

DroidPhone的专栏

欢迎各位大虾交流，本人联系方式：droid.phx@gmail.com

目录视图

摘要视图

RSS 订阅

个人资料



DroidPhone

访问：1072912次
积分：8742
等级：
排名：第1446名
原创：51篇 转载：0篇
译文：4篇 评论：536条

文章搜索

文章分类

移动开发之Android (11)
Linux内核架构 (15)
Linux设备驱动 (20)
Linux电源管理 (3)
Linux音频子系统 (15)
Linux中断子系统 (5)
Linux时间管理系统 (8)
Linux输入子系统 (4)

文章存档

2014年07月 (1)
2014年04月 (4)
2013年11月 (4)
2013年10月 (3)
2013年07月 (3)

展开

阅读排行

Linux ALSA声卡驱动之一 (73868)
Android Audio System 之 (58676)
Linux ALSA声卡驱动之二 (46250)
Android Audio System 之 (42628)
Linux ALSA声卡驱动之三 (41413)
Linux ALSA声卡驱动之

【公告】博客系统优化升级 【收藏】Scala 资源一应俱全 博乐招募开始啦 程序员七夕表白礼品指南

Linux ALSA声卡驱动之二：声卡的创建

标签：linux struct module structure list

2011-03-30 19:15 46260人阅读 评论(19) 收藏 举报

分类：Linux音频子系统 (14) Linux设备驱动 (19)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?) [+]

声明：本博内容均由<http://blog.csdn.net/droidphone>原创，转载请注明出处，谢谢！

1. struct snd_card

1.1. snd_card是什么

snd_card可以说是整个ALSA音频驱动最顶层的一个结构，整个声卡的软件逻辑结构开始于该结构，几乎所有与声音相关的逻辑设备都是在snd_card的管理之下，声卡驱动的第一个动作通常就是创建一个snd_card结构体。正因为如此，本节中，我们也从 struct snd_card开始吧。

1.2. snd_card的定义

snd_card的定义位于改头文件中：include/sound/core.h

```
[c-sharp]
01. /* main structure for soundcard */
02.
03. struct snd_card {
04.     int number;          /* number of soundcard (index to
05.                          snd_cards) */
06.
07.     char id[16];          /* id string of this card */
08.     char driver[16];      /* driver name */
09.     char shortname[32];   /* short name of this soundcard */
10.     char longname[80];    /* name of this soundcard */
11.     char mixername[80];   /* mixer name */
12.     char components[128]; /* card components delimited with
13.                          space */
14.     struct module *module; /* top-level module */
15.
16.     void *private_data;   /* private data for soundcard */
17.     void (*private_free) (struct snd_card *card); /* callback for freeing of
18.                          private data */
19.     struct list_head devices; /* devices */
20.
21.     unsigned int last_numid; /* last used numeric ID */
22.     struct rw_semaphore controls_rwsem; /* controls list lock */
23.     rwlock_t ctl_files_rwlock; /* ctl_files list lock */
24.     int controls_count; /* count of all controls */
25.     int user_ctl_count; /* count of all user controls */
26.     struct list_head controls; /* all controls for this card */
27.     struct list_head ctl_files; /* active control files */
28. }
```

[Android Audio System 之](#) (37372)
[Linux时间子系统之六：hrtimer](#) (36277)
[Linux ALSA声卡驱动之十](#) (36208)
[Linux ALSA声卡驱动之四](#) (36031)
[Linux ALSA声卡驱动之四](#) (31602)

评论排行

[Android Audio System 之](#) (56)
[Linux ALSA声卡驱动之三](#) (42)
[Linux ALSA声卡驱动之八](#) (35)
[Linux时间子系统之六：hrtimer](#) (25)
[Linux中断（interrupt）子](#) (24)
[Android SurfaceFlinger中](#) (21)
[Linux ALSA声卡驱动之二](#) (19)
[Android Audio System 之](#) (18)
[Linux ALSA声卡驱动之十](#) (17)
[Linux中断（interrupt）子](#) (17)

推荐文章

* 致JavaScript也将征服的物联网世界
* 从苏宁电器到卡斯基：难忘的三年硕士时光
* 作为一名基层管理者如何利用情商管理自己和团队（一）
* Android CircleImageView圆形ImageView
* 高质量代码的命名法则

