Android4.4

comip\_set\_usb\_info(&comip\_usb\_info)

comip\_usb\_info

.usb\_power\_set = usb\_power\_set

--->

--->1:comip\_register\_device(&comip\_device\_hcd\_hsic,&info->hcd);

comip\_resource\_hcd\_hsic

.id = HSIC\_HW,

static struct resource comip\_resource\_hcd\_hsic[] = {

[0] = {

.start = USB\_HSIC\_BASE,//0xA0440000

.end = USB\_HSIC\_BASE + 0x3ffff,

.flags = IORESOURCE\_MEM,

},

[1] = {

.start = CTL\_HSIC\_PHY\_POR\_CTRL,//(USB\_CTL + 0x60)=0xA0480000+0x60

.end = CTL\_HSIC\_PHY\_POR\_CTRL + 0x30,

.flags = IORESOURCE\_MEM,

},

[2] = {

.start = INT\_USB\_HSIC,//(INT\_PRI\_BASE + 18)

.end = INT\_USB\_HSIC,

.flags = IORESOURCE\_IRQ,

},

};

comip\_hsic\_hcd\_driver\_probe

--->1:struct device\_node \*np = \_dev->dev.of\_node;

--->2: struct resource \*resbase, \*resphy;//获取资源

--->3: resbase = platform\_get\_resource\_byname(\_dev,

IORESOURCE\_MEM, "com\_hsic\_base");//对应reg-names = "com\_hsic\_base", "com\_hsic\_phy";

--->4:comip\_hsic\_hcd\_dev->os\_dep.base[0] = ioremap(resbase->start, resource\_size(resbase));

--->5: resphy = platform\_get\_resource\_byname(\_dev,

IORESOURCE\_MEM, "com\_hsic\_phy");//对应com\_hsic\_phy

--->6:comip\_hsic\_hcd\_dev->os\_dep.base[1] = ioremap(resphy->start, resource\_size(resphy));

//申请中断资源

--->7:comip\_hsic\_hcd\_dev->irq = platform\_get\_irq(\_dev, 0);

-->8:comip\_hsic\_hcd\_dev->core\_if = comip\_hsic\_cil\_init(comip\_hsic\_hcd\_dev->os\_dep.base[0],

comip\_hsic\_hcd\_dev->os\_dep.base[1],\_dev);

\_dev->id = HSIC\_HW;

8: comip\_hsic\_cil\_init

--->8.1:core\_if->core\_global\_regs = (comip\_hsic\_core\_global\_regs\_t \*) reg\_base;//对应:com\_hsic\_base

--->8.2:core\_if->ctl\_regs = (comip\_hsic\_core\_ctl\_regs\_t \*)\_ctl\_reg\_base\_addr;//对应com\_hsic\_phy

--->8.3:host\_if->host\_global\_regs = (comip\_hsic\_host\_global\_regs\_t \*)

(reg\_base + COMIP\_HSIC\_HOST\_GLOBAL\_REG\_OFFSET);

--->8.4:host\_if->hprt0 =

(uint32\_t \*) (reg\_base + COMIP\_HSIC\_HOST\_PORT\_REGS\_OFFSET);

--->8.5:host\_if->hc\_regs[i] = (comip\_hsic\_hc\_regs\_t \*)

(reg\_base + COMIP\_HSIC\_HOST\_CHAN\_REGS\_OFFSET +

(i \* COMIP\_HSIC\_CHAN\_REGS\_OFFSET));

--->8.6:case HSIC\_HW:

--->8.6.1:clk\_id = "usbhsic\_12m\_clk";

--->8.6.2:core\_if->clk\_hsic = clk\_get(&\_dev->dev, clk\_id);

clk\_id = "usb\_hsic\_clk";

core\_if->usb\_hsic\_clk = clk\_get(&\_dev->dev, clk\_id);

clk\_id = "ap\_sw4\_hsic\_clk";

core\_if->ap\_sw4\_hsic\_clk = clk\_get(&\_dev->dev, clk\_id);

Lc1881 usb时钟设置

usb\_phy: usb\_phy@0xFA340000 {

compatible = "leadcore,usb-phy";

reg = <0xFA340000 0xFFF>;

usbvbus-flag = <0>;

usb3v3-supply = <&dldo6>;

usb0v9-supply = <&aldo12>;

clocks = <&usb\_24m\_clk>;

clock-names = "usb\_24m\_clk";

};

--->usb\_24m\_clk: usb\_24m\_clk {

compatible = "comip,clk-div";

