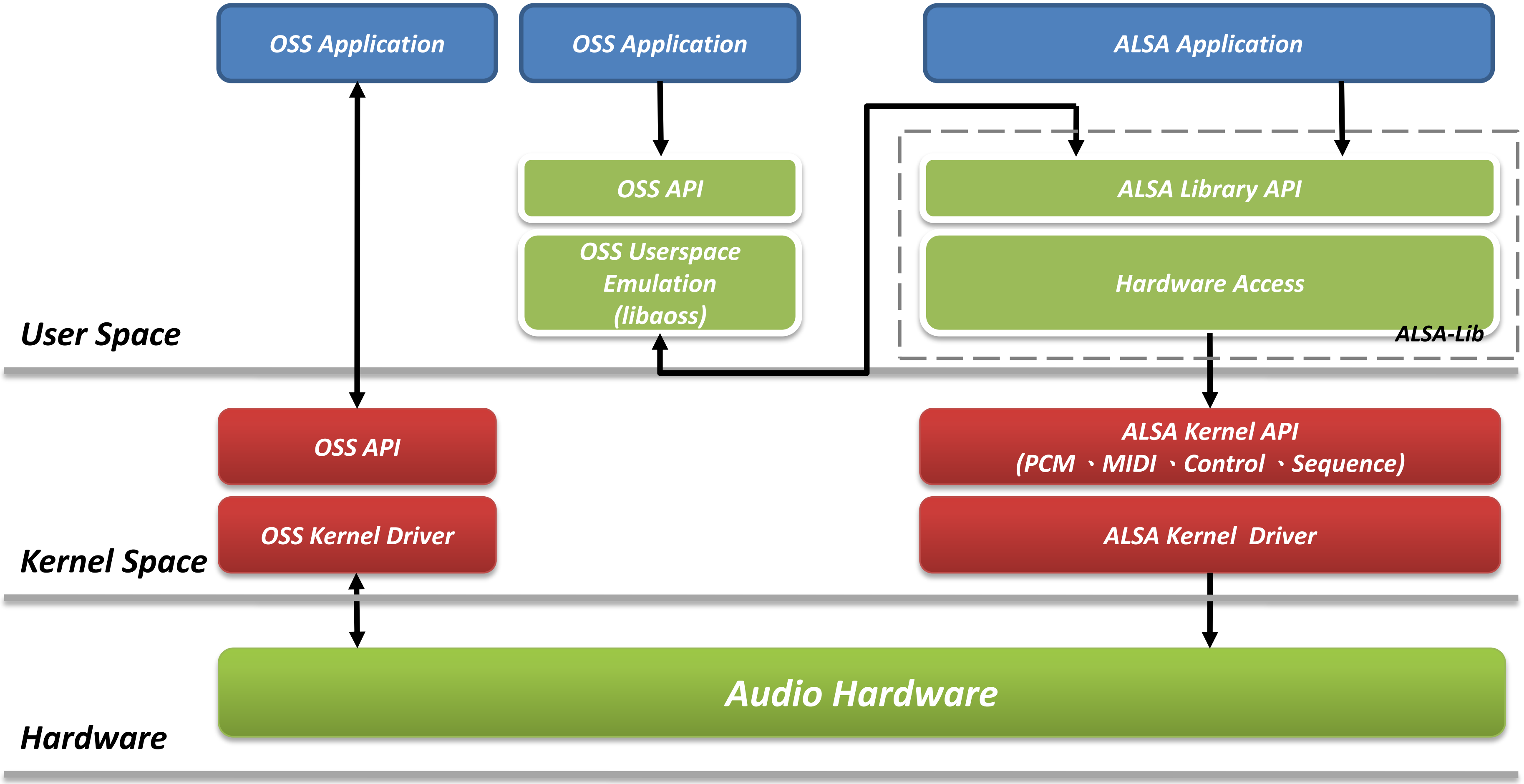
- Linux Sound Card Type
  - Roland MPU-401
    - MPU-401 compatible MIDI interfaces , with a useful collection of utilities including a Standard MIDI File player and recorder
  - SoundBlaster Live!
    - Creative Labs developed Linux drivers for several cards
  - PC Speaker
    - An alternate sound driver for PC speaker , which provides much lower quality output and has much more CPU overhead

MMN Lab.

Main Linux Audio Driver

- Open Sound System (OSS)
  - 2.4 kernel below
  - Old sound card interface
- Advanced Linux Sound Architecture (ALSA)
  - 2.6 kernel above
  - New Age Sound card driver
  - APIs for APP Layer programming

Main Linux Audio Driver



MMN Lab.

- *What is Multimedia*
  - *Codec and Multimedia Format*
  - *Android Multimedia System Introduction*
- *Sound Subsystem in Kernel*
- *Alternate Sound Drivers*
- ***Main Linux Audio Driver***
  - ***Open Sound System, OSS***
  - *Advanced Linux Sound Architecture, ALSA*
- *Related Embedded Player*
- *Lab*

MMN Lab.

Open Sound System, OSS

- OSS Feature
  - Open Sound System (OSS) is an audio interface for linux system
  - OSS provide standard linux device for programming (POSIX read , write , ioctl)
  - Developed by 4Front Technologies teams
    - http://www.4fronttech.com/oss.html
  - Old sound card support system in Linux versions up to 2.4. Still used for some cards in 2.6 (porting to ALSA in progress).

Open Sound System, OSS

- OSS Driver Device
  - The major and minor numbers for these devices are defined in ***Documentation/devices.txt*** in the kernel sources.
  - Use ***mknod*** command to create the device file

```
Host$ sudo mknod /dev/dsp c 14 3
Host$ sudo mknod /dev/mixer c 14 0
```

14 Char Open Sound System (OSS)

0 = /dev/mixer	*Mixer control*	*16 = /dev/mixer1*	*Second soundcard mixer control*
1 = /dev/sequencer	*Audio sequencer*	*17 = /dev/patmgr0*	*Sequencer patch manager*
2 = /dev/midi00	*First MIDI port*	*18 = /dev/midi01*	*Second MIDI port*
3 = /dev/dsp	*Digital audio*	*19 = /dev/dsp1*	*Second soundcard digital audio*
4 = /dev/audio	*Sun-compatible digital audio*	*20 = /dev/audio1*	*Second soundcard Sun digital audio*
6 =		*33 = /dev/patmgr1*	*Sequencer patch manager*
7 = /dev/audioctl	*SPARC audio control device*	*34 = /dev/midi02*	*Third MIDI port*
8 = /dev/sequencer2	*Sequencer -- alternate device*	*50 = /dev/midi03*	*Fourth MIDI port*

Open Sound System, OSS

- OSS Driver Device
  - /dev/dsp
    - API for accessing playback and capture controls
    - The main ***capture*** and ***playback*** device in oss system
      - » Capture function is also implemented by read data from the device (Mic-Reading from /dev/dsp)
      - » Playback function is implemented by writing data to such device (Speaker-Writing to /dev/dsp)

```
Host$ cat /dev/dsp > mmn          -  recodeing
Host$ cat /mmn >  /dev/dsp        -  playback
```

    - Only one application can open /dev/dsp at time
    - Available ***ioctl*** settings: ***sample size*** and ***sample rate***, **number of read** or ***write channels*** (1: mono, 2: stereo).
    - Writing to the device accesses the ***D/A converter*** to produce sound. Reading the device activates the ***A/D converter*** for sound recording and analysis.