最新评论

[Linux时间子系统之一：clock_source](#)
zhqh100: 1.2 read回调函数时钟源本身不会产生中断，要获得时钟源的当前计数，只能通过主动调用它的read...
[Linux中断（interrupt）子系统之六](#)
liunix61: 大神！必须顶@！！！！
[Linux时间子系统之三：时间的维护](#)
Kevin_Smart: 学习了
[Linux时间子系统之六：高精度定时器](#)
Kevin_Smart: 像楼主说的，原则上，hrtimer是利用一个硬件计数器来实现的，所以精度才可以做到ns级别。硬件的计...
[Linux中断（interrupt）子系统之七](#)
12期-马金兴: 恩，虽然这么多字但是我要好好学习一下
已知二叉树的前序遍历和中序遍历
重修月: 很喜欢博主的文章，刚刚用豆瓣博客备份专家备份了您的全部博文。
[Linux ALSA声卡驱动之三：PCM](#)
灿哥哥: 学习了
[Android Audio System 之三：ALSA](#)
ss0429: 楼主的文章写的很精炼，多谢分享~
[Linux中断（interrupt）子系统之七](#)
KrisFei: 针对这句话有两个问题想讨论下：1. disable_irq()放在中断上半部会导致死锁。2. 如果...
[Linux ALSA声卡驱动之七：ASoC](#)
Wit-Z-Joy: 博主您好，我现在需要解决苹果耳机麦克风不能使用的问题。我的设备也是美标的耳机口，市面上常见的魅族，小...

```
29. struct snd_info_entry *proc_root; /* root for soundcard specific files */
30. struct snd_info_entry *proc_id; /* the card id */
31. struct proc_dir_entry *proc_root_link; /* number link to real id */
32.
33. struct list_head files_list; /* all files associated to this card */
34. struct snd_shutdown_f_ops *s_f_ops; /* file operations in the shutdown
35. state */
36. spinlock_t files_lock; /* lock the files for this card */
37. int shutdown; /* this card is going down */
38. int free_on_last_close; /* free in context of file_release */
39. wait_queue_head_t shutdown_sleep;
40. struct device *dev; /* device assigned to this card */
41. #ifndef CONFIG_SYSFS_DEPRECATED
42. struct device *card_dev; /* cardX object for sysfs */
43. #endif
44.
45. #ifdef CONFIG_PM
46. unsigned int power_state; /* power state */
47. struct mutex power_lock; /* power lock */
48. wait_queue_head_t power_sleep;
49. #endif
50.
51. #if defined(CONFIG_SND_MIXER_OSS) || defined(CONFIG_SND_MIXER_OSS_MODULE)
52. struct snd_mixer_oss *mixer_oss;
53. int mixer_oss_change_count;
54. #endif
55. };
```

- struct list_head devices 记录该声卡下所有逻辑设备的链表
- struct list_head controls 记录该声卡下所有的控制单元的链表
- void *private_data 声卡的私有数据，可以在创建声卡时通过参数指定数据的大小

2. 声卡的建立流程

2.1.1. 第一步，创建snd_card的一个实例

```
[c-sharp]
01. struct snd_card *card;
02. int err;
03. ....
04. err = snd_card_create(index, id, THIS_MODULE, 0, &card);
```

- index 一个整数值，该声卡的编号
- id 字符串，声卡的标识符
- 第四个参数 该参数决定在创建snd_card实例时，需要同时额外分配的私有数据的大小，该数据的指针最终会赋值给snd_card的private_data数据成员
- card 返回所创建的snd_card实例的指针

2.1.2. 第二步，创建声卡的芯片专用数据

声卡的专用数据主要用于存放该声卡的一些资源信息，例如中断资源、io资源、dma资源等。可以有两种创建方法：

- 通过上一步中snd_card_create()中的第四个参数，让snd_card_create自己创建

```
[c-sharp]
01. // struct mychip 用于保存专用数据
02. err = snd_card_create(index, id, THIS_MODULE,
03. sizeof(struct mychip), &card);
04. // 从private_data中取出
05. struct mychip *chip = card->private_data;
```

- 自己创建：

```
[c-sharp]
01. struct mychip {
```

```

02.     struct snd_card *card;
03.     ....
04. };
05. struct snd_card *card;
06. struct mychip *chip;
07.
08. chip = kzalloc(sizeof(*chip), GFP_KERNEL);
09. ....
10. err = snd_card_create(index[dev], id[dev], THIS_MODULE, 0, &card);
11. // 专用数据记录snd_card实例
12. chip->card = card;
13. ....

