蜗窝科技

慢下来，享受技术。

Linux PWM framework(1)_简介和API描述

作者：wowo 发布于：2015-10-11 15:45 分类：通信类协议

1. 前言

PWM是Pulse Width Modulation（脉冲宽度调制）的缩写，是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，其本质是一种对模拟信号电平进行数字编码的方法。在嵌入式设备中，PWM多用于控制马达、LED、振动器等模拟器件。

PWM framework是kernel为了方便PWM driver开发、PWM使用而抽象出来的一套通用API，之所以要分析该framework，原因如下：

> 1）PWM接口，本质上一种通信协议，和I2C、SPI、USB、WIFI等没有任何差别。因此，本文将会是kernel通信协议有关framework的分析文章的第一篇。
>
> 2）它太简单了！但是，虽然简单，思路却大同小异，因而非常适合做第一篇。
>
> 3）我计划整理显示子系统的分析文章，而PWM，是显示子系统中最基础的那一个。

闲话少说，言归正传！

2. 软件框架及API汇整

PWM framework非常简单，但它同样具备framework的基本特性：对上，为内核其它driver（Consumer）提供使用PWM功能的统一接口；对下，为PWM driver（Provider）提供driver开发的通用方法和API；内部，抽象并实现公共逻辑，屏蔽技术细节。下面我们通过它所提供的API，进一步认识PWM framework。

2.1 向PWM consumer提供的APIs

对consumer而言，关注PWM的如下参数：

1）频率

> PWM的频率决定了所模拟出来的模拟电平的平滑度，通俗的讲，就是逼真度。不同的模拟器件，对期待的频率是有要求的，因此需要具体情况具体对待。
>
> 另外，人耳能感知的频率范围是20Hz~16KHz，因此要注意PWM的频率不要落在这个范围，否则可能会产生莫名其妙的噪声。

2）占空比

占空比，决定了一个周期内PWM信号高低的比率，进而决定了一个周期内的平均电压，也即所模拟的模拟电平的电平值。

3）极性

简单的说，一个PWM信号的极性，决定了是高占空比的信号输出电平高，还是低占空比信号输出电平高。假设一个信号的占空比为100%，如果为正常极性，则输出电平最大，如果为翻转的极性，则输出电平为0。

4）开关

站内搜索

功能

留言板
评论列表

最新评论

温柔海洋
@linuxer：你说的是对的，标准的同步原语我之前只使用过...
linuxer
@温柔海洋：一年前，我也是信心满满的杀入内核同步的领域，试图...
wowo
@hangkeke052：我觉得的你这个场景很有意思，要怎么...
温柔海洋
看完对SMP架构下，内核态并行编程做个总结： 如果是多...
linuxer
@REGA：在linux中，thread<----->str...
linuxer
@江南书生：网站上有写的，呵呵 ~ ~ 联系我们 QQ:...

文章分类

Linux内核分析(7)
订阅该分类
统一设备模型(14)
订阅该分类
电源管理子系统(40)
订阅该分类
中断子系统(14)
订阅该分类
进程管理(5)
订阅该分类
内核同步机制(17)
订阅该分类
GPIO子系统(3)
订阅该分类
时间子系统(13)
订阅该分类
通信类协议(4)
订阅该分类
内存管理(11)
订阅该分类
图形子系统(1)
订阅该分类
文件系统(1)
订阅该分类

控制PWM信号是否输出。

基于上述需求，linux pwm framework向consumer提供了如下API：

```
1: /* include/linux/pwm.h */
2:
3: /*
4:  * pwm_config - change a PWM device configuration
5:  */
6: int pwm_config(struct pwm_device *pwm, int duty_ns, int period_ns);
```

> pwm_config，用于控制PWM输出信号的频率和占空比，其中频率是以周期（period_ns）的形式配置的，占空比是以有效时间（duty_ns）的形式配置的。
>
> pwm_enable/pwm_disable，用于控制PWM信号输出与否。
>
> pwm_set_polarity，可以更改pwm信号的极性，可选参数包括normal（PWM_POLARITY_NORMAL）和inversed（极性翻转，PWM_POLARITY_INVERSED）两种。

