Clock-lc1860.h (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\mach-lc186x\clock)

static struct clk com\_pcm\_clk = {

.name = "pcm\_clk",

.parent = &com\_pre\_clk,

.cust = &clk\_com\_pcm\_table,

.mclk\_reg = (void \_\_iomem \*)io\_p2v(DDR\_PWR\_CLKEN0),

.mclk\_bit = 4,

.mclk\_we\_bit = 20,

.ifclk\_reg = (void \_\_iomem \*)io\_p2v(DDR\_PWR\_PCLKEN),

.ifclk\_bit = 4,

.ifclk\_we\_bit = 20,

.divclk\_reg = (void \_\_iomem \*)io\_p2v(DDR\_PWR\_PCMCLKDIV),

.init = &fdiv\_init\_by\_table,

.enable = &clk\_enable\_generic,

.disable = &clk\_disable\_generic,

.set\_rate = &fdiv\_set\_rate\_by\_table,

};

Board-common.c (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\mach-lc186x\board)

MACHINE\_START(LC186X, "Leadcore Innopower")

--->.init\_machine = comip\_init,xxxx

--->comip\_init

--->comip\_init\_virtual\_pcm();

--->comip\_init\_pcm();

--->comip\_set\_pcm\_info();

--->comip\_register\_device(&comip\_device\_pcm, NULL);

--->comip\_init\_common\_devices();

--->platform\_add\_devices(common\_devices, ARRAY\_SIZE(common\_devices));

--->&comip\_codec\_device,

--->&snd\_lowpower\_device,

--->&comip\_snd\_soc\_device,

platform\_driver平台驱动comip-pcm

Comip-pcm.c (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\plat-lc\drivers\sound)

comip\_pcm\_init

--->platform\_driver\_probe(&comip\_pcm\_driver, comip\_pcm\_probe);

----> comip\_pcm\_driver = {

.remove = comip\_pcm\_remove,

.driver = {

.name = "comip-pcm",//与设备comip\_device\_pcm对应

.owner = THIS\_MODULE,

},

};

--->comip\_pcm\_probe

--->mmres = platform\_get\_resource(pdev, IORESOURCE\_MEM, 0);//获取设备地址PCM\_BASE

---->dmarxres = platform\_get\_resource(pdev, IORESOURCE\_DMA, 0);//dma地址

--->dmatxres = platform\_get\_resource(pdev, IORESOURCE\_DMA, 1);

--->pcm->reg\_base = io\_p2v(mmres->start);//pcm地址映射

--->platform\_set\_drvdata(pdev, pcm);

--->comip\_pcm\_stereo[0].channel = dmatxres->start;//pcm out dma通道

--->comip\_pcm\_stereo[0].daddr = mmres->start + PCM\_TRAN\_FIFO;//pcm out dma 目标物理地址

--->comip\_pcm\_stereo[1].channel = dmarxres->start;//pcm in dma通道

--->comip\_pcm\_stereo[1].saddr = mmres->start + PCM\_REC\_FIFO; //pcm in dma 源物理地址

声卡component注册

--->snd\_soc\_register\_component(&pdev->dev, &lc186x\_pcm\_component, --->.name = "comip-pcm",

&comip\_pcm\_dai, 1);

--->comip\_pcm\_dai

--->struct snd\_soc\_dai\_driver comip\_pcm\_dai = {

.playback = {

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_PCM\_RATES,

.formats = SNDRV\_PCM\_FMTBIT\_S8 | SNDRV\_PCM\_FMTBIT\_S16\_LE,

},

.capture = {

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_PCM\_RATES,

.formats = SNDRV\_PCM\_FMTBIT\_S8 | SNDRV\_PCM\_FMTBIT\_S16\_LE,

},

.ops = &comip\_pcm\_dai\_ops,

};

--->comip\_pcm\_dai\_ops = {

.trigger = comip\_pcm\_trigger,

.hw\_params = comip\_pcm\_hw\_params,

--->cpu\_dai->playback\_dma\_data = &comip\_pcm\_stereo[0];

--->cpu\_dai->capture\_dma\_data = &comip\_pcm\_stereo[1];

