全志科技 Allwinner Technology

Tina Linux

音频模块使用文档 v1.0

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2018.03.12		初始版本，整合 R40/R16/R6/F35/R18 内容

目　录

1. 概述

1.1. 编写目的

介绍 Tina 平台音频模块的使用方法。

1.2. 适用范围

适用于 Tina SDK。

1.3. 相关人员

音频相关开发人员。

1.4. 相关术语

Audio Driver: Acronyms	
Acronym	Definition
ALSA	Advanced Linux Sound Architecture;
DMA	即直接内存存取，指数据不经 cpu，直接在设备和内存，内存和内存，设备和设备之间传输;
ASoC	ALSA System on Chip;
样本长度 sample	样本是记录音频数据最基本的单位，常见的有 8 位和 16 位;
通道数 channel	该参数为 1 表示单声道，2 则是立体声;
帧 frame	帧记录了一个声音单元，其长度为样本长度与通道数的乘积;
采样率 rate	每秒钟采样次数，该次数是针对帧而言;
周期 period	音频设备一次处理所需要的帧数，对于音频设备的数据访问以及音频数据的存储，都是以此为单位;
交错模式 interleaved	是一种音频数据的记录模式，在交错模式下，数据以连续帧的形式存放，即首先记录完帧 1 的左声道样本和右声道样本（假设为立体声格式），再开始帧 2 的记录，而在非交错模式下，首先记录的是一个周期内所有帧的左声道样本，再记录右声道样本，数据是以连续通道的方式存储。不过多数情况下，我们只需要使用交错模式就可以了;
audiocodec	芯片内置音频接口;
daudio	数字音频接口，可配置成 I2S/PCM 标准音频接口;

2. 模块整体介绍

Linux 中的音频子系统采用 ALSA 架构实现。ALSA 目前已经成为了 Linux 的主流音频体系结构。在内核设备驱动层，ALSA 提供了 alsa-driver，同时在应用层，ALSA 为我们提供了 alsa-lib，应用程序只要调用 alsa-lib 提供的 API，即可以完成对底层音频硬件的控制。

2.1. 驱动架构

Tina SDK 对各个平台的音频设备驱动均采用 ASoC 架构实现。ASoC 是建立在标准 alsa 驱动层上，为了更好地支持嵌入式处理器和移动设备中的音频 codec 的一套软件体系，ASoC 将音频系统分为 3 部分：Codec，Platform 和 Machine。

1. Codec 驱动

   asoc 中的一个重要设计原则就是要求 Codec 驱动是平台无关的，一般提供以下特性：

   Mixer 和其他的音频控件；

   Codec 的 ALSA 音频操作接口；

2. Platform 驱动

   它包含了该 SoC 平台的音频 DMA 和音频接口的配置和控制（I2S，PCM，AC97 等等）；一般不包含与板子或 codec 相关的代码。

3. Machine 驱动

   单独的 Platform 和 Codec 驱动是不能工作的，它必须由 Machine 驱动把它们结合在一起才能完成整个设备的音频处理工作。

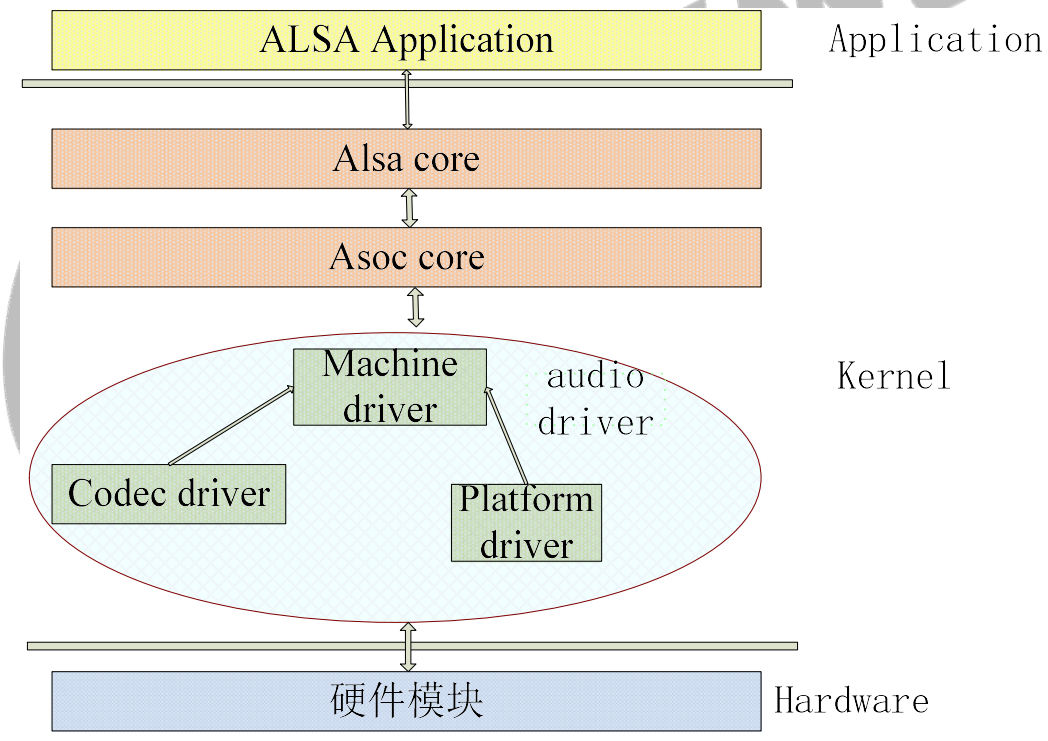


图 2 - 1 ASoC 软件架构图

2.2. 接口使用描述

应用程序使用 alsa-lib 的接口使用音频驱动。在 Tina SDK 中，提供了 3 个应用程序 amixer，aplay，arecord 用于音频的测试。

2.2.1. Mixer 控件接口

对音频通路、功放音量等 mixer 控件（control）接口的操作可通过对 amixer 进行封装来实现。如下所示为获取/设置音量的接口封装 demo。

2.2.1.1. 获取音量 demo

```
int AudioManager::mixer_get(const char* shell,const char* name){
    int bytes;
    char buf[10];

    char cmd[500];
    sprintf(cmd,shell,name);
    FILE    *stream;
    TLOGD("%s\n",cmd);
    stream = popen( cmd, "r" );
    if(!stream) return -1;
    bytes = fread( buf, sizeof(char), sizeof(buf), stream);
    pclose(stream);
    if(bytes > 0){
            return atoi(buf);
    }else {
            TLOGE("%s --> failed\n",cmd);
            return -1;
    }
}

int AudioManager::mixer_getcurvolume(const char* name){
    const char* shell = "volume=`amixer cget name='%s' | grep ': values='`;volume=${volume#*=};echo $volume";
    return mixer_get(shell,name);
}
```

2.2.1.2. 设置音量 demo

```
void AudioManager::mixer_set(const char* name, int value){
    char cmd[100];
    sprintf(cmd,"amixer cset name='%s' %d",name,value);
system(cmd);
}
```

2.2.1.3. amixer 使用

运行命令

```
amixer -h
```

可获取 amixer 的帮助信息：

```
root@TinaLinux:/# amixer -h
amixer -h
Usage: amixer <options> [command]

Available options:
  -h,--help       this help
  -c,--card N     select the card
  -D,--device N   select the device, default 'default'
  -d,--debug      debug mode
  -n,--nocheck    do not perform range checking
  -v,--version    print version of this program
  -q,--quiet      be quiet
  -i,--inactive   show also inactive controls
  -a,--abstract L select abstraction level (none or basic)
  -s,--stdin      Read and execute commands from stdin sequentia
  -R,--raw-volume Use the raw value (default)
  -M,--mapped-volume Use the mapped volume

Available commands:
  scontrols       show all mixer simple controls
  scontents       show contents of all mixer simple controls (de
  sset sID P      set contents for one mixer simple control
```

图 2 - 2 amixer -h

amixer 默认操作的为 card0，若要操作 card1，需要加上参数 -c 1。

不同平台的 mixer 控件的名字可能会有所不同，可使用

```
amixer contents
```

查看当前平台可用的 mixer 控件及其当前值。

假设当前平台 speaker 音量调节的 mixer 控件名字为“speaker volume control”,可用以下命令获取 speaker 当前音量：

```
amixer -c 1 cget iface=MIXER,name='speaker volume control'
```

若要设置 speaker 音量为 40 可用：

```
amixer -c 1 cset iface=MIXER,name='speaker volume control' 40
```

2.2.2. Play 接口

2.2.2.1.snd_pcm_open

函数原型	int snd_pcm_open(snd_pcm_t **pcm, const char *name, snd_pcm_stream_t stream, int mode);
参数说明	pcm：声卡句柄； name：声卡名称； default 默认是（card0，device0）； ”hw:1,0”表示（card0，device0）； “plug:dmix”表示使用混音设备接口； stream：音频流名称； SND_PCM_STREAM_PLAYBACK 播放流； SND_PCM_STREAM_CAPTURE:录音流； mode: 播放模式选择；0：阻塞模式；1：非阻塞模式；
返回说明	Return 0：声卡 open 成功；
功能描述	打开 pcm

2.2.2.2.snd_pcm_hw_params

函数原型	int snd_pcm_hw_params(snd_pcm_t *pcm, snd_pcm_hw_params_t *params);
参数说明	pcm：声卡句柄； params：声卡设备相关参数； params 中需要设置数据格式，采样率，通道数，音频数据模式（比如交错模式）； 音频相关 params 设置，参考备注
返回说明	Return 0：声卡设置参数成功
功能描述	申请 pcm 硬件参数
备注	
音频参数设置步骤如下： 1.申请 param 参数存储地址 snd_pcm_hw_params_malloc snd_pcm_hw_params_any 2.设置音频数据格式：SND_PCM_ACCESS_RW_INTERLEAVED snd_pcm_hw_params_set_access 3.设置音频采样精度 snd_pcm_hw_params_set_format 4.设置音频采样率 snd_pcm_hw_params_set_rate 5.设置音频通道数据 snd_pcm_hw_params_set_channels 6.将以上设置参数，写入到 alsa lib 里面 snd_pcm_hw_params 7.释放 param 存储地址 snd_pcm_hw_params_free	

2.2.2.3.snd_pcm_writei

函数原型	snd_pcm_sframes_t snd_pcm_writei(snd_pcm_t *pcm, const void *buffer, snd_pcm_uframes_t size);
参数说明	pcm：声卡句柄； buffer：音频数据地址； size：音频数据大小；
返回说明	Return 0：声卡写数据成功；
功能描述	pcm 写数据

2.2.2.4.snd_pcm_close

函数原型	int snd_pcm_close(snd_pcm_t *pcm);
参数说明	pcm：声卡句柄
返回说明	Return 0：关闭声卡设备成功
功能描述	关闭声卡设备

2.2.2.5.播放测试

用 aplay 播放一个音频文件：

```
aplay -D "hw:0,0" record.wav
```

其中-D "hw:0,0"表示“使用声卡 0，设备 0”

2.2.2.6.播放的 demo 代码

```
int play_stereo_test(int sample_rate)
{
    int i;
    int err;
```

```
    snd_pcm_t *playback_handle;
    snd_pcm_hw_params_t *hw_params;
    FILE *fp = NULL;
    fprintf(stderr, "sample_rate is %d\n", sample_rate);

    if((err = snd_pcm_open(&playback_handle,    "default", SND_PCM_STREAM_PLAYBACK, 0)) < 0)
    {
        fprintf(stderr, "cannot open audio device record.pcm (%s)\n",    snd_strerror(err));
        return -1;
    }
    if((err = snd_pcm_hw_params_malloc(&hw_params)) < 0)
    {
        fprintf(stderr, "cannot allocate hardware parameter structure (%s)\n",snd_strerror(err));
        return -1;
    }
    if((err = snd_pcm_hw_params_any(playback_handle, hw_params)) < 0)
    {
        fprintf(stderr, "cannot initialize hardware parameter structure (%s)\n", snd_strerror(err));
        return -1;
    }

    if((err = snd_pcm_hw_params_set_access(playback_handle, hw_params,
                SND_PCM_ACCESS_RW_INTERLEAVED)) < 0)
    {
        fprintf(stderr, "cannot allocate hardware parameter structure (%s)\n",snd_strerror(err));
        return -1;
    }

    if((err = snd_pcm_hw_params_set_format(playback_handle, hw_params, SND_PCM_FORMAT_S16_LE)) <
0)
    {
        fprintf(stderr, "cannot allocate hardware parameter structure (%s)\n",snd_strerror(err));
        return -1;
    }

    if((err = snd_pcm_hw_params_set_rate(playback_handle, hw_params, sample_rate, 0)) < 0)
    {
        fprintf(stderr , "cannot set sample rate (%s)\n", snd_strerror(err));
        return -1;
    }
    if((err = snd_pcm_hw_params_set_channels(playback_handle, hw_params, 2)) <0)
    {
        fprintf(stderr, "cannot set channel count (%s)\n", snd_strerror(err));
        return -1;
    }
    if((err = snd_pcm_hw_params(playback_handle, hw_params)) < 0)
    {
        fprintf(stderr, "cannot set parameters (%s)\n", snd_strerror(err));
        return -1;
    }
    snd_pcm_hw_params_free(hw_params);
    fprintf(stderr, "open file : record.pcm\n");
    fp = fopen(filename, "r");
    if (fp == NULL) {
        fprintf(stderr, "open test pcm file err\n");
        return -1;
    }
```

```
    for (i = 0; i < REPLAY_TIME; i++) {
        while (!feof(fp))
        {
            err = fread(buf, 1, BUF_LEN, fp);
            if (err < 0)
            {
                fprintf(stderr, "read pcm from file err\n");
                return -1;
            }
            err = snd_pcm_writei(playback_handle, buf , BUF_LEN/4);
            if (err < 0)
            {
                fprintf(stderr, "write to audio interface failed (%s)\n",
                snd_strerror(err));
                return -1;
            }
            if (feof(fp)) {
                break;
            }
        }
        fprintf(stderr, "%d\n", i);
    }

    fprintf(stderr, "close file\n");
    fclose(fp);

    fprintf(stderr, "close dev\n");
    snd_pcm_close(playback_handle);
    fprintf(stderr, "ok\n");
}
```

2.2.3. Record 接口

2.2.3.1.snd_pcm_open

函数原型	int snd_pcm_open(snd_pcm_t **pcm, const char *name, snd_pcm_stream_t stream, int mode);
参数说明	pcm：声卡句柄； name：声卡名称； default 默认是（card0，device0）；”hw:1,0”表示（card0，device0）；“plug:dmix”表示使用混音设备接口； stream：音频流名称； SND_PCM_STREAM_PLAYBACK 播放流； SND_PCM_STREAM_CAPTURE:录音流； mode: 播放模式选择；0：阻塞模式；1：非阻塞模式；
返回说明	Return 0：声卡 open 成功；
功能描述	打开声卡设备

2.2.3.2.snd_pcm_hw_params

函数原型	int snd_pcm_hw_params(snd_pcm_t *pcm, snd_pcm_hw_params_t *params);
参数说明	pcm：声卡句柄；

	params：声卡设备相关参数； params 中需要设置数据格式，采样率，通道数，音频数据模式（比如交错模式）； 音频相关 params 设置，参考 note：
返回说明	Return 0：声卡设置参数成功；
功能描述	设置 pcm 硬件参数
备注：	
音频参数设置步骤如下： 1.申请 param 参数存储地址 snd_pcm_hw_params_malloc snd_pcm_hw_params_any 2.设置音频数据格式：SND_PCM_ACCESS_RW_INTERLEAVED snd_pcm_hw_params_set_access 3.设置音频采样精度 snd_pcm_hw_params_set_format 4.设置音频采样率 snd_pcm_hw_params_set_rate 5.设置音频通道数据 snd_pcm_hw_params_set_channels 6.将以上设置参数，写入到 alsa lib 里面 snd_pcm_hw_params 7.释放 param 存储地址 snd_pcm_hw_params_free	

2.2.3.3.snd_pcm_readi

函数原型	snd_pcm_sframes_t snd_pcm_readi(snd_pcm_t *pcm, const void *buffer, snd_pcm_uframes_t size);
参数说明	pcm：声卡句柄； buffer：音频数据地址； size：音频数据大小；
返回说明	Ret > 0 读取音频数据帧数； Ret < 0 读取音频数据失败；
功能描述	pcm 读数据

2.2.3.4.snd_pcm_close

函数原型	int snd_pcm_close(snd_pcm_t *pcm);
参数说明	pcm：声卡句柄；
返回说明	Return 0：关闭声卡设备成功；
功能描述	关闭 pcm 设备

2.2.3.5.录音测试

使用 arecord 进行录音：

```
arecord -D "hw:0,0" -r 48000 -c 2 -f "S16_LE" -d 5 record.wav
```

其中，-D "hw:0,0"表示“使用声卡 0，设备 0”，-r 48000 表示采样率为 48k Hz，-c 2 表示声道数为 2，-f "S16_LE"表示每个采样点保存为小端序的有符号 16 位数据，-d 5 表示录音时长为 5 秒。

2.2.3.6.录音的 demo 代码

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <alsa/asoundlib.h>

#define BUF_LEN           (4096)
#define REPLAY_TIME    (1)

char buf[BUF_LEN];
static char filename[64] = "test.pcm";
int main(void)
{
	int i;
	int err;
	snd_pcm_t *capture_handle;
	snd_pcm_hw_params_t *hw_params;
	int dtime = 1;
	int r;

	FILE *fp = NULL;
	int loop;
	int sample_rate = 48000;


	fprintf(stderr, "sample_rate is %d\n", sample_rate);

	if((err = snd_pcm_open(&capture_handle,    "default", SND_PCM_STREAM_CAPTURE, 0)) < 0)
	//if((err = snd_pcm_open(&capture_handle,    "default", SND_PCM_OPEN_DUPLEX, 0)) < 0)
		{
		fprintf(stderr, "cannot open audio device (%s)\n",    snd_strerror(err));
		return -1;
	}

	if((err = snd_pcm_hw_params_malloc(&hw_params)) < 0)
	{
		fprintf(stderr, "cannot allocate hardware parameter structure (%s)\n",snd_strerror(err));
		return -1;
	}

	if((err = snd_pcm_hw_params_any(capture_handle, hw_params)) < 0)
	{
		fprintf(stderr, "cannot initialize hardware parameter structure (%s)\n", snd_strerror(err));
		return -1;
	}
	if((err = snd_pcm_hw_params_set_access(capture_handle, hw_params,
			SND_PCM_ACCESS_RW_INTERLEAVED)) < 0)
	{
		fprintf(stderr, "cannot allocate hardware parameter structure (%s)\n",snd_strerror(err));
		return -1;
	}

	if((err = snd_pcm_hw_params_set_format(capture_handle, hw_params, SND_PCM_FORMAT_S16_LE)) <
0)
	{
		fprintf(stderr, "cannot allocate hardware parameter structure (%s)\n",snd_strerror(err));
		return -1;
	}

	if((err = snd_pcm_hw_params_set_rate(capture_handle, hw_params, sample_rate, 0)) < 0)
	{
```