上面的API都以struct pwm_device类型的指针为操作句柄，该指针抽象了一个PWM设备（consumer不需要关心其内部构成），那么怎么获得PWM句柄呢？使用如下的API：

注1：本文只介绍基于DTS的、新的pwm request系列接口，对于那些旧接口，让它随风而去吧。

```
1: /* include/linux/pwm.h */
2:
3: struct pwm_device *pwm_get(struct device *dev, const char *con_id);
4: struct pwm_device *of_pwm_get(struct device_node *np, const char *con_id);
5: void pwm_put(struct pwm_device *pwm);
6:
7: struct pwm_device *devm_pwm_get(struct device *dev, const char *con_id);
8: struct pwm_device *devm_of_pwm_get(struct device *dev, struct device_node *np,
9:                                     const char *con_id);
10: void devm_pwm_put(struct device *dev, struct pwm_device *pwm);
```

> pwm_get/devm_pwm_get，从指定设备（dev）的DTS节点中，获得对应的PWM句柄。可以通过con_id指定一个名称，或者会获取和该设备绑定的第一个PWM句柄。设备的DTS文件需要用这样的格式指定所使用的PWM device（具体的形式，还依赖pwm driver的具体实现，后面会再介绍）：
> bl: backlight {
> pwms = <&pwm 0 5000000 PWM_POLARITY_INVERTED>;
> pwm-names = "backlight";
> };
> 如果“con_id”为NULL，则返回DTS中“pwms”字段所指定的第一个PWM device；如果“con_id”不为空，如是“backlight”，则返回和“pwm-names ”字段所指定的name对应的PWM device。
> 上面“pwms”字段各个域的含义如下：
> 1）&pwm，对DTS中pwm节点的引用；
> 2）0，pwm device的设备号，具体需要参考SOC以及pwm driver的实际情况；
> 3）5000000，PWM信号默认的周期，单位是纳秒（ns）；
> 4）PWM_POLARITY_INVERTED，可选字段，是否提供由pwm driver决定，表示pwm信号的极性，若为0，则正常极性，若为PWM_POLARITY_INVERTED，则反转极性。
>
> of_pwm_get/devm_of_pwm_get，和pwm_get/devm_pwm_get类似，区别是可以指定需要从中解析PWM信息的device node，而不是直接指定device指针。

2.2 向PWM provider提供的APIs

TTY子系统(3)
订阅该分类
订阅该分类
u-boot分析(3)
Linux应用技巧(11)
订阅该分类
订阅该分类
软件开发(6)
订阅该分类
基础技术(8)
订阅该分类
蓝牙(10)
ARMv8A Arch(13)
订阅该分类
订阅该分类
显示(3)
订阅该分类
基础学科(9)
订阅该分类
技术漫谈(10)
订阅该分类
项目专区(0)
X Project(14)
订阅该分类

随机文章
X-001-PRE-git介绍及操作记录
Linux时间子系统之（五）：POSIX Clock
arm64 linux移植
linux cpufreq framework(5)_ARM big Little driver
Linux内核同步机制之（六）：Seqlock

文章存档
2016年10月(5)
2016年9月(6)
2016年8月(9)
2016年7月(5)
2016年6月(10)
2016年5月(8)
2016年4月(7)
2016年3月(5)
2016年2月(5)
2016年1月(6)
2015年12月(6)
2015年11月(9)
2015年10月(9)
2015年9月(4)
2015年8月(3)
2015年7月(7)
2015年6月(3)
2015年5月(7)
2015年4月(9)
2015年3月(9)
2015年2月(6)
2015年1月(6)
2014年12月(17)
2014年11月(8)
2014年10月(9)
2014年9月(7)
2014年8月(12)
2014年7月(6)
2014年6月(6)
2014年5月(9)
2014年4月(9)
2014年3月(7)
2014年2月(3)
2014年1月(4)