.set\_fmt = comip\_pcm\_set\_fmt,

.startup = comip\_pcm\_startup,

--->pcm->clk = clk\_get(dai->dev, "pcm\_clk");//时钟使能

--->clk\_set\_rate(pcm->clk, COMIP\_PCM\_CLK\_RATE);

--->clk\_enable(pcm->clk);

.shutdown = comip\_pcm\_shutdown,

};

---> leadcore\_asoc\_pcm\_dma\_register(&pdev->dev);//dma

platform注册

--->snd\_soc\_register\_platform(dev, &leadcore\_asoc\_platform);

--->struct snd\_soc\_platform\_driver leadcore\_asoc\_platform = {

.ops = &comip\_pcm\_dma\_ops,

--->comip\_pcm\_dma\_ops = {

.open = comip\_pcm\_open,

--->snd\_soc\_set\_runtime\_hwparams(substream, &comip\_pcm\_hardware);

--->snd\_pcm\_hw\_constraint\_integer(runtime, SNDRV\_PCM\_HW\_PARAM\_PERIODS);

--->runtime->private\_data = prtd;

.close = comip\_pcm\_close,

.ioctl = snd\_pcm\_lib\_ioctl,

.hw\_params = comip\_pcm\_hw\_params,

--->comip\_dmas\_request(prtd->params->name, dma->channel);

--->snd\_pcm\_set\_runtime\_buffer(substream, &substream->dma\_buffer);

.hw\_free = comip\_pcm\_hw\_free,

--->comip\_dmas\_free(prtd->dma\_ch);

--->snd\_pcm\_set\_runtime\_buffer(substream, NULL);

.prepare = comip\_pcm\_prepare,

--->config = &pcm\_dmas\_config[1];

--->config->src\_addr = runtime->dma\_addr;

--->config->dst\_addr = dma->daddr;

--->config->irq\_handler = comip\_pcm\_dma\_irq

--->comip\_dmas\_config(dma\_ch,config);

--->comip\_dmas\_start(dma\_ch);

--->snd\_pcm\_period\_elapsed(substream);

--->config->irq\_data = substream;

--->comip\_dmas\_config(dma->channel,config);

.trigger = comip\_pcm\_trigger,

--->SNDRV\_PCM\_TRIGGER\_START:comip\_dmas\_start(prtd->dma\_ch);

--->SNDRV\_PCM\_TRIGGER\_STOP:comip\_dmas\_stop(prtd->dma\_ch);

--->comip\_dmas\_intr\_disable(prtd->dma\_ch, DMAS\_INT\_DONE);

--->SNDRV\_PCM\_TRIGGER\_SUSPEND:comip\_dmas\_stop(prtd->dma\_ch);

--->SNDRV\_PCM\_TRIGGER\_PAUSE\_PUSH:comip\_dmas\_stop(prtd->dma\_ch);

--->SNDRV\_PCM\_TRIGGER\_RESUME:comip\_dmas\_start(prtd->dma\_ch);

.pointer = comip\_pcm\_pointer,

--->comip\_dmas\_get(prtd->dma\_ch, &ptr);

--->x = bytes\_to\_frames(runtime, dma\_ptr - runtime->dma\_addr);

.mmap = comip\_pcm\_mmap,

--->dma\_mmap\_writecombine(substream->pcm->card->dev, vma,

runtime->dma\_area,

runtime->dma\_addr,

runtime->dma\_bytes);

};

.pcm\_new = comip\_pcm\_dma\_new,

.pcm\_free = comip\_pcm\_free\_dma\_buffers,

};

platform\_device --->平台设备comip-pcm

Devices.c (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\mach-lc186x)

comip\_set\_pcm\_info

--->comip\_register\_device(&comip\_device\_pcm, NULL);

--->struct platform\_device comip\_device\_pcm = {

.name = "comip-pcm",//注册设备驱动

.id = 0,

.dev = {

.dma\_mask = &comip\_pcm\_dma\_mask,

.coherent\_dma\_mask = COMIP\_DMA\_BIT\_MASK,

},

.resource = comip\_resource\_pcm,

.num\_resources = ARRAY\_SIZE(comip\_resource\_pcm),

};