```
        fprintf(stderr , "cannot set sample rate (%s)\n", snd_strerror(err));
        return -1;
    }

    if((err = snd_pcm_hw_params_set_channels(capture_handle, hw_params, 2)) <0)
    {
        fprintf(stderr, "cannot set channel count (%s)\n", snd_strerror(err));
        return -1;
    }

    if((err = snd_pcm_hw_params(capture_handle, hw_params)) < 0)
    {
        fprintf(stderr, "cannot set parameters (%s)\n", snd_strerror(err));
        return -1;
    }

    snd_pcm_hw_params_free(hw_params);

    fprintf(stderr, "open file : %s\n",filename);
    dtime = 60;

    fp = fopen(filename, "wb+");
    if (fp == NULL) {
        fprintf(stderr, "open test pcm file err\n");
        return -1;
    }
    loop = dtime*sample_rate/1024;

        for(loop;loop > 0;loop--)
        {
            r = snd_pcm_readi( capture_handle, buf , 1024);
            if(r>0)
            {
                err = fwrite(buf, 1, r*4, fp);

            if(err < 0)
            {
                fprintf(stderr, "write to audio interface failed (%s)\n",snd_strerror(err));
                return err;
            }
            }

        }

    fprintf(stderr, "close file\n");
    fclose(fp);

    fprintf(stderr, "close dev\n");
    snd_pcm_close(capture_handle);
    fprintf(stderr, "ok\n");

    return 0;
}
```

3. 读写寄存器的方法

对于数字寄存器，可使用以下方法进行读/写：

```
cd /sys/class/sunxi_dump
echo 0xf1c22000,0xf1c2203c > dump;cat dump;        # 读
echo 0xf1c22c00 0xf > write;cat write;             # 写（如将 0xf 写入到 0xf1c22c00）
```

4. R40 音频模块

4.1. 硬件框架

R40 中包含 5 个声卡设备，分别为内置 audiocodec，daudio0（tdm 模式），daudio1（tdm 模式），HDMI，S/PDIF（owa）。

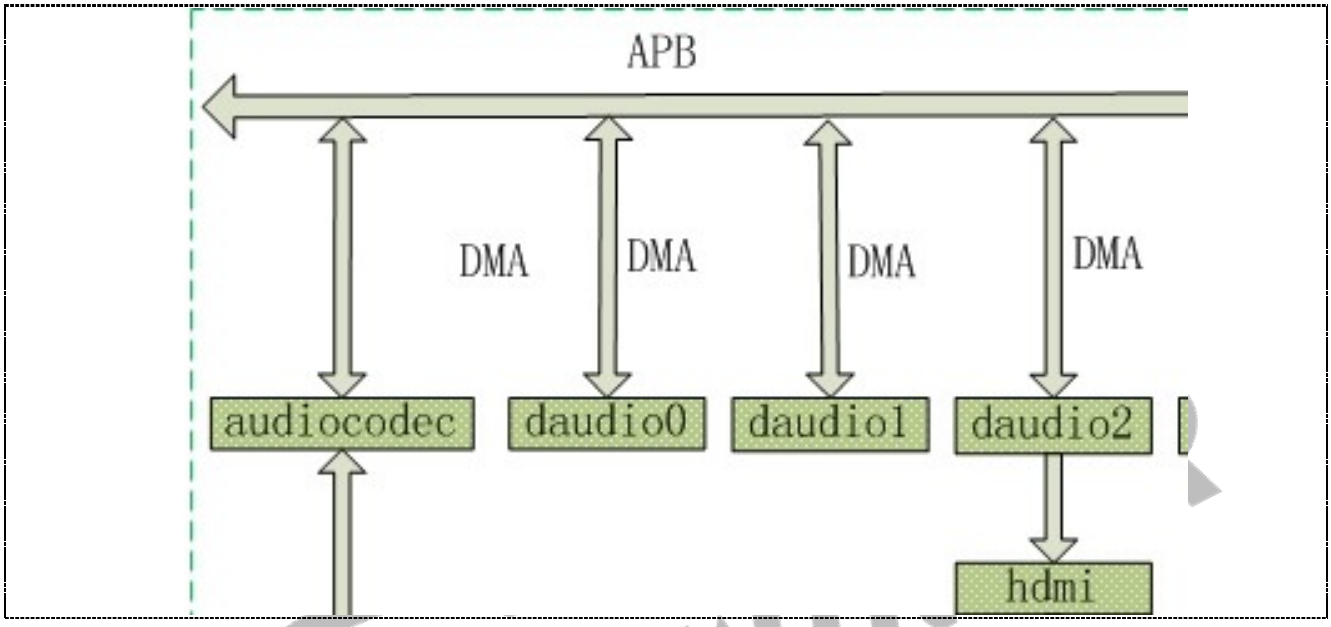


图 4 - 1 R40 音频硬件框架

4.2. 软件框架

R40 中 5 个声卡设备均采用 ASoC 架构实现。

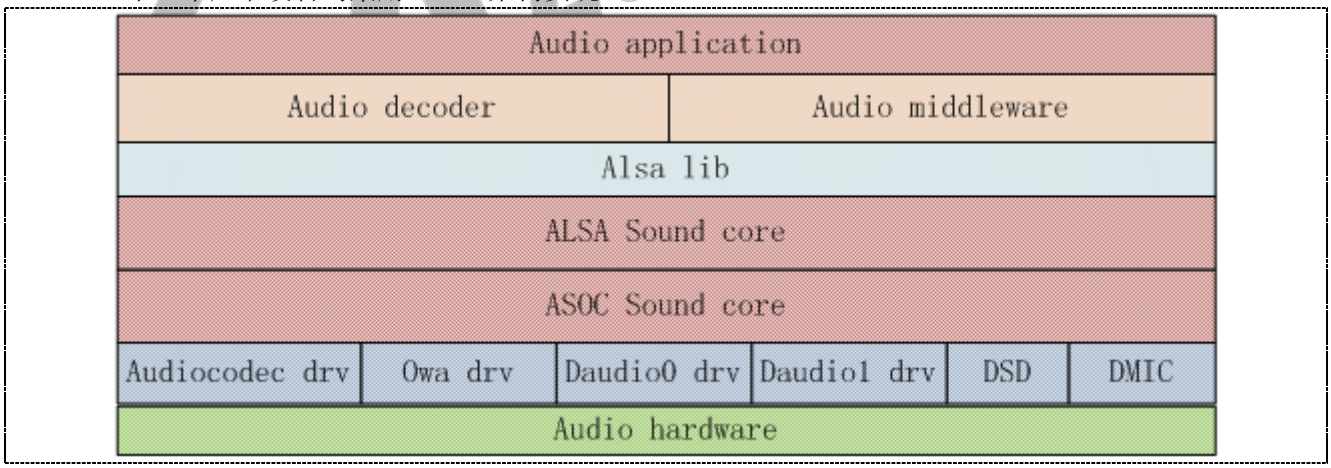


图 4 - 2 R40 音频软件框架

4.3. 时钟源

在 R40 中，存在 3 个音频模块（audiocodec，daudio0(i2s0)，daudio1(i2s1)），音频的时钟源都来自 pll2clk;

4.3.1. 音频时钟源信息

pll2clk 可以输出 24.576M 或者 22.5792M 的时钟，分别支持 48k 系列，44.1k 系列的播放录音；

audiocodec，daudio0，daudio1 内部都有自己的 mclk，bclk；audiocodec 的 mclk 也可以通过 PH0（跟 pwm 输出口共用）输出合适的 mclk 提供给外挂 codec 使用。

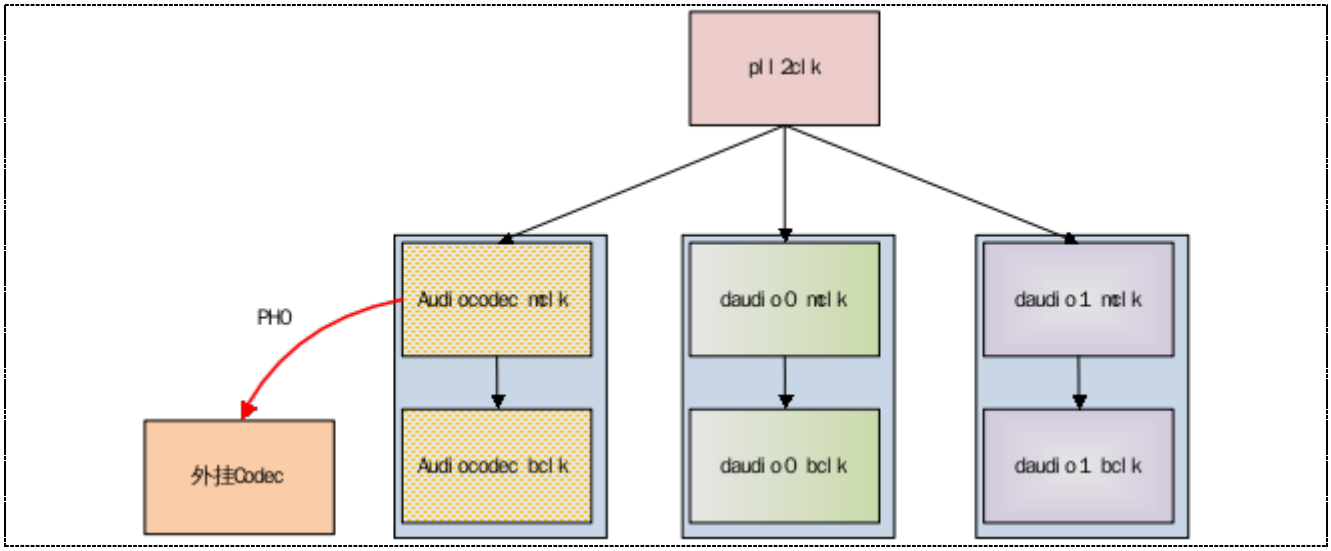


图 4 - 3 R40 音频时钟源

注：audiocodec 的 mclk 如果提供给外挂 codec 使用，需要注意 audiocodec 的 mclk 分频系数；

4.3.2. audiocodec mclk table

mclkdiv 支持 1，2，4，6，8，12，16，24，32，48，64；对应的 mclk 输出频率如下表所示：

pll2clk 频率	audiocodec mclk 频率	mclk_div
24.576M	24.576M	1
24.576M	12.288M	2
24.576M	6.144M	4
24.576M	4.096M	6
24.576M	3.072M	8
24.576M	2.048M	12
24.576M	1.536M	16
24.576M	1.024M	24
24.576M	768k	32
24.576M	512k	48
24.576M	384k	64
22.5792M	22.5792M	1
22.5792M	11.2896M	2
22.5792M	5.6448M	4
22.5792M	3.7632M	6
22.5792M	2.8224M	8
22.5792M	1.8816M	12
22.5792M	1.4112M	16
22.5792M	940.8k	24
22.5792M	705.6k	32
22.5792M	470.7k	48
22.5792M	352.8k	64

比如 audiocodec 输出 12.288M 的时钟给外挂的 codec5707 或者 apa3165 使用所设置的代码如下，具体寄存器值需要根据 datasheet 进行设置：

sun8iw11_sndcodec.c

```
static void codec_init(void)
{
......
#if defined CONFIG_SND_SOC_APA3165 || defined CONFIG_SND_SOC_TAS5707
```

```
        /*set mclk from audiocodec*/
        reg_val = readl(0xf1c22f60);//0x01c22c00+0x360
        reg_val &= ~0x7;
        reg_val |= (0x1<<2);
        writel(reg_val, 0xf1c22f60);

        /*set audiocodec mclk gpio config*/
        reg_val = readl(0xf1c208fc);
        reg_val &= ~(0x7);
        reg_val |= (0x3);
        writel(reg_val, 0xf1c208fc);

        /*enable global en*/
        writel(0x1, 0xf1c22c00);

        /*
        * 0x80: enable mclk and set mclk_div = 1; mclk = 24.576M/22.5792M
        * 0x81: enable mclk and set mclk_div = 2; mclk = 12.288M/11.2896M
        */
        writel(0x81, 0xf1c22c24);
#endif
}
```

4.4. 代码结构

```
tina/lichee/linux-3.10/sound/soc/sunxi $tree
.
├── built-in.o
├── codec_utils.c
├── codec-utils.h
├── codec_utils.o
├── cs4385.c
├── cs4385.h
├── Kconfig
├── Makefile
├── modules.builtin
├── modules.order
├── snddaudio.c
├── sndhdmi.c
├── snd-sunxi-soc.o
├── spdif-utils.c
├── sun50iw2_codec.c
├── sun50iw2_codec.h
├── sun50iw2_sndcodec.c
├── sun8iw10_codec.c
├── sun8iw10_codec.h
├── sun8iw10_sndcodec.c
├── sun8iw11_codec.c
├── sun8iw11_codec.h
├── sun8iw11_codec.o
├── sun8iw11_sndcodec.c
├── sun8iw11_sndcodec.o
├── sunxi_codec.c
├── sunxi_codec.h
├── sunxi_cpudai.c
```

```
├── sunxi_cpudai.h
├── sunxi_cpudai.o
├── sunxi_daudio.c
├── sunxi_daudio.h
├── sunxi_dma.c
├── sunxi_dma.h
├── sunxi_dma.o
├── sunxi_dmic.c
├── sunxi_dmic.h
├── sunxi_dsd.c
├── sunxi_dsd.h
├── sunxi_i2s.c
├── sunxi_i2s.h
├── sunxi_rw_func.c
├── sunxi_rw_func.h
├── sunxi_rw_func.o
├── sunxi_sndcodec.c
├── sunxi-snddaudio0.c
├── sunxi-snddaudio1.c
├── sunxi-snddmic.c
├── sunxi-snddsd.c
├── sunxi-sndhdmi.c
├── sunxi-sndspdif.c
├── sunxi_spdif.c
├── sunxi_spdif.h
├── sunxi_tdmhdmi.c
├── sunxi_tdm_utils.c
└── sunxi_tdm_utils.h
```

4.4.1. 公共部分

Platform（DMA 注册）：
sunxi_dma.c /*该文件内处理 DMA 部分，主要负责提供注册 platform 设备的公共函数*/

4.4.2. audiocodec

sun8iw11_codec.c /*该文件处理 audiocodec 部分，在 ASoC 中框架中设计为 codec 模型*/
sun8iw11_sndcodec.c /*该文件处理 audiocodec 部分,在 ASoC 中框架中设计为 machine 模型*/
sunxi_cpudai.c /*该文件 DMA 和内部声卡的数据传输，在 ASoC 中框架中设计为 cpu_dai 模型，其中 platform 也在此注册*/
sunxi_rw_func.c /*该文件处理 audiocodec 读写模拟寄存器部分*/

4.4.3. daudio

sunxi_daudio.c /*该文件处理 daudio 部分，在 ASoC 中框架中设计为 cpu_dai 模型，其中 platform 也在此注册*/
sunxi-snddaudio0.c /*该文件处理 daudio0 部分，在 ASoC 中框架中设计为 machine 模型*/
sunxi-snddaudio1.c /*该文件处理 daudio1 部分，在 ASoC 中框架中设计为 machine 模型*/

4.4.4. HDMI

sunxi_daudio.c /*该文件处理 daudio<for HDMI>部分，在 ASoC 中框架中设计为 cpu_dai 模型，其中 platform 也在此注册*/
sndhdmi.c /*该文件处理 HDMI 解码库接口设置部分，在 ASoC 中框架中设计为 codec 模型*/
sunxi-sndhdmi.c /*该文件处理 daudio1 部分，在 ASoC 中框架中设计为 machine 模型*/

4.4.5. S/PDIF

sunxi-sndspdif.c /*该文件处理 S/PDIF 部分，在 ASoC 中框架中设计为 machine 模型*/

sunxi_spdif.c /*该文件处理 S/PDIF 部分，在 ASoC 中框架中设计为 cpu_dai 模型，其中 platform 也在此注册*/

4.5. menuconfig 配置

在命令行中进入 Tina 根目录，执行命令进入配置主界面：

```
source build/envsetup.sh (详见①)
lunch                    (详见②)
make kernel_menuconfig   (详见③)
```

详注：

① 加载环境变量及 tina 提供的命令

② 输入编号，选择方案，注意，R40 对应方案为：azalea_m2ultra-tina

③ 进入内核配置主界面(对一个 shell 而言，前两个命令只需要执行一次)

配置路径：

```
Device Drivers
  └──>Sound card support
      └──>Advanced Linux Sound Architecture
          └──>ALSA for SoC audio support
```

配置说明 ：

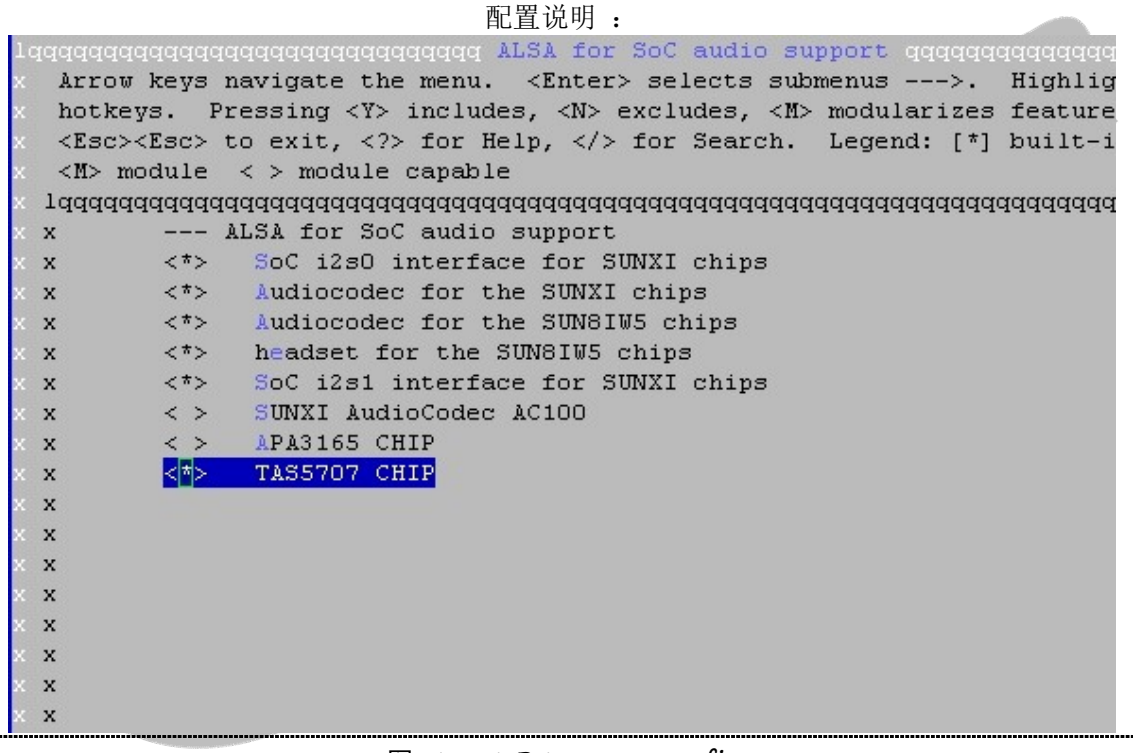


图 4 - 4 R40 menuconfig

声卡选项说明：

配置选项\声卡	audiocodec	i2s0	i2s1	i2s0-apa3165	i2s0-tas5707
Soc i2s0 interface for SUNXI chips		√		√	√
Audiocodec for the SUNXI chips	√				
Audiocodec for the SUN8IW5 chips	√				
headset for the SUN8IW5 chips	√(耳机检测)				
SoC i2s1 interface for SUNXI cihps			√		
SUNXI AudioCodec AC100					
APA3165 CIHP				√	
TAS5707					√

4.6. sys_config 配置

4.6.1. i2s0