MMN Lab.

Open Sound System, OSS

- OSS Driver Device
  - /dev/mixer
    - C API for accessing mixer controls: mainly setting channel volume (left, right or mono), and selecting recording sources.
    - allow user configure sound card feature, like speaker, mic and midi via ***ioctl*** command
    - Applications don't have to open ***/dev/mixer*** to issue these ioctls. They can also use ***/dev/dsp*** if it is already open.
    - Settings are kept even after the applications exit.
  - /dev/audio
    - Same as /dev/dsp, but it is only for Sun compatible digital audio(.au fileformat)

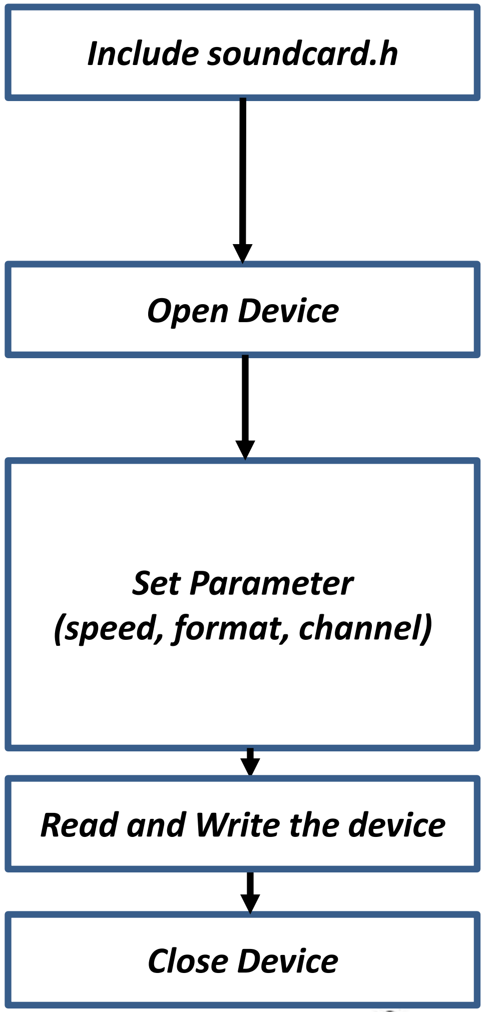
Open Sound System, OSS

- Example for Advance Audio Programming Flow
  1. Include OSS API <soundcard.h>
  2. Open a device file, get the file descriptor
     - Open /dev/mixer
     - Open /dev/dsp
  3. Use ioctl function to control the device property
     - Set parameters
     - Set mixer
  4. Read from device, write to device or do nothing
  5. Close the device

Open Sound System, OSS

- Playback Programming

```
#include <sys/ioctl.h>
#include <sys/soundcard.h>

int main() {
   int audio_fd;
   int music_fd = open(argv[1],O_RDONLY, 0);
   if ((audio_fd = open("/dev/dsp",omode,0)) == -1) {
                 perror("/dev/dsp");
                 exit(1);
   }

   int format = AFMT_S16_NE;
   ioctl(audio_fd,SNDCTL_DSP_SETFMT, &format);
   int channels = 2;
   ioctl(audio_fd, SNDCTL_DSP_CHANNELS, & channels);
   int speed = 44100;
   ioctl(audio_fd, SNDCTL_DSP_SPEED, &speed);
   while ((count = read(music_fd, applicbuf, 2048)) > 0) {
                 write(audio_fd, applicbuf, count);
   }

   close(audio_fd)
}
```


Include soundcard.h
Open Device
Set Parameter (speed, format, channel)
Read and Write the device
Close Device

MMN Lab.

Open Sound System, OSS

- Channel Number (mono, stereo, multiple)
  - Methods for handle multiple channel (depends on hardware) :
    1. ***Interleaved multi channel audio*** : only one device file，record or play the N channels samples one after one for each time slot.
    2. ***Multiple audio device method*** : several device files : one file for one channel sample (/dev/dspM for channel 1, /dev/dspM+1 for channel 2 ….)
- Audio Format
  - Actually most applications need to support just a 16 bit data format. OSS can convert it to any other format if necessary

Name	**Description**
AFMT_U8	8 bit unsigned sample format
AFMT_S8	8 bit signed sample format
AFMT_S16_BE	16 bit signed big endian sample forma
AFMT_S16_LE	16 bit little endian sample format
AFMT_A_LAW	A-Law encoded logarithmic sample format
AFMT_MU_LAW	encoded logarithmic sample format

Supported audio formats

Open Sound System, OSS

- Sample rate
  - Telephone Quality：8 KHz or 11.025 KHz
  - Radio Quality：22.05 KHz
  - CD Quality：44.1KHz
  - DVD Quality：98KHz
- It's very important that applications accept up to ***10%*** differences between the requested and the granted sampling rates. OSS tries to return the real sampling rate as precisely as possible.
  - For example if the requested rate is 22050 Hz the granted value may be something like 22024 Hz. This means just that the nearest rate supported by the the device is 22024 instead of 22050. It's better to tolerate this error instead of refusing to work with the device.

Open Sound System, OSS

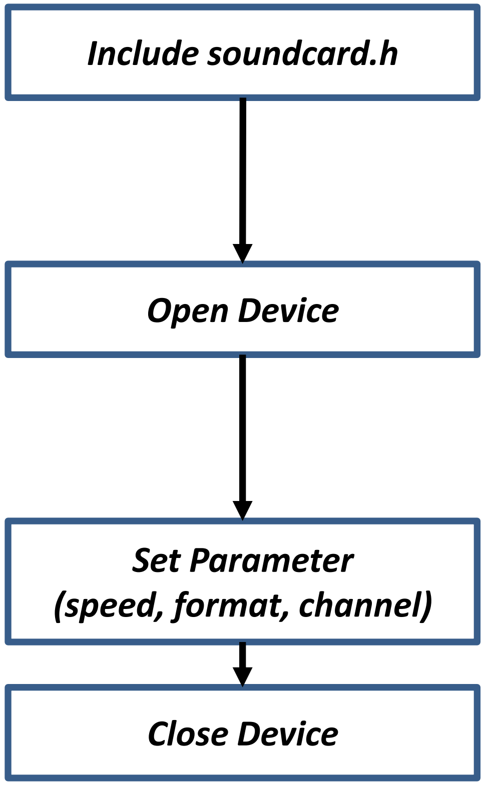
- Ioctl calls available on audio devices

command	Description
SNDCTL_DSP_CHANNELS	Set the number of audio channels
SNDCTL_DSP_SETFMT	Select the sample format
SNDCTL_DSP_SPEED	Set the sampling rate
SNDCTL_DSP_GETPLAYVOL	Returns the current audio playback volume
SNDCTL_DSP_GETFMTS	Returns a list of natively supported sample formats
SNDCTL_DSP_SILENCE	Clears the playback buffer with silence
SNDCTL_DSP_GETOPEAKS	The peak levels for all playback channels
SNDCTL_DSP_GETERROR	Returns audio device error information

Open Sound System, OSS

- Playback Programming

```
#include <sys/ioctl.h>
#include <sys/soundcard.h>

int main() {

int mixer_fd;
if ((mixer_fd = open("/dev/mixer",O_RDONLY,0)) == -1) {
                error("mixer");
                exit(1);
                }

int vol = 0x3f3f;
ioctl(mixer_fd,
MIXER_WRITE(SOUND_MIXER_VOLUME),&vol);

close(mixer_fd);
return 0;

}
```



MMN Lab.

