

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：1077196次
积分：8768
等级：
排名：第1448名
原创：51篇 转载：0篇
译文：4篇 评论：537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 (74187)
Android Audio System 之 (58758)
Linux ALSA声卡驱动之二 (46387)
Android Audio System 之 (42720)
Linux ALSA声卡驱动之三 (41553)
Linux ALSA声卡驱动之四

【公告】博客系统优化升级 Unity3D学习，离VR开发还有一步 博乐招募开始啦 虚拟现实，一探究竟

ALSA声卡驱动中的DAPM详解之二：widget-具备路径和电源管理信息的kcontrol

2013-10-23 20:31

10804人阅读

评论(2)

收藏

举报

分类：Linux音频子系统 (14)

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(?)

上一篇文章中，我们介绍了音频驱动中对基本控制单元的封装：kcontrol。利用kcontrol，我们可以完成对音频系统中的mixer，mux，音量控制，音效控制，以及各种开关量的控制，通过对各种kcontrol的控制，使得音频硬件能够按照我们预想的结果进行工作。同时我们可以看到，kcontrol还是有以下几点不足：

- 只能描述自身，无法描述各个kcontrol之间的连接关系；
- 没有相应的电源管理机制；
- 没有相应的时间处理机制来响应播放、停止、上电、下电等音频事件；
- 为了防止pop-pop声，需要用户程序关注各个kcontrol上电和下电的顺序；
- 当一个音频路径不再有效时，不能自动关闭该路径上的所有的kcontrol；

/*****

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

/*****

为此，DAPM框架正是为了解决以上这些问题而诞生的，DAPM目前已经是ASoc中的重要组成部分，让我们先从DAPM的数据结构开始，了解它的设计思想和工作原理。

DAPM的基本单元：widget

文章的开头，我们说明了一下目前kcontrol的一些不足，而DAPM框架为了解决这些问题，引入了widget这一概念，所谓widget，其实可以理解为是kcontrol的进一步升级和封装，她同样是指音频系统中的某个部件，比如mixer，mux，输入输出引脚，电源供应器等等，甚至，我们可以定义虚拟的widget，例如playback stream widget。widget把kcontrol和动态电源管理进行了有机的结合，同时还具备音频路径的连接功能，一个widget可以与它相邻的widget有某种动态的连接关系。在DAPM框架中，widget用结构体snd_soc_dapm_widget来描述：

```
[cpp]
01. struct snd_soc_dapm_widget {
02.     enum snd_soc_dapm_type id;
03.     const char *name;          /* widget name */
04.
05.     .....
06.     /* dapm control */
07.     int reg;                   /* negative reg = no direct dapm */
08.     unsigned char shift;      /* bits to shift */
09.     unsigned int value;       /* widget current value */
10.     unsigned int mask;        /* non-shifted mask */
11.     .....
12.
13.     int (*power_check)(struct snd_soc_dapm_widget *w);
14.
15.     int (*event)(struct snd_soc_dapm_widget*, struct snd_kcontrol *, int);
16. }
```

Linux时间子系统之六：[高精度定时器](#) (37505)
[Android Audio System 之三：ASoc](#) (36411)
[Linux ALSA声卡驱动之十一：PCM](#) (36317)
[Linux ALSA声卡驱动之四：卡](#) (36146)
[Linux时间子系统之一：clock source](#) (31714)

评论排行

[Android Audio System 之三：ASoc](#) (56)
[Linux ALSA声卡驱动之十一：PCM](#) (42)
[Linux ALSA声卡驱动之九：卡](#) (35)
[Linux时间子系统之六：高精度定时器](#) (25)
[Linux中断（interrupt）子系统之十二：马金兴](#) (24)
[Android SurfaceFlinger](#) (21)
[Linux ALSA声卡驱动之十二：马金兴](#) (19)
[Android Audio System 之三：ASoc](#) (18)
[Linux ALSA声卡驱动之十一：PCM](#) (17)
[Linux中断（interrupt）子系统之十二：马金兴](#) (17)