```
[i2s0]
i2s0_used               = 0
i2s0_channel            = 2
i2s0_master             = 4
i2s0_select             = 1
audio_format            = 1
signal_inversion        = 1
over_sample_rate        = 512
sample_resolution       = 16
word_select_size        = 32
pcm_sync_period         = 256
msb_lsb_first           = 0
sign_extend             = 0
slot_index              = 0
slot_width              = 16
frame_width             = 1
tx_data_mode            = 1
rx_data_mode            = 1
i2s0_mclk               =
i2s0_bclk               = port:PB04<2><1><default><default>
i2s0_lrclk              = port:PB05<2><1><default><default>
i2s0_dout0              = port:PB06<2><1><default><default>
i2s0_dout1              =
i2s0_dout2              =
i2s0_dout3              =
i2s0_din                = port:PB07<2><1><default><default>
```

i2s 的配置需要熟悉 i2s、pcm 相关数字音频总线原理，一般对接 codec 的时候需要使用到；

i2s 配置	i2s 配置说明
i2s0_used	是否使用 i2s0 驱动 0：不使用；1：使用
i2s0_master	1：SND_SOC_DAIFMT_CBM_CFM(codec clk & FRM master) 表示 codec 做 master，i2s 做 slave 2：SND_SOC_DAIFMT_CBS_CFM(codec clk slave & FRM master) 3：SND_SOC_DAIFMT_CBM_CFS(codec clk master & frame slave) 4:SND_SOC_DAIFMT_CBS_CFS(codec clk & FRM slave) use 表示 codec 做 slave，i2s 做 master
i2s0_select	0 is pcm.1 is i2s 0 配置成 pcm 模式；1 配置成 i2s 模式
audio_format	1：SND_SOC_DAIFMT_I2S(standard i2s format). use 表示标准 i2s 格式 2：SND_SOC_DAIFMT_RIGHT_J(right justfied format). 表示右对齐格式 3：SND_SOC_DAIFMT_LEFT_J(left justfied format) 表示左对齐格式 4：SND_SOC_DAIFMT_DSP_A 表示 pcm 短帧模式 offset 为 2. 短帧模式需要设置 frame_width 为 1. 长帧模式，设置 frame_width 为 0 即可. 长帧模式 offset 不起作用。 5：SND_SOC_DAIFMT_DSP_B 表示 pcm 短帧模式 offset 为 1. 短帧模式需要设置 frame_width 为 1.
signal_inversion	1:SND_SOC_DAIFMT_NB_NF(normal bit clock + frame) 表示 bclk 采用正常模式，lrck 采用正常模式 2:SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM) 表示 bclk 采用正常模式，lrck 采用翻转模式 3:SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM) 表示 bclk 采用翻转模式，

	lrck 采用正常模式 4:SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM) 表示 bclk 采用翻转模式，lrck 采用翻转模式 信号的翻转，比如标准的 I2S 模式，如果 lrck 翻转是模式，那么用示波器测量，左右声道是跟标准 i2s 模式相反的。如果 bclk 是翻转模式，那么用示波器测量，BCLK 信号是翻转的。参考上面的标准 i2s 时序图。
over_sample_rate	support 128fs/192fs/256fs/384fs/512fs/768fs 过采样率选择。跟 mclk 有关。 比如在 48k 情况下，over_sample_rate 配置成 128fs，那么 mclk = 128*48000 = 6.144M。在 pll_audio = 24.576M 情况下，就可以算出 mclk_div = pll_audio/mclk = 4; bclk = 2* word_select_size*fs;bclk_div = mclk/bclk;
word_select_size	16, 24,32 数据 word 的宽度。word_select_size 必须大于等于采样精度，要不然会出现数据丢失。相当于 word_select_size 装载 sample_resolution 传输， 所以 word_select_size > sample_resolution;只对 i2s 有效；
sample_resolution	16bits/20bits/24bits 采样精度选择
pcm_sync_period	16/32/64/128/256 pcm 模式中一个 lrck 有多少个 BCLK。用于 pcm 模式中。比如采样率 8k（也即是 LRCK=8k），bclk 在 2.048M 情况下，pcm_sync_period 的大小：bclk = lrck * pcm_sync_period;pcm_sync_period = 2.048M/8k = 256;其它以此类推；并需要确保 channel*slot_width<=pcm_sync_period.
msb_lsb_first	0: msb first; 1: lsb first 对应每一个 slot，数据有效位从 lsb 算（数据的低位传开始）还是从 msb 算（数据的高位传开始）。
sign_extend	0: zero pending; 1: sign extend 数据扩展位是 0 还是最后一位
slot_index	slot index: 0: the 1st slot - 3: the 4th slot slot 的索引号。pcm 传输模式中，从第几个 slot 开始传输，查看 pcm short frame timing diagram 或者 pcm long frame timing diagram；
slot_width	8 bit width / 16 bit width slot 的宽度。比如 pcm 模式的短帧（short frame）传输模式，选择 16 slot。查看 PCM short frame timing diagram；pcm 模式的长帧传输模式中，选择 8 slot；只对 pcm 有效；
frame_width	0: long frame = 2 clock width; 1: short frame 长帧还是短帧。2 个 bclk 代表长帧，1 个 bclk 代表短帧（SND_SOC_DAIFMT_DSP_B 模式）。用于 pcm 模式的配置中，i2s 无效
tx_data_mode	0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law pcm 传输模式中选择数据的格式；注意：如果数据格式选择的是 u-law,a-law，而 slot_width 选择 16bit，但是 u-law，a-law 是 8bit，后续 8bit 用 0 填充。
rx_data_mode	0: 16bit linear PCM; 1: 8bit linear PCM; 2: 8bit u-law; 3: 8bit a-law pcm 传输模式中选择数据的格式

4.6.2. audio0

```
[audio0]
audio_used          = 1
audio_hp_ldo        = "none"
headphone_vol       = 0x3b
earpiece_vol        = 0x3e
cap_vol             = 0x5
pa_single_vol       = 0x3e
pa_double_used      = 0
pa_double_vol       = 0x3e
headphone_direct_used = 1
headset_mic_vol     = 3
main_mic_vol        = 1
mic1_used               = 1
mic2_used               = 0
```

```
linein_to_spk_used      = 0
linein_to_hp_used       = 0
linein_to_aif2_used     = 0
audio_pa_ctrl           = port:PD11<1><default><default><0>
audio_linein_ctrl       = port:PE14<0><default><default><0>
aif2_used               = 0
aif3_used               = 0
headphone_mute_used     = 0
DAC_VOL_CTRL_SPK= 0xa0a0
DAC_VOL_CTRL_HEADPHONE      = 0xa0a0
agc_used                = 0
drc_used                = 0
```

audiocodec 配置	audiocodec 配置说明
audio_used	是否使用 audiocodec 驱动；0：使用；1：不使用；
audio_hp_ldo	耳机电压配置，默认不配置；
headphone_vol	耳机音量大小(0x0~0x3f)
earpiece_vol	听筒音量大小(0x0~0x3f)
cap_vol	录音音量大小(0x0~0x7)
pa_single_vol	单喇叭音量大小(0x0~0x3f)
pa_double_used	是否使用双喇叭
pa_double_vol	双喇叭音量大小(0x0~0x3f)
headphone_direct_used	是否使用耳机直驱模式
headset_mic_vol	耳机 mic 音量大小(0x0~0x7)
main_mic_vol	主 mic 音量大小(0x0~0x7)
mic1_used	是否使用 mic1. 如果 mic1.mic2 同时使用，mic1，mic2 都使能； 如果只使用 mic1，那么只使能 mic1； 如果只使用 mic2，那么只使能 mic2；
mic2_used	是否使用 mic2.
linein_to_spk_used	是否开启 Linein 输入到 audiocodec 端的喇叭输出
linein_to_hp_used	是否开启 linein 输入到 audiocodec 端的耳机输出
linein_to_aif2_used	是否开启 linein 输入到 audiocodec 端的 aif2 输出； 一般用于 linein 输入到外挂 codec 喇叭输出情况； 音频播放从 i2s0 接外挂 codec 输出；linein 音频输入通过 aif2 接外挂的 codec。因为 i2s0 跟 aif2 共用 pin 脚；
audio_pa_ctrl	喇叭 PA gpio 输出控制口
audio_linein_ctrl	Linein gpio 检测控制口
aif2_used	是否使用 aif2；
aif3_used	是否使用 aif3；
headphone_mute_used	没有使用；
DAC_VOL_CTRL_SPK	播放 dac 喇叭输出数字端音量大小
DAC_VOL_CTRL_HEADPHONE	播放 dac 耳机输出数字端音量大小
agc_used	是否使用 agc;录音自动增益控制
drc_used	是否使用 drc;播放动态范围控制

4.7. 读写寄存器方法

注意：audiocodec 寄存器读写方式（建议模拟部分寄存器使用此方式，数字部分寄存器使用 sunxi_dump 的模式）：

4.7.1. audiocodec 模拟寄存器读写

4.7.1.1.方式 1

1．执行“cd /sys/bus/platform/devices/sunxi-pcm-codec/audio_reg_debug”
2．执行 “cat audio_reg”
注明：通过上述命令可以将所有模拟相关的寄存器打印出来

4.7.1.2.方式 2

1．执行“cd /sys/bus/platform/devices/sunxi-pcm-codec/audio_reg_debug”
2．执行 “echo 0,2,0x00> audio_reg” #0：表示该操作为读，2：表示操作类型为模拟部分寄存器，0x00 表示寄存器地址。
注明：使用第二种只能每次操作一个寄存器。
注明：读寄存器方式可以一次读出一个寄存器的值，也可以读出多个寄存器的值.

4.7.1.3.写寄存器步骤

1．执行“cd /sys/bus/platform/devices/sunxi-pcm-codec/audio_reg_debug”
2．执行 “echo 1,2,0x00,0xff > audio_reg” #1：表示该操作为写，2：表示操作类型为模拟部分寄存器，0x00：表示寄存器地址，0xff：表示要写入寄存器的值。
如果无法打印寄存器值：请将打印优先级调高；echo 8 > /proc/sys/kernel/printk

4.7.2. audiocodec 数字寄存器读写

audiocodec 寄存器读：echo 0xf1c22c00,0xf1c22f60 > dump;cat dump;
audiocodec 寄存器写（比如将 0xf 写入到 0xf1c22c00）：echo 0xf1c22c00 0xf > write;cat write;

4.7.3. daudio0/daudio1 寄存器读写

daudio 的寄存器需要通过 sunxi_dump 进行寄存器读写：

```
cd /sys/class/sunxi_dump
```

daudio0 寄存器读方式：

```
echo 0xf1c22000,0xf1c2203c > dump;cat dump;
```

daudio0 寄存器写方式（比如将 0x3c 写入到 0xf1c22000 中）：

```
echo 0xf1c22000 0x3c > write;cat write;
```

daudio1 寄存器读方式：

```
echo 0xf1c22400,0xf1c2243c > dump;cat dump;
```

daudio1 寄存器写方式（比如将 0x3c 写入到 0xf1c22000 中）：

```
echo 0xf1c22400 0x3c > write;cat write;
```

4.8. linein 接口检测

4.8.1. linein 检测实现方案

linein 检测中，有两种实现；linein 检测具体的实现需要根据硬件原理图进行，跟具体的 codec 无关；

4.8.1.1. linein 插拔检测以及 linein 信号检测

此模式下包含三种状态，以 tas5707 方案供参考；

```
STATUS_IN
STATUS_SIGNAL
STATUS_OUT
```

tas5707 挂载在 i2s0 上，tas5707 驱动中实现了 linein 检测，那么 linein 检测的 input event 事件的节点名称是 sndi2s0 sunxi linein Jack;

```
root@TinaLinux:/# cat /sys/class/input/input*/name
cat /sys/class/input/input*/name
```

```
axp22-supplyer
gpio-keys-polled
sunxi-keyboard
sndi2s0 sunxi linein Jack
audiocodec linein Jack
headset
sunxi-ths
```

4.8.1.2. 只有 linein 插拔检测

此模式下包含两种状态，以内部 audiocodec 方案为参考：

```
STATUS_IN
STATUS_OUT
```

内部 audiocodec 的 linein 检测，第三方 codec 驱动中没有实现 linein 检测，那么 linein 检测的 input event 事件的节点名称是 audiocodec linein Jack；

```
root@TinaLinux:/# cat /sys/class/input/input*/name
cat /sys/class/input/input*/name
axp22-supplyer
gpio-keys-polled
sunxi-keyboard
sndi2s0 sunxi linein Jack
audiocodec linein Jack
headset
sunxi-ths
```

4.8.2. 检测接口使用方法

4.8.2.1. 节点查看方式

（1） 支持 linein 插拔检测和 linein 信号检测，以 tas5707 方案实现 linein 检测做参考：

当 linein 拔出的时候，查阅 AUX_JACK_DETECT 为 N；

当 linein 插入的时候，查阅 AUX_JACK_DETECT 为 Y;

当 linein 中没有信号的时候，查阅 AUX_SIGNAL_DETECT 为 N；

当 linein 有信号的时候，查阅 AUX_SIGNAL_DETECT 为 Y;

如下所示：

```
root@TinaLinux:/sys/module/tas5707/parameters# cat AUX_S
cat AUX_SIGNAL_DETECT
N
root@TinaLinux:/sys/module/tas5707/parameters# cat AUX_J
cat AUX_JACK_DETECT
N
```

图 4 - 5 linein 插拔和信号检测

（2） 只支持 linein 检测，以内部 audiocodec 实现 linein 检测做参考：

当 linein 拔出的时候，查阅 AUX_JACK_DETECT 为 N；

当 linein 插入的时候，查阅 AUX_JACK_DETECT 为 Y;

如下所示：

```
root@TinaLinux:/sys/module/sun8iw5_sndcodec/parameters# cat A
cat AUX_JACK_DETECT
Y
root@TinaLinux:/sys/module/sun8iw5_sndcodec/parameters# cat A
cat AUX_JACK_DETECT
N
```

图 4 - 6 linein 插拔检测

4.8.2.2. input event 接口方式

通过 input event 接口查看插拔 linein 线上报的信息判断 linein 的插拔；

- 支持 linein 插拔检测和 linein 信号检测，以 tas5707 方案实现 linein 检测做参考：

```
拔出状态：128；插入状态（无信号）：32；插入有信号状态：64；
STATUS_OUT = 128,
STATUS_IN = 32,
STATUS_SIGNAL = 64,
```

- 支持 linein 插拔检测，以内部 audiocodec 方案实现 linein 检测做参考：

```
拔出状态：128；插入状态（无信号）：32；
STATUS_OUT = 128,
STATUS_IN = 32,
```

linein 检测 input event 接口 demo 参考./package/utils/test_r16_linein/src/linein_test.c；

注意 linein 检测实现的接口名称；

1.以 audiocodec 的 linein 检测为 demo：

```
#define PRESS_DATA_NAME       "audiocodec linein Jack"
```

2.以第三方功放驱动 tas5707 的 linein 检测为 demo：

```
#define PRESS_DATA_NAME       "sndi2s0 sunxi linein Jack"
```

4.8.3. linein 接口切换

检测到 linein 插入或者拔出后，需要调用 codec_set_lineinin 接口进行切换；

利用 amixer 接口调试命令如下所示:

```
root@TinaLinux:/sys/module/sun8iw5_sndcodec/parameters# amixer -c 1 cset iface=MIXER,nam
amixer -c 1 cset iface=M
IXER,name='Audio linein in' 1
numid=26,iface=MIXER,name='Audio linein in'
  ; type=BOOLEAN,access=rw------,values=1
  : values=on
root@TinaLinux:/sys/module/sun8iw5_sndcodec/parameters# amixer -c 1 cset iface=MIXER,nam
amixer -c 1 cset iface=M
IXER,name='Audio linein in' 0
numid=26,iface=MIXER,name='Audio linein in'
```

图 4 - 7 amixer 接口调试演示

如果需要封装成函数调用形式，可以参考 AudioManager\src\AudioManager.cpp 对 amixer 的接口封装。

检测到 linein 插入后，设置 Audio linein in 为 1，那么就可以打通 linein 到 spk 或者 linein 到 aif2（接第三方功放）的通路；

移植第三方功放驱动，可以参考 sun8iw5_sndcodec.c 中对 TAS5707 的接口实现；

```
#ifdef CONFIG_SND_SOC_TAS5707
        tas5707_linein_play(linein_plugin);
#endif
```

4.8.4. linein input event 接口 demo 代码示例

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <linux/input.h>
#include <stdlib.h>

#include <fcntl.h>
#include <errno.h>
#include <math.h>
```

```
#include <stdlib.h>
#include <poll.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/select.h>
#include <dlfcn.h>

#define PRESS_DATA_NAME       "audiocodec linein Jack"
#define PRESS_SYSFS_PATH      "/sys/class/input"

struct input_event buff;

int fd;

int linein_get_class_path(char *class_path)
{
    char dirname[] = PRESS_SYSFS_PATH;
    char buf[256];
    int res;
    DIR *dir;
    struct dirent *de;
    int fd = -1;
    int found = 0;

    dir = opendir(dirname);
    if (dir == NULL)
        return -1;

    while((de = readdir(dir))) {
        if (strncmp(de->d_name, "input", strlen("input")) != 0) {
            continue;
        }

        sprintf(class_path, "%s/%s", dirname, de->d_name);
        snprintf(buf, sizeof(buf), "%s/name", class_path);

        fd = open(buf, O_RDONLY);
        if (fd < 0) {
            continue;
        }
        if ((res = read(fd, buf, sizeof(buf))) < 0) {
            close(fd);
            continue;
        }
        buf[res - 1] = '\0';
        if (strcmp(buf, PRESS_DATA_NAME) == 0) {
            found = 1;
            close(fd);
            break;
        }

        close(fd);
        fd = -1;
    }
    closedir(dir);
    if (found) {
        return 0;
```

```
    }else {
        *class_path = '\0';
        return -1;
    }
}

int test_linein()
{
    int i = 0;
    int len = 0;
    char class_path[100];

    memset(class_path,0,sizeof(class_path));
    linein_get_class_path(class_path);
    len = strlen(class_path);
    printf("index: %c\n",class_path[len - 1]);

    sprintf(class_path, "/dev/input/event%c", class_path[len - 1]);
    printf("path: %s\n",class_path);

    fd = open(class_path, O_RDONLY); //may be the powerlinein is /dev/input/event1
    if (fd < 0) {
        perror("can not open device usblineinboard!");
        exit(1);
    }
    printf("--fd:%d--\n",fd);

    while(1)
    {
        while(read(fd,&buff,sizeof(struct input_event))==0)
        {
            ;
        }

        printf("%s,l:%d,d buff.code:%d, buff.value:%d\n", __FUNCTION__, __LINE__, buff.code, buff.value);
        if(buff.code != 0 && buff.value == 32) {
            printf("linein plug in\n");
        } else if (buff.code != 0 && buff.value == 64) {
            printf("linein signal\n");
        } else if (buff.code != 0 && buff.value == 128) {
            printf("linein plug out\n");
        }
    }

    close(fd);
    return 0;
}

int main()
{
    test_linein();
    return 0;
}
```

