## Audio驱动和框架笔记

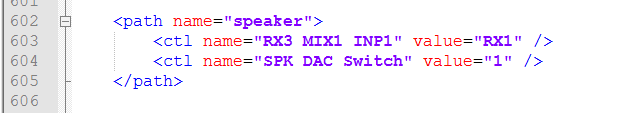
## 框架相关

关注system/media/audio\_route目录。主要是audio\_route.c文件。

int audio\_route\_apply\_path(struct audio\_route \*ar, const char \*name)

实现相关path对应的ctrl的设置。path\_print可以打印设置的ctrl。

设置是根据hardware/qcom/audio/configs/msm8909/mixer\_paths.xml文件来的，比如当前播放mp3，打开的通道path对应：



那么在：

static void path\_print(struct audio\_route \*ar, struct mixer\_path \*path)

{

unsigned int i;

unsigned int j;

ALOGE("Path: %s, length: %d", path->name, path->length);

for (i = 0; i < path->length; i++) {

struct mixer\_ctl \*ctl = index\_to\_ctl(ar, path->setting[i].ctl\_index);

ALOGE(" id=%d: ctl=%s", i, mixer\_ctl\_get\_name(ctl));

if (mixer\_ctl\_get\_type(ctl) == MIXER\_CTL\_TYPE\_BYTE) {

for (j = 0; j < path->setting[i].num\_values; j++)

ALOGE(" id=%d value=0x%02x", j, path->setting[i].value.bytes[j]);

} else if (mixer\_ctl\_get\_type(ctl) == MIXER\_CTL\_TYPE\_ENUM) {

for (j = 0; j < path->setting[i].num\_values; j++)

ALOGE(" id=%d value=%d", j, path->setting[i].value.enumerated[j]);

} else {

for (j = 0; j < path->setting[i].num\_values; j++)

ALOGE(" id=%d value=%ld", j, path->setting[i].value.integer[j]);

}

}

}

中，path->name就是“speaker”，ctl分别对应上面的ctl name。值就是对应上面的xml的value。

至于xml中的字符串，代码里面对应的是根据字符串在数组的序号来设的，可以在msm8x16-wcd.c中查到，或者直接tinymix命令获取。

关注hardware/qcom/audio/hal/msm8916/下的：

hw\_info.c platform.c

## 驱动相关

相关log打开：

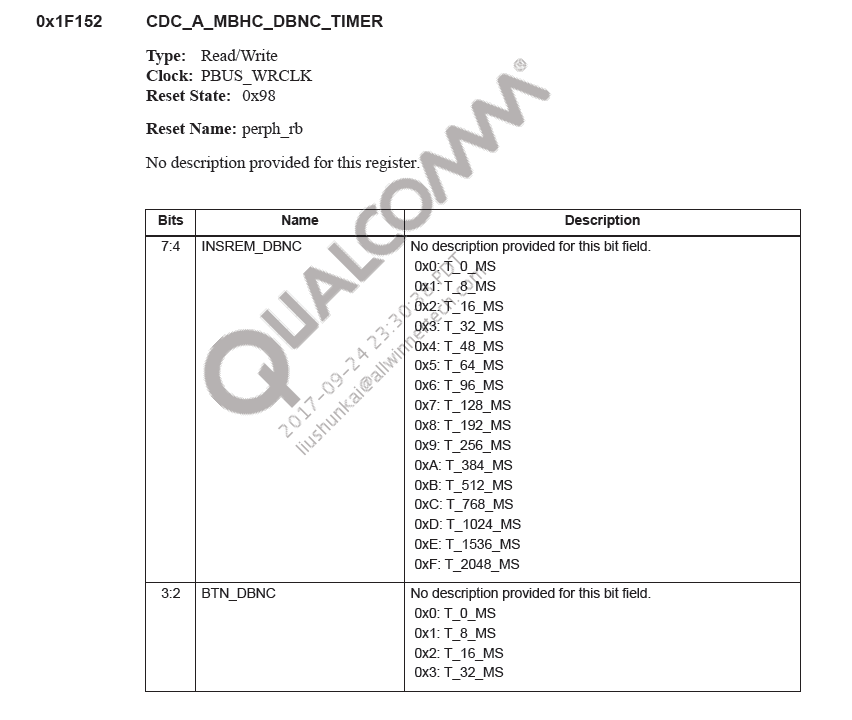
echo “file wcd-mbhc-v2.c +p” > /sys/kernel/debug/dynamic\_debug/control   
echo “file msm8x16-wcd.c +p” > /sys/kernel/debug/dynamic\_debug/control   
echo “file msm8916-wcd-irq.c +p” > /sys/kernel/debug/dynamic\_debug/control   
echo "file spmi-pmic-arb.c +p" > /sys/kernel/debug/dynamic\_debug/control   
echo “file qpnp-int.c +p” > /sys/kernel/debug/dynamic\_debug/control

模拟编解码端（pmic）的驱动在wcd-mbhc-v2.c：

/\* Insertion debounce set to 256ms \*/

WCD\_MBHC\_REG\_UPDATE\_BITS(WCD\_MBHC\_INSREM\_DBNC, 9);

深圳硬件debounce时间，即多久后插入/拔出动作才识别到。



耳机插拔irq入口：

wcd\_mbhc\_mech\_plug\_detect\_irq

msm8x16-wcd.c:

msm8x16\_wcd\_codec\_enable\_spk\_pa,每次开启或者关闭speaker前，都会调用此函数。

开启的事件：

#define SND\_SOC\_DAPM\_PRE\_PMU 0x1 /\* before widget power up \*/

#define SND\_SOC\_DAPM\_POST\_PMU 0x2 /\* after widget power up \*/

关闭的事件：

#define SND\_SOC\_DAPM\_PRE\_PMD 0x4 /\* before widget power down \*/

#define SND\_SOC\_DAPM\_POST\_PMD 0x8 /\* after widget power down \*/