订阅Rss

接着从PWM provider的角度，看一下PWM framework为provider编写PWM驱动提供了哪些API。

2.2.1 pwm chip

PWM framework使用struct pwm_chip抽象PWM控制器。通常情况下，在一个SOC中，可以同时支持多路PWM输出（如6路），以便同时控制多个PWM设备。这样每一路PWM输出，可以看做一个PWM设备（由上面struct pwm_device抽象），没有意外的话，这些PWM设备的控制方式应该类似。PWM framework会统一管理这些PWM设备，将它们归类为一个PWM chip。

struct pwm_chip的定义如下：

```
1: /* include/linux/pwm.h */
2:
3: /**
4:  * struct pwm_chip - abstract a PWM controller
5:  * @dev: device providing the PWMs
```

> dev，该pwm chip对应的设备，一般由pwm driver对应的platform驱动指定。必须提供！
>
> ops，操作PWM设备的回调函数，后面会详细介绍。必须提供！
>
> npwm，该pwm chip可以支持的pwm channel（也可以称作pwm device由struct pwm_device表示）个数，kernel会根据该number，分配相应个数的struct pwm_device结构，保存在pwms指针中。必须提供！
>
> pwms，保存所有pwm device的数组，kernel会自行分配，不需要driver关心。
>
> base，在将该chip下所有pwm device组成radix tree时使用，只有旧的pwm_request接口会使用，因此忽略它吧，编写pwm driver不需要关心。
>
> of_pwm_n_cells，该PWM chip所提供的DTS node的cell，一般是2或者3，例如：为3时，consumer需要在DTS指定pwm number、pwm period和pwm flag三种信息（如2.1中的介绍）；为2时，没有flag信息。
>
> of_xlate，用于解析consumer中指定的、pwm信息的DTS node的回调函数（如2.1中介绍的，pwms = <&pwm 0 5000000 PWM_POLARITY_INVERTED>）。
>
> 注2：一般情况下，of_pwm_n_cells取值为3，或者2（不关心极性），of_xlate则可以使用kernel提供的of_pwm_xlate_with_flags（解析of_pwm_n_cells为3的chip）或者of_pwm_simple_xlate（解析of_pwm_n_cells为2的情况）。具体的driver可以根据实际情况修改上述规则，但不到万不得已的时候，不要做这种非标准的、掏力不讨好的事情！（有关of_xlate的流程，会在下一篇流程分析的文章中介绍。）
>
> can_sleep，如果ops回调函数中，.config()，.enable()或者.disable()操作会sleep，则要设置该变量。

2.2.2 pwm ops

struct pwm_ops结构是pwm device有关的操作函数集，如下：

```
1: /**
2:  * struct pwm_ops - PWM controller operations
3:  * @request: optional hook for requesting a PWM
4:  * @free: optional hook for freeing a PWM
```

> 这些回调函数的操作对象是具体的pwm device（由struct pwm_device类型的指针表示），包括：
>
> config，配置pwm device的频率、占空比。必须提供！
>
> enable/disable，使能/禁止pwm信号输出。必须提供！
>
> request/free，不再使用。

> set_polarity，设置pwm信号的极性。可选，具体需要参考of_pwm_n_cells的定义。

2.2.3 pwm device

struct pwm_device是pwm device的操作句柄，consumer的API调用，会中转到provider的pwm ops回调函数上，provider（及pwm driver）根据pwm device的信息，进行相应的寄存器操作。如下：

```
1: struct pwm_device {

2:         const char              *label;

3:         unsigned long           flags;

4:         unsigned int            hwpwm;
```

> pwm driver比较关心的字段是：
>
> hwpwm，该pwm device对应的hardware pwm number，可用于寄存器的寻址操作。
>
> period、duty_cycle、polarity，pwm信号的周期、占空比、极性等信息。