推荐文章

[* 致JavaScript也将征服的物联网世界](#)
[* 从苏宁电器到卡巴斯基：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输入子系统：输入设备编程u012839187](#): 看着没有问题。你要弄清楚的是新的内核有许多函数宏的变动。
[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rea...
[Linux中断（interrupt）子系统之十二：马金兴](#): 大神！必须顶@！！！
[Linux时间子系统之三：时间的维度 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）子系统之十二：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，刚刚用豆约翰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：ASoc ss0429](#): 楼主的文章写的很精炼，多谢分享~
[Linux中断（interrupt）子系统之十二：马金兴](#): 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...

```
17.         /* kcontrols that relate to this widget */
18.         int num_kcontrols;
19.         const struct snd_kcontrol_new *kcontrol_news;
20.         struct snd_kcontrol **kcontrols;
21.
22.         /* widget input and outputs */
23.         struct list_head sources;
24.         struct list_head sinks;
25.         .....
26.     };
```

snd_soc_dapm_widget结构比较大，为了简洁一些，这里我没有列出该结构体的完整字段，不过不用担心，下面我会说明每个字段的意义：

id 该widget的类型值，比如snd_soc_dapm_output，snd_soc_dapm_mixer等等。

***name** 该widget的名字

***sname** 代表该widget所在stream的名字，比如对于snd_soc_dapm_dai_in类型的widget，会使用该字段。

***codec *platform** 指向该widget所属的codec和platform。

list 所有注册到系统中的widget都会通过该list，链接到代表声卡的snd_soc_card结构的widgets链表头字段中。

***dapm** snd_soc_dapm_context结构指针，ASoc把系统划分为多个dapm域，每个widget属于某个dapm域，同一个域代表着同样的偏置电压供电策略，比如，同一个codec中的widget通常位于同一个dapm域，而平台上的widget可能会位于另外一个platform域中。

***priv** 有些widget可能需要一些专有的数据，可以使用该字段来保存，像snd_soc_dapm_dai_in类型的widget，会使用该字段来记住与之相关联的snd_soc_dai结构指针。

***regulator** 对于snd_soc_dapm_regulator_supply类型的widget，该字段指向与之相关的regulator结构指针。

***params** 目前对于snd_soc_dapm_dai_link类型的widget，指向该dai的配置信息的snd_soc_pcm_stream结构。

reg shift mask 这3个字段用来控制该widget的电源状态，分别对应控制信息所在的寄存器地址，位移值和屏蔽值。

value on_val off_val 电源状态的当前只，开启时和关闭时所对应的值。

power invert 用于指示该widget当前是否处于上电状态，invert则用于表明power字段是否需要逻辑反转。

active connected 分别表示该widget是否处于激活状态和连接状态，当和相邻的widget有连接关系时，connected位会被置1，否则置0。

new 我们定义好的widget（snd_soc_dapm_widget结构），在注册到声卡中时需要进行实例化，该字段用来表示该widget是否已经被实例化。

ext 表示该widget当前是否有外部连接，比如连接mic，耳机，喇叭等等。

force 该位被设置后，将会不管widget当前的状态，强制更新至新的电源状态。

ignore_suspend new_power power_checked 这些电源管理相关的字段。

subseq 该widget目前在上电或下电队列中的排序编号，为了防止在上下电的过程中出现pop-pop声，DAPM会给每个widget分配合理的上下电顺序。

***power_check** 用于检查该widget是否应该上电或下电的回调函数指针。

event_flags 该字段是一个位或字段，每个位代表该widget会关注某个DAPM事件通知。只有被关注的通知事件会被发送到widget的事件处理回调函数中。

***event** DAPM事件处理回调函数指针。

num_kcontrols *kcontrol_news **kcontrols 这3个字段用来描述与该widget所包含的kcontrol控件，例如一个mixer控件或者是一个mux控件。

sources sinks 两个链表字段，两个widget如果有连接关系，会通过一个snd_soc_dapm_path结构进行连接，sources链表用于链接所有的输入path，sinks链表用于链接所有的输出path。



