Linux3.0.8平台搭建移植文档——alsa声卡驱动移植

1）添加 ac97控制器时钟

ac97.c 文件对应着 s5pv210的 ac97控制器驱动程序，并且采用了 platform 机制，在 platform_driver 的 probe 回调函数中会执行 clk_get(&pdev->dev, "ac97");函数获取 ac97的时钟，因此我们需要进行 ac97时钟的添加。

#vi arch/arm/mach-s5pv210/clock.c

在 init_clocks_off 数组开头添加如下成员：

```
{
        .name           = "ac97",
        .id             = -1,
        .parent         = &clk_pclk_psys.clk,
        .enable         = s5pv210_clk_ip3_ctrl,
        .ctrlbit        = (1 << 1),
},
```

2）修改 wm9713 codec 驱动

#vi sound/soc/codecs/wm9713.c

Step1：修改 wm9713_set_pll()函数中 ac98_write 函数参数：

```
static int wm9713_set_pll(struct snd_soc_codec *codec,int pll_id, unsigned int freq_in, unsigned int freq_out)
{
    ...
    /* turn PLL on and select as source */
    reg = ac97_read(codec, AC97_EXTENDED_MID);
    ac97_write(codec, AC97_EXTENDED_MID, reg & 0x7dff); //修改参数
    reg = ac97_read(codec, AC97_HANDSET_RATE);
    ac97_write(codec, AC97_HANDSET_RATE, reg & 0xff7f);
    wm9713->pll_in = freq_in;
    /* wait 10ms AC97 link frames for the link to stabilise */
    schedule_timeout_interruptible(msecs_to_jiffies(10));
    return 0;
}
```

Step2：在 wm9713_pcm_hw_params()函数前面,添加以下函数:

```
static int wm9713_hifi_hw_params(struct snd_pcm_substream *substream,
                    struct snd_pcm_hw_params *params,
                    struct snd_soc_dai *dai)
{
        struct snd_soc_codec *codec = dai->codec;
        ac97_write(codec, AC97_POWERDOWN, 0x0000);
        ac97_write(codec, AC97_PHONE, 0x0808);
        /* hcj for dbg */
        ac97_write(codec, AC97_EXTENDED_MID, 0x7803);
        ac97_write(codec, AC97_EXTENDED_MSTATUS, 0xb990);
```

```
        ac97_write(codec, AC97_MASTER, 0x8080);
        ac97_write(codec, AC97_HEADPHONE, 0x0606);
        ac97_write(codec, AC97_REC_GAIN, 0x00aa);
#ifdef CONFIG_SOUND_WM9713_INPUT_STREAM_MIC
        ac97_write(codec, 0x5c, 0x0002);
        ac97_write(codec, AC97_LINE, 0x0068);
        ac97_write(codec, AC97_VIDEO, 0xfe00);
#else
        ac97_write(codec, AC97_VIDEO, 0xd612);
#endif
        return 0;
}
```

Step3：在 wm9713_dai_ops_hifi 结构体成员初始化中添加如下内容：

```
static struct snd_soc_dai_ops wm9713_dai_ops_hifi = {
        .hw_params      = wm9713_hifi_hw_params,    //add by sunpluss
        .prepare      = ac97_hifi_prepare,
        .set_clkdiv      = wm9713_set_dai_clkdiv,
        .set_pll      = wm9713_set_dai_pll,
};
```

Step4：修改 wm9713_reset()函数如下：

```
int wm9713_reset(struct snd_soc_codec *codec, int try_warm)
{
/*      if (try_warm && soc_ac97_ops.warm_reset) {
                soc_ac97_ops.warm_reset(codec->ac97);
                if (ac97_read(codec, 0) == wm9713_reg[0])
                        return 1;
        }
        soc_ac97_ops.reset(codec->ac97);
        if (soc_ac97_ops.warm_reset)
                soc_ac97_ops.warm_reset(codec->ac97);
        if (ac97_read(codec, 0) != wm9713_reg[0])
                return -EIO;
*/
        if (try_warm && soc_ac97_ops.warm_reset) {
                while (ac97_read(codec, 0) != wm9713_reg[0])
                {
                        soc_ac97_ops.warm_reset(codec->ac97);
                }

                //if (ac97_read(codec, 0) == wm9713_reg[0])
                        return 1;
        }
```

```
        return 0;
    }
```

Step5: 修改函数 wm9713_set_bias_level()如下：

```
static int wm9713_set_bias_level(struct snd_soc_codec *codec, enum snd_soc_bias_level level)
{
    ...
    case SND_SOC_BIAS_OFF:
    /* disable everything including AC link */
     ac97_write(codec, AC97_EXTENDED_MID, 0x7fff);
     ac97_write(codec, AC97_EXTENDED_MSTATUS, 0xffff);
     ac97_write(codec, AC97_POWERDOWN, 0xffff);
    ...
}
```

3)make menuconfig 配置 alsa 声卡驱动

```
#vi make menuconfig

Device Drivers  --->
    [*]Sound card support  --->
        <*>   Advanced Linux Sound Architecture  --->//选中该选项支持 alsa 总线驱动
        < >   Open Sound System (DEPRECATED)  --->
```

进入 Advanced Linux Sound Architecture 选项配置如下：

```
--- Advanced Linux Sound Architecture
< >   Sequencer support
<*>   OSS Mixer API
<*>   OSS PCM (digital audio) API
[*]     OSS PCM (digital audio) API - Include plugin system
< >   HR-timer backend support
[ ]   Dynamic device file minor numbers
[*]   Support old ALSA API
[*]   Verbose procfs contents
[ ]   Verbose printk
[ ]   Debug
[ ]   Generic sound devices  --->
[ ]   ARM sound devices  --->
<*>   ALSA for SoC audio support  --->
          --- ALSA for SoC audio support
          [ ]   Support LZO compression for register caches
          <*>   ASoC support for Samsung
          < >   SoC I2S Audio support for WM8580 on SMDK
          <*>   SoC AC97 Audio support for SMDK with WM9713
          < >   SoC S/PDIF Audio support for SMDK
          < >   SoC PCM Audio support for WM8580 on SMDK
           < >   Build all ASoC CODEC drivers
```

4)make

将在arch/arm/boot/下生成编译好的可执行程序zImage下载到开发板即可，成功移植后会在看到ls /dev/dsp这个设备文件，执行命令“echo /dev/zero > /dev/dsp”可以听到“吱”的声音，也可以通过mplayer播放一首音乐进行测试