RK3566 EBOOK休眠模式说明及相关驱动处理方法

文件标识：RK-SM-YF-291

发布版本：V1.0.0

日期：2021-05-25

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文主要介绍RK3566 EBOOK休眠模式以及相关驱动处理方法。

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	刘益星	2021-05-19	初始版本
V1.0.1	刘益星	2021-05-25	增加电源控制说明和模式获取说明

目录

RK3566 EBOOK休眠模式说明及相关驱动处理方法
1 **休眠模式**
1.1 deep
1.2 Ultra (新增)
1.3 Lite（新增）
2 **不同待机模式的电源控制**
3 **相关驱动**
3.1 TP驱动
3.2 背光
3.3 hall sensor
3.4 驱动获取休眠模式的说明

1 休眠模式

1.1 deep

简称：普通待机

简述：系统灭屏的情况下才会有条件地选择是否进入deep休眠；

详细过程：系统灭屏后，检测Wi-Fi/BT是否打开，是否是AC charge，如果有，则选择deep待机，如果没有打开，则选择ultra待机；

进入条件：系统灭屏，Wi-Fi/BT有打开，或有AC charge，系统没有wake_lock；

特点：可为某些外设保留主控的IO供电，避免某些外设唤醒后异常，如Wi-Fi/BT；如有其他外设有这种需求，需要参照处理；IO可唤醒系统；

缺点：待机功耗相对ultra更高；

手动强制进入deep待机的命令：

```
echo deep > /sys/power/mem_sleep

echo mem > /sys/power/state
```

1.2 Ultra (新增)

简称：超低功耗待机

简述：系统灭屏的情况下才会有条件地选择是否进入ultra休眠；

详细过程：系统灭屏后，检测Wi-Fi/BT是否打开，是否有AC charge，如果有，则选择deep待机，如果没有，则选择ultra待机；

进入条件：系统灭屏，Wi-Fi/BT没有打开，没有AC charge，系统没有wake_lock；

特点：超低待机功耗，核心部分只留DDR部分供电，主控全断电；

缺点：只有PMIC的中断可以唤醒，如rtc，power key，usb det；其他外设如果有唤醒需求，需要特殊电路支持，如hall sensor的设计，如有其他外设需求，需联系我们；

手动强制进入ultra待机的命令：

```
echo ultra > /sys/power/mem_sleep

echo mem > /sys/power/state
```

1.3 Lite（新增）

简称：浅休眠或亮屏待机

简述：亮屏情况下，X秒内没有交互，系统会进入lite休眠

详细过程：亮屏情况下，计时X秒内没有交互，lite休眠开始生效；中间如果有交互，则取消本次lite待机，重新计时；

进入条件：系统亮屏，X秒内没有交互，系统没有wake_lock

作用或特点：lite待机主要用于减少系统运行功耗；IO可唤醒系统；

手动强制进入lite待机的命令：

```
echo lite > /sys/power/mem_sleep

echo mem > /sys/power/state
```

修改lite时间：

可通过属性修改进入lite待机的时间：persist.sys.idle-delay 单位ms；

修改persist.sys.idle-delay属性值后，要有交互，才会马上生效，如果没有交互，会在下一次lite唤醒后生效；

当persist.sys.idle-delay属性值为0时，lite机制失效，不会进入lite模式休眠；

2 不同待机模式的电源控制

不同的待机模式下可以对各路电源的开关进行配置；

配置方法：修改kernel dts的rockchip_suspend节点，参考arch/arm64/boot/dts/rockchip/rk3566-rk817-eink-w103.dts文件

```
&rockchip_suspend {
    status = "okay";

    rockchip,regulator-off-in-mem-lite =
        <&vdd_cpu>, <&vdd_logic>, <&vdd_gpu>, <&vcc_3v3>, <&vdda_0v9>,
<&vcc_1v8>,
        <&vccio_acodec>, <&vccio_sd>, <&vcc1v8_dvp>, <&dcdc_boost>,
<&otg_switch>,
        <&sleep_sta_ctl>;
    rockchip,regulator-on-in-mem-lite =
        <&vcc_ddr>, <&vdda0v9_pmu>, <&vcca1v8_pmu>, <&vcc3v3_pmu>;
```

```
    rockchip,regulator-off-in-mem =
        <&vdd_cpu>, <&vdd_logic>, <&vdd_gpu>, <&vcc_3v3>, <&vdda_0v9>,
<&vcc_1v8>,
        <&vccio_acodec>, <&vccio_sd>, <&vcc1v8_dvp>, <&dcdc_boost>,
<&otg_switch>,
        <&sleep_sta_ctl>;
    rockchip,regulator-on-in-mem =
        <&vcc_ddr>, <&vdda0v9_pmu>, <&vcca1v8_pmu>, <&vcc3v3_pmu>;

    rockchip,regulator-off-in-mem-ultra =
        <&vdd_logic>, <&vdd_gpu>, <&vcc_ddr>, <&vcc_3v3>, <&vdda_0v9>,
<&vcc_1v8>,
        <&vdda0v9_pmu>, <&vcca1v8_pmu>, <&vcc3v3_pmu>, <&vccio_acodec>,
<&vccio_sd>,
        <&vcc1v8_dvp>, <&dcdc_boost>, <&otg_switch>;
    rockchip,regulator-on-in-mem-ultra = <&vdd_cpu>, <&sleep_sta_ctl>;
};
```