声卡驱动



pos机刷卡手续



上海专升本



呼叫中心系统

power_list 每次更新整个dapm的电源状态时，会根据一定的**算法**扫描所有的widget，然后把需要变更电源状态的widget利用该字段链接到一个上电或下电的链表中，扫描完毕后，dapm系统会遍历这两个链表执行相应的上电或下电操作。

dirty 链表字段，widget的状态变更后，dapm系统会利用该字段，把该widget加入到一个dirty链表中，稍后会对dirty链表进行扫描，以执行整个路径的更新。

inputs 该widget的所有有效路径中，连接到输入端的路径数量。

outputs 该widget的所有有效路径中，连接到输出端的路径数量。

***clk** 对于snd_soc_dapm_clock_supply类型的widget，指向相关联的clk结构指针。

以上我们对snd_soc_dapm_widget结构的各个字段所代表的意义一一做出了说明，这里只是让大家现有个概念，至于每个字段的详细作用，我们会在以后相关的章节中提及。

widget的种类

在DAPM框架中，把各种不同的widget划分为不同的种类，snd_soc_dapm_widget结构中的id字段用来表示该widget的种类，可选的种类都定义在一个枚举中：

```
[cpp]
01. /* dapm widget types */
02. enum snd_soc_dapm_type {.....}
```

下面我们逐个解释一下这些widget的种类：

- **snd_soc_dapm_input** 该widget对应一个输入引脚。
- **snd_soc_dapm_output** 该widget对应一个输出引脚。
- **snd_soc_dapm_mux** 该widget对应一个mux控件。
- **snd_soc_dapm_virt_mux** 该widget对应一个虚拟的mux控件。
- **snd_soc_dapm_value_mux** 该widget对应一个value类型的mux控件。
- **snd_soc_dapm_mixer** 该widget对应一个mixer控件。
- **snd_soc_dapm_mixer_named_ctl** 该widget对应一个mixer控件，但是对应的kcontrol的名字不会加入widget的名字作为前缀。
- **snd_soc_dapm_pga** 该widget对应一个pga控件（可编程增益控件）。
- **snd_soc_dapm_out_drv** 该widget对应一个输出驱动控件
- **snd_soc_dapm_adc** 该widget对应一个ADC
- **snd_soc_dapm_dac** 该widget对应一个DAC
- **snd_soc_dapm_micbias** 该widget对应一个麦克风偏置电压控件
- **snd_soc_dapm_mic** 该widget对应一个麦克风。
- **snd_soc_dapm_hp** 该widget对应一个耳机。
- **snd_soc_dapm_spk** 该widget对应一个扬声器。
- **snd_soc_dapm_line** 该widget对应一个线路输入。
- **snd_soc_dapm_switch** 该widget对应一个模拟开关。
- **snd_soc_dapm_vmid** 该widget对应一个codec的vmid偏置电压。
- **snd_soc_dapm_pre** machine级别的专用widget，会先于其它widget执行检查操作。
- **snd_soc_dapm_post** machine级别的专用widget，会后于其它widget执行检查操作。
- **snd_soc_dapm_supply** 对应一个电源或是时钟源。
- **snd_soc_dapm_regulator_supply** 对应一个外部regulator稳压器。
- **snd_soc_dapm_clock_supply** 对应一个外部时钟源。
- **snd_soc_dapm_aif_in** 对应一个数字音频输入接口，比如I2S接口的输入端。
- **snd_soc_dapm_aif_out** 对应一个数字音频输出接口，比如I2S接口的输出端。
- **snd_soc_dapm_siggen** 对应一个信号发生器。
- **snd_soc_dapm_dai_in** 对应一个platform或codec域的输入DAI结构。
- **snd_soc_dapm_dai_out** 对应一个platform或codec域的输出DAI结构。
- **snd_soc_dapm_dai_link** 用于链接一对输入/输出DAI结构。