```

然后，把芯片的专有数据注册为声卡的一个低阶设备：

```

[c-sharp]
01. static int snd_mychip_dev_free(struct snd_device *device)
02. {
03.     return snd_mychip_free(device->device_data);
04. }
05.
06. static struct snd_device_ops ops = {
07.     .dev_free = snd_mychip_dev_free,
08. };
09. ....
10. snd_device_new(card, SNDRV_DEV_LOWLEVEL, chip, &ops);

```

注册为低阶设备主要是为了当声卡被注销时，芯片专用数据所占用的内存可以被自动地释放。

2.1.3. 第三步，设置Driver的ID和名字

```

[c-sharp]
01. strcpy(card->driver, "My Chip");
02. strcpy(card->shortname, "My Own Chip 123");
03. sprintf(card->longname, "%s at 0x%lx irq %i",
04.         card->shortname, chip->ioport, chip->irq);

```

snd_card的driver字段保存着芯片的ID字符串，user空间的alsa-lib会使用到该字符串，所以必须要保证该ID的唯一性。shortname字段更多地用于打印信息，longname字段则会出现在/proc/asound/cards中。

2.1.4. 第四步，创建声卡的功能部件（逻辑设备），例如PCM，Mixer，MIDI等

这时候可以创建声卡的各种功能部件了，还记得开头的snd_card结构体的devices字段吗？每一种部件的创建最终会调用snd_device_new()来生成一个snd_device实例，并把该实例链接到snd_card的devices链表中。

通常，alsa-driver的已经提供了一些常用的部件的创建函数，而不必直接调用snd_device_new()，比如：

```

PCM ---- snd_pcm_new()

RAWMIDI -- snd_rawmidi_new()

CONTROL -- snd_ctl_create()

TIMER  -- snd_timer_new()

INFO   -- snd_card_proc_new()

JACK   -- snd_jack_new()

```

2.1.5. 第五步，注册声卡

```

[c-sharp]
01. err = snd_card_register(card);
02. if (err < 0) {
03.     snd_card_free(card);
04.     return err;
05. }