Open Sound System, OSS

- Ioctl calls Mixer channel

command	Description
SOUND_MIXER_VOLUME	Master output level (headphone/line out volume)
SOUND_MIXER_TREBLE	Treble level of all of the output channels
SOUND_MIXER_BASS	Bass level of all of the output channels
SOUND_MIXER_SYNTH	Volume of the synthesizer input (FM, wavetable). In some cases may be connected to other inputs too.
SOUND_MIXER_PCM	Output level for the audio (Codec, PCM, ADC) device (/dev/dsp and /dev/audio)
SOUND_MIXER_SPEAKER	Output volume for the PC speaker signals. Works only if the speaker output is connected directly to the sound card

Open Sound System, OSS

- Each device has a volume setting, which can be ***read/written*** using the following ***macros***. The setting ranges from 0 to 100 and is scaled by the driver. In the case of stereo devices, the least significant byte is the volume for the left channel, the next byte is the volume for the right channel
    - macro ***MIXER_READ()*** returns IOCTL command to read setting
    - macro ***MIXER_WRITE(<--->)*** returns IOCTL command to write setting

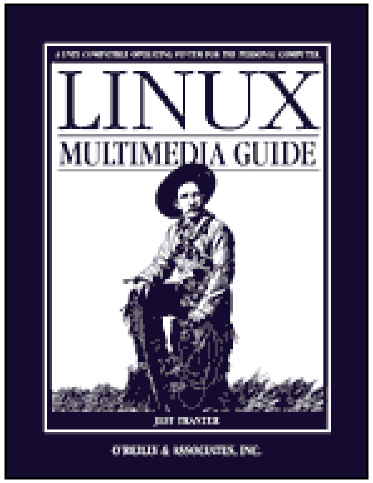
      ioctl(fd, MIXER_READ(SOUND_MIXER_VOLUME), &var)

Open Sound System, OSS

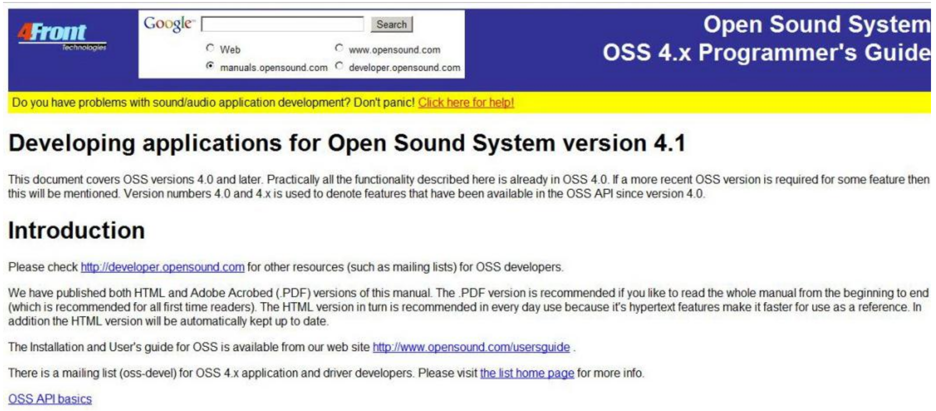
- ***OSS Advantages (users)***
  - Per-application volume control.
  - Some legacy cards are supported better.
  - Initial response time in sound applications is usually better.
  - Better support for applications that use the OSS API. Many applications still use this API, and do not require an emulation layer like ALSA uses.
- ***OSS Advantages (developers)***
  - Cleaner and easier to use API, and better API documentation.
  - Support for drivers in userspace.
  - Cross-platform. OSS runs on BSDs and Solaris.

Open Sound System, OSS

- Useful References
  - O'Reilly's Multimedia Guide
    - http://www.oreilly.de/catalog/multilinux/
  - Developing applications for Open Sound System version 4.1
    - http://manuals.opensound.com/developer/


A UNIX-COMPATIBLE OPERATING SYSTEM FOR THE PERSONAL COMPUTER
LINUX
MULTIMEDIA GUIDE
JEFF TRANTER
O'REILLY & ASSOCIATES, INC.

Linux Multimedia Guide



Do you have problems with sound/audio application development? Don't panic! Click here for help!

Developing applications for Open Sound System version 4.1

This document covers OSS versions 4.0 and later. Practically all the functionality described here is already in OSS 4.0. If a more recent OSS version is required for some feature then this will be mentioned. Version numbers 4.0 and 4.x is used to denote features that have been available in the OSS API since version 4.0.

Introduction

Please check http://developer.opensound.com for other resources (such as mailing lists) for OSS developers.

We have published both HTML and Adobe Acrobed (.PDF) versions of this manual. The .PDF version is recommended if you like to read the whole manual from the beginning to end (which is recommended for all first time readers). The HTML version in turn is recommended in every day use because it's hypertext features make it faster for use as a reference. In addition the HTML version will be automatically kept up to date.

The Installation and User's guide for OSS is available from our web site http://www.opensound.com/usersguide .

There is a mailing list (oss-devel) for OSS 4.x application and driver developers. Please visit the list home page for more info.

OSS API basics

OSS Official Website – 4Front

MMN Lab.

- What is Multimedia
  - Codec and Multimedia Format
  - Android Multimedia System Introduction
- Sound Subsystem in Kernel
- Alternate Sound Drivers
- Main Linux Audio Driver
  - Open Sound System, OSS
  - Advanced Linux Sound Architecture, ALSA
- Related Embedded Player
- Lab

Advanced Linux Sound Architecture, ALSA

- ***Why will be replaced OSS by ALSA?***
  - Better support for USB audio devices. With OSS output is experimental, input is not implemented.
  - Support for Bluetooth audio devices.
  - Support for AC'97 and HDAudio dial-up soft-modems such as Si3055.
  - Better support for MIDI devices. With OSS you'll have to use a software synthesizer such as Timidity or Fluidsynth.
  - ALSA can handle multiple source of audio
  - With support new sound cards, programs for multi-track recording and playback for musicians
  - ALSA provides various ways to support programs that want to use OSS
  - Provide flexible ALSA library in user space
  - OSS Emulation Support

Advanced Linux Sound Architecture, ALSA

- ALSA Feature
  - Efficient support for different types of audio interfaces, from consumer sound cards to multichannel audio interfaces
  - Fully modularized driver
  - SMP and thread-safe design
  - User space library (alsa-lib) to simplify application programming and provide higher level functionality
  - Support for the older Open Sound System (OSS) API, providing binary compatibility for most OSS programs.

