struct imx\_chip->struct pwm\_chip chip;

 ->int (\*config)(struct pwm\_chip \*chip,

 struct pwm\_device \*pwm, int duty\_ns, int period\_ns);

 ->void (\*set\_enable)(struct pwm\_chip \*chip, bool enable);

static struct imx\_pwm\_data imx\_pwm\_data\_v2 = {

 .config = imx\_pwm\_config\_v2,

 .set\_enable = imx\_pwm\_set\_enable\_v2,

};

最终，imx\_chip-》config= imx\_pwm\_config\_v2

 imx\_chip-》set\_enabl = imx\_pwm\_set\_enable\_v2

struct pwm\_chip-> const struct pwm\_ops \*ops;

 注意pwm\_ops需要驱动编写人员编写，实现

 pwm\_ops-> int (\*config)(struct pwm\_chip \*chip,

 struct pwm\_device \*pwm,

 int duty\_ns, int period\_ns);

 pwm\_ops-> int (\*enable)(struct pwm\_chip \*chip,

 struct pwm\_device \*pwm);

# PWM驱动流程

1、pwm\_chip初始化，

2、向内核注册pwm\_chip，使用函数pwmchip\_add

# 6ULL的PWM驱动

1、在probe函数：

 imx->chip.ops = &imx\_pwm\_ops; -》pwm\_chip->ops

 imx->chip.dev = &pdev->dev;

 imx->chip.base = -1;

 imx->chip.npwm = 1;

 imx->chip.can\_sleep = true;

2、imx->mmio\_base保存PWM控制器的起始地址，

3、imx\_pwm\_data \*data= of\_id->data= imx\_pwm\_data\_v2

4、struct imx\_chip \*imx.->config = imx\_pwm\_config\_v2

5、struct imx\_chip \*imx.> set\_enable = imx\_pwm\_set\_enable\_v2

可以得到，pwm\_ops为：

static struct pwm\_ops imx\_pwm\_ops = {

 .enable = imx\_pwm\_enable,

 .disable = imx\_pwm\_disable,

 .config = imx\_pwm\_config,

 .owner = THIS\_MODULE,

};

经过分析，可知：imx\_pwm\_enable,最终调用imx\_pwm\_set\_enable\_v2

imx\_pwm\_disable也调用imx\_pwm\_set\_enable\_v2

imx\_pwm\_config最终调用的是imx\_pwm\_config\_v2