4.9. 耳机检测接口

耳机检测有在 tina 中有两种方式实现。1.uevent 机制。2.input event 机制；

4.9.1. uevent 机制

通过查看耳机驱动相关节点/sys/class/switch/h2w/state 可以判断当前耳机状态。

0：无耳机插入状态;
1：4 段耳机插入状态;
2：3 段耳机插入状态。

查看方式（以 R40 调试为例），监控/sys/class/switch/h2w/state 节点：

```
shell@octopus-f1:/ # cd /sys/class/switch/h2w/
shell@octopus-f1:/sys/class/switch/h2w # ls
name
power
state
subsystem
uevent
shell@octopus-f1:/sys/class/switch/h2w # cat state
0
```

4.9.2. input event 机制

耳机检测在原有 uevnet 的机制上增加了 input evnet 的接口扩展；不区分 3 节，4 节耳机；
相关接口可以参考 linein_test.c 的实现；

```
#define PRESS_DATA_NAME        "headset"
read(fd,&buff,sizeof(struct input_event)
```

耳机插入的时候：

```
(buff.code == 211 && buff.value == 1)
```

耳机拔出的时候：

```
(buff.code == 211 && buff.value == 0)
```

4.10.音频通路配置

R40 codec 驱动中所有通路的切换配置都需要通过操作控件来完成。以下为通过 tinymix（由 tinyalsa 提供）得到的 R40 codec 的所有控件：

```
# tinymix contents
Number of controls: 47
ctl  type    num   name                                value
0    INT     1     digital volume                      63
1    INT     1     Headphone volume                    59
2    INT     2     LINEIN Mixer volume                 3 3
3    INT     1     FM gain volume                      3
4    INT     1     LINEIN gain volume                  3
5    INT     2     MIC gain volume                     3 3
6    INT     1     phoneout volume                     3
7    INT     1     MIC1 boost volume                   4
8    INT     1     MIC2 boost volume                   4
9    INT     1     ADC gain volume                     3
10   ENUM    1     MIC2 Mux                            MIC2IN
11   ENUM    1     HPL Mux                             DAC
12   ENUM    1     HPR Mux                             DAC
13   BOOL    1     Phone Out Mixer LOMIX Switch        Off
14   BOOL    1     Phone Out Mixer ROMIX Switch        Off
15   BOOL    1     Phone Out Mixer MIC2 Boost Switch   Off
16   BOOL    1     Phone Out Mixer MIC1 Boost Switch   Off
17   BOOL    1     Right Input Mixer LOMIX Switch      Off
18   BOOL    1     Right Input Mixer ROMIX Switch      Off
19   BOOL    1     Right Input Mixer FMR Switch        Off
20   BOOL    1     Right Input Mixer LINEINR Switch    Off
21   BOOL    1     Right Input Mixer LINEINLR Switch   Off
22   BOOL    1     Right Input Mixer MIC2 Boost Switch Off
23   BOOL    1     Right Input Mixer MIC1 Boost Switch Off
24   BOOL    1     Left Input Mixer ROMIX Switch       Off
25   BOOL    1     Left Input Mixer LOMIX Switch       Off
26   BOOL    1     Left Input Mixer FML Switch         Off
27   BOOL    1     Left Input Mixer LINEINL Switch     Off
28   BOOL    1     Left Input Mixer LINEINLR Switch    Off
29   BOOL    1     Left Input Mixer MIC2 Boost Switch  Off
30   BOOL    1     Left Input Mixer MIC1 Boost Switch  Off
31   BOOL    1     Right Output Mixer DACL Switch      Off
32   BOOL    1     Right Output Mixer DACR Switch      Off
33   BOOL    1     Right Output Mixer FMR Switch       Off
34   BOOL    1     Right Output Mixer LINEINR Switch   Off
35   BOOL    1     Right Output Mixer LINEINLR Switch  Off
36   BOOL    1     Right Output Mixer MIC2 Boost Switch Off
37   BOOL    1     Right Output Mixer MIC1 Boost Switch Off
38   BOOL    1     Left Output Mixer DACR Switch       Off
39   BOOL    1     Left Output Mixer DACL Switch       Off
40   BOOL    1     Left Output Mixer FML Switch        Off
41   BOOL    1     Left Output Mixer LINEINL Switch    Off
42   BOOL    1     Left Output Mixer LINEINLR Switch   Off
43   BOOL    1     Left Output Mixer MIC2 Boost Switch Off
44   BOOL    1     Left Output Mixer MIC1 Boost Switch Off
45   BOOL    1     Headphone Switch                    Off
46   BOOL    1     Phoneout Speaker Switch             Off
```

以下为各情景的音频通路配置：

4.10.1. 耳机输出

DAC 输出经过 Output Mixer 再连接到耳机输出时的配置：

number	control_name	value
1	Headphone volume	0------63
2	Right Output Mixer DACR Switch	1
3	Left Output Mixer DACL Switch	1
4	HPR Mux	OMIX
5	HPL Mux	OMIX
6	Headphone Switch	1

DAC 输出不经过 Output Mixer 而直接连接到耳机输出的配置：

number	control_name	value
1	Headphone volume	0------63
2	HPR Mux	DAC
3	HPL Mux	DAC
4	Headphone Switch	1

4.10.2. PHONEOUT 输出

number	control_name	value
1	phoneout volume	0------7
2	Phone Out Mixer LOMIX Switch	1
3	Phone Out Mixer ROMIX Switch	1
4	Right Output Mixer DACR Switch	1
5	Left Output Mixer DACL Switch	1
6	Phoneout Speaker Switch	1

4.10.3. LINEIN 输入

number	control_name	value
1	Right Input Mixer LINEINR Switch	1
2	Left Input Mixer LINEINL Switch	1

4.10.4. MIC1 输入

number	control_name	value
1	MIC1 boost volume	0------7
2	Left Input Mixer MIC1 Boost Switch	1
3	Right Input Mixer MIC1 Boost Switch	1

4.10.5. MIC2 输入

number	control_name	value
1	MIC2 boost volume	0------7
2	Left Input Mixer MIC2 Boost Switch	1
3	Right Input Mixer MIC2 Boost Switch	1

4.10.6. MIC1 + MIC2 立体声输入

<MIC1 左声道，MIC2 右声道>

number	control_name	value
1	MIC1 boost volume	0-------7
2	MIC2 boost volume	0-------7
3	Right Input Mixer MIC2 Boost Switch	1

4	Left Input Mixer MIC1 Boost Switch	1

<MIC2 左声道，MIC1 右声道>

number	control_name	value
1	MIC1 boost volume	0-------7
2	MIC2 boost volume	0-------7
3	Right Input Mixer MIC1 Boost Switch	1
4	Left Input Mixer MIC2 Boost Switch	1

4.10.7. MIC1 到 HPOUT（耳机）通路

number	control_name	value
1	MIC1 boost volume	0-------7
2	Headphone volume	0-------63
3	Right Output Mixer MIC1 Boost Switch	1
4	Left Output Mixer MIC1 Boost Switch	1
5	HPL Mux	OMIX
6	HPR Mux	OMIX
7	Headphone Switch	1

5. R16 音频模块

5.1. 支持的特性

5.1.1. audiocodec

（1）支持多种采样率（8 kHz，11.025 kHz，12 kHz，16 kHz，22.05 kHz，24 kHz，32 kHz，44.1 kHz，48 kHz，96KHz，192KHz）；
（2）支持 mono 和 stereo 模式；
（3）支持同时 playback 和 record（全双工模式）；
（4）支持 3、4 段耳机插拔检测；
（5）支持 linein 插拔检测（通过 GPIO 检测）；
（6）播放录音采样精度支持 16bit、24bit；

5.1.2. I2S

（1）支持多种采样率(8 kHz，11.025 kHz，16 kHz，22.05 kHz，24 kHz，32 kHz，44.1 kHz，48 kHz，88.2 kHz，96 kHz，176.4 kHz，192 kHz)；
（2）支持 mono 和 stereo 模式；
（3）支持同时 playback 和 record（全双工模式）；
（4）支持 I2S、PCM 两种配置；
（5）播放录音采样精度支持 16bit、24bit；

5.2. sys_config 配置

```
[audio0]
audio_used              = 1
audio_hp_ldo                = "none"
headphone_vol               = 0x3b
pa_single_vol               = 0x3a
pa_double_used              = 1
pa_double_vol           = 0x3e
headphone_direct_used       = 1
headset_mic_vol             = 3
main_mic_vol            = 1
audio_lowlevel_detect       = 1
;audio_linein_detect        = port:PB07<0><default><default><0>
audio_pa_ctrl               = port:PD11<1><default><default><0>
aif2_used                   = 0
aif3_used                   = 0
headphone_mute_used   = 0
pa_gpio_reverse             = 0
agc_used                    = 1
drc_used                    = 0
aif1_lrlk_div               = 0x40

[codec_aif2]
aif2_lrlk_div = 0x20
aif2master = 1
aif2fmt = 1
aif2_bclk               = port:PB05<3><1><default><default>
aif2_lrclk                = port:PB04<3><1><default><default>
aif2_dout               = port:PB06<3><1><default><default>
aif2_din                = port:PB07<3><1><default><default>
```

```
[codec_aif3]
aif3fmt = 4
aif3_bclk          = port:PG11<3><1><default><default>
aif3_lrclk         = port:PG10<3><1><default><default>
aif3_dout          = port:PG12<3><1><default><default>
aif3_din           = port:PG13<3><1><default><default>
```

audio_used	内置声卡加载：1，不加载：0
headphone_vol	耳机音量调节（0 --- 63）
main_mic_vol	MIC1 录音音量调节（0--7）
headset_mic_vol	MIC2 录音音量调节（0--7）
headphone_direct_used	耳机电流直驱交驱配置，0：交驱；1：直驱
audio_pa_ctrl	模拟功放芯片的 pa 使能引脚配置
aif2_used	使用 AIF2
aif3_used	使用 AIF3
pa_gpio_reverse	功放芯片电平使能配置，1：低电平使能；0：高电平使能
aif1_lrlk_div	AIF1 bclk/lrck 的分频值，一般不需要修改
aif2_lrlk_div	AIF2 bclk/lrlk 的分频值。 假如 8k 采样率，256k bclk。 运算公式：256k/8k = 32，aif2_lrlk_div = 0x20 （一般 AIF2 可以外挂功放芯片，bclk & lrck 供给的话，参数为 16，32，64，128，256）
aif2master	1: codec 做 master，i2s 做 slave 4：codec 做 slave，i2s 作 master
aif2fmt	1: SND_SOC_DAIFMT_NB_NF(normal bit clock + frame) use 表示 bclk 采用正常模式，lrck 也正常模式 2：SND_SOC_DAIFMT_NB_IF(normal BCLK + inv FRM) 表示 bclk 采用正常模式，lrck 采用翻转模式 3：SND_SOC_DAIFMT_IB_NF(invert BCLK + nor FRM) use 表示 bclk 采用翻转模式，lrck 采用正常模式 4：SND_SOC_DAIFMT_IB_IF(invert BCLK + FRM) 表示 bclk 采用翻转模式，lrck 采用翻转模式 信号的翻转，比如标准的 I2S 模式，如果 lrck 是翻转模式，那么用示波器测量，左右声道是跟标准 i2s 模式相反的。如果 bclk 是翻转模式，那么用示波器测量，BCLK 信号是翻转的。参考上面的标准 i2s 时序图。
aif3fmt	同“aif2fmt”

5.3. menuconfig 配置

在 tina 根目录下输入 make kernel_menuconfig 进行声卡配置：

1、选择 Device Drivers：

```
r> selects submenus --->.  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M
h.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                [*] Patch physical to virtual translations at runtime
                    General setup  --->
                [*] Enable loadable module support  --->
                [*] Enable the block layer  --->
                    System Type  --->
                [ ] FIQ Mode Serial Debugger
                    Bus support  --->
                    Kernel Features  --->
                    Boot options  --->
                    CPU Power Management  --->
                    Floating point emulation  --->
                    Userspace binary formats  --->
                    Power management options  --->
                [*] Networking support  --->
                    Device Drivers  --->
                    File systems  --->
                    Kernel hacking  --->
                    Security options  --->
                -*- Cryptographic API  --->
                    Library routines  --->
                ---
                    Load an Alternate Configuration File
                    Save an Alternate Configuration File

                          <Select>    < Exit >    < Help >
```

图 5-1 R16 menuconfig（1）

2、选择 sound card support：

```
      ^(-)
      [*] Pulse-Width Modulation (PWM) Support  --->
      <*> Multimedia support  --->
          Graphics support  --->
      <*> Sound card support  --->
      [*] HID Devices  --->
      [*] USB support  --->
      <*> MMC/SD/SDIO card support  --->
      < > Sony MemoryStick card support (EXPERIMENTAL)  --->
      [*] LED Support  --->
      <*> Switch class support  --->
      [ ] Accessibility support  --->
      [*] Real Time Clock  --->
      [*] DMA Engine support  --->
      [ ] Auxiliary Display support  --->
      < > Userspace I/O drivers  --->
          Virtio drivers  --->
          Microsoft Hyper-V guest support  --->
      [*] Staging drivers  --->
          Common Clock Framework  --->
          Hardware Spinlock drivers  --->
      [ ] IOMMU Hardware Support  --->
          Remoteproc drivers (EXPERIMENTAL)  --->
          Rpmsg drivers (EXPERIMENTAL)  --->
      [ ] Virtualization drivers  --->
      [*] Generic Dynamic Voltage and Frequency Scaling (DVFS) support  --->
      < > Gator driver for DS-5
```

图 5 - 2 R16 menuconfig（2）

3、选择 Advanced Linux Sound Architecture：

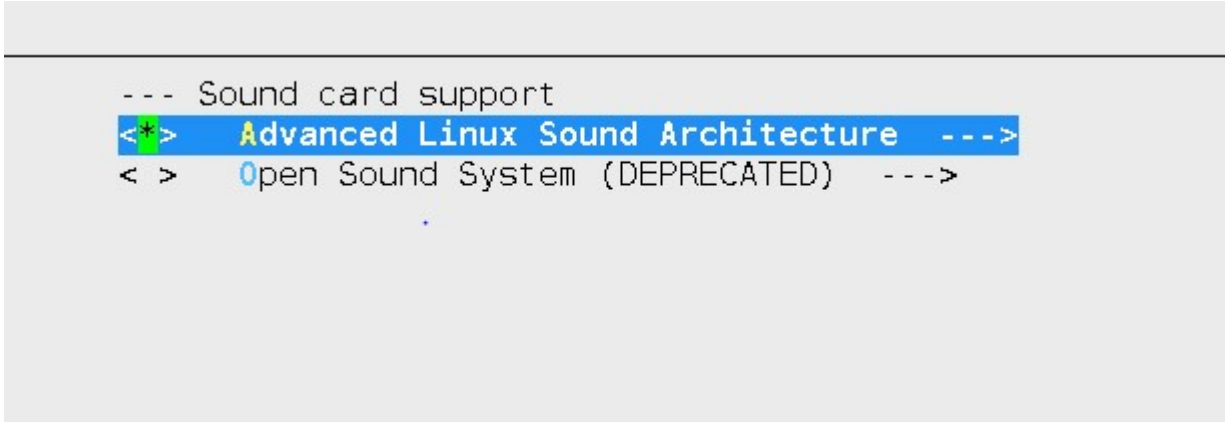


图 5 - 3 R16 menuconfig（3）

4、选择 ALSA for SoC audio support：

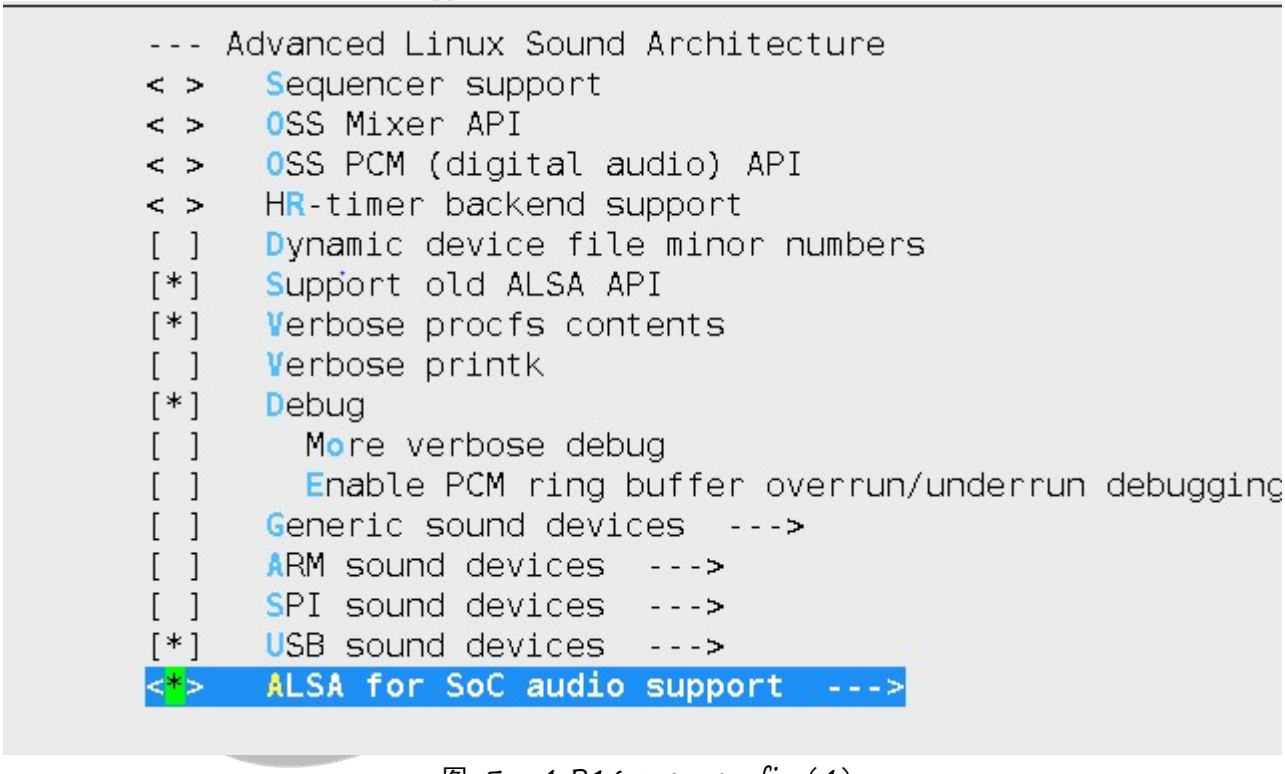


图 5 - 4 R16 menuconfig（4）

5、选择你想要的声卡设备：

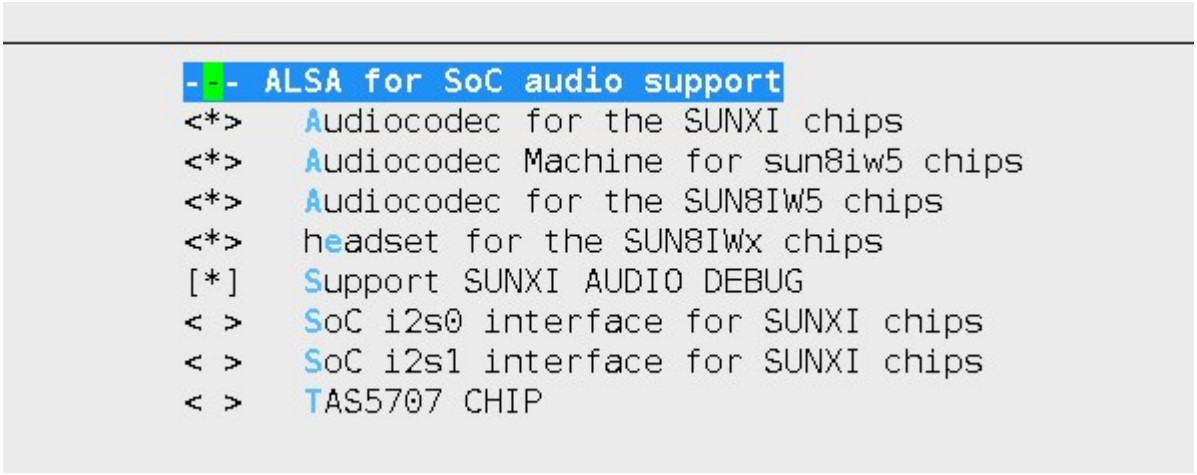


图 5 - 5 R16 menuconfig（5）

配置选项\声卡	audiocodec	I2s0	I2s1
Audiocodec for the SUNXI chips	√		
Audiocodec Machine for sun8iw5 chips	√		
Audiocodec for the SUN8IW5 chips	√		
headset for the SUN8IWx chips			
Support SUNXI AUDIO DEBUG			
SoC i2s0 interface for SUNXIchips		√	
SoC i2s1 interface for SUNXI chips			√
TAS5707 CHIP			