http://www.alsa-project.org/main/index.php/Main_Page

Advanced Linux Sound Architecture, ALSA

- ALSA Feature
  - Efficient support for different types of audio interfaces, from consumer
  - OSS emulation: fully supports applications originally created for OSS (still accessing ***/dev/sound***, ***/dev/dsp*** or ***/dev/mixer***).
  - Device files in ***/dev/snd/***.You don't need to use them directly. Use alsalib instead.
- ALSA proc interface – located at /proc
  - ALSA uses for device information and for some control purposes.
    - /proc/asound/oss/
    - /proc/asound/version
    - /proc/asound/devices

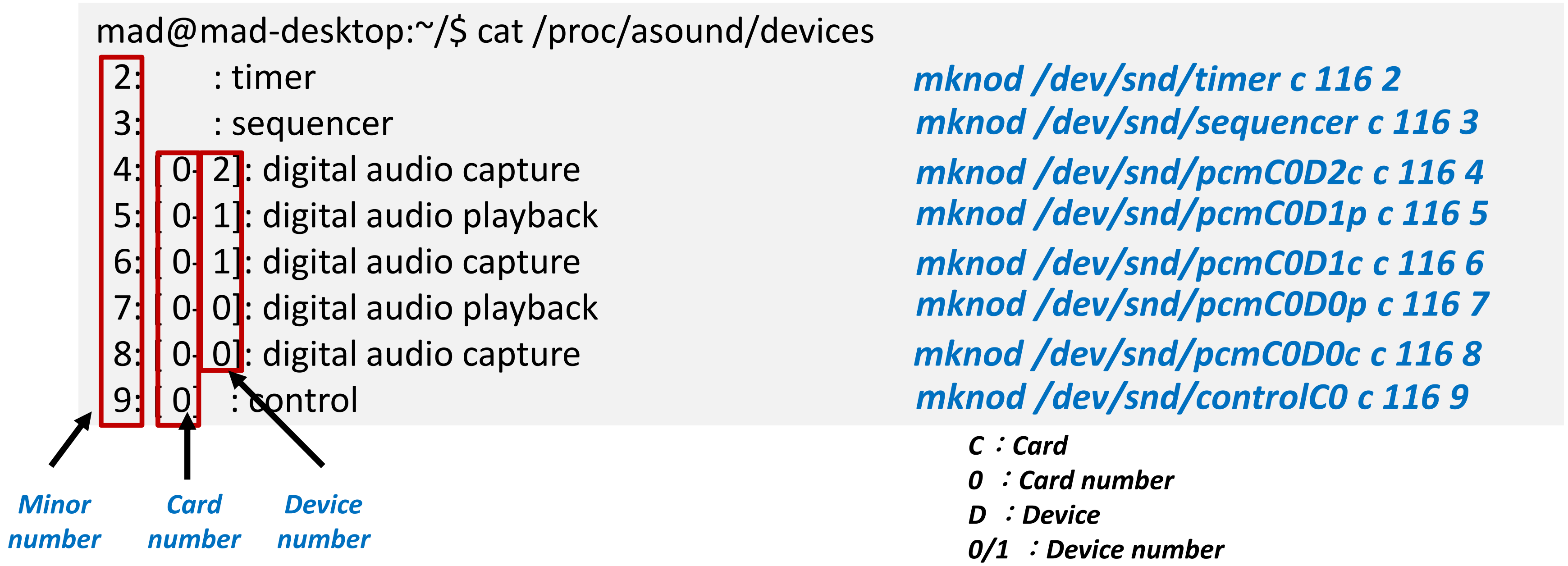
      one control channel
      two PCM playback devices (DAC's)
      two PCM capture devices (ADC's)
      a MIDI sequencer
      a timer

```
mad@mad-desktop:~/$ cat /proc/asound/devices
  2:        : timer
  3:        : sequencer
  4: [ 0- 2]: digital audio capture
  5: [ 0- 1]: digital audio playback
  6: [ 0- 1]: digital audio capture
  7: [ 0- 0]: digital audio playback
  8: [ 0- 0]: digital audio capture
  9: [ 0]   : control
```

Reference：http://alsa.opensrc.org/Proc_asound_documentation#The_.2Fproc.2Fasound.2Foss.2F_directory

Advanced Linux Sound Architecture, ALSA

- ALSA Device
  - Device files in ***/dev/snd/***.You don't need to use them directly. Use alsalib instead.
- how to create ALSA device
  - Use cat /proc/asound/devices

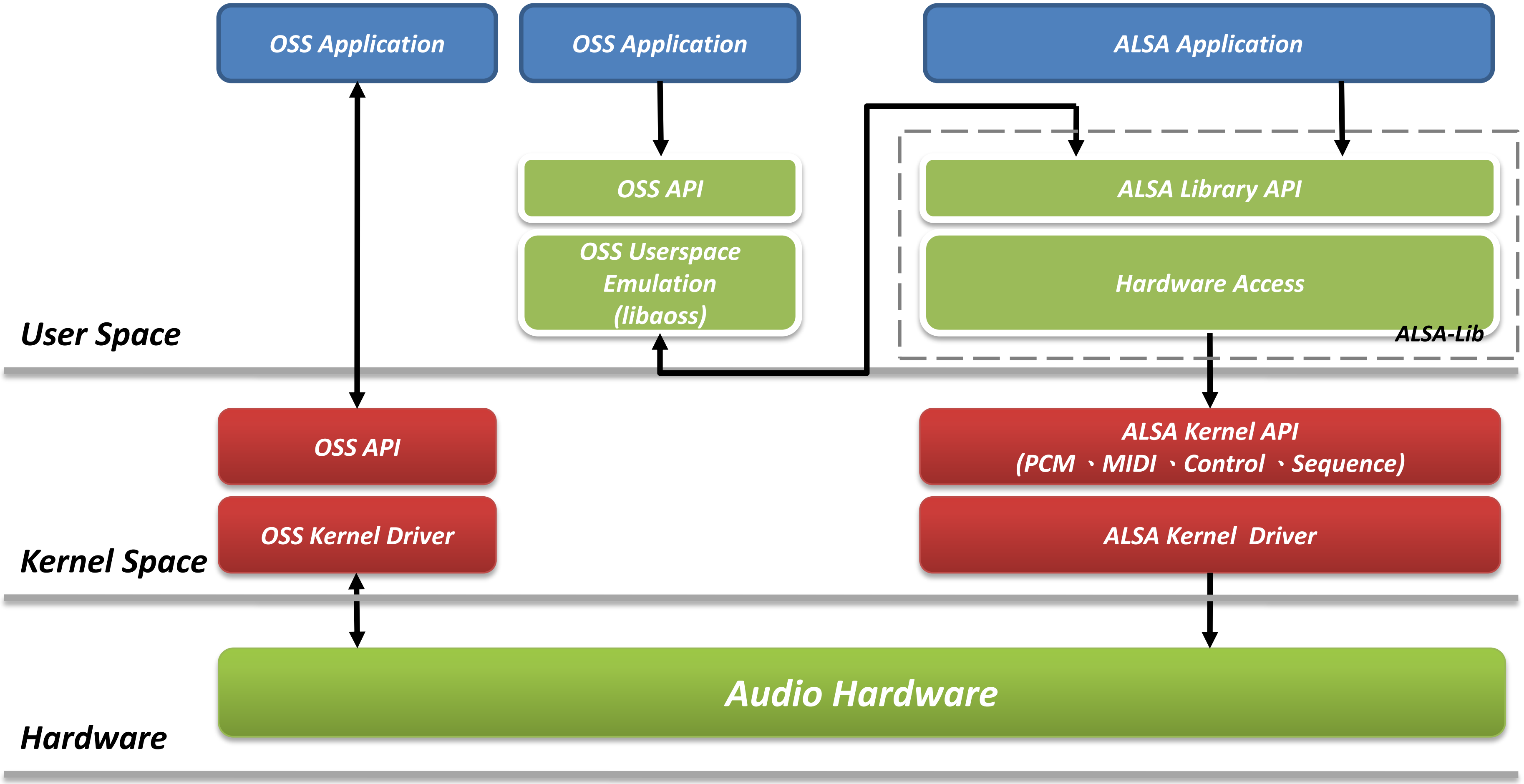


```
mad@mad-desktop:~/$ cat /proc/asound/devices
 2:        : timer                              mknod /dev/snd/timer c 116 2
 3:        : sequencer                          mknod /dev/snd/sequencer c 116 3
 4: [ 0- 2]: digital audio capture              mknod /dev/snd/pcmC0D2c c 116 4
 5: [ 0- 1]: digital audio playback             mknod /dev/snd/pcmC0D1p c 116 5
 6: [ 0- 1]: digital audio capture              mknod /dev/snd/pcmC0D1c c 116 6
 7: [ 0- 0]: digital audio playback             mknod /dev/snd/pcmC0D0p c 116 7
 8: [ 0- 0]: digital audio capture              mknod /dev/snd/pcmC0D0c c 116 8
 9: [ 0]   : control                            mknod /dev/snd/controlC0 c 116 9
```