```

2.2. 一个实际的例子

我把/sound/arm/pxa2xx-ac97.c的部分代码贴上来:

```
[cpp]
01. static int __devinit pxa2xx_ac97_probe(struct platform_device *dev)
02. {
03.     struct snd_card *card;
04.     struct snd_ac97_bus *ac97_bus;
05.     struct snd_ac97_template ac97_template;
06.     int ret;
07.     pxa2xx_audio_ops_t *pdata = dev->dev.platform_data;
08.
09.     if (dev->id >= 0) {
10.         dev_err(&dev->dev, "PXA2xx has only one AC97 port./n");
11.         ret = -ENXIO;
12.         goto err_dev;
13.     }
14.     ///(1)///
15.     ret = snd_card_create(SNDRV_DEFAULT_IDX1, SNDRV_DEFAULT_STR1,
16.         THIS_MODULE, 0, &card);
17.     if (ret < 0)
18.         goto err;
19.
20.     card->dev = &dev->dev;
21.     ///(3)///
22.     strncpy(card->driver, dev->dev.driver->name, sizeof(card->driver));
23.
24.     ///(4)///
25.     ret = pxa2xx_pcm_new(card, &pxa2xx_ac97_pcm_client, &pxa2xx_ac97_pcm);
26.     if (ret)
27.         goto err;
28.     ///(2)///
29.     ret = pxa2xx_ac97_hw_probe(dev);
30.     if (ret)
31.         goto err;
32.
33.     ///(4)///
34.     ret = snd_ac97_bus(card, 0, &pxa2xx_ac97_ops, NULL, &ac97_bus);
35.     if (ret)
36.         goto err_remove;
37.     memset(&ac97_template, 0, sizeof(ac97_template));
38.     ret = snd_ac97_mixer(ac97_bus, &ac97_template, &pxa2xx_ac97_ac97);
39.     if (ret)
40.         goto err_remove;
41.     ///(3)///
42.     snprintf(card->shortname, sizeof(card->shortname),
43.         "%s", snd_ac97_get_short_name(pxa2xx_ac97_ac97));
44.     snprintf(card->longname, sizeof(card->longname),
45.         "%s (%s)", dev->dev.driver->name, card->mixername);
46.
47.     if (pdata && pdata->codec_pdata[0])
48.         snd_ac97_dev_add_pdata(ac97_bus->codec[0], pdata->codec_pdata[0]);
49.     snd_card_set_dev(card, &dev->dev);
50.     ///(5)///
51.     ret = snd_card_register(card);
52.     if (ret == 0) {
53.         platform_set_drvdata(dev, card);
54.         return 0;
55.     }
56.
57. err_remove:
58.     pxa2xx_ac97_hw_remove(dev);
59. err:
60.     if (card)
61.         snd_card_free(card);
62. err_dev:
63.     return ret;
64. }
65.
66. static int __devexit pxa2xx_ac97_remove(struct platform_device *dev)
67. {
68.     struct snd_card *card = platform_get_drvdata(dev);
```

```

69.
70.     if (card) {
71.         snd_card_free(card);
72.         platform_set_drvdata(dev, NULL);
73.         pxa2xx_ac97_hw_remove(dev);
74.     }
75.
76.     return 0;
77. }
78.
79. static struct platform_driver pxa2xx_ac97_driver = {
80.     .probe      = pxa2xx_ac97_probe,
81.     .remove     = __devexit_p(pxa2xx_ac97_remove),
82.     .driver     = {
83.         .name    = "pxa2xx-ac97",
84.         .owner   = THIS_MODULE,
85. #ifdef CONFIG_PM
86.         .pm      = &pxa2xx_ac97_pm_ops,
87. #endif
88.     },
89. };
90.
91. static int __init pxa2xx_ac97_init(void)
92. {
93.     return platform_driver_register(&pxa2xx_ac97_driver);
94. }
95.
96. static void __exit pxa2xx_ac97_exit(void)
97. {
98.     platform_driver_unregister(&pxa2xx_ac97_driver);
99. }
100.
101. module_init(pxa2xx_ac97_init);
102. module_exit(pxa2xx_ac97_exit);
103.
104. MODULE_AUTHOR("Nicolas Pitre");
105. MODULE_DESCRIPTION("AC97 driver for the Intel PXA2xx chip");

```

驱动程序通常由probe回调函数开始，对一下2.1中的步骤，是否有相似之处？

经过以上的创建步骤之后，声卡的逻辑结构如下图所示：

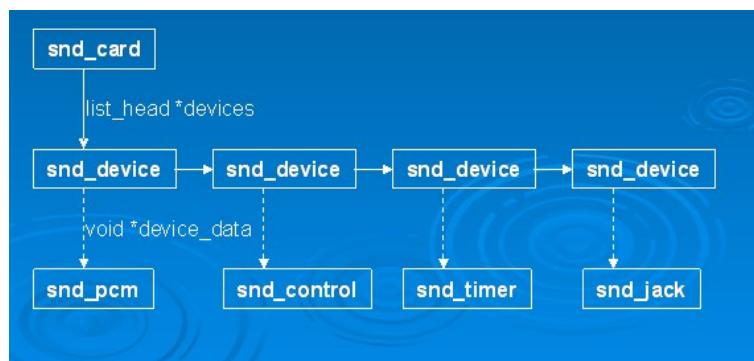


图 2.2.1 声卡的软件逻辑结构

下面的章节里我们分别讨论一下snd_card_create()和snd_card_register()这两个函数。

3. snd_card_create()

snd_card_create()在/sound/core/init.c中定义。

[cpp]

```

01.  /**
02.   * snd_card_create - create and initialize a soundcard structure
03.   * @idx: card index (address) [0 ... (SNDRV_CARDS-1)]
04.   * @xid: card identification (ASCII string)
05.   * @module: top level module for locking
06.   * @extra_size: allocate this extra size after the main soundcard structure
07.   * @card_ret: the pointer to store the created card instance
08.   *
09.   * Creates and initializes a soundcard structure.
10.   *
11.   * The function allocates snd_card instance via kzalloc with the given
12.   * space for the driver to use freely. The allocated struct is stored
13.   * in the given card_ret pointer.
14.   *
15.   * Returns zero if successful or a negative error code.
16.   */
17. int snd_card_create(int idx, const char *xid,
18.                    struct module *module, int extra_size,
19.                    struct snd_card **card_ret)

