sprd\_headset\_parse

-->priv->gpio\_hp\_det = of\_get\_named\_gpio\_flags(node,

"sprd-audio-card,hp-det-gpio",

0, &flags);

--->priv->gpio\_mic\_det = of\_get\_named\_gpio\_flags(node,

"sprd-audio-card,mic-det-gpio",

0, &flags);//没有打印

sprd\_headset\_probe

--->ret = sprd\_headset\_parse\_dt(hdst);

--->ret = of\_property\_read\_u32(np, "sprd,jack-type", &val);

--->pdata->jack\_type = val ? JACK\_TYPE\_NC : JACK\_TYPE\_NO;

--->index = of\_property\_match\_string(np, "gpio-names", name);

--->index = of\_property\_match\_string(np, "gpio-names", name);

--->ret = of\_get\_gpio\_flags(np, index, NULL);

--->pdata->gpios[type] = (u32)ret;

--->ret = of\_property\_read\_u32(np, "sprd,3pole-adc-threshold",

&pdata->threshold\_3pole);

--->et = of\_property\_read\_u32(np, "sprd,adc-gnd",

&pdata->sprd\_adc\_gnd);

--->ret = of\_property\_read\_u32(np, "sprd,half-adc-gnd",

&pdata->sprd\_half\_adc\_gnd);

--->ret = of\_property\_read\_u32(np, "sprd,coefficient",

&pdata->coefficient);

--->ret = of\_property\_read\_u32(

np, "sprd,irq-threshold-button", &pdata->irq\_threshold\_button);

--->pdata->sprd\_one\_half\_adc\_gnd = pdata->sprd\_adc\_gnd +

pdata->sprd\_half\_adc\_gnd;

--->/\* Get the adc channels of headset. \*/

--->hdst->adc\_chan = iio\_channel\_get(dev, "headmic\_in\_little");

--->devm\_gpio\_request(dev, pdata->gpios[type], name);

--->sprd\_hdst = hdst;

ret = sprd\_headset\_soc\_probe(codec);

--->et = sprd\_headset\_parse(card);

--->hdst->codec = codec;

--->adie\_chip\_id = sci\_get\_ana\_chip\_id() >> 16;

--->ret = sprd\_headset\_power\_init(&hdst->power\_manager, pdev);

--->ret = sprd\_detect\_reg\_init();

--->if (pdata->jack\_type == JACK\_TYPE\_NO)

headset\_reg\_clr\_bits(ANA\_HDT0, HEDET\_JACK\_TYPE);

else if (pdata->jack\_type == JACK\_TYPE\_NC)

headset\_reg\_set\_bits(ANA\_HDT0, HEDET\_JACK\_TYPE);

--->ret = snd\_soc\_card\_jack\_new(card, "Headset Jack",

SPRD\_HEADSET\_JACK\_MASK, &hdst->hdst\_jack, NULL, 0);

--->ret = snd\_soc\_card\_jack\_new(card, "Headset Keyboard",

SPRD\_BUTTON\_JACK\_MASK, &hdst->btn\_jack, NULL, 0);

--->hdst->irq\_detect\_int\_all =

gpio\_to\_irq(pdata->gpios[HDST\_GPIO\_AUD\_DET\_INT\_ALL]);

--->INIT\_DELAYED\_WORK(&hdst->det\_mic\_work, sprd\_mdet\_eic\_work);

--->hdst->det\_mic\_work\_q = create\_singlethread\_workqueue("headset\_mic");

--->INIT\_DELAYED\_WORK(&hdst->btn\_work, headset\_button\_work\_func);//耳机检测

--->hdst->btn\_work\_q = create\_singlethread\_workqueue("headset\_button");//按键检测

--->INIT\_DELAYED\_WORK(&hdst->det\_all\_work, headset\_detect\_all\_work\_func);

--->hdst->det\_all\_work\_q =

create\_singlethread\_workqueue("headset\_detect\_all");

--->INIT\_DELAYED\_WORK(&hdst->ldetl\_work, headset\_ldetl\_work\_func);

--->hdst->ldetl\_work\_q =

create\_singlethread\_workqueue("headset\_ldetl");

--->ret = devm\_request\_threaded\_irq(

dev, hdst->irq\_detect\_int\_all, NULL,

sprd\_headset\_top\_eic\_handler,

IRQF\_TRIGGER\_HIGH | IRQF\_NO\_SUSPEND | IRQF\_ONESHOT,

"head\_aud\_det\_int\_all", hdst);

--->sprd\_hmicbias\_hw\_control\_enable(true, pdata);

--->headset\_reg\_set\_bits(ANA\_HDT2, HEDET\_MDET\_EN);--->D8

--->sprd\_headset\_eic\_plugin\_enable();

1:sprd\_mdet\_eic\_work

--->mdet\_insert = sprd\_headset\_part\_is\_inserted(HDST\_INSERT\_MDET);

--->/\* disable MDET \*/

sprd\_headset\_eic\_enable(HDST\_MDET\_EIC, false);

sprd\_headset\_eic\_clear(HDST\_MDET\_EIC);

sprd\_intc\_force\_clear(false, ANA\_INT\_CLR);

--->headset\_reg\_read(ANA\_STS0, &val);