C	*: Card*
0	*: Card number*
D	*: Device*
0/1	*: Device number*
P/C	*: playback/control*

MMN Lab.

Advanced Linux Sound Architecture, ALSA



Advanced Linux Sound Architecture, ALSA

- ALSA Related Resource
  - ALSA Drivers - Kernel drivers
  - ALSA Lib - Userspace library
  - ALSA Utility - Utilities aplay,arecord,amixer etc
  - ALSA Tools - Tools
  - ALSA firmware - Firmware for cards that require it
  - ALSA Plugin - Additional library plugins Eg.jack, pulse, maemo ..
  - ALSA OSS - OSS compatibility library

Advanced Linux Sound Architecture, ALSA

- ALSA Driver
  - Universal control API
    - New enhanced and flexible API for applications
    - Support for unlimited number of controls
    - support for mixer events
  - Let two or more applications to be synchronized
    - Digital audio (PCM)
    - Support for all types of hardware
    - Full real duplex support
    - Stream start synchronization

Advanced Linux Sound Architecture, ALSA

- ALSA Driver
  - Sequencer
    - support multiple event queues
  - Hardware dependent API
    - support hw specific things
  - OSS/Free kernel emulation
    - Mixer
    - PCM (/dev/dsp) compatibility
- Useful references
  - Write an ALSA driver
    - http://www.alsa-project.org/~tiwai/writing-an-alsa-driver/
  - AlSA Driver API
    - http://www.alsa-project.org/~tiwai/alsa-driver-api/

Advanced Linux Sound Architecture, ALSA

- ALSA Library – Alsa-oss library
  - Alsa-oss is a package that uses a different means of providing OSS emulation
  - One of the aims of ALSA is to provide full OSS compatibility for OSS applications
  - ALSA has two alternative methods of emulating the old OSS sound driver
    1. load ALSA's kernel ***OSSEmulation*** modules: ***snd-pcm-oss***, ***snd-mixer-oss***, and ***snd-seq-oss***
    2. use the *aoss script* included in the alsa-oss package

Advanced Linux Sound Architecture, ALSA

- ALSA Library
  - The ALSA library API is the interface to the ALSA drivers
  - Developers need to use the functions in this API to achieve native ALSA support for their applications
  - The currently designed interfaces are listed below
    1. Information Interface (/proc/asound)
    2. Control Interface (/dev/snd/controlCX)
    3. Mixer Interface (/dev/snd/mixerCXDX)
    4. PCM Interface (/dev/snd/pcmCXDX)
    5. Raw MIDI Interface (/dev/snd/midiCXDX)
    6. Sequencer Interface (/dev/snd/seq)
    7. Timer Interface (/dev/snd/timer)
  - Useful references
    - http://www.alsa-project.org/alsa-doc/alsa-lib/

Advanced Linux Sound Architecture, ALSA

- ALSA Library
  - Control interface
    - General-purpose facility for managing registers of sound cards and querying the available devices
  - High level control interface
    - The high-level primitive controls API
  - Mixer interface
    - controls the devices on sound cards that route signals and control volume levels
  - PCM (digital audio) interface
    - The interface for managing digital audio capture and playback , as it is the one most commonly used for digital audio applications

Advanced Linux Sound Architecture, ALSA

- ALSA Library
  - PCM External Plugin SDK
    - The external PCM plugin SDK
  - External Control Plugin SDK
    - The external control plugin SDK
  - RawMidi interface
    - supports MIDI (Musical Instrument Digital Interface), a standard for electronic musical instruments
  - Timer interface
    - Provides access to timing hardware on sound cards for synchronizing sound events
  - Sequencer interface
    - A higher-level interface for MIDI programming and sound synthesis than the raw MIDI interface

Advanced Linux Sound Architecture, ALSA

- ALSA Library programming flow

More detail in LAB time

Advanced Linux Sound Architecture, ALSA

- ALSA Utility
  - Alsaconf
    - The ALSA driver configurator script
  - Alsactl
    - An utility for soundcard settings management
  - Aplay/Arecord
    - An utility for the playback / capture of .wav,.voc,.au files
  - Amixer
    - A command line mixe
  - Alsamixer
    - A ncurses mixer
  - Amidi
    - A utility to send/receive sysex dumps or other MIDI data

Advanced Linux Sound Architecture, ALSA

- ALSA Utility
  - amixer – set volume control, microphone input level(use command line)

```
Host$: amixer --help
Usage: amixer <options> [command]
Available options:
-c,--card N     select the card
-D,--device N   select the device, default 'default‘
-v,--version    print version of this program

Sequentially
controls        show all controls for given card
contents        show contents of all controls for given card
cset cID P      set control contents for one control
cget cID        get control contents for one control
```