```

首先，根据extra_size参数的大小分配内存，该内存区可以作为芯片的专有数据使用（见前面的介绍）：

```

[c-sharp]
01. card = kzalloc(sizeof(*card) + extra_size, GFP_KERNEL);
02. if (!card)
03.     return -ENOMEM;

```

拷贝声卡的ID字符串：

```

[c-sharp]
01. if (xid)
02.     strcpy(card->id, xid, sizeof(card->id));

```

如果传入的声卡编号为-1，自动分配一个索引编号：

```

[c-sharp]
01. if (idx < 0) {
02.     for (idx2 = 0; idx2 < SNDRV_CARDS; idx2++)
03.         /* idx == -1 == 0xffff means: take any free slot */
04.         if (~snd_cards_lock & idx & 1<idx2) {
05.             if (module_slot_match(module, idx2)) {
06.                 idx = idx2;
07.                 break;
08.             }
09.         }
10. }
11. if (idx < 0) {
12.     for (idx2 = 0; idx2 < SNDRV_CARDS; idx2++)
13.         /* idx == -1 == 0xffff means: take any free slot */
14.         if (~snd_cards_lock & idx & 1<idx2) {
15.             if (!slots[idx2] || !*slots[idx2]) {
16.                 idx = idx2;
17.                 break;
18.             }
19.         }
20. }

```

初始化snd_card结构中必要的字段：

```

[c-sharp]
01. card->number = idx;
02. card->module = module;
03. INIT_LIST_HEAD(&card->devices);
04. init_rwsem(&card->controls_rwsem);
05. rwlock_init(&card->ctl_files_rwlock);
06. INIT_LIST_HEAD(&card->controls);
07. INIT_LIST_HEAD(&card->ctl_files);
08. spin_lock_init(&card->files_lock);
09. INIT_LIST_HEAD(&card->files_list);
10. init_waitqueue_head(&card->shutdown_sleep);
11. #ifdef CONFIG_PM
12.     mutex_init(&card->power_lock);
13.     init_waitqueue_head(&card->power_sleep);
14. #endif

```



建立逻辑设备: Control

```
[c-sharp]
01. /* the control interface cannot be accessed from the user space until */
02. /* snd_cards_bitmask and snd_cards are set with snd_card_register */
03. err = snd_ctl_create(card);
```

建立proc文件中的info节点: 通常就是/proc/asound/card0

```
[c-sharp]
01. err = snd_info_card_create(card);
```

把第一步分配的内存指针放入private_data字段中:

```
[c-sharp]
01. if (extra_size > 0)
02.     card->private_data = (char *)card + sizeof(struct snd_card);
```

4. snd_card_register()

snd_card_create()在/sound/core/init.c中定义。

```
[c-sharp]
01. /**
02.  * snd_card_register - register the soundcard
03.  * @card: soundcard structure
04.  *
05.  * This function registers all the devices assigned to the soundcard.
06.  * Until calling this, the ALSA control interface is blocked from the
07.  * external accesses. Thus, you should call this function at the end
08.  * of the initialization of the card.
09.  *
10.  * Returns zero otherwise a negative error code if the register failed.
11.  */
12. int snd_card_register(struct snd_card *card)
```

首先, 创建sysfs下的设备:

```
[c-sharp]
01. if (!card->card_dev) {
02.     card->card_dev = device_create(sound_class, card->dev,
03.                                   MKDEV(0, 0), card,
04.                                   "card%i", card->number);
05.     if (IS_ERR(card->card_dev))
06.         card->card_dev = NULL;
07. }
```

其中, sound_class是在/sound/sound_core.c中创建的:

```
[c-sharp]
01. static char *sound_devnode(struct device *dev, mode_t *mode)
02. {
03.     if (MAJOR(dev->devt) == SOUND_MAJOR)
04.         return NULL;
05.     return kasprintf(GFP_KERNEL, "snd/%s", dev_name(dev));
06. }
07. static int __init init_soundcore(void)
08. {
09.     int rc;
10.
11.     rc = init_oss_soundcore();
12.     if (rc)
13.         return rc;
14.
15.     sound_class = class_create(THIS_MODULE, "sound");
16.     if (IS_ERR(sound_class)) {
17.         cleanup_oss_soundcore();
18.         return PTR_ERR(sound_class);
19.     }
20.
21.     sound_class->devnode = sound_devnode;
22. }
```

```
23.         return 0;
24.     }
```