5.4. 音频通路配置

R16 audiocodec 的 mixer 控件（control）如下所示：

```
numid=105,iface=MIXER,name='Headphone Switch'
numid=106,iface=MIXER,name='Linein_detect Switch'
numid=74,iface=MIXER,name='Phoneout Mixer Left Output Mixer Switch'
numid=76,iface=MIXER,name='Phoneout Mixer MIC2Booststage Switch'
numid=75,iface=MIXER,name='Phoneout Mixer Right Output Mixer Switch'
numid=22,iface=MIXER,name='ADC input gain'
numid=18,iface=MIXER,name='ADC volume'
numid=25,iface=MIXER,name='ADCL Mux'
numid=24,iface=MIXER,name='ADCR Mux'
numid=51,iface=MIXER,name='AIF1 AD0L Mixer ADCL Switch'
numid=49,iface=MIXER,name='AIF1 AD0L Mixer AIF1 DA0L Switch'
numid=50,iface=MIXER,name='AIF1 AD0L Mixer AIF2 DACL Switch'
numid=52,iface=MIXER,name='AIF1 AD0L Mixer AIF2 DACR Switch'
numid=47,iface=MIXER,name='AIF1 AD0R Mixer ADCR Switch'
numid=45,iface=MIXER,name='AIF1 AD0R Mixer AIF1 DA0R Switch'
numid=48,iface=MIXER,name='AIF1 AD0R Mixer AIF2 DACL Switch'
numid=46,iface=MIXER,name='AIF1 AD0R Mixer AIF2 DACR Switch'
numid=44,iface=MIXER,name='AIF1 AD1L Mixer ADCL Switch'
numid=43,iface=MIXER,name='AIF1 AD1L Mixer AIF2 DACL Switch'
numid=42,iface=MIXER,name='AIF1 AD1R Mixer ADCR Switch'
numid=41,iface=MIXER,name='AIF1 AD1R Mixer AIF2 DACR Switch'
numid=13,iface=MIXER,name='AIF1 ADC timeslot 0 mixer gain'
numid=9,iface=MIXER,name='AIF1 ADC timeslot 0 volume'
```

```
numid=14,iface=MIXER,name='AIF1 ADC timeslot 1 mixer gain'
numid=10,iface=MIXER,name='AIF1 ADC timeslot 1 volume'
numid=11,iface=MIXER,name='AIF1 DAC timeslot 0 volume'
numid=12,iface=MIXER,name='AIF1 DAC timeslot 1 volume'
numid=104,iface=MIXER,name='AIF1IN0L Mux'
numid=103,iface=MIXER,name='AIF1IN0R Mux'
numid=102,iface=MIXER,name='AIF1IN1L Mux'
numid=101,iface=MIXER,name='AIF1IN1R Mux'
numid=56,iface=MIXER,name='AIF1OUT0L Mux'
numid=55,iface=MIXER,name='AIF1OUT0R Mux'
numid=54,iface=MIXER,name='AIF1OUT1L Mux'
numid=53,iface=MIXER,name='AIF1OUT1R Mux'
numid=17,iface=MIXER,name='AIF2 ADC mixer gain'
numid=15,iface=MIXER,name='AIF2 ADC volume'
numid=65,iface=MIXER,name='AIF2 ADL Mixer ADCL Switch'
numid=62,iface=MIXER,name='AIF2 ADL Mixer AIF1 DA0L Switch'
numid=63,iface=MIXER,name='AIF2 ADL Mixer AIF1 DA1L Switch'
numid=64,iface=MIXER,name='AIF2 ADL Mixer AIF2 DACR Switch'
numid=61,iface=MIXER,name='AIF2 ADR Mixer ADCR Switch'
numid=58,iface=MIXER,name='AIF2 ADR Mixer AIF1 DA0R Switch'
numid=59,iface=MIXER,name='AIF2 ADR Mixer AIF1 DA1R Switch'
numid=60,iface=MIXER,name='AIF2 ADR Mixer AIF2 DACL Switch'
numid=16,iface=MIXER,name='AIF2 DAC volume'
numid=69,iface=MIXER,name='AIF2INL Mux'
numid=71,iface=MIXER,name='AIF2INL Mux VIR switch aif2inl aif3'
numid=73,iface=MIXER,name='AIF2INL Mux switch aif2inl aif2'
numid=68,iface=MIXER,name='AIF2INR Mux'
numid=70,iface=MIXER,name='AIF2INR Mux VIR switch aif2inr aif3'
numid=72,iface=MIXER,name='AIF2INR Mux switch aif2inr aif2'
numid=67,iface=MIXER,name='AIF2OUTL Mux'
numid=66,iface=MIXER,name='AIF2OUTR Mux'
numid=57,iface=MIXER,name='AIF3OUT Mux'
numid=20,iface=MIXER,name='DAC mixer gain'
numid=19,iface=MIXER,name='DAC volume'
numid=97,iface=MIXER,name='DACL Mixer ADCL Switch'
numid=100,iface=MIXER,name='DACL Mixer AIF1DA0L Switch'
numid=99,iface=MIXER,name='DACL Mixer AIF1DA1L Switch'
numid=98,iface=MIXER,name='DACL Mixer AIF2DACL Switch'
numid=93,iface=MIXER,name='DACR Mixer ADCR Switch'
numid=96,iface=MIXER,name='DACR Mixer AIF1DA0R Switch'
numid=95,iface=MIXER,name='DACR Mixer AIF1DA1R Switch'
numid=94,iface=MIXER,name='DACR Mixer AIF2DACR Switch'
numid=77,iface=MIXER,name='HP_L Mux'
numid=78,iface=MIXER,name='HP_R Mux'
numid=38,iface=MIXER,name='LEFT ADC input Mixer LINEINL Switch'
numid=39,iface=MIXER,name='LEFT ADC input Mixer Lout_Mixer_Switch'
numid=34,iface=MIXER,name='LEFT ADC input Mixer MIC1 boost Switch'
numid=35,iface=MIXER,name='LEFT ADC input Mixer MIC2 boost Switch'
numid=37,iface=MIXER,name='LEFT ADC input Mixer PHONEN Switch'
numid=36,iface=MIXER,name='LEFT ADC input Mixer PHONEP-PHONEN Switch'
numid=40,iface=MIXER,name='LEFT ADC input Mixer Rout_Mixer_Switch'
numid=6,iface=MIXER,name='LINEINL/R to L_R output mixer gain'
numid=87,iface=MIXER,name='Left Output Mixer DACL Switch'
numid=86,iface=MIXER,name='Left Output Mixer DACR Switch'
numid=88,iface=MIXER,name='Left Output Mixer LINEINL Switch'
numid=92,iface=MIXER,name='Left Output Mixer MIC1Booststage Switch'
numid=91,iface=MIXER,name='Left Output Mixer MIC2Booststage Switch'
```

```
numid=89,iface=MIXER,name='Left Output Mixer PHONEN Switch'
numid=90,iface=MIXER,name='Left Output Mixer PHONEP-PHONEN Switch'
numid=4,iface=MIXER,name='MIC1 boost amplifier gain'
numid=26,iface=MIXER,name='MIC2 SRC'
numid=5,iface=MIXER,name='MIC2 boost amplifier gain'
numid=31,iface=MIXER,name='RIGHT ADC input Mixer LINEINR Switch'
numid=33,iface=MIXER,name='RIGHT ADC input Mixer Lout_Mixer_Switch'
numid=27,iface=MIXER,name='RIGHT ADC input Mixer MIC1 boost Switch'
numid=28,iface=MIXER,name='RIGHT ADC input Mixer MIC2 boost Switch'
numid=30,iface=MIXER,name='RIGHT ADC input Mixer PHONEP Switch'
numid=29,iface=MIXER,name='RIGHT ADC input Mixer PHONEP-PHONEN Switch'
numid=32,iface=MIXER,name='RIGHT ADC input Mixer Rout_Mixer_Switch'
numid=79,iface=MIXER,name='Right Output Mixer DACL Switch'
numid=80,iface=MIXER,name='Right Output Mixer DACR Switch'
numid=83,iface=MIXER,name='Right Output Mixer LINEINR Switch'
numid=85,iface=MIXER,name='Right Output Mixer MIC1Booststage Switch'
numid=84,iface=MIXER,name='Right Output Mixer MIC2Booststage Switch'
numid=82,iface=MIXER,name='Right Output Mixer PHONEP Switch'
numid=81,iface=MIXER,name='Right Output Mixer PHONEP-PHONEN Switch'
numid=23,iface=MIXER,name='SRC FUCTION'
numid=21,iface=MIXER,name='digital volume'
numid=1,iface=MIXER,name='headphone volume'
numid=2,iface=MIXER,name='lineout volume'
numid=7,iface=MIXER,name='phonein pre-amplifier gain'
numid=8,iface=MIXER,name='phonein(p-n) to L_R output mixer gain'
numid=3,iface=MIXER,name='phoneout volume'
```

5.4.1. headphone 播放

不经过 Left/Right Output Mixer：

control_name	value
headphone volume	0 --- 63
DACL Mixer AIF1DA0L Switch	1
DACR Mixer AIF1DA0R Switch	1
HP_L Mux	0
HP_R Mux	0
Headphone Switch	1

经过 Left/Right Output Mixer：

control_name	value
headphone volume	0 --- 63
DACL Mixer AIF1DA0L Switch	1
DACR Mixer AIF1DA0R Switch	1
Left Output Mixer DACL Switch	1
Right Output Mixer DACR Switch	1
HP_L Mux	1
HP_R Mux	1
Headphone Switch	1

5.4.2. speaker 播放

不经过 Left/Right Output Mixer：

control_name	value
headphone volume	0 --- 63
DACL Mixer AIF1DA0L Switch	1
DACR Mixer AIF1DA0R Switch	1

HP_L Mux	0
HP_R Mux	0
External Speaker Switch	1

经过 Left/Right Output Mixer：

control_name	value
headphone volume	0 --- 63
DACL Mixer AIF1DA0L Switch	1
DACR Mixer AIF1DA0R Switch	1
Left Output Mixer DACL Switch	1
Right Output Mixer DACR Switch	1
HP_L Mux	1
HP_R Mux	1
External Speaker Switch	1

5.4.3. MIC1 录音

control_name	value
MIC1 boost amplifier gain	1 --- 7
LEFT ADC input Mixer MIC1 boost Switch	1
RIGHT ADC input Mixer MIC1 boost Switch	1
AIF1 AD0L Mixer ADCL Switch	1
AIF1 AD0R Mixer ADCR Switch	1

5.4.4. MIC1 到 headphone

control_name	value
Left Output Mixer MIC1Booststage Switch	1
Right Output Mixer MIC1Booststage Switch	1
HP_R Mux	1
HP_L Mux	1
Headphone Switch	1

5.4.5. LINEIN 到 headphone

control_name	value
Left Output Mixer LINEINL Switch	1
Right Output Mixer LINEINR Switch	1
HP_R Mux	1
HP_L Mux	1
Headphone Switch	1

5.5. 案例：蓝牙语音

5.5.1. 案例信息

蓝牙模组支持格式为声道数（1），采样精度（16bit），采样路（8k），格式 4），主从关系（从），bclk（256kM），短帧。

注意：目前我们的 aif2/aif3 只支持短帧模式,并且 aif3 只能做主。

5.5.2. 配置

（1）配置 sys_config 参数：

```
[codec_aif2]
aif2_lrlk_div = 0x20
aif2master = 1
aif2fmt = 1
```

```
aif2_bclk                = port:PB05<3><1><default><default>
aif2_lrclk                 = port:PB04<3><1><default><default>
aif2_dout                = port:PB06<3><1><default><default>
aif2_din                 = port:PB07<3><1><default><default>

[codec_aif3]
aif3fmt = 4
aif3_bclk                = port:PG11<3><1><default><default>
aif3_lrclk               = port:PG10<3><1><default><default>
aif3_dout                = port:PG12<3><1><default><default>
aif3_din                 = port:PG13<3><1><default><default>
```

（2）配置 audiocodec 音频通路，具体配置如下：

MIC1 至 AIF3OUT 通路设置：

MIC1 boost amplifier gain	0 -- 7
LEFT ADC input Mixer MIC1 boost Switch	1
RIGHT ADC input Mixer MIC1 boost Switch	1
AIF2 ADR Mixer ADCR Switch	1
AIF2 ADL Mixer ADCL Switch	1
AIF3OUT Mux	2 (AIF2 ADC right channel)

AIF3IN 至 headphone 输出：

AIF2INR Mux VIR switch aif2inr aif3	1
DACR Mixer AIF2DACR Switch	1
DACL Mixer AIF2DACL Switch	1
Headphone Switch	1

寄存器配置：

```
SUNXI_DA_CTL 0xf1c22c00: 0x1
SUNXI_DA_FAT0 0xf1c22c04: 0xc
SUNXI_DA_FAT1 0xf1c22c08: 0x4020
SUNXI_DA_TXFIFO 0xf1c22c0c: 0x0
SUNXI_DA_RXFIFO 0xf1c22c10: 0x0
SUNXI_DA_FCTL 0xf1c22c14: 0x400f0
SUNXI_DA_FSTA 0xf1c22c18: 0x10800000
SUNXI_DA_INT 0xf1c22c1c: 0x0
SUNXI_DA_ISTA 0xf1c22c20: 0x10
SUNXI_DA_CLKD 0xf1c22c24: 0x0
SUNXI_DA_TXCNT 0xf1c22c28: 0x0
SUNXI_DA_RXCNT 0xf1c22c2c: 0x0
SUNXI_DA_TXCHSEL 0xf1c22c30: 0x1
SUNXI_DA_TXCHMAP 0xf1c22c34: 0x76543210
SUNXI_DA_RXCHSEL 0xf1c22c38: 0x1
SUNXI_DA_RXCHMAP 0xf1c22c3c: 0x3210
SUNXI_CHIP_AUDIO_RST 0xf1c22e00: 0x101
SUNXI_SYSCLK_CTL 0xf1c22e0c: 0x3b9
SUNXI_MOD_CLK_ENA 0xf1c22e10: 0x600c
SUNXI_MOD_RST_CTL 0xf1c22e14: 0x600c
SUNXI_SYS_SR_CTRL 0xf1c22e18: 0x0
SUNXI_SYS_SRC_CLK 0xf1c22e1c: 0x0
SUNXI_AIF1_CLK_CTRL 0xf1c22e40: 0x0
SUNXI_AIF1_ADCDAT_CTRL 0xf1c22e44: 0x0
SUNXI_AIF1_DACDAT_CTRL 0xf1c22e48: 0x0
SUNXI_AIF1_MXR_SRC 0xf1c22e4c: 0x0
SUNXI_AIF1_VOL_CTRL1 0xf1c22e50: 0xa0a0
SUNXI_AIF1_VOL_CTRL2 0xf1c22e54: 0xa0a0
```

```
SUNXI_AIF1_VOL_CTRL3 0xf1c22e58: 0xa0a0
SUNXI_AIF1_VOL_CTRL4 0xf1c22e5c: 0xa0a0
SUNXI_AIF1_MXR_GAIN 0xf1c22e60: 0x0
SUNXI_AIF1_RXD_CTRL 0xf1c22e64: 0x800
SUNXI_AIF2_CLK_CTRL 0xf1c22e80: 0x165e
SUNXI_AIF2_ADCDAT_CTRL 0xf1c22e84: 0xc000
SUNXI_AIF2_DACDAT_CTRL 0xf1c22e88: 0x0
SUNXI_AIF2_MXR_SRC 0xf1c22e8c: 0x1100
SUNXI_AIF2_VOL_CTRL1 0xf1c22e90: 0xa0a0
SUNXI_AIF2_VOL_CTRL2 0xf1c22e98: 0xa0a0
SUNXI_AIF2_MXR_GAIN 0xf1c22ea0: 0x0
SUNXI_AIF2_RXD_CTRL 0xf1c22ea4: 0x0
SUNXI_AIF3_CLK_CTRL 0xf1c22ec0: 0x6011
SUNXI_AIF3_ADCDAT_CTRL 0xf1c22ec4: 0x0
SUNXI_AIF3_DACDAT_CTRL 0xf1c22ec8: 0x0
SUNXI_AIF3_SGP_CTRL 0xf1c22ecc: 0xa00
SUNXI_AIF3_RXD_CTRL 0xf1c22ee4: 0x0
SUNXI  ADC  DIG  CTRL 0xf1c22f00: 0x8000
SUNXI_ADC_VOL_CTRL 0xf1c22f04: 0xa0a0
SUNXI_ADC_DBG_CTRL 0xf1c22f08: 0x0
SUNXI_HMIC_CTRL1 0xf1c22f0c: 0x0
SUNXI_HMIC_CTRL2 0xf1c22f14: 0x0
SUNXI_HMIC_STS 0xf1c22f18: 0x0
SUNXI_DAC_DIG_CTRL 0xf1c22f20: 0x8000
SUNXI_DAC_VOL_CTRL 0xf1c22f24: 0xa0a0
SUNXI_DAC_DBG_CTRL 0xf1c22f2c: 0x0
SUNXI_DAC_MXR_SRC 0xf1c22f30: 0x2200
SUNXI_DAC_MXR_GAIN 0xf1c22f34: 0x0
SUNXI_AC_DAPHHPFC 0xf1c2304c: 0xff
SUNXI_AC_DAPLHPFC 0xf1c23050: 0xfac1
SUNXI_AC_DAPOPT 0xf1c23054: 0x0
SUNXI_AGC_ENA 0xf1c230d0: 0x0
SUNXI_DRC_ENA 0xf1c230d4: 0x0
SUNXI_SRC_BISTCR 0xf1c230d8: 0x0
SUNXI_SRC_BISTST 0xf1c230dc: 0x202
SUNXI_SRC1_CTRL1 0xf1c230e0: 0x0
SUNXI_SRC1_CTROL2 0xf1c230e4: 0x0
SUNXI_SRC1_CTRL3 0xf1c230e8: 0x40
SUNXI_SRC1_CTRL4 0xf1c230ec: 0x0
SUNXI_SRC2_CTRL1 0xf1c230f0: 0x0
SUNXI_SRC2_CTRL2 0xf1c230f4: 0x0
SUNXI_SRC2_CTRL3 0xf1c230f8: 0x40
SUNXI_SRC2_CTRL4 0xf1c230fc: 0x0
HP_VOLC 0x0: 0x3b
LOMIXSC 0x1: 0x0
ROMIXSC 0x2: 0x0
DAC_PA_SRC 0x3: 0x8c
PHONEIN_GCTRL 0x4: 0x33
LINEIN_GCTRL 0x5: 0x33
MICIN_GCTRL 0x6: 0x33
PAEN  HP  CTRL 0x7: 0x84
PHONEOUT_CTRL 0x8: 0x60
LINEOUT_VOLC 0x9: 0x4
MIC2G_LINEEN_CTRL 0xa: 0x0
MIC1G_MICBIAS_CTRL 0xb: 0xfd
LADCMIXSC 0xc: 0x40
RADCMIXSC 0xd: 0x40
```

```
ADC_AP_EN 0xf: 0xc3
ADDA_APT0 0x10: 0x55
ADDA_APT1 0x11: 0x55
ADDA_APT2 0x12: 0x42
BIAS_AD16_CAL_CTRL 0x13: 0xd6
BIAS_DA16_CAL_CTRL 0x14: 0x0
DA16CALI 0x15: 0x66
DA16VERIFY 0x16: 0x80
BIASCALI 0x17: 0x1f
BIASVERIFY 0x18: 0x20
```