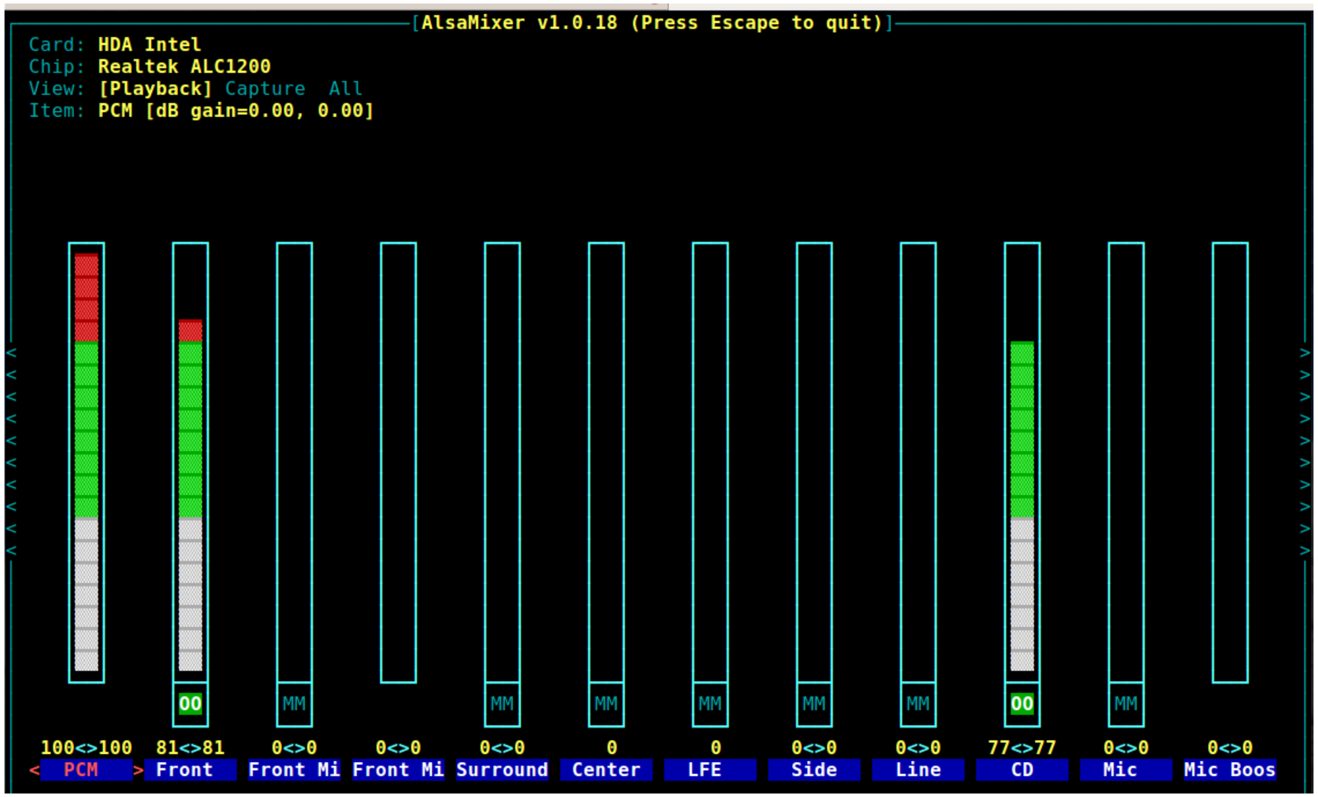
Advanced Linux Sound Architecture, ALSA

- ALSA Utility
  - amixer- See all control interface supported by driver

```
Host$: amixer controls

numid=2,iface=MIXER,name='Master Switch'
numid=1,iface=MIXER,name='Master Volume'
numid=7,iface=MIXER,name='PCM Mode Switch'
numid=6,iface=MIXER,name='PCM Switch'
numid=5,iface=MIXER,name='PCM Volume
……

Host$ amixer cget numid=5,iface=MIXER,name='PCM Volume'
numid=5,iface=MIXER,name='PCM Volume'
; type=INTEGER,access=rw---R--,values=2,min=0,max=27,step=0
: values=27,27
| dBscale-min=-40.50dB,step=1.50dB,mute=0

Host$ amixer cset numid=5,iface=MIXER,name='PCM Volume' 25
numid=5,iface=MIXER,name='PCM Volume'
; type=INTEGER,access=rw---R--,values=2,min=0,max=27,step=0
: values=25,25
| dBscale-min=-40.50dB,step=1.50dB,mute=0
```

Advanced Linux Sound Architecture, ALSA

- ALSA Utility
  - alsamixer



```
[AlsaMixer v1.0.18 (Press Escape to quit)]
Card: HDA Intel
Chip: Realtek ALC1200
View: [Playback] Capture  All
Item: PCM [dB gain=0.00, 0.00]

          OO     MM           MM      MM      MM      MM      MM      OO      MM
100<>100  81<>81  0<>0    0<>0    0<>0    0       0       0<>0    0<>0    77<>77  0<>0    0<>0
< PCM   > Front  Front Mi Front Mi Surround Center  LFE     Side    Line    CD      Mic    Mic Boos
```

MMN Lab.

Advanced Linux Sound Architecture, ALSA

- ASoC (Alsa System on Chip)
  - ASoC subsystem is proposed by Wolfson, support portable embedded system
  - Before ASoC proposed, linux kernel support SoC audio but with some limitation
  - Codec driver is CPU dependent
  - No inform standard for audio event
  - Incomplete power management

Advanced Linux Sound Architecture, ALSA

- ASoC Design Feature
  - Codec independence : different platform use same code driver
  - Simple interface (AC97 / I2S / PCM) between CPU and codec
  - DAPM (Dynamic Audio Power Management)
  - Driver layer
    - Codec Driver
    - Platform Driver
    - Machine Driver
  - DAI (digital/audio interface)

Advanced Linux Sound Architecture, ALSA

- Machine Driver
  - The ASoC machine (or board) driver is the code that tights together the platform and codec drivers
  - Machine driver contains codec and platform specific code
  - It registers the audio subsystem with the kernel as a platform device
- Codec Driver
  - Codec driver is generic and hardware independent code
  - Configures the audio codec to provide audio capture/playback mechanism
  - It should contain no code that is specific to the target platform or machine