--->static struct resource comip\_resource\_pcm[] = {

[0] = {

.start = PCM\_BASE,

.end = PCM\_BASE + 0x28,

.flags = IORESOURCE\_MEM,

},TOP\_RAM1\_BASE  ---》语音通话时modem侧的语音。

[1] = { /\* RX \*/

.start = TOP\_DMAS\_CH10,

.end = TOP\_DMAS\_CH10,

.flags = IORESOURCE\_DMA,

},

[2] = { /\* TX \*/

.start = TOP\_DMAS\_CH2,

.end = TOP\_DMAS\_CH2,

.flags = IORESOURCE\_DMA,

},

};

--->dev->dev.platform\_data = data;

--->platform\_device\_register(dev);

--->device\_initialize(&pdev->dev);

--->arch\_setup\_pdev\_archdata(pdev);

--->return platform\_device\_add(pdev);

.pcm\_new = comip\_pcm\_dma\_new

--->card->dev->dma\_mask = &comip\_pcm\_dmamask;

--->card->dev->coherent\_dma\_mask = COMIP\_DMA\_BIT\_MASK;

--->comip\_pcm\_preallocate\_dma\_buffer(pcm, SNDRV\_PCM\_STREAM\_PLAYBACK);

--->buf->area = dma\_alloc\_writecombine(pcm->card->dev, size,

&buf->addr, GFP\_KERNEL);

--->comip\_pcm\_preallocate\_dma\_buffer(pcm, SNDRV\_PCM\_STREAM\_CAPTURE);

.pcm\_free = comip\_pcm\_free\_dma\_buffers

--->buf = &substream->dma\_buffer;

--->dma\_free\_writecombine(pcm->card->dev, buf->bytes,

buf->area, buf->addr);

Soc-core.c (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\sound\soc)

/\* ASoC platform driver \*/

platform\_driver "soc-audio" 平台驱动

struct platform\_driver soc\_driver = {

.driver = {

.name = "soc-audio",//和平台device platform\_device "soc-audio"对应

.owner = THIS\_MODULE,

.pm = &snd\_soc\_pm\_ops,

},

.probe = soc\_probe,

.remove = soc\_remove,

};

snd\_soc\_init

--->snd\_soc\_util\_init();

--->soc\_dummy\_dev = platform\_device\_alloc("snd-soc-dummy", -1);

--->platform\_device\_add(soc\_dummy\_dev);

--->platform\_driver\_register(&soc\_dummy\_driver);

--->struct platform\_driver soc\_dummy\_driver = {

.driver = {

.name = "snd-soc-dummy",

.owner = THIS\_MODULE,

},

.probe = snd\_soc\_dummy\_probe,

.remove = snd\_soc\_dummy\_remove,

};

--->platform\_driver\_register(&soc\_driver);//匹配 name为 "soc-audio"的设备

--->.probe = soc\_probe

--->snd\_soc\_card \*card = platform\_get\_drvdata(pdev);

--->card->dev = &pdev->dev;

--->snd\_soc\_register\_card(card);

--->dev\_set\_drvdata(card->dev, card);

--->snd\_soc\_initialize\_card\_lists(card);

--->soc\_init\_card\_debugfs(card);

--->snd\_soc\_instantiate\_card(card);

--->

--->.pm = &snd\_soc\_pm\_ops

platform\_driver "comip\_snd\_soc" 平台驱动

平台设备platform\_device "soc-audio"

平台设备添加snd\_soc\_card comip\_snd\_soc

Lc186x-snd-soc.c (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\mach-lc186x\sound)

comip\_snd\_soc\_init

--->platform\_driver\_register(&comip\_snd\_soc\_driver);

--->struct platform\_driver comip\_snd\_soc\_driver = {

.probe = comip\_snd\_soc\_probe,

.remove = comip\_snd\_soc\_remove,

.driver = {

.name = "comip\_snd\_soc",

.owner = THIS\_MODULE,

},

};

--->comip\_snd\_soc\_probe

//注册平台设备platform\_device "soc-audio"