由此可见，声卡的class将会出现在文件系统的/sys/class/sound/下面，并且，sound_devnode()也决定了相应的设备节点也将会出现在/dev/snd/下面。

接下来的步骤，通过snd_device_register_all()注册所有挂在该声卡下的逻辑设备，snd_device_register_all()实际上是通过snd_card的devices链表，遍历所有的snd_device，并且调用snd_device的ops->dev_register()来实现各自设备的注册的。

```
[c-sharp]
01. if ((err = snd_device_register_all(card)) < 0)
02.     return err;
```

最后就是建立一些相应的proc和sysfs下的文件或属性节点，代码就不贴了。

至此，整个声卡完成了建立过程。

顶

3

踩

0

上一篇

Linux ALSA声卡驱动之一：ALSA架构简介

下一篇

Linux ALSA声卡驱动之三：PCM设备的创建

我的同类文章

Linux音频子系统（14）				Linux设备驱动（19）			
•	ALSA声卡驱动中的DAPM...	2013-11-09	阅读 8908	•	ALSA声卡驱动中的DAPM...	2013-11-04	阅读 9881
•	ALSA声卡驱动中的DAPM...	2013-11-04	阅读 12874	•	ALSA声卡驱动中的DAPM...	2013-11-01	阅读 10150
•	ALSA声卡驱动中的DAPM...	2013-10-24	阅读 12490	•	ALSA声卡驱动中的DAPM...	2013-10-23	阅读 10704
•	ALSA声卡驱动中的DAPM...	2013-10-18	阅读 17011	•	Linux ALSA声卡驱动之八：...	2012-03-13	阅读 31171
•	Linux ALSA声卡驱动之七：...	2012-02-23	阅读 36031	•	Linux ALSA声卡驱动之六：...	2012-02-03	阅读 37375
•	Linux ALSA声卡驱动之五：...	2012-01-17	阅读 26608				
更多文章							

猜你在找

- Android底层技术：Linux驱动框架与开发
- Linux ALSA声卡驱动之二声卡的创建
- linux嵌入式开发+驱动开发
- Linux ALSA声卡驱动之二声卡的创建
- Linux设备驱动开发入门
- Linux ALSA声卡驱动之二声卡的创建
- 嵌入式Linux项目实战：三个大项目（数码相机、摄像头驱动Linux ALSA声卡驱动之二声卡的创建
- Linux ALSA声卡驱动之二声卡的创建
- 嵌入式Linux高级驱动教程（韦东山2期）
- Linux ALSA声卡驱动之二声卡的创建

上海二手别墅

阿迪正品折扣

德云社门票

linux声卡


热水器除垢

厨宝热水器

留学生公寓

查看评论

- 14楼 elfofd 2014-11-29 16:23发表

我想买个支持Linux红帽6.4系统声卡，PCI-E接口，有没有合适推荐下？？还是我随便买个PCIE接口的声卡，下载ALSA驱动就可以了？？希望大神帮帮我这个菜鸟，谢谢~~~
- 13楼 辉捺天韵 2013-12-10 11:58发表

谢谢！！！！



12楼 [shallot0000](#) 2013-08-15 11:39发表



真的不错，很希望自己有一天也能写出类似的文章跟大家分享，膜拜中。。。

11楼 [ljf10000](#) 2013-08-14 16:03发表



2.1.2 里有个错误（自己创建）

```
01.struct mychip {
02. struct snd_card *card;
03. ....
04.};
05.struct snd_card *card;
06.struct mychip *chip;
07.err = snd_card_create(index[dev], id[dev], THIS_MODULE, 0, &card);
08.// 专用数据记录snd_card实例
09.chip->card = card; // 这一步chip还没分配内存，
10.....
11.chip = kzalloc(sizeof(*chip), GFP_KERNEL);
```

