# 一、rtc\_device结构体

 1、RTC也是一个标准字符设备驱动

 2、rtc\_device是RTC设备在内核中的具体实现，找到RTC相关节点

 snvs\_rtc: snvs-rtc-lp {

 compatible = "fsl,sec-v4.0-mon-rtc-lp";

 regmap = <&snvs>;

 offset = <0x34>;

 interrupts = <GIC\_SPI 19 IRQ\_TYPE\_LEVEL\_HIGH>, <GIC\_SPI 20 IRQ\_TYPE\_LEVEL\_HIGH>;

 };

 根据compatible找到驱动文件：rtc\_snvs.c，驱动文件里面就是 初始化rtc\_device并注册。

 3、rtc\_device结构体里面重点是rtc\_class\_ops操作集，

# 二、I.MX6U RTC驱动简析

 1、驱动和设备匹配以后，snvs\_rtc\_probe函数会执行，NXP为6ULL的RTC外设创建了一个结构体snvs\_rtc\_data，此结构体里面包含了rtc\_device成员变量。

struct snvs\_rtc\_data {

 struct rtc\_device \*rtc;

 struct regmap \*regmap;

 int offset;

 int irq;

 struct clk \*clk;

};

 地址就是：0x020cc000+0x34 = 0x020cc034

 首先从设备树里面获取SNVS RTC外设寄存器，初始化RTC，申请中断处理闹钟，中断处理函数是snvs\_rtc\_irq\_handler，最后通过devm\_rtc\_device\_register函数向内核注册rtc\_device，重点是注册的时候设置了snvs\_rtc\_ops，此结构体内容如下：

static const struct rtc\_class\_ops snvs\_rtc\_ops = {

 .read\_time = snvs\_rtc\_read\_time,

 .set\_time = snvs\_rtc\_set\_time,

 .read\_alarm = snvs\_rtc\_read\_alarm,

 .set\_alarm = snvs\_rtc\_set\_alarm,

 .alarm\_irq\_enable = snvs\_rtc\_alarm\_irq\_enable,

};

 当应用通过ioctl读取RTC时间的时候，RTC核心层的rtc\_dev\_ioctl会执行，通过cmd来决定具体操作，比如RTC\_ALM\_READ就是 读取闹钟，此时rtc\_read\_alarm就会执行，rtc\_read\_alarm函数就会找到具体的rtc\_deice，运行其下的ops里面的read\_alarm。

 RTC\_ALM\_SET就是设置闹

 rtc\_set\_alarm

 -》rtc\_timer\_enqueue

 -> \_\_rtc\_set\_alarm

 ->rtc->ops->set\_alarm

# 三、RTC测试