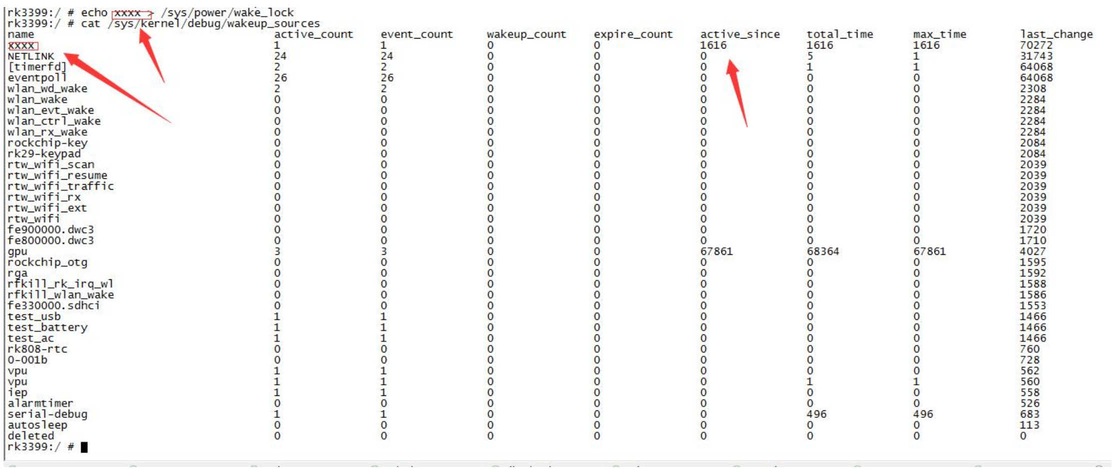
目录

1、休眠问题定位

1.1、wake_lock 导致无法休眠

可以通过 cat /sys/power/wake_lock（只能看到 Android 的锁无法看到 kernel 中设置的锁）