--->platform\_device comip\_snd\_soc\_device = platform\_device\_alloc("soc-audio", -1);

--->platform\_set\_drvdata(comip\_snd\_soc\_device, &comip\_snd\_soc);

--->platform\_device\_add(comip\_snd\_soc\_device);

--->struct snd\_soc\_card comip\_snd\_soc = {

.name = "comip\_snd\_soc",

.dai\_link = comip\_snd\_soc\_dai\_link,

.num\_links = ARRAY\_SIZE(comip\_snd\_soc\_dai\_link),

};

---> struct snd\_soc\_dai\_link comip\_snd\_soc\_dai\_link[] = {

{

.name = "comip\_audio",

.stream\_name = "comip\_au",

.codec\_name = "comip\_codec",

.codec\_dai\_name = "comip\_hifi",

.cpu\_dai\_name = "comip-i2s.0",

.platform\_name = "comip-i2s.0",

.ops = &comip\_snd\_soc\_ops,

.init = comip\_soc\_init,

.ignore\_suspend = 1,

.dai\_fmt = SND\_SOC\_DAIFMT\_CBM\_CFM,

},

{

.name = "comip\_voice",

.stream\_name = "comip\_vx",

.codec\_name = "comip\_codec",

.codec\_dai\_name = "comip\_voice",

.cpu\_dai\_name = "virtual-pcm",//通话语音

.ops = &comip\_snd\_soc\_ops,

.ignore\_suspend = 1,

.dai\_fmt = SND\_SOC\_DAIFMT\_CBM\_CFM,

},

#if defined(CONFIG\_SND\_COMIP\_PCM)

{

.name = "comip\_voip",

.stream\_name = "voip",

.codec\_name = "comip\_codec",

.codec\_dai\_name = "virtual\_codec",

.cpu\_dai\_name = "comip-pcm.0",

.platform\_name = "comip-pcm.0",

.ops = &comip\_snd\_soc\_ops,

.ignore\_suspend = 1,

.dai\_fmt = SND\_SOC\_DAIFMT\_CBM\_CFM,

},

{

.name = "voice\_link",

.stream\_name = "voice\_loop",

.codec\_name = "comip\_codec",

.codec\_dai\_name = "comip\_voice",

.cpu\_dai\_name = "comip-pcm.0",

.platform\_name = "comip-pcm.0",

.ops = &comip\_snd\_soc\_ops,

.ignore\_suspend = 1,

.dai\_fmt = SND\_SOC\_DAIFMT\_CBM\_CFM,设置为codec pcm master

cpu为slave:

SND\_SOC\_DAIFMT\_CBM\_CFM /\* codec clk & FRM master \*/ codec为master,frame也是master cpu为slave

SND\_SOC\_DAIFMT\_CBS\_CFS /\* codec clk & FRM slave \*/ codec frame也为从 cpu master

SND\_SOC\_DAIFMT\_CBS\_CFM /\* codec clk slave & FRM master \*/ 一般不支持

SND\_SOC\_DAIFMT\_CBM\_CFS /\* codec clk master & frame slave \*/一般不支持

}

#endif

};

Board-common.c

platform\_device “comip\_snd\_soc” 平台设设备

(g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\mach-lc186x\board)

platform\_device comip\_snd\_soc\_device = {

.name = "comip\_snd\_soc",

.id = 0,

};

--->platform\_add\_devices(common\_devices, ARRAY\_SIZE(common\_devices));

|  |
| --- |
| snd\_soc\_dai\_link |
| \*name:Codec name |
| stream\_name:Stream name |
| cpu\_name |
| cpu\_of\_node |
| cpu\_dai\_name |
| codec\_name |
| codec\_of\_node |
| codec\_dai\_name |
| platform\_name |
| platform\_of\_node |
| be\_id |
| params |
| dai\_fmt |
| trigger[2] |
| ignore\_suspend:1 |
| symmetric\_rates:1 |
| no\_pcm:1 |
| dynamic:1 |
| ignore\_pmdown\_time:1 |
| init |
| be\_hw\_params\_fixup |
| ops |
| compr\_ops |