#clock-cells = <0>;

clocks = <&usb\_24m\_clkgt>;

comip,sdiv;

comip,clkdiv = <AP\_PWR\_USB\_CLK24M\_CTL PWR\_MASK(4) PWR\_BIT\_SFT(4) PWR\_WEBIT\_DEF(4)>;

comip,val\_min\_max = <1 7>;

clock-output-names = "usb\_24m\_clk";

};

--->usb\_24m\_clkgt: usb\_24m\_clkgt {

compatible = "comip,clk-gate";

#clock-cells = <0>;

clocks = <&pll3\_out\_div13>;

comip,clkgt = <AP\_PWR\_USB\_CLK24M\_CTL PWR\_BIT\_SFT(0) PWR\_WEBIT\_DEF(0)>;

clock-output-names = "usb\_24m\_clkgt";

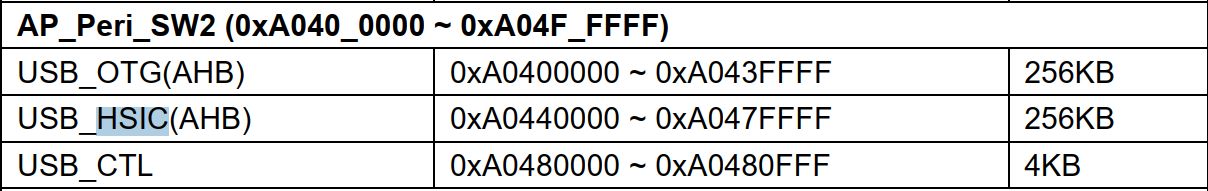
};

执行流程:

usb3\_phy->clk = clk\_get(&pdev->dev, "usb\_24m\_clk");

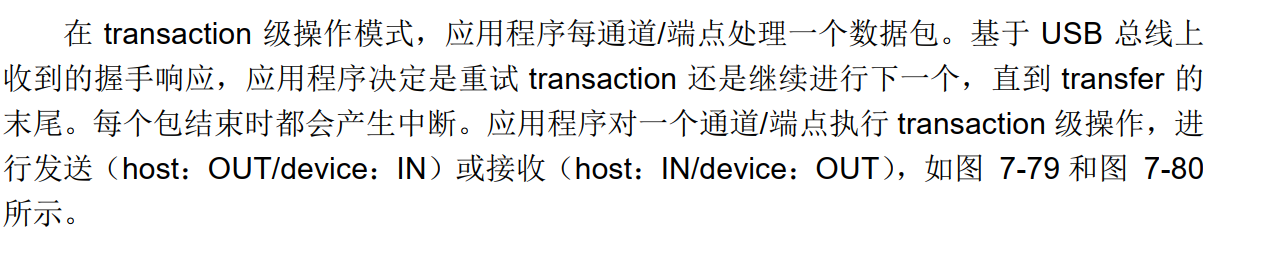
==







transaction级操作



usb控制器的工作模式：

1：DMA模式

2：slave模式

操作模式:

1：transaction:传输长度等于包的长度

2: transfer:传输长度大于包的长度

slave模式：

transaction级操作模式:

1:每个通道 端点处理一个数据包

2：每个包介绍都产生中断

Transaction

host模式:

1:传输长度等于一个最大包长度：每个通道处理一个数据包 包结束产生一个中断

写完一个完整的包后才能切换到其他通道fifo. 否则将产生错误

一个out transaction: 使能通道 数据包写入通道fifo

一个 in transaction: 使能通道 通道号写入请求队列 产生包接收中断 读空fifo数据

2：传输长度大于最大包长度

整个传输结束 usb控制器才产生中断

一个out transaction: 建立传输 使能通道

A:对同一个通道 连续把多个包写到发送fifo

B:流水化多个通道的Out传输。每次写HCCHARn寄存器 并向该通道写入一个包 usb控制器按通道的写入顺序在usb总线上安排传输。

一个 in transaction传输：

请求队列的空间 足够请求队列顶端的通道的至少一个最大包长 usb控制器在usb总线上发出一个IN令牌

建立传输 使能通道 usb控制器把通道号写入请求队列 只要请求队列 有空间可用 可同时在多个通道上按安排 In transaction

Device 模式:

1:传输长度等于一个最大包长度

一个 In transaction: 使能端点 数据包写入相应的发送fifo 等待usb控制器发出包完成中断

一个out transaction:使能端点 等待usb控制器发出包接收中断 把包从接收fifo中读空。

切换不同的端点 fifo前 必须写完一个完整的包 否则产生错误

2:传输长度大于一个最大包长度情况

一个 in transaction:建立传输 使能端点 在fifo空间足够的情况下 可以对同一个端点连续写多个包

流水化操作：写寄存器DIEPCTLn 并向该端点写一个包 当usb总线上收到IN令牌时 usb控制器就发送TXfifo中的数据