Re: [DroidPhone](#) 2013-08-14 21:39发表



回复ljf10000：嗯，是的，已修改。
Thanks！

10楼 [vito_coleone](#) 2013-05-24 14:32发表



分析的好

9楼 [9Th](#) 2013-03-19 16:57发表



漂亮~

8楼 [hua3x](#) 2012-10-30 16:03发表



很牛呀，没有几年的经验写不出来呀，佩服。

7楼 [wulong117](#) 2012-09-12 09:41发表



分析的非常好，大哥是牛人啊！！！

6楼 [haokaihaohe110](#) 2012-05-17 09:24发表



很好，很详细。

博主，你好，我想问一下，我在android下使用alsa时，在asound.conf中设置扬声器音量为100，为什么在启动时会出现暴音呢，这个问题我一直找不到解决办法，也不知道怎样在代码中修改。求博主帮忙啊，下面是我发的求助帖子：

Android2.3上面的音频系统ALSA的asound.conf中有一段设置扬声器音量的位置，即：

```
pcm.AndroidPlayback_Speaker_normal {
type hooks
slave.pcm {
type hw
card 1
device 0 # Must be of type "digital audio playback"
}
hooks.0 {
type ctl_elems
hook_args [

{
name 'Line DAC Playback Volume'
value.0 100 //这个位置
value.1 100
}
{
name 'HP DAC Playback Volume'
value.0 0
value.1 0
}
{
name 'Telephony'
value Off
}

]
}
```

}

将 name 'Line DAC Playback Volume'下面的value设置为最大值100或者小于100的时候，启动时总会听到扬声器出现一声“噗”的爆音，设置为0的时候，系统就没有声音了。
请大侠们帮帮忙，应该怎么解决这个问题？

Re: [DroidPhone](#) 2012-05-17 10:34发表



回复haokaihaohe110：这个应该查一下你的驱动中开启speaker时，codec中各个子部件的上电和unmute顺序，功放最好在最后打开，否则设置dac等参数时，可能造成异响。

Re: [haokaihaohe110](#) 2012-05-17 11:04发表



回复DroidPhone：博主，您好，谢谢您的回复。我使用的是TLV320的codec芯片，在aic3x_init函数里面有这么一段代码，好像是unmute的（我不是很懂驱动代码）：

```
/* unmute all outputs */
reg = snd_soc_read(codec, LLOPM_CTRL);
snd_soc_write(codec, LLOPM_CTRL, reg | UNMUTE);
reg = snd_soc_read(codec, RLOPM_CTRL);
snd_soc_write(codec, RLOPM_CTRL, reg | UNMUTE);
reg = snd_soc_read(codec, MONOLOPM_CTRL);
snd_soc_write(codec, MONOLOPM_CTRL, reg | UNMUTE);
reg = snd_soc_read(codec, HPLOUT_CTRL);
snd_soc_write(codec, HPLOUT_CTRL, reg | UNMUTE);
reg = snd_soc_read(codec, HPROUT_CTRL);
snd_soc_write(codec, HPROUT_CTRL, reg | UNMUTE);
reg = snd_soc_read(codec, HPLCOM_CTRL);
snd_soc_write(codec, HPLCOM_CTRL, reg | UNMUTE);
reg = snd_soc_read(codec, HPRCOM_CTRL);
snd_soc_write(codec, HPRCOM_CTRL, reg | UNMUTE);
```

我不清楚打开speaker和设置音量的位置。
还请博主帮帮忙啊。多谢了。

5楼 [jzp0409](#) 2012-05-08 11:22发表



请问一下您所介绍的这些资料所引用的内核版本是哪一个版本啊？

Re: [DroidPhone](#) 2012-05-08 12:20发表



回复jzp0409：如果没有记错，这篇好像是基于2.6.35的吧。

4楼 [hainei_](#) 2012-02-15 19:15发表



谢谢楼主了！

3楼 [allen6268198](#) 2011-09-19 10:08发表



so nice

2楼 [xubo830711](#) 2011-05-10 15:31发表



[e01]

1楼 [sxjyx2009](#) 2011-04-28 08:54发表



[e03]，持续关注中。。。。

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题	Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack
VPN	Spark	ERP	IE10	Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP
jQuery	BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS	Fedora
XML	LBS	Unity	Splashtop	UML	components	Windows Mobile	Rails	QEMU	KDE	Cassandra
CloudStack	FTC	coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo
Compuware	大数据	aptech	Perl	Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr
Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap					

[公司简介](#) | [招贤纳士](#) | [广告服务](#) | [银行汇款帐号](#) | [联系方式](#) | [版权声明](#) | [法律顾问](#) | [问题报告](#) | [合作伙伴](#) | [论坛反馈](#)

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持
京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