（3）打开相关的接口设备

```
struct pcm_config config = {
        .channels = 1,
        .rate = SAMPLING_RATE_8K,
        .period_size = 160,
        .period_count = 2,
        .format = PCM_FORMAT_S16_LE,
};

Play_sample (file,0,2,1,8000,16 ,period,period_count);
        pcm = pcm_open(card, 1, PCM_OUT, &config);
        if (!pcm || !pcm_is_ready(pcm)) {
                fprintf(stderr, "Unable to open PCM device %u (%s)\n",
                                device, pcm_get_error(pcm));
                return;
        }

        pcm1 = pcm_open(card, 2, PCM_OUT, &config);
        if (!pcm1 || !pcm_is_ready(pcm1)) {
                fprintf(stderr, "Unable to open PCM device %u (%s)\n",
                                device, pcm_get_error(pcm1));
                return;
        }

        pcm2 = pcm_open(card, 1, PCM_IN, &config);
        if (!pcm || !pcm_is_ready(pcm2)) {
                fprintf(stderr, "Unable to open PCM device %u (%s)\n",
                                device, pcm_get_error(pcm));
                return;
        }

        pcm3 = pcm_open(card, 2, PCM_IN, &config);
        if (!pcm1 || !pcm_is_ready(pcm3)) {
                fprintf(stderr, "Unable to open PCM device %u (%s)\n",
                                device, pcm_get_error(pcm1));
                return;
        }


        pcm_start(pcm);
        pcm_start(pcm1);
        pcm_start(pcm2);
        pcm_start(pcm3);
```

6. R6 音频模块

6.1. 硬件框架

在 R6 中，存在 2 个音频设备，音频基本硬件框架图如下所示：

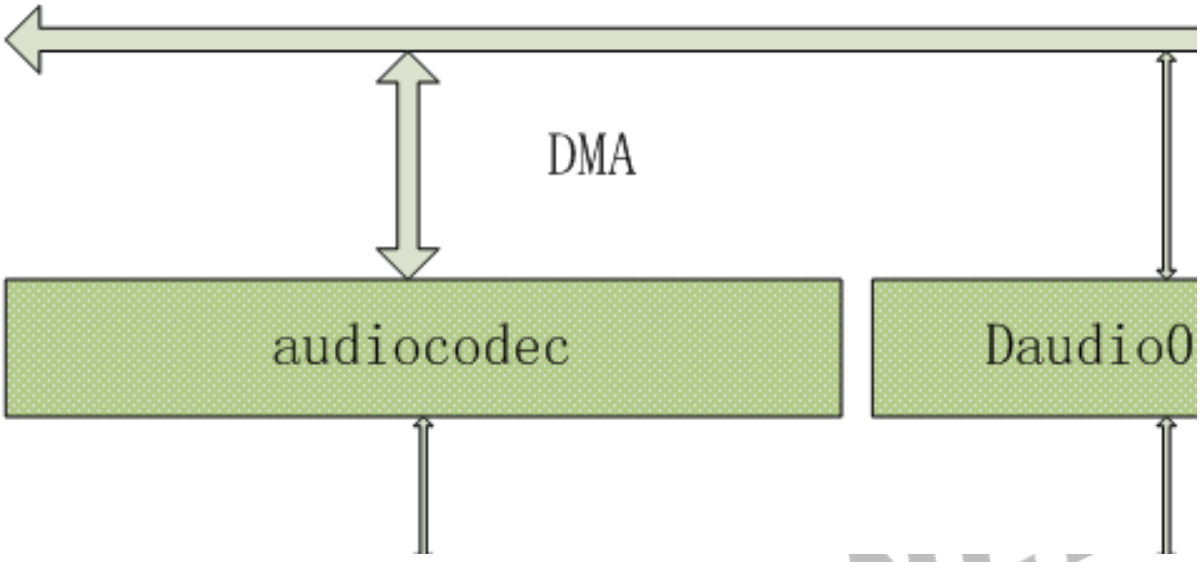


图 6 - 1 R6 音频硬件框架图

6.2. 软件框架

R6 的 2 个音频设备均采用 ASoC 框架实现，软件结构图如下所示：

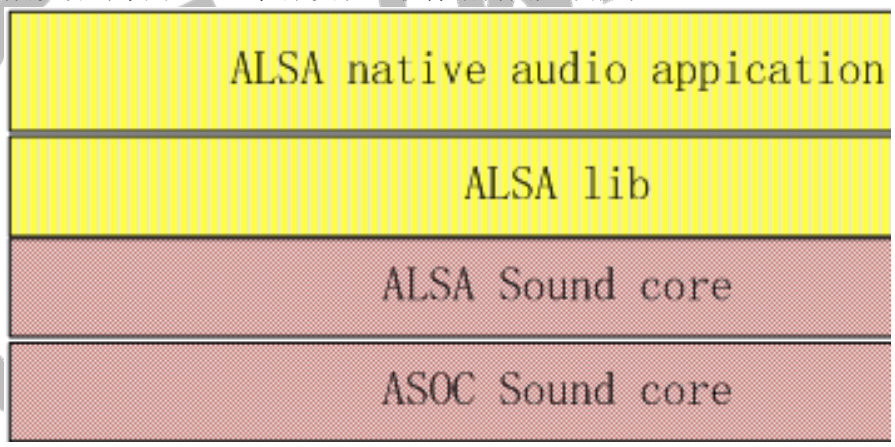


图 6 - 2 R6 音频软件框架图

6.3. 代码结构

```
└── sunxi
    ├── sun3iw1_codec.c
    ├── Kconfig
    ├── Makefile
    ├── sunxi_cpudai.c
    ├── sun3iw1_sndcodec.c
    ├── daudio0
    ├── sun3iw1_daudio.c
    ├── sunxi-snddaudio.c
    └── nau8520.c
```

audiocodec 的代码目录在 sunxi 下面；
（1）codec 部分：sun3iw1_codec.c
（2）platform 部分：sunxi_cpudai.c
（3）machine 部分：sun3iw1_sndcodec.c
daudio0 的代码目录在 sunxi 下面；
（1）codec 部分：nau8520.c
（2）platform 部分：sun3iw1_daudio.c
（3）machine 部分：sunxi-snddaudio.c

6.4. menuconfig 配置

在 tina 根目录下执行 make kernel_menuconfig 进入配置项：
Device Drivers ---> <*> Sound card support ---> <*> Advanced Linux Sound Architecture ---> <*> ALSA for SoC audio support ---> ASoC support for SUNXI

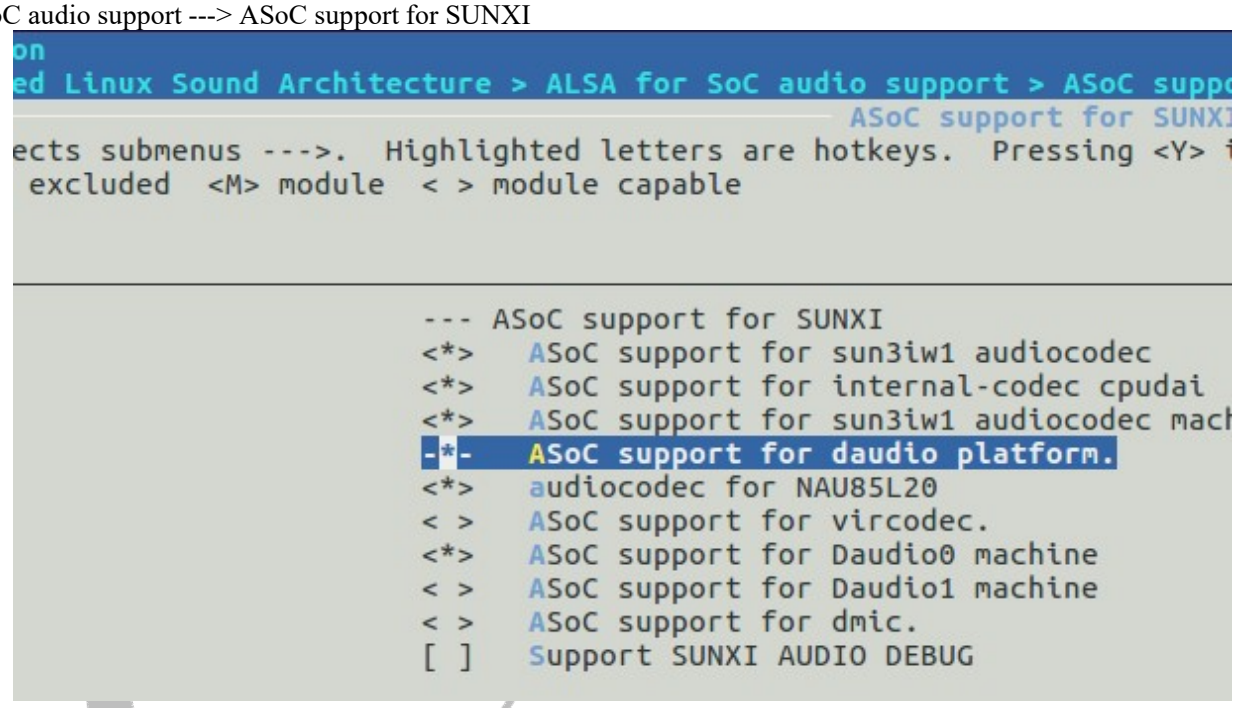


图 6 - 3 R6 menuconfig 配置

配置选项\声卡	audiocodec	daudio0
Asoc support for sun3iw1 audiocodec	*	
Asoc support for internal-codec cpudai	*	
AsoC support for sun3iw1 audio codec machine	*	
Asoc support for daudio platform		*
Audiocodec for NAU85L20		*
AsoC support for Daudio0 machine		*

6.5. 音频通路配置

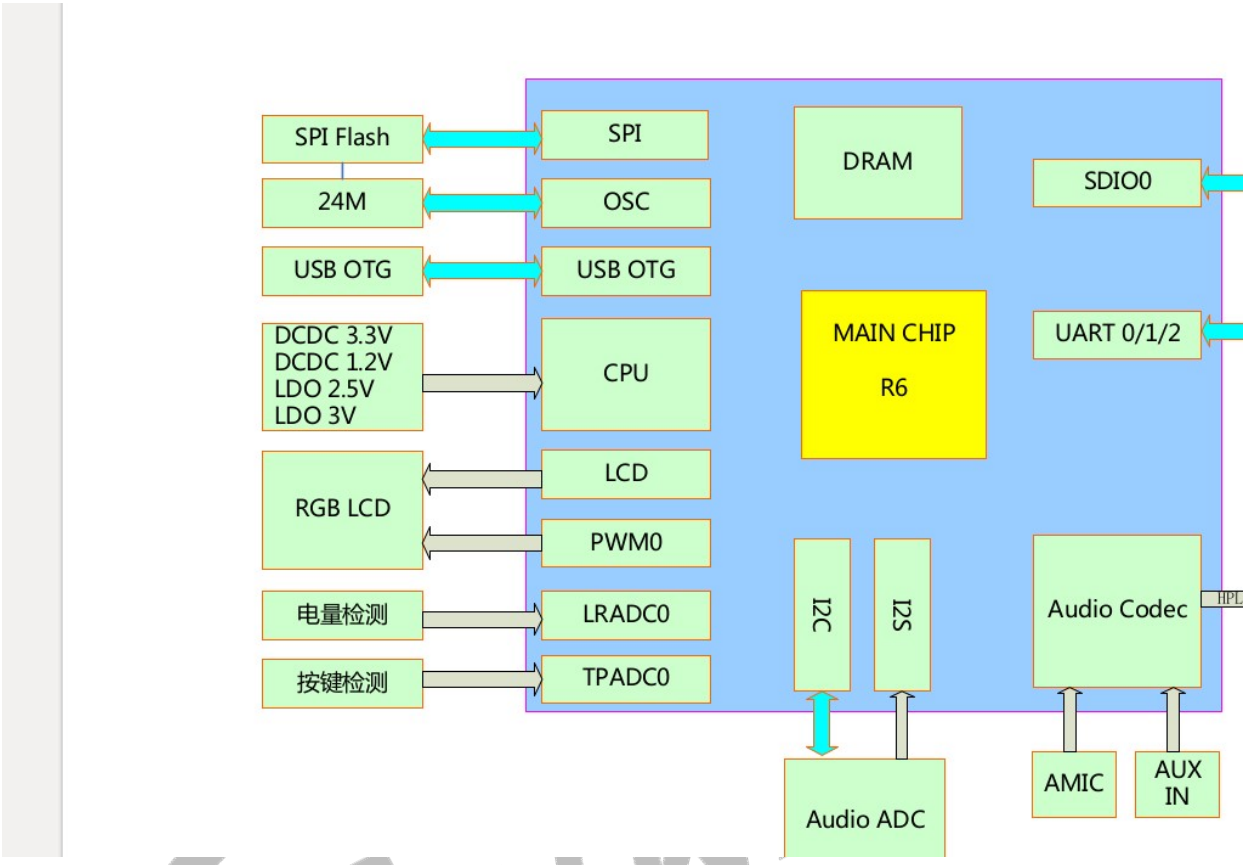


图 6 - 4 R6 音频通路示意图

如上图所示，R6 上有两个音频 codec,一个为内置，即图中的 Audio Codec。另外一个为外挂，即图中的 Audio ADC。内置 Audio Codec 可以用来录制和播放，外挂 codec 只能用来录制。

内置的 Audio Codec 可输出音频到 HPL/HPR，此处与耳机（插孔为 J1）及喇叭（SP1、SP2）相连，其中 SP1 仅与 HPL 相连，而 SP2 与 HPL 和 HPR 均相连；另一方面，音频可通过 MIC1 或 LINEIN（即图中的 AUX IN，对应插孔为 J2）输入给内置的 Audio Codec。

外挂的 Audio ADC 支持两路麦克风（MIC2 和 MIC3）输入，其中 MIC3 与喇叭 SP2 的输出相连而形成回路。

6.5.1. 内置 codec 音频通路设置

内置 codec 为 card0，amixer 默认配置的为 card0，因此配置控件时不需要特意指定 card 的编号。

6.5.1.1. HPL/HPR（耳机&喇叭）输出

耳机和喇叭均与 HPL/HPR 相连，使用它们播放声音时所需要设置的控件是相同的。（使用喇叭播放时必须拔掉耳机，否则声音会从耳机输出而不经过喇叭）

control_name	value
head phone volume	0 --- 63
head phone power	1

6.5.1.2. MIC1 输入

control_name	value
MICIN GAIN control	0 --- 7
ADC MIC Boost AMP en	1

ADC mixer mute for mic	1

6.5.1.3.MIC1 到 HPL/HPR 通路

control_name	value
ADC MIC Boost AMP en	0 --- 7
dac: left analog output mixer en	1
dac: right analog output mixer en	1
dac: left mute	1
dac: right mute	1
dac: left mixer mute: mic	1
dac: right mixer mute: mic	1
hp left source select: 0-dac, 1-mixer	1
hp right source select: 0-dac, 1-mixer	1
SPK_L Mux Left Mixer en	1
SPK_R Mux Right Mixer en	1
head phone power	1

6.5.1.4.LINEIN 到 HPL/HPR 通路

control_name	value
dac: left analog output mixer en	0 --- 7
dac: right analog output mixer en	1
dac: left mute	1
dac: right mute	1
dac: right mixer mute: FM	1
dac: left mixer mute: FM	1
hp left source select: 0-dac, 1-mixer	1
hp right source select: 0-dac, 1-mixer	1
SPK_L Mux Left Mixer en	1
SPK_R Mux Right Mixer en	1
head phone power	1

6.5.2. 外挂 codec 音频通路配置

外挂 codec 为 card1，使用 amixer 配置控件时需要添加指定 card 编号的参数：“-c 1”

6.5.2.1.MIC2 输入

control_name	value
Mic1 Volume	0-1312
Frontend PGA1 Volume	0-37
Digital CH1 Mux	0 (ADC channel 1)

6.5.2.2.MIC3 输入

control_name	value
Mic4 Volume	0-1312
Frontend PGA4 Volume	0-37
Digital CH2 Mux	3 (ADC channel 4)