平台驱动platform\_driver "comip\_codec"

Lc1160\_codec.c (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\plat-lc\drivers\sound\codecs)

lc1160\_codec\_init

--->platform\_driver\_probe(&lc1160\_codec\_driver, lc1160\_codec\_probe)

--->platform\_driver lc1160\_codec\_driver = {

.driver = {

.name = "comip\_codec",

.owner = THIS\_MODULE,

},

.remove = \_\_exit\_p(lc1160\_codec\_remove),

};

--->lc1160\_codec\_probe

--->snd\_soc\_register\_codec(&pdev->dev,&soc\_codec\_dev\_lc1160,lc1160\_dai, ARRAY\_SIZE(lc1160\_dai))

--->lc1160\_dai

--->static struct snd\_soc\_dai\_driver lc1160\_dai[] = {

{

.name = "comip\_hifi",

.playback = {

.stream\_name = "Playback",

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_1160\_RATES,

.formats = COMIP\_1160\_FORMATS,

},

.capture = {

.stream\_name = "Capture",

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_1160\_RATES,

.formats = COMIP\_1160\_FORMATS,

},

.ops = &lc1160\_dai\_ops,

},

{

.name = "comip\_voice",

.playback = {

.stream\_name = "VxDL",

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_1160\_RATES,

.formats = COMIP\_1160\_FORMATS,

},

.capture = {

.stream\_name = "VxUL",

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_1160\_RATES,

.formats = COMIP\_1160\_FORMATS,

},

.ops = &lc1160\_dai\_ops,

},

{

.name = "virtual\_codec",

.playback = {

.stream\_name = "Play",

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_1160\_RATES,

.formats = COMIP\_1160\_FORMATS,

},

.capture = {

.stream\_name = "Cap",

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_1160\_RATES,

.formats = COMIP\_1160\_FORMATS,

},

}

};

---> struct snd\_soc\_codec\_driver soc\_codec\_dev\_lc1160 = {

.probe = lc1160\_probe,

.remove = lc1160\_remove,

.read = lc1160\_read\_reg\_cache,

.write = lc1160\_write,

.set\_bias\_level = lc1160\_set\_bias\_level,

.reg\_cache\_size = sizeof(lc1160\_codec\_reg),

.reg\_word\_size = sizeof(u8),

.reg\_cache\_default = lc1160\_codec\_reg,

.ignore\_pmdown\_time = true,

.controls = lc1160\_snd\_controls,

.num\_controls = ARRAY\_SIZE(lc1160\_snd\_controls),

.dapm\_widgets = lc1160\_dapm\_widgets,

.num\_dapm\_widgets = ARRAY\_SIZE(lc1160\_dapm\_widgets),

.dapm\_routes = intercon,

.num\_dapm\_routes = ARRAY\_SIZE(intercon),

};

--->snd\_soc\_register\_dais(dev, dai\_drv, num\_dai);

平台设置 platform\_device "comip\_codec"

comip\_codec\_device

--->static struct platform\_device comip\_codec\_device = {

.name = "comip\_codec",

.id = -1,

.dev = {

.platform\_data = &lc1160\_codec\_platform\_data,

--->struct comip\_codec\_platform\_data lc1160\_codec\_platform\_data = {

.pcm\_switch = codec\_bt\_switch,

.codec\_power\_save = lc1160\_power\_save,

.pa\_enable=pa\_enable,

#if defined (LC1160\_INT2\_PIN)

.irq\_gpio = mfp\_to\_gpio(LC1160\_INT2\_PIN),

#endif

.mclk\_id = "clkout3\_clk",

.eclk\_id = "clkout4\_clk",

#if defined (CODEC\_JACK\_POLARITY)

.jack\_polarity = CODEC\_JACK\_POLARITY,

#endif

.vth1 = 0x01,

.vth2 = 0x00,

.vth3 = 0x00,

};

}

};

平台驱动: platform\_driver "virtual-pcm"

Comip-snd-virtual.c (g:\code-pj\source\lc1860-4.4\kernel\linux-3.10\arch\arm\plat-lc\drivers\sound)

comip\_snd\_virtual\_init

--->platform\_driver\_probe(&comip\_pcm\_virtual\_driver, comip\_pcm\_virtual\_probe);