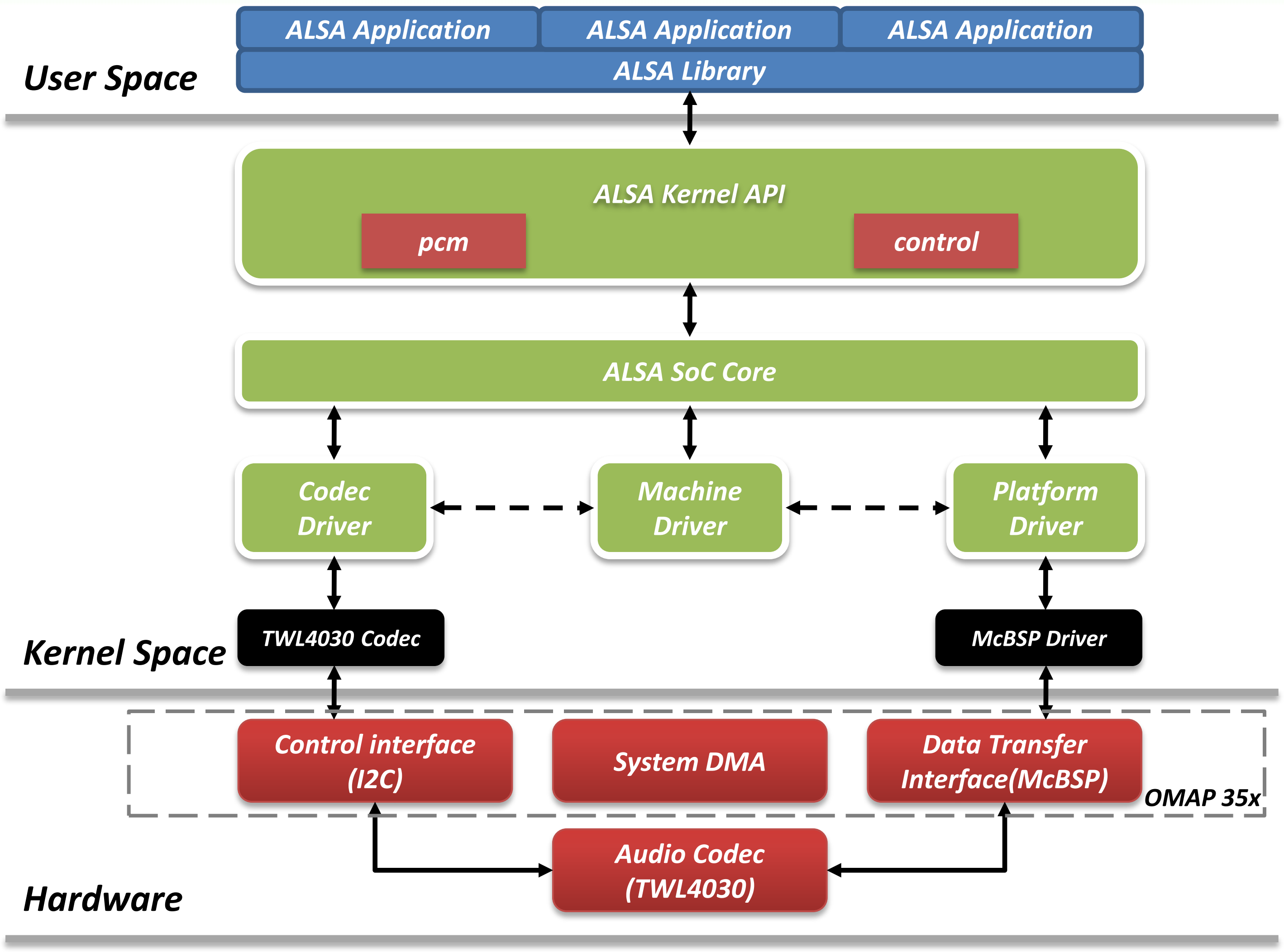
Advanced Linux Sound Architecture, ALSA

- Platform Driver
  - The platform driver can be divided into audio DMA and SoC Digital Audio Interface (DAI) configuration and control
  - The platform driver only targets the SoC CPU and must have no board specific code

Advanced Linux Sound Architecture, ALSA

- ALSA SoC Architecture for OMAP3530
  - Supports TWL4030 audio codec in ALSA SoC framework.
  - Multiple sample rate support (8 KHz, 11.025 KHz, 12 KHz, 16 KHz, 22.05 KHz, 24 KHz, 32 KHz,
  - 44.1 KHz and 48 KHz) for both capture and playback.
  - Supports audio in both mono and stereo modes.
  - Supports simultaneous playback and record (full-duplex mode).
  - Start, stop, pause and resume feature.
  - Supports mixer interface for TWL4030 audio codec

Advanced Linux Sound Architecture, ALSA



ALSA SoC Architecture

MMN Lab.

Embedded Player - Gstreamer

- GStreamer is a pipeline-based multimedia framework written in the C programming language
- Each element is provided by a plug-in
- TI support Dvsdk plug-in component for Gstreamer

Embedded Player - MPlayer

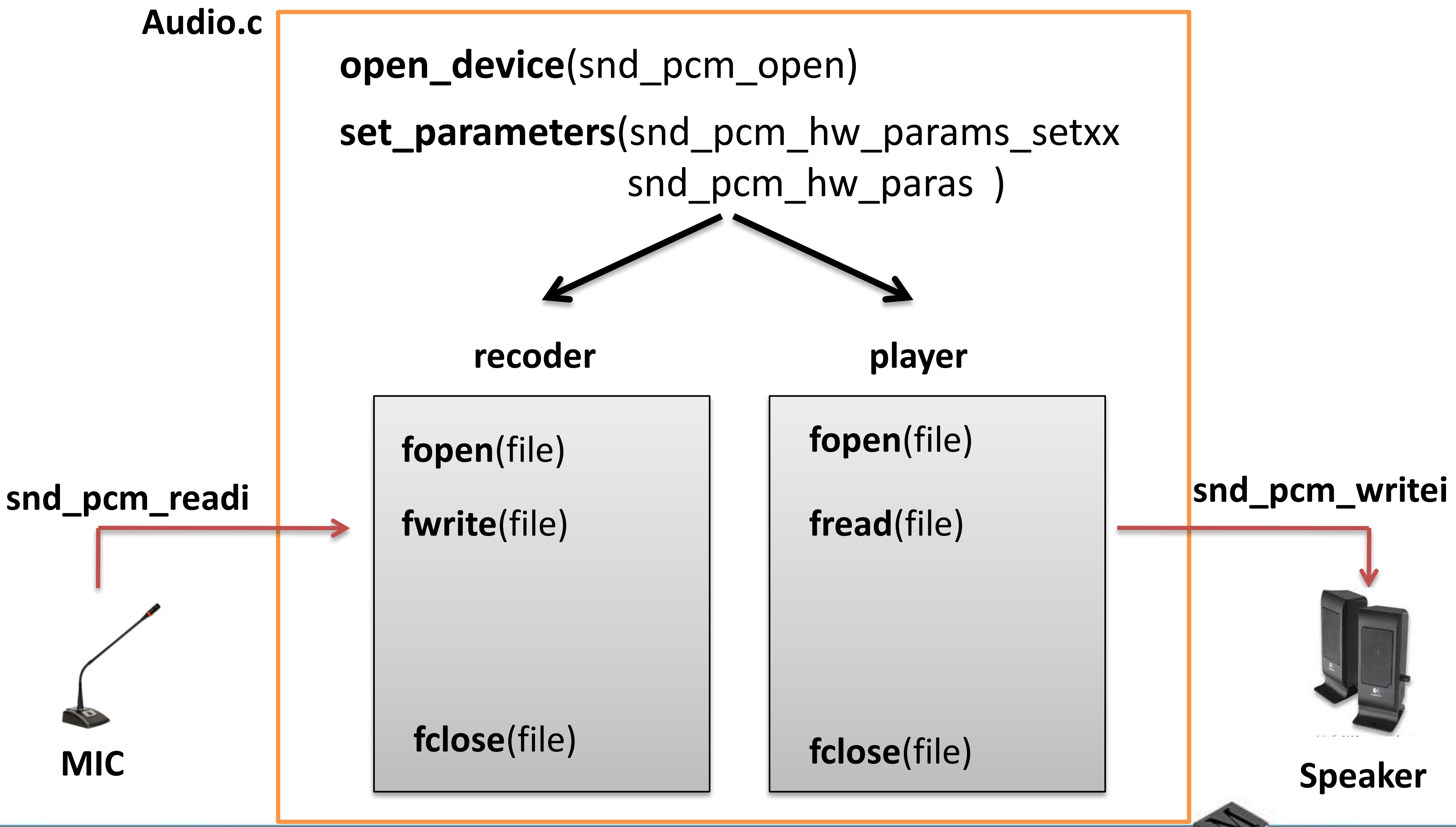
- MPlayer is a free and open source media player
- MPlayer supports various media formats and also save all streamed content to a file
- High Portable flexibility