rockchip,regulator-off-in-mem-lite：lite待机模式下需要关闭的电源

rockchip,regulator-on-in-mem-lite：lite待机模式下需要打开的电源

rockchip,regulator-off-in-mem：deep待机模式下需要关闭的电源

rockchip,regulator-on-in-mem：deep待机模式下需要打开的电源

rockchip,regulator-off-in-mem-ultra：ultra待机模式下需要关闭的电源

rockchip,regulator-on-in-mem-ultra：ultra待机模式下需要打开的电源

注：ultra待机下，基本电都是关闭的，不建议做修改。

3 相关驱动

3.1 TP驱动

由于增加了lite待机机制，需要对TP驱动做一些处理；

目的：TP驱动在lite待机的时候需要能唤醒系统，并上报事件；

处理方法：TP驱动只有在灭屏的时候才需要关闭，亮屏的时候打开；

具体过程（例子drivers/input/touchscreen/cyttsp5/cyttsp5_core.c）：

1）驱动注册时，需要设置驱动中断可唤醒功能：

```
 rc = device_init_wakeup(dev, 1);

     if (rc < 0)

          dev_err(dev, "%s: Error, device_init_wakeup rc:%d\n",

                  __func__, rc);

     enable_irq_wake(cd->irq);
```

2）更改TP的休眠唤醒机制：

把原来的休眠唤醒注册去掉，改成注册一级休眠唤醒函数（在灭屏和亮屏的时候会调用，一级休眠的时候就可以关闭TP和TP的供电了）

```
     cd->tp.tp_resume = cyttsp5_core_late_resume;

     cd->tp.tp_suspend = cyttsp5_core_early_suspend;

     tp_register_fb(&cd->tp);
```

以上两步，保证了lite待机的时候，TP仍然可用，并可唤醒系统进行响应；

3）如果TP在lite休眠的时候还需要做一些什么工作，比如进入TP的低功耗模式之类的动作，可以在原来的suspend和resume函数中来做；

3.2 背光

背光请选用pmuio2 上的 pwm0 ~ pwm7，参考SDK背光补丁：

```
RKDocs/android/patches/ebook/0001-arm64-dts-rk3566-rk817-eink-w103-add-backlight-suppo.patch
```

3.3 hall sensor

Hall sensor（皮套）驱动参考SDK hall sensor补丁：

```
RKDocs/android/patches/ebook/0001-drivers-input-sensor-hall-sensor-mh248-support-ebook.patch
```

3.4 驱动获取休眠模式的说明

kernel驱动中，每次待机时都可以通过获取mem_sleep_current的值来判断此次休眠是哪个模式；

mem_sleep_current的值比较多，正常只需关注三种休眠模式的定义值即可：

PM_SUSPEND_MEM：deep sleep

PM_SUSPEND_MEM_LITE：lite sleep

PM_SUSPEND_MEM_ULTRA：ultra sleep

具体值定义在include/linux/suspend.h中，驱动中需要引用该头文件。

例子：

```
static int pwm_backlight_suspend(struct device *dev)
{
    struct backlight_device *bl = dev_get_drvdata(dev);
    struct pwm_bl_data *pb = bl_get_data(bl);

    if (mem_sleep_current == PM_SUSPEND_MEM_LITE)
        return 0;

    if (pb->notify)
        pb->notify(pb->dev, 0);

    pwm_backlight_power_off(pb);
```

```
    if (pb->notify_after)
        pb->notify_after(pb->dev, 0);

    return 0;
}

static int pwm_backlight_resume(struct device *dev)
{
    struct backlight_device *bl = dev_get_drvdata(dev);

    if (mem_sleep_current == PM_SUSPEND_MEM_LITE)
        return 0;

    backlight_update_status(bl);

    return 0;
}
```