**2:headset\_button\_work\_func**

**--->btn\_irq\_trig\_level = sprd\_headset\_button\_status();**

**--->if (btn\_irq\_trig\_level == BTN\_PRESS) {**

**sprd\_set\_eic\_trig\_level(15, 0);**

**sprd\_enable\_hmicbias\_polling(false, false);**

**mutex\_lock(&hdst->btn\_detecting\_lock);**

**hdst->btn\_detecting = true;**

**mutex\_unlock(&hdst->btn\_detecting\_lock);**

**} else if (btn\_irq\_trig\_level == BTN\_RELEASE) {**

**sprd\_set\_eic\_trig\_level(15, 1);**

**mutex\_lock(&hdst->btn\_detecting\_lock);**

**hdst->btn\_detecting = false;**

**mutex\_unlock(&hdst->btn\_detecting\_lock);**

**mutex\_lock(&hdst->audio\_on\_lock);**

**if (!hdst->audio\_on)**

**sprd\_enable\_hmicbias\_polling(true, false);**

**mutex\_unlock(&hdst->audio\_on\_lock);**

**}**

**--->if (btn\_irq\_trig\_level == BTN\_PRESS)**

**sprd\_headset\_button\_press(hdst);**

**else if (btn\_irq\_trig\_level == BTN\_RELEASE)**

**sprd\_headset\_button\_release(hdst);**

**sprd\_headset\_button\_eic\_reenable();**

**/\* wake\_unlock(&hdst->btn\_wakelock); \*/**

**up(&hdst->sem);**

**3:headset\_detect\_all\_work\_func**

**--->insert\_all\_data\_last =**

**sprd\_headset\_eic\_get\_data(HDST\_INSERT\_ALL\_EIC);**

**--->return (BIT(eic\_type) & sprd\_read\_reg\_value(ANA\_INT0)) > 0;**

**--->ret = sprd\_headset\_valid\_insert\_all(hdst, 40);**

**--->val = sprd\_get\_eic\_mis\_status(HDST\_INSERT\_ALL\_EIC);**

**--->data\_last = sprd\_headset\_eic\_get\_data(HDST\_INSERT\_ALL\_EIC);**

**--->data\_current = sprd\_headset\_eic\_get\_data(HDST\_INSERT\_ALL\_EIC);**

**--->trig\_level = sprd\_get\_eic\_trig\_level(10);**

**--->insert\_status =**

**sprd\_headset\_part\_is\_inserted(HDST\_INSERT\_ALL);**

**--->sprd\_headset\_set\_hw\_status(hdst, pdata);**

**--->sprd\_enable\_hmicbias\_polling(false, true);**

**--->sprd\_headset\_type\_report(hdst);**

**--->headset\_type = sprd\_headset\_get\_type();**

**--->case HEADSET\_NO\_MIC:**

**--->queue\_delayed\_work(hdst->det\_all\_work\_q,**

**&hdst->det\_all\_work,**

**msecs\_to\_jiffies(1000));**

**4:headset\_ldetl\_work\_func**

**--->sprd\_codec\_intc\_irq(hdst->codec, codec\_intc);**

**--->val = sprd\_get\_eic\_mis\_status(12);**

**--->ldetl\_data\_last = sprd\_headset\_eic\_get\_data(12);**

**---.>ldetl\_data\_current = sprd\_headset\_eic\_get\_data(12);**

**--->hdst->ldetl\_trig\_val\_last = sprd\_get\_eic\_trig\_level(12);**

**--->insert\_status = sprd\_headset\_part\_is\_inserted(HDST\_INSERT\_LDETL);**

**---.>sprd\_hmicbias\_hw\_control\_enable(false, pdata);**

**---》sprd\_headset\_eic\_enable(12, 0);**

**--->sprd\_set\_eic\_trig\_level(12, 0);**

**--->sprd\_hmicbias\_hw\_control\_enable(true, pdata);**

**--->sprd\_headset\_prepare\_ldetl();**

**5:sprd\_headset\_top\_eic\_handler**

**--->val\_intc = sprd\_intc\_irq\_status();**

**--->eic\_mis = sprd\_get\_all\_eic\_mis\_status();**

**--->sprd\_intc\_force\_clear(true, val\_intc);**

**--->irq\_set\_irq\_type(hdst->irq\_detect\_int\_all, IRQF\_TRIGGER\_HIGH);**

**--->sprd\_codec\_intc\_status\_check(val\_intc)**

**--->queue\_delayed\_work(hdst->ldetl\_work\_q, &hdst->ldetl\_work, 0);**

**--->sprd\_headset\_eic\_mis\_check(eic\_mis)**

**--->sprd\_headset\_reset(hdst);**

**--->queue\_delayed\_work(hdst->det\_all\_work\_q,**

**&hdst->det\_all\_work, msecs\_to\_jiffies(0));**

**--->queue\_delayed\_work(hdst->det\_mic\_work\_q,**

**&hdst->det\_mic\_work, msecs\_to\_jiffies(5));**

**--->queue\_delayed\_work(hdst->ldetl\_work\_q,**

**&hdst->ldetl\_work, msecs\_to\_jiffies(0));**

**--->queue\_delayed\_work(hdst->btn\_work\_q,**

**&hdst->btn\_work, msecs\_to\_jiffies(0));**

**---**