--->struct platform\_driver comip\_pcm\_virtual\_driver = {

.remove = comip\_pcm\_virtual\_remove,

.driver = {

.name = "virtual-pcm",

.owner = THIS\_MODULE,

},

};

--->comip\_pcm\_virtual\_probe

--->snd\_soc\_register\_component(&pdev->dev, &comip\_pcm\_virtual\_component,

&comip\_pcm\_virtual\_dai, 1);

--->snd\_soc\_register\_platform(&pdev->dev, &leadcore\_asoc\_platform);

--->struct snd\_soc\_component\_driver comip\_pcm\_virtual\_component = {

.name = "virtual-pcm",

};

--->struct snd\_soc\_dai\_driver comip\_pcm\_virtual\_dai = {

.playback = {

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_PCM\_RATES,

.formats = SNDRV\_PCM\_FMTBIT\_S16\_LE,

},

.capture = {

.channels\_min = 1,

.channels\_max = 2,

.rates = COMIP\_PCM\_RATES,

.formats = SNDRV\_PCM\_FMTBIT\_S16\_LE,

},

.ops = &comip\_pcm\_virtual\_dai\_ops,

};

--->struct snd\_soc\_dai\_ops comip\_pcm\_virtual\_dai\_ops = {

.hw\_params = comip\_pcm\_virtual\_dai\_hw\_params,

};

--->struct snd\_soc\_component\_driver comip\_pcm\_virtual\_component = {

.name = "virtual-pcm",

};

--->struct snd\_soc\_platform\_driver leadcore\_asoc\_platform = {

.ops = &comip\_pcm\_virtual\_ops,

.pcm\_new = comip\_pcm\_virtual\_new,

.pcm\_free = comip\_pcm\_virtual\_free\_dma\_buffers,

};

平台设备platform\_device "virtual-pcm"

comip\_set\_virtual\_pcm\_info

--->comip\_register\_device(&comip\_device\_virtual\_pcm, NULL)

--->platform\_device comip\_device\_virtual\_pcm = {

.name = "virtual-pcm",

.id = -1,

};

snd\_soc\_codec\_driver

.set\_bias\_level = lc1160\_set\_bias\_level

最终调用流程

dapm\_post\_sequence\_async

snd\_soc\_dapm\_set\_bias\_level

--->card->set\_bias\_level(card, dapm, level);

--->dapm->codec->driver->set\_bias\_level(dapm->codec,level);

Log 流程：

SND\_SOC\_BIAS\_OFF = 0,

SND\_SOC\_BIAS\_STANDBY = 1,

SND\_SOC\_BIAS\_PREPARE = 2,

SND\_SOC\_BIAS\_ON = 3,

lc1160\_codec:lc1160\_set\_bias\_level:538: comip\_codec comip\_codec: lc1160\_set\_bias\_level :level is 1

lc1160\_codec:lc1160\_set\_bias\_level:538: comip\_codec comip\_codec: lc1160\_set\_bias\_level :level is 2

lc1160\_codec:lc1160\_set\_bias\_level:538: comip\_codec comip\_codec: lc1160\_set\_bias\_level :level is 3

1：设置lc1160处于i2s或者pcm模式

lc1160\_set\_dai\_fmt

--->pcm模式:

/\* pcm,send data after one clk,sel extclk,

short frame sync, 16fs,master mode \*/

snd\_soc\_write(codec, LC1160\_R12, 0x51);

2:lc1160 hw 设置

--->lc1160\_hw\_params

根据dai设置lc1160 clk

---> clk\_set\_rate(lc1160->eclk, LC1160\_ECLK\_RATE \*multiple);

--->clk\_enable(lc1160->eclk);

3:lc1160上电使能

--->lc1160\_set\_bias\_level

--->lc1160\_power(codec, 1);

4: pcm switch

--->默认天通电话选择CODEC\_ON\_S1710 = 3

---》gpio\_direction\_output(pcm\_sw\_1, 1);//gpio82拉高