widget之间的连接器：path

之前已经提到，一个widget是有输入和输出的，而且widget之间是可以动态地进行连接的，那它们是用什么来连接两个widget的呢？DAPM为我们提出了path这一概念，path相当于电路中的一根跳线，它把一个widget的输出端和另一个widget的输入端连接在一起，path用snd_soc_dapm_path结构来描述：

```
[cpp]
01. struct snd_soc_dapm_path {
02.     const char *name;
03.
04.     /* source (input) and sink (output) widgets */
05.     struct snd_soc_dapm_widget *source;
06.     struct snd_soc_dapm_widget *sink;
07.     struct snd_kcontrol *kcontrol;
08.
09.     /* status */
10.     u32 connect:1; /* source and sink widgets are connected */
11.     u32 walked:1; /* path has been walked */
12.     u32 walking:1; /* path is in the process of being walked */
13.     u32 weak:1; /* path ignored for power management */
14.
15.     int (*connected)(struct snd_soc_dapm_widget *source,
16.                     struct snd_soc_dapm_widget *sink);
17.
18.     struct list_head list_source;
19.     struct list_head list_sink;
20.     struct list_head list;
21. };
```

当widget之间发生连接关系时，snd_soc_dapm_path作为连接者，它的source字段会指向该连接的起始端widget，而它的sink字段会指向该连接的到达端widget，还记得前面snd_soc_dapm_widget结构中的两个链表头字段：sources和sinks么？widget的输入端和输出端可能连接着多个path，所有输入端的snd_soc_dapm_path结构通过list_sink字段挂在widget的sources链表中，同样，所有输出端的snd_soc_dapm_path结构通过list_source字段挂在widget的sinks链表中。这里可能大家会被搞得晕呼呼的，一会source，一会sink，不要紧，只要记住，连接的路径是这样的：起始端widget的输出-->path的输入-->path的输出-->到达端widget输入。

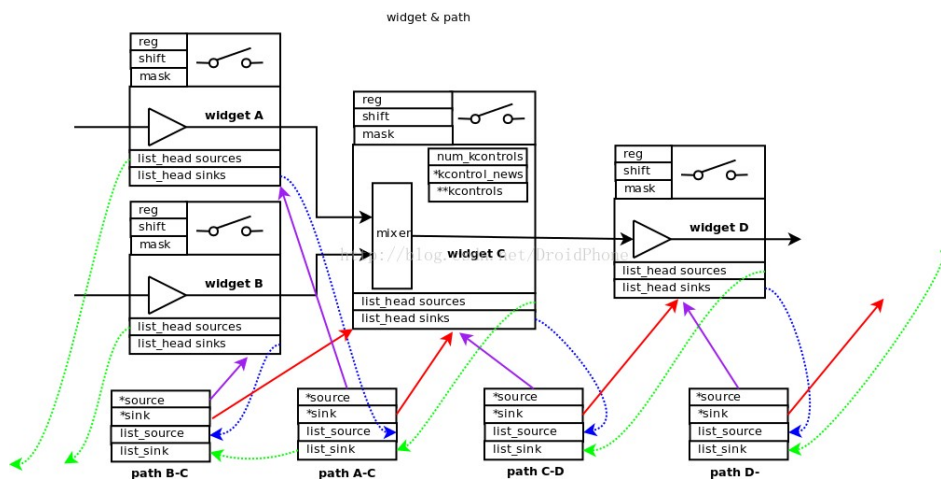


图1 widget通过path进行连接

另外，snd_soc_dapm_path结构的list字段用于把所有的path注册到声卡中，其实就是挂在snd_soc_card结构的paths链表头字段中。如果你要自己定义方法来检查path的当前连接状态，你可以提供自己的connected回调函数指针。