6.6. 音频通路测试

以下介绍音频通路的测试方法，以 ALSA 提供的工具 amixer、arecord、aplay 为例。

6.6.1. 内置 codec 功能测试

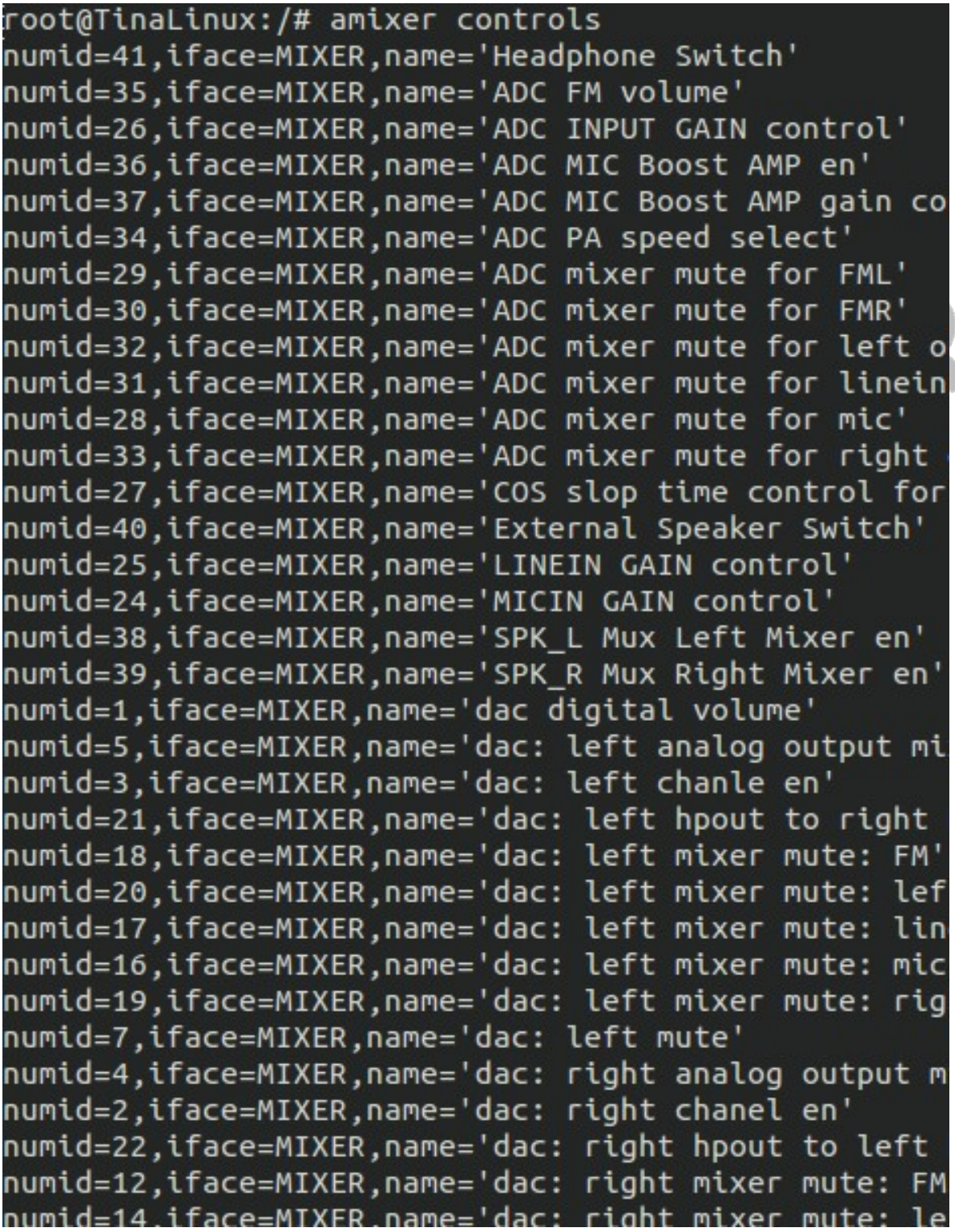
运行

```
# amixer controls
```

可得到如下内置 codec 的 mixer 控件列表，若使用

```
# amixer contents
```

还可以得到各控件当前值以及允许设置的值的范围等更为详细的信息。



6.6.1.1. 播放功能测试

设置音量及音频通路后即可播放：

```
amixer cset iface=MIXER,name='head phone volume' 50
amixer cset iface=MIXER,name='head phone power' 1
aplay test.wav
```

播放的音频可以通过耳机或喇叭听到（使用喇叭时需要把耳机拔掉）。

6.6.1.2. 录制功能测试

通过 MIC1 来录制音频：

```
amixer cset iface=MIXER,name='MICIN GAIN control' 3
amixer cset iface=MIXER,name='ADC MIC Boost AMP Enable' 1
amixer cset iface=MIXER,name='ADC Mixer Mute for Mic' 1
arecord -Dhw:0,0 -r 16000 -c 2 -f "S16_LE" -d 5 record.wav
```

上述例子录制一段时长为 5 秒的 16k Hz、两声道、16-bit 的音频。

6.6.1.3. LINEIN 到 HPL/HPR 通路测试

测试 LINEIN 到 HPL/HPR 的音频通路：

```
amixer cset iface=MIXER,name='dac: left analog output mixer en' 1
amixer cset iface=MIXER,name='dac: right analog output mixer en' 1
amixer cset iface=MIXER,name='dac: left mute' 1
amixer cset iface=MIXER,name='dac: right mute' 1
amixer cset iface=MIXER,name='dac: right mixer mute: FM' 1
amixer cset iface=MIXER,name='dac: left mixer mute: FM' 1
amixer cset iface=MIXER,name='hp left source select: 0-dac, 1-mixer' 1
amixer cset iface=MIXER,name='hp right source select: 0-dac, 1-mixer' 1
amixer cset iface=MIXER,name='SPK_L Mux Left Mixer en' 1
amixer cset iface=MIXER,name='SPK_R Mux Right Mixer en' 1
amixer cset iface=MIXER,name='head phone power' 1
```

此时使用 AUX 线外接音源，即可通过耳机或喇叭听到声音。

6.6.1.4. MIC1 到 HPL/HPR 通路测试

测试 MIC1 到 HPL/HPR 的音频通路：

```
amixer cset iface=MIXER,name='ADC MIC Boost AMP en' 1
amixer cset iface=MIXER,name='dac: left analog output mixer en' 1
amixer cset iface=MIXER,name='dac: right analog output mixer en' 1
amixer cset iface=MIXER,name='dac: left mute' 1
amixer cset iface=MIXER,name='dac: right mute' 1
amixer cset iface=MIXER,name='dac: left mixer mute: mic' 1
amixer cset iface=MIXER,name='dac: right mixer mute: mic' 1
amixer cset iface=MIXER,name='hp left source select: 0-dac, 1-mixer' 1
amixer cset iface=MIXER,name='hp right source select: 0-dac, 1-mixer' 1
amixer cset iface=MIXER,name='SPK_L Mux Left Mixer en' 1
amixer cset iface=MIXER,name='SPK_R Mux Right Mixer en' 1
amixer cset iface=MIXER,name='head phone power' 1
```

此时即可通过耳机或喇叭听到从 MIC1 输入的声音。

6.6.2. 外挂 codec 功能测试

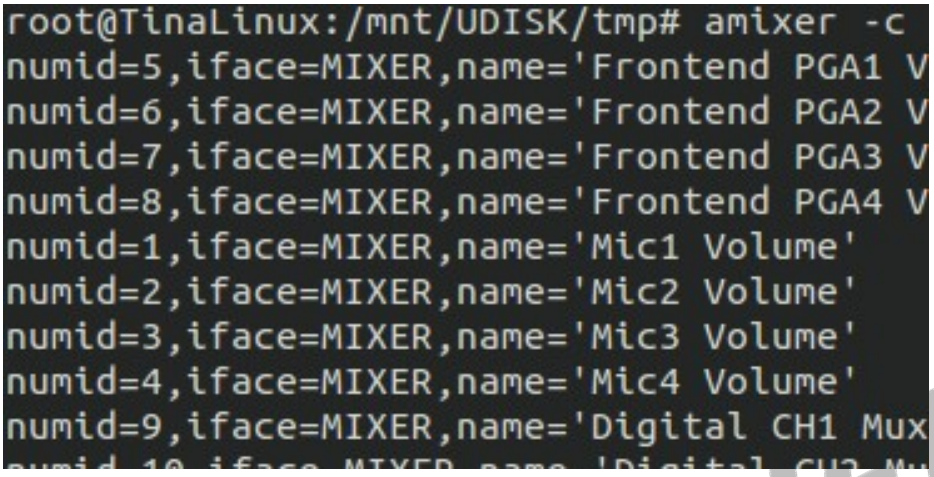
运行

```
amixer -c 1 controls
```

可得到如下外挂 codec 的 mixer 控件列表，同理若使用

```
amixer -c 1 contents
```

可以得到各控件当前值以及允许设置的值的范围等更为详细的信息。



6.6.2.1. 录制功能测试

通过 MIC2 进行录制：

```
amixer -c 1 cset iface=MIXER,name='Mic1 Volume' 1024
amixer -c 1 cset iface=MIXER,name='Frontend PGA1 Volume' 30
amixer -c 1 cset iface=MIXER,name='Digital CH1 Mux' 0
arecord -Dhw:1,0 -r 16000 -c 2 -f "S16_LE" -d 5 record.wav
```

上述例子录制一段时长为 5 秒的 16k Hz、两声道、16-bit 的音频。

6.6.2.2. SP2 至 MIC3 的回路测试

由于 MIC3 的输入与 SP2 的输出相连，因此测试此项一定要在 SP2 处接上喇叭，并且拔掉耳机，然后设置以下音量与音频通路：

```
amixer -c 1 cset iface=MIXER,name='Mic4 Volume' 1024
amixer -c 1 cset iface=MIXER,name='Frontend PGA4 Volume' 30
amixer -c 1 cset iface=MIXER,name='Digital CH2 Mux' 3
```

然后在后台播放音乐的同时进行录音：

```
aplay -Dhw:0,0 test.wav &
arecord -Dhw:1,0 -r 16000 -c 2 -f "S16_LE" -d 5 record.wav
```

即可把播放的声音录制下来（仅一个声道有声音）。

6.6.3. 混合功能测试

完整通路的测试方法如下：

1. 通过 AUX 线从 LINEIN 输入外接的音源；
2. 把内置 codec 设置为 MIC1 和 LINEIN 同时输出到 HPL/HPR，方法如下：

```
amixer cset iface=MIXER,name='ADC MIC Boost AMP en' 1
amixer cset iface=MIXER,name='dac: left analog output mixer en' 1
amixer cset iface=MIXER,name='dac: right analog output mixer en' 1
amixer cset iface=MIXER,name='dac: left mute' 1
amixer cset iface=MIXER,name='dac: right mute' 1
amixer cset iface=MIXER,name='dac: right mixer mute: FM' 1
```

```
amixer cset iface=MIXER,name='dac: left mixer mute: FM' 1
amixer cset iface=MIXER,name='dac: left mixer mute: mic' 1
amixer cset iface=MIXER,name='dac: right mixer mute: mic' 1
amixer cset iface=MIXER,name='hp left source select: 0-dac, 1-mixer' 1
amixer cset iface=MIXER,name='hp right source select: 0-dac, 1-mixer' 1
amixer cset iface=MIXER,name='SPK_L Mux Left Mixer en' 1
amixer cset iface=MIXER,name='SPK_R Mux Right Mixer en' 1
amixer cset iface=MIXER,name='head phone power' 1
```

3. 将喇叭连接到 SP2，拔掉耳机；

4. 设置外挂 codec 的输入增益和音频通路，打开 MIC2 和 MIC3 的输入，方法如下：

```
amixer -c 1 cset iface=MIXER,name='Mic1 Volume' 1024
amixer -c 1 cset iface=MIXER,name='Frontend PGA1 Volume' 30
amixer -c 1 cset iface=MIXER,name='Digital CH1 Mux' 0
amixer -c 1 cset iface=MIXER,name='Mic4 Volume' 1024
amixer -c 1 cset iface=MIXER,name='Frontend PGA4 Volume' 30
amixer -c 1 cset iface=MIXER,name='Digital CH2 Mux' 3
```

5. 通过外挂 codec 进行录音，同时对着 MIC2 讲话

```
arecord -Dhw:1,0 -r 16000 -c 2 -f "S16_LE" -d 5 record.wav
```

得到的录音其中一个声道为通过 MIC2 录制的说话声，另一个声道为经 MIC1 & LINEIN -> 内置 codec -> HPL/HPR(SP2) -> MIC3 -> 外挂 codec 传输的声音。

7. F35 音频模块

7.1. 硬件框架

在 F35 中，存在 2 个音频设备，其基本音频硬件框架图如下图所示：

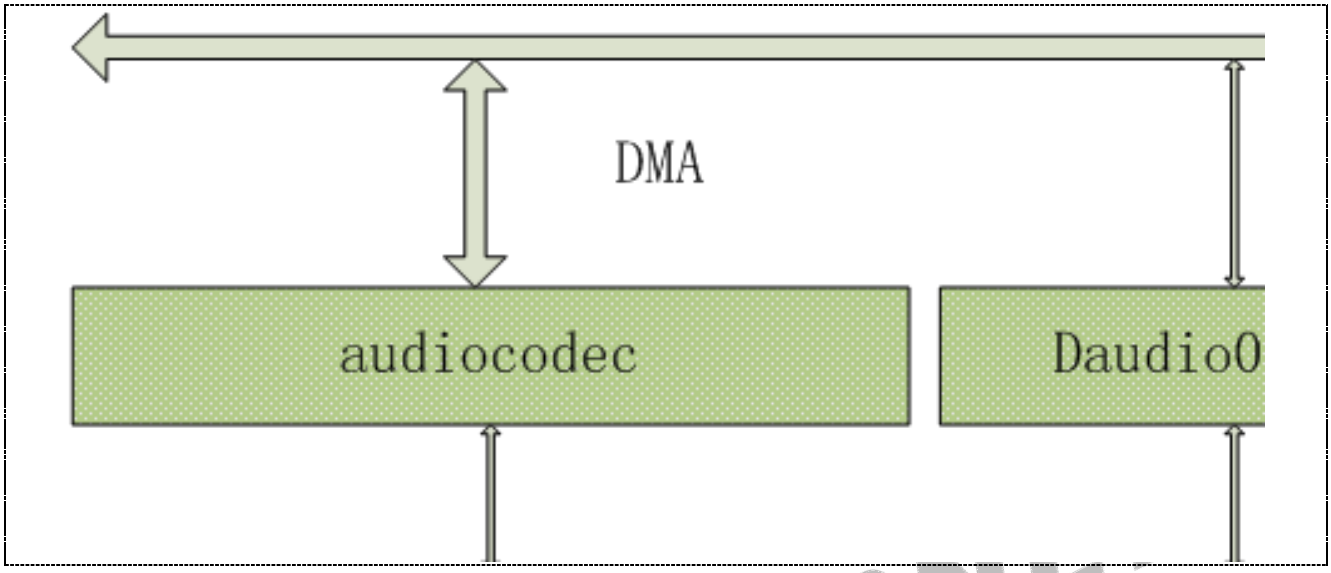


图 7 - 1 F35 音频硬件框架图

7.2. 软件框架

F35 的 2 个音频设备均采用 ASoC 框架实现，软件结构图如下所示：

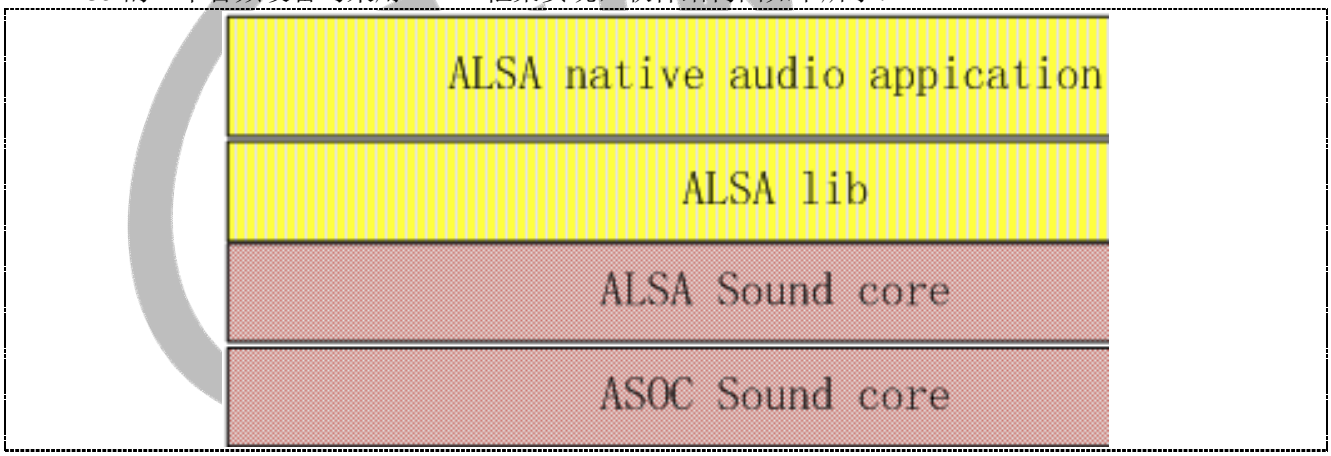


图 7 - 2 F35 音频软件框架图

7.3. 代码结构

```
└── sunxi
    ├── audiocodec
    │   ├── sun8iw8_sndcodec.c
    │   ├── sunxi_codec.c
    │   ├── Kconfig
    │   ├── Makefile
    │   ├── sunxi_codecdma.c
    │   ├── sunxi_sndcodec.c
    │   └── switch_hdset.c
    ├── daudio0
    │   ├── Kconfig
    │   ├── Makefile
    │   ├── sunxi-daudio0.c
    │   ├── sunxi-daudiodma0.c
    │   ├── snddaudio0.c
    │   └── sunxi-snddaudio0.c
```

audiocodec 的代码：

（1）codec 部分：sun8iw8_sndcodec.c

（2）platform 部分：sunxi_codec.c、sunxi_codecdma.c

（3）machine 部分：sunxi_sndcodec.c

daudio0 的代码：

（1）codec 部分：snddaudio0.c

（2）platform 部分：sunxi-daudio0.c、sunxi-daudiodma0.c

（3）machine 部分：sunxi-snddaudio0.c

7.4. menuconfig 配置

在 tina 根目录下执行 make kernel_menuconfig 进入配置：

Device Drivers ---> <*> Sound card support ---> <*> Advanced Linux Sound Architecture ---> <*> ALSA for SoC audio support --->