LAB

- How to use ALSA API
- ALSA API Programming Flow
    1. open playback or record device
    2. set parameters’s structure
    3. write parameters structure to driver
    4. Two function
        1. Audio Record
        2. Audio Playback

LAB

- ALSA API Programming Flow


Audio.c

open_device(snd_pcm_open)

set_parameters(snd_pcm_hw_params_setxx
snd_pcm_hw_paras)

recoder

fopen(file)
fwrite(file)
fclose(file)

player

fopen(file)
fread(file)
fclose(file)

LAB

- ALSA Library API
  - snd_pcm_open ()
    - Open audio device
  - snd_pcm_hw_params_set_XX ()
    - Set parameter's structure
  - snd_pcm_hw_params ()
    - write parameters structure to driver
  - snd_pcm_readi ()
    - read data from device
  - snd_pcm_writei ()
    - write buffer to device

LAB

- ALSA Library API
  - Recording Stage
    - Use ***snd_pcm_readi()*** to read data from device
    - Use ***fwrite()*** to write buffer to file
  - Playback Stage
    - Use ***fread()*** to read data from file to buffer
    - Use ***snd_pcm_writei()*** to write buffer to device

LAB

- ALSA Library API
  - int ***snd_pcm_open***(snd_pcm_t** pcmp,   const char * name, snd_pcm_stream_t stream, int mode)
  - Parameters:
    - pcmp   Returned PCM handle
      - &handle[??]
    - name   ASCII identifier of the PCM handle
      - We choose “default”
    - stream   Wanted stream
      - SND_PCM_STREAM_PLAYBACK
      - SND_PCM_STREAM_CAPTURE
    - mode   Open mode
      - 0

LAB

- ALSA Library API
  - snd_pcm_hw_params_set_access
  - snd_pcm_hw_params_set_format
  - snd_pcm_hw_params_set_channels
  - snd_pcm_hw_params_set_rate_near

LAB

- ALSA Library API
  - int ***snd_pcm_hw_params_set_access***(snd_pcm_t * pcm, snd_pcm_hw_params_t * params,snd_pcm_access_t access)
  - Restrict a configuration space to contain only one access type
  - Parameters:
    - pcm   PCM handle
      - » handle[??]
    - params   Configuration space
      - » We fill declared snd_snd_pcm_hw_params_t object
    - access   access type
      - » SND_PCM_ACCESS_RW_INTERLEAVED

LAB

- ALSA Library API
  - int ***snd_pcm_hw_params_set_format***(snd_pcm_t * pcm, snd_pcm_hw_params_t * params, snd_pcm_format_t format)
  - Restrict a configuration space to contain only one format
  - Parameters:
    - pcm   PCM handle
      - » handle[??]
    - params   Configuration space
      - » We fill declared snd_snd_pcm_hw_params_t object
    - format   format
      - » SND_PCM_FORMAT_S16_LE

LAB

- ALSA Library API
  - int ***snd_pcm_hw_params_set_channels***(snd_pcm_t * pcm, snd_pcm_hw_params_t * params, unsigned int val)
  - Restrict a configuration space to contain only one channels count
  - Parameters:
    - pcm PCM handle
      - » handle[??]
    - params Configuration space
      - » We fill declared snd_snd_pcm_hw_params_t object
    - val channels count
      - » 2

LAB

- ALSA Library API
  - int ***snd_pcm_hw_params_set_rate_near***(snd_pcm_t * pcm, snd_pcm_hw_params_t * params, unsigned int * val,int * dir)
  - Restrict a configuration space to have rate nearest to a target
  - Parameters:
    - pcm PCM handle
      - » handle[??]
    - params Configuration space
      - » We fill declared snd_snd_pcm_hw_params_t object
    - val approximate target rate / returned approximate set rate
      - » val pointer
    - dir Sub unit direction
      - » dir pointer

LAB

- ALSA Library API
  - int ***snd_pcm_hw_params***(snd_pcm_t * pcm, snd_pcm_hw_params_t * params)
  - Install one PCM hardware configuration chosen from a configuration space
    - pcm PCM handle
      - » handle[??]
    - Params Configuration space definition container
      - » We fill declared ***snd_snd_pcm_hw_params_t*** object

LAB

- ALSA Library API
  - snd_pcm_sframes_t ***snd_pcm_readi***(snd_pcm_t * pcm, void * buffer, snd_pcm_uframes_t size)
  - Read interleaved frames from a PCM
  - Parameters:
    - – pcm PCM handle
      - » handle[??]
    - – buffer frames containing buffer
      - » data buffer
    - – size frames to be read
      - » Declared snd_pcm_uframes_t object

LAB

- ALSA Library API
  - snd_pcm_sframes_t ***snd_pcm_writei***(snd_pcm_t * pcm, const void * buffer, snd_pcm_uframes_t size)
  - Write interleaved frames to a PCM
  - Parameters:
    - pcm PCM handle
      - » handle[??]
    - buffer frames containing buffer
      - » data buffer
    - size frames to be written
      - » Declared snd_pcm_uframes_t object

MMN Lab.

LAB

- Audio Driver LAB
  - Use ALSA APIs to Implement following functions
  - Recoder
    - Capture Audio Data from Mic
    - Write to xxx.raw
  - Player
    - Read from xxx.raw
    - Push data to playback device

LAB

- Audio Driver LAB program tip
  - Revise ***audio/audio_playback.c***
  - Revise ***Rule.make***
    - ***KERNEL_DIR (kernel source path)***
    - ***LIB_DIR (External Library Dir)***
    - ***LIB_INC (External Headers Dir)***
  - Fill the blank with right code
  - ***make***
  - copy ***audio_playback*** binary file to target file system
  - Try & debug

LAB

- Appendix - PCM Main APIs-States
  - The ALSA PCM API design uses the states to determine the communication phase between application and library
    - SND_PCM_STATE_OPEN
    - SND_PCM_STATE_SETUP
    - SND_PCM_STATE_PREPARE
    - SND_PCM_STATE_RUNNING
    - SND_PCM_STATE_PAUSED
    - SND_PCM_STATE_DISCONNECTED

LAB

- Appendix - PCM Main APIs-States
  - SND_PCM_STATE_OPEN
    - the PCM device is in the open state. After the ***snd_pcm_open()*** open call, the device is in this state. Also, when ***snd_pcm_hw_params()***
  - SND_PCM_STATE_SETUP
    - he PCM device has accepted communication parameters and it is waiting for ***snd_pcm_prepare()***
  - SND_PCM_STATE_PREPARE
    - The PCM device is prepared for operation. Application can use ***snd_pcm_start()*** call, write or read data to start the operation.
  - SND_PCM_STATE_RUNNING
    - The PCM device has been started and is running. It processes the samples. The stream can be stopped using the ***snd_pcm_drop()*** or ***snd_pcm_drain()*** calls

LAB

- Appendix - PCM Main APIs-States
  - SND_PCM_STATE_PAUSED
    - The device is in this state when application called the ***snd_pcm_pause()*** function until the pause is released
  - SND_PCM_STATE_DISCONNECTED
    - The device is physicaly disconnected. It does not accept any I/O calls in this state.