connect, walked, walking, weak是几个辅助字段，用于帮助所有path的遍历。

widget的连接关系：route

通过上一节的内容，我们知道，一个路径的连接至少包含以下几个元素：起始端widget，跳线path，到达端widget，在DAPM中，用snd_soc_dapm_route结构来描述这样一个连接关系：

```
[cpp]
01. struct snd_soc_dapm_route {
02.     const char *sink;
```

```
03.         const char *control;
04.         const char *source;
05.         int (*connected)(struct snd_soc_dapm_widget *source,
06.                           struct snd_soc_dapm_widget *sink);
07.     };
```

sink指向到达端widget的名字字符串，source指向起始端widget的名字字符串，control指向负责控制该连接所对应的kcontrol名字字符串，connected回调则定义了上一节所提到的自定义连接检查回调函数。该结构的意义很明显就是：source通过一个kcontrol，和sink连接在一起，现在是否处于连接状态，请调用connected回调函数检查。

这里直接使用名字字符串来描述连接关系，所有定义好的route，最后都要注册到dapm系统中，dapm会根据这些名字找出相应的widget，并动态地生成所需要的snd_soc_dapm_path结构，正确地处理各个链表和指针的关系，实现两个widget之间的连接，具体的连接代码分析，我们留到以后的章节中讨论。

顶 踩
1 0

上一篇 ALSA声卡驱动中的DAPM详解之一：kcontrol
下一篇 ALSA声卡驱动中的DAPM详解之三：如何定义各种widget

我的同类文章

Linux音频子系统（14）

• ALSA声卡驱动中的DAPM详... 2013-11-09 阅读 8979

• ALSA声卡驱动中的DAPM详... 2013-11-04 阅读 9950

• ALSA声卡驱动中的DAPM详... 2013-11-04 阅读 12941

• ALSA声卡驱动中的DAPM详... 2013-11-01 阅读 10222

• ALSA声卡驱动中的DAPM详... 2013-10-24 阅读 12568

• ALSA声卡驱动中的DAPM详... 2013-10-18 阅读 17125

• Linux ALSA声卡驱动之八：... 2012-03-13 阅读 31315

• Linux ALSA声卡驱动之七：... 2012-02-23 阅读 36156

• Linux ALSA声卡驱动之六：... 2012-02-03 阅读 37515

• Linux ALSA声卡驱动之五：... 2012-01-17 阅读 26715

更多文章

参考知识库



算法与数据结构知识库

2034 关注 | 3500 收录


猜你在找

- 【直通华为HCNA/HCNP系列R篇3】路由器接口配置与管理
- ALSA声卡驱动中的DAPM详解之一kcontrol
- 嵌入式Linux项目实战：三个大项目（数码相框、摄像头驱动）
- ALSA声卡驱动中的DAPM详解之一kcontrol
- Struts实战-使用SSH框架技术开发学籍管理系统
- ALSA声卡驱动中的DAPM详解之一kcontrol
- 系统集成项目管理工程师考试感性理性认识
- ALSA声卡驱动中的DAPM详解之一kcontrol
- 2016软考系统集成项目管理工程师视频教程精讲 基础知
- ALSA声卡驱动中的DAPM详解之一kcontrol



查看评论

2楼 jeffreylieu 2013-11-05 20:40发表



博主如遇到LINE IN输入的时候(用作模拟输入通道非MIC) 在切换到该通道的时候发现该条路径上的寄存器值不能更新过来只有当打开录音流的时候才会更新寄存器的值,组件注册如下：
/* Input path */

SND_SOC_DAPM_ADC("ADC Left", "Capture", CS42L52_PWRCTL1, 1, 1),
SND_SOC_DAPM_ADC("ADC Right", "Capture", CS42L52_PWRCTL1, 2, 1),
如果去掉Capture填充NULL参数CS42L52_PWRCTL1寄存器的值会更新但是该PATH的下游PATH又不能导通 请问这是什么情况？

1楼 [Color_Fly](#) 2013-11-01 09:37发表



抢个沙发，感谢博主分享！！！

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 