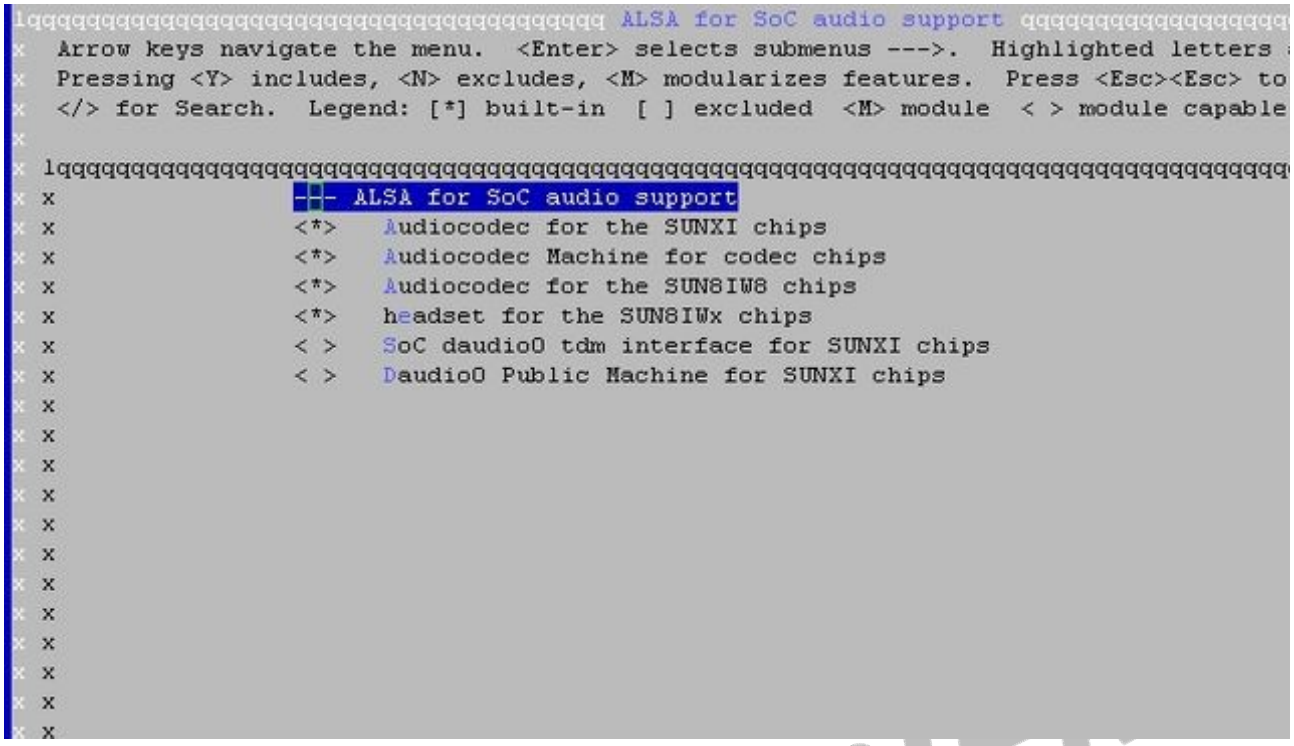


图 7 - 3 F35 menuconfig 配置

配置选项\声卡	audiocodec	daudio0
Audiocodec for the SUNXI chips	*	
Audiocodec Machine for codec chips	*	
Audiocodec for the SUN8IW8 chips	*	
headset for the SUN8IWx chips	*	
SoC daudio0 tdm interface for SUNXI chips		*
Daudio0 Public Machine for SUNXI chips		*

7.5. 音频通路配置

7.5.1. headphone 输出

control_name	value
Master Playback Volume	0~63
Speaker Function	headset
Audio headphone out	on

7.5.2. speaker 输出

control_name	value
Master Playback Volume	0~63
Speaker Function	spk
Audio speaker out	on

7.5.3. mic 输入

control_name	value
Audio record source	mic1
Audio main mic	on

8. R18 音频模块

8.1. 硬件框架

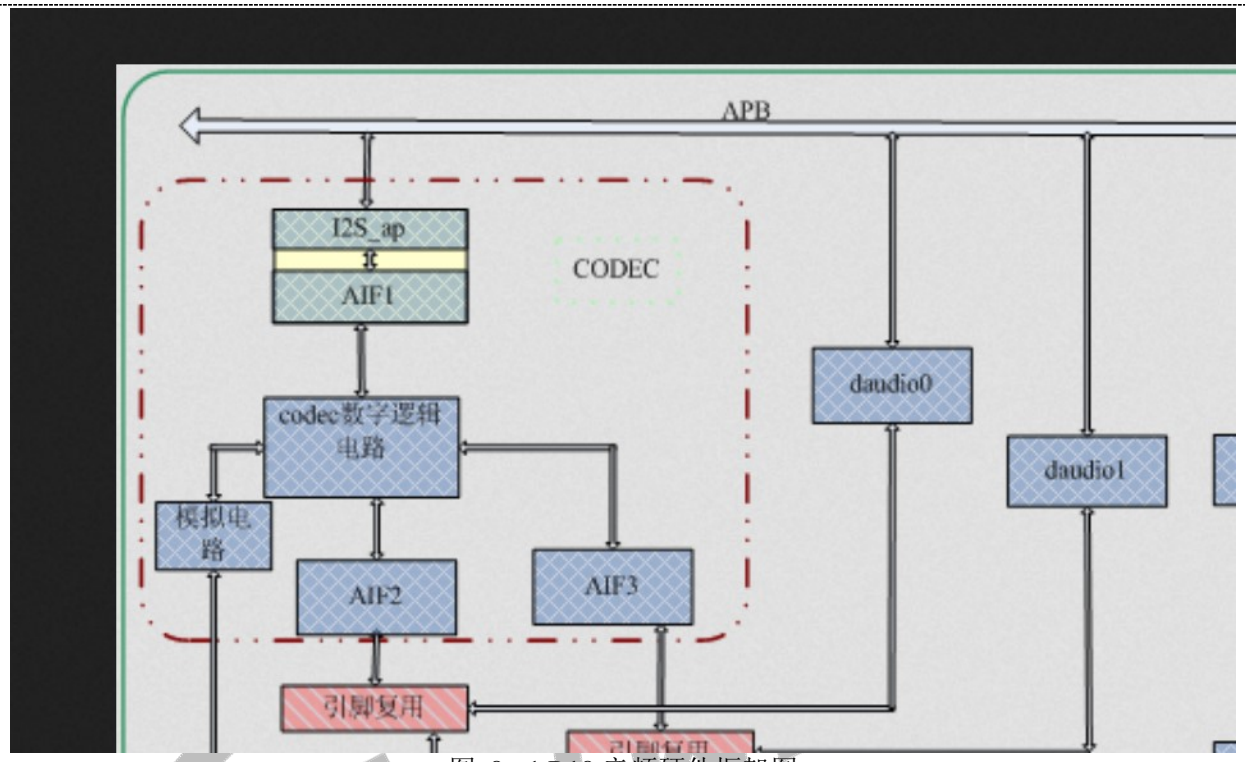


图 8 - 1 R18 音频硬件框架图

8.2. 软件框架

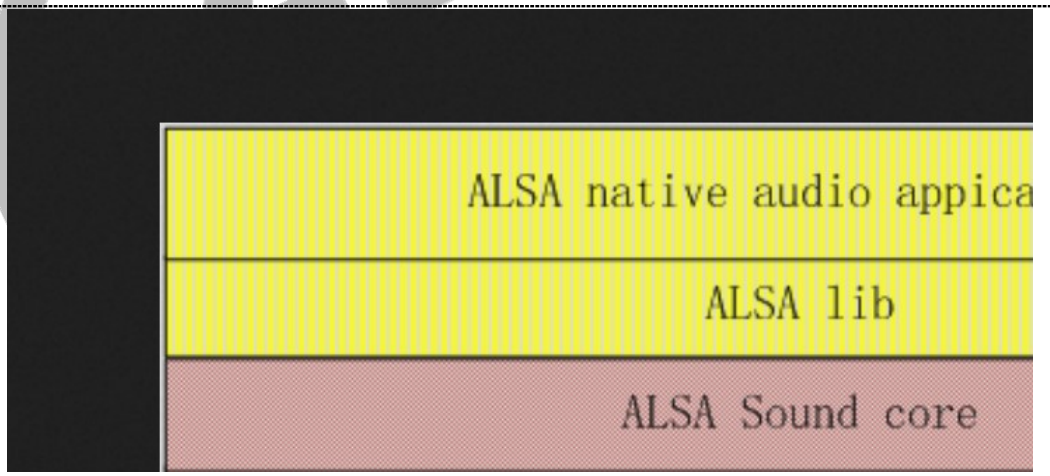


图 8 - 2 R18 音频软件框架图

8.3. 代码结构

```
└── sunxi
    ├── sun50iw1-codec.c
    ├── Kconfig
    ├── Makefile
    ├── sunxi_cpudai.c
    ├── sun50w1-sndcodec.c
    ├── daudio0
    ├── sunxi_daudio.c
    ├── sunxi-inter-i2s.c
    └── sunxi-snddaudio.c
```

audiocodec 的代码目录在 sunxi 下面；

codec 部分：sun50iw1-codec.c

platform 部分：sunxi_cpudai.c

machine 部分：sun50iw1-sndcodec.c

daudio0 的代码目录在 sunxi 下面；

- platform 部分：sunxi_daudio.c
- machine 部分：sunxi-snddaudio.c

8.4. menuconfig 配置

make kernel_menuconfig 进入配置项：

Device Drivers ---> <*> Sound card support ---> <*> Advanced Linux Sound Architecture ---> <*> ALSA for SoC audio support ---> Allwinner SoC Audio Support --->

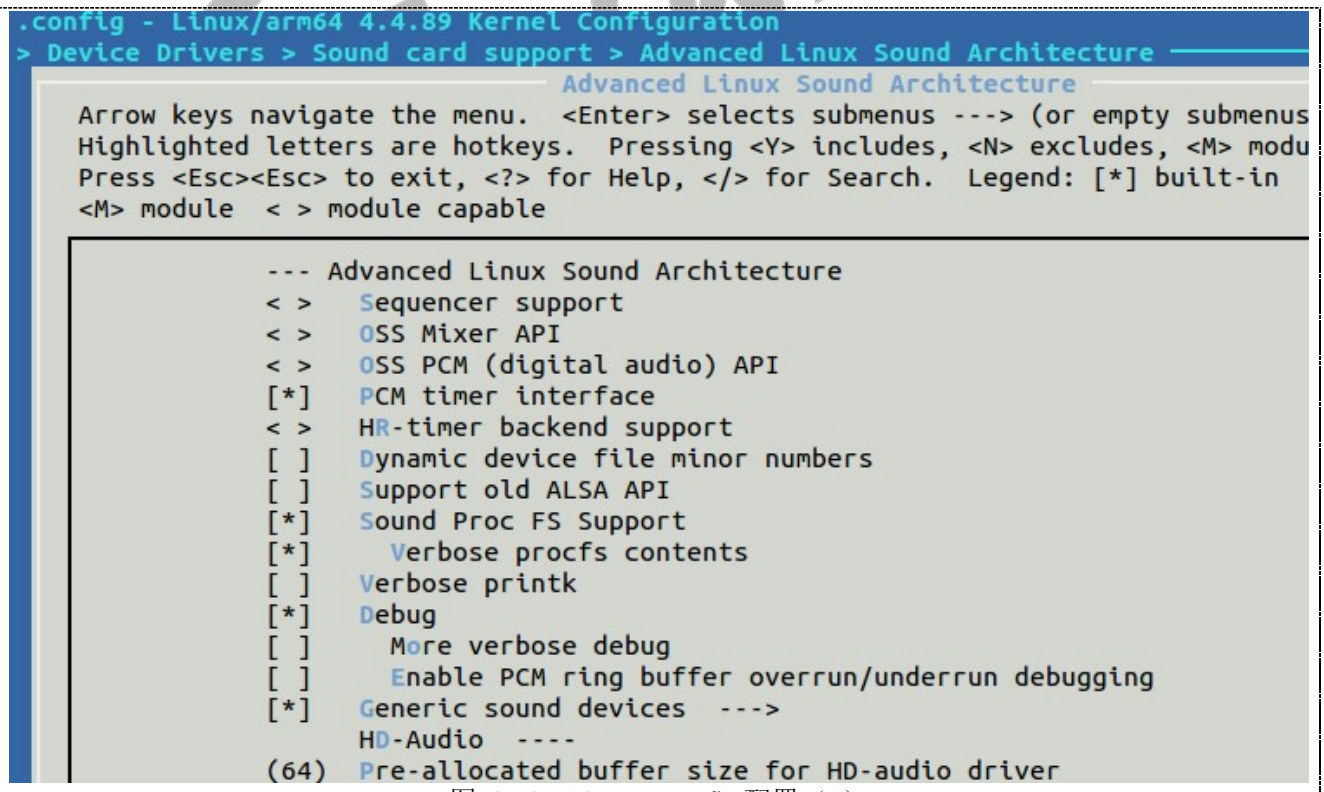


图 8 - 3 R18 menuconfig 配置（1）

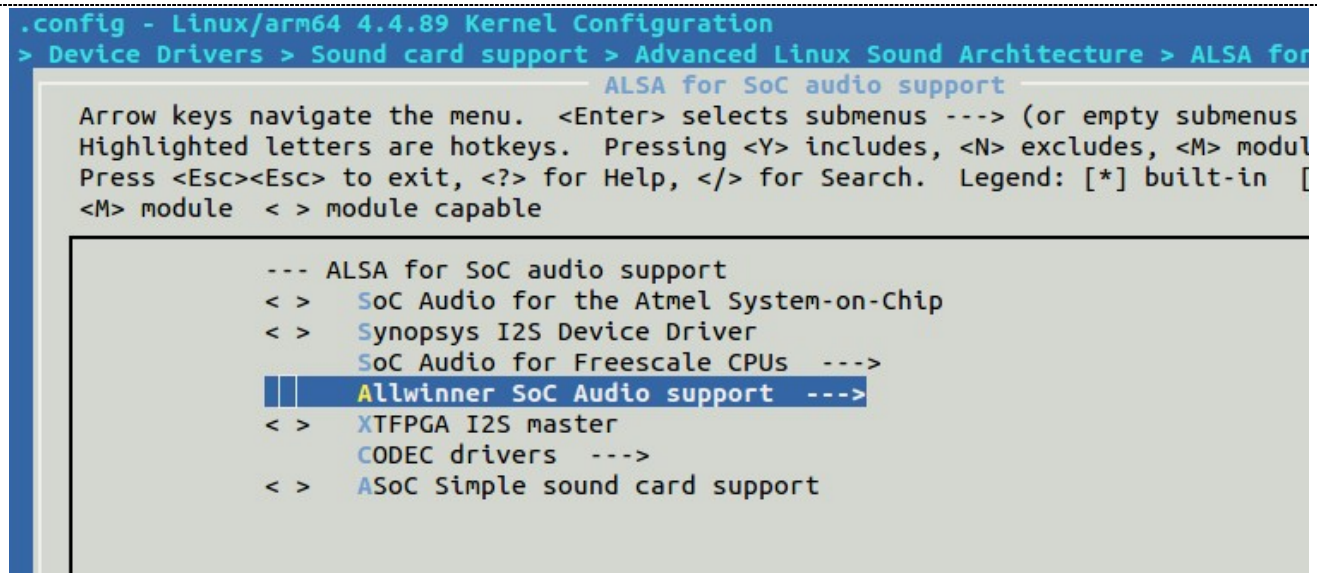


图 8 - 4 R18 menuconfig 配置（2）

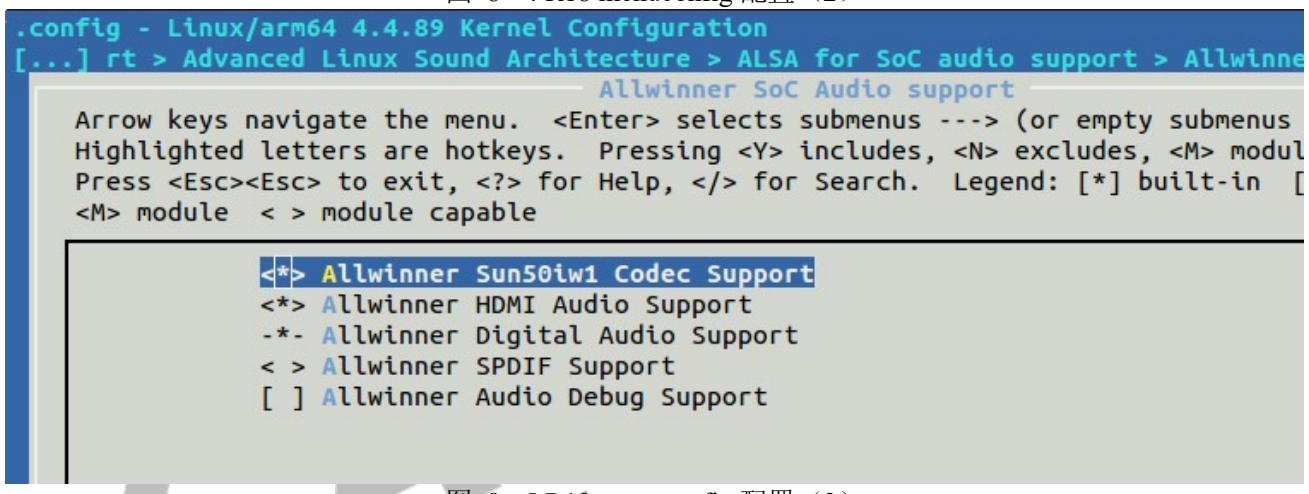


图 8 - 5 R18 menuconfig 配置（3）

配置选项\声卡	audiocodec	daudio0	HDMI audio
Allwinner Sun50iw1 Codec Support	*		
Allwinner HDMI Audio Support			*
Allwinner Digital Audio Support		*	

8.5. 音频通路配置

8.5.1. Headphone 输出

Headphone 音量（0~63）：

```
amixer cset name='headphone volume' 40
```

Headphone 通路：

```
amixer cset name='AIF1IN0R Mux' 'AIF1_DA0R'
amixer cset name='AIF1IN0L Mux' 'AIF1_DA0L'
amixer cset name='DACR Mixer AIF1DA0R Switch' 1
amixer cset name='DACL Mixer AIF1DA0L Switch' 1
amixer cset name='HP_R Mux' 'DACR HPR Switch'
amixer cset name='HP_L Mux' 'DACL HPL Switch'
amixer cset name='Headphone Switch' 1
```

8.5.2. MIC 输入

MIC 音量（0~7）：

```
amixer cset iface=MIXER,name='MIC1 boost AMP gain control' 4
```

MIC 通路：

```
amixer cset name='LADC input Mixer MIC1 boost Switch' 1
amixer cset name='RADC input Mixer MIC1 boost Switch' 1
amixer cset name='ADCR Mux' 'ADC'
amixer cset name='ADCL Mux' 'ADC'
amixer cset name='AIF1 AD0L Mixer ADCL Switch' 1
amixer cset name='AIF1 AD0R Mixer ADCR Switch' 1
amixer cset name='AIF1OUT0L Mux' 'AIF1_AD0L'
amixer cset name='AIF1OUT0R Mux' 'AIF1_AD0R'
```

8.5.3. Line in 输入

Line in 通路：

```
amixer cset name='LADC input Mixer LINEINL Switch' 1
amixer cset name='RADC input Mixer LINEINR Switch' 1
amixer cset name='ADCL Mux' 'ADC'
amixer cset name='ADCR Mux' 'ADC'
amixer cset name='AIF1 AD0L Mixer ADCL Switch' 1
amixer cset name='AIF1 AD0R Mixer ADCR Switch' 1
amixer cset name='AIF1OUT0L Mux' 'AIF1_AD0L'
amixer cset name='AIF1OUT0R Mux' 'AIF1_AD0R'
```

9. Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgment to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.