2.2.4 pwmchip_add/pwmchip_remove

初始化完成后的pwm chip可以通过pwmchip_add接口注册到kernel中，之后的事情，pwm driver就不用操心了。该接口的原型如下：

```
1: int pwmchip_add(struct pwm_chip *chip);

2: int pwmchip_remove(struct pwm_chip *chip);
```

3. API使用指南

3.1 consumer使用PWM的步骤

基于2.1章节描述的API，可以得到pwm consumer（如pwm backlight driver）使用pwm framework的方法和步骤如下：

1）查看pwm provider所提供的pwm dts binding信息（一般会在“Documentation/devicetree/bindings/pwm”目录中），并以此在该device所在的dts node中添加“pwms ”以及“pwm-names ”相关的配置。例如：

```
/* arch\arm\boot\dts\imx23-evk.dts */
backlight {
        compatible = "pwm-backlight";
        pwms = <&pwm 2 5000000>;
        brightness-levels = <0 4 8 16 32 64 128 255>;
        default-brightness-level = <6>;
};
```

其中，&pwm，表示对pwm driver的DTS节点的引用，具体可参考下面3.2章节的介绍。

2）在driver的probe接口中，调用devm_pwm_get接口，获取pwm device句柄，并保存起来。

3）devm_pwm_get成功后，该pwm信号已经具备初始的周期和极性。后续根据需要，可以调用pwm_config和pwm_set_polarity更改该pwm信号的周期、占空比和极性。

4）driver可以根据需要，调用pwm_enable/pwm_disable接口，打开或者关闭pwm信号的输出。

3.2 provider编写PWM driver的步骤

基于2.2章节描述的API，可以得到pwm provider（即具体的PWM驱动）使用pwm framework的方法和步骤如下：

1）创建代表该pwm driver的DTS节点，并提供platform device有关的资源信息，例如：

```
/* arch\arm\boot\dts\imx23.dtsi */
pwm: pwm@80064000 {
        compatible = "fsl,imx23-pwm";
        reg = <0x80064000 0x2000>;
```

```
        clocks = <&clks 30>;
        #pwm-cells = <2>;
        fsl,pwm-number = <5>;
        status = "disabled";
};

/* arch\arm\boot\dts\imx23-evk.dts */
pwm: pwm@80064000 {
        pinctrl-names = "default";
        pinctrl-0 = <&pwm2_pins_a>;
        status = "okay";
};
```

2）定义一个pwm chip变量

3）注册相应的platform driver，并在driver的.probe()接口中，初始化pwm chip变量，至少要包括如下字段：

> dev，使用platform device中的dev指针即可；npwm；ops，至少包括config、enable、disable三个回调函数。
>
> 如果该pwm chip支持额外的flag（如PWM极性，或者自定义的flag），将PWM cell指定为3（of_pwm_n_cells），of_xlate指定为of_pwm_xlate_with_flags。

初始化完成调用pwmchip_add接口，将chip添加到kernel中。

4）每当consumer有API调用时，kernel会以pwm device为参数，调用pwm driver提供的pwm ops，相应的回调函数可以从pwm device中取出pwm number（该number的意义driver自行解释），并操作对应的寄存器即可。

原创文章，转发请注明出处。蜗窝科技，http://www.wowotech.net/comm/pwm_overview.html。

标签: Linux driver pwm

« ARM64的启动过程之（二）：创建启动阶段的页表 | ARM64的启动过程之（一）：内核第一个脚印»

评论：

linux_emb
2015-12-24 17:30

麻雀虽小，五脏俱全。

回复

ushineme
2015-10-31 11:15

啊呀呀呀，要整理显示子系统了吗，我就是刚毕业在一个手机公司学习lcd的驱动，好期待！！

回复

wowo
2015-10-31 12:13

@ushineme：是在准备，不过还没有开始呢，最近事情比较多，就耽搁了。

回复

ushineme
2015-10-31 16:18

@wowo：嗯嗯，知道啦，我会慢慢等的

回复

发表评论：

昵称

邮件地址 (选填)

个人主页 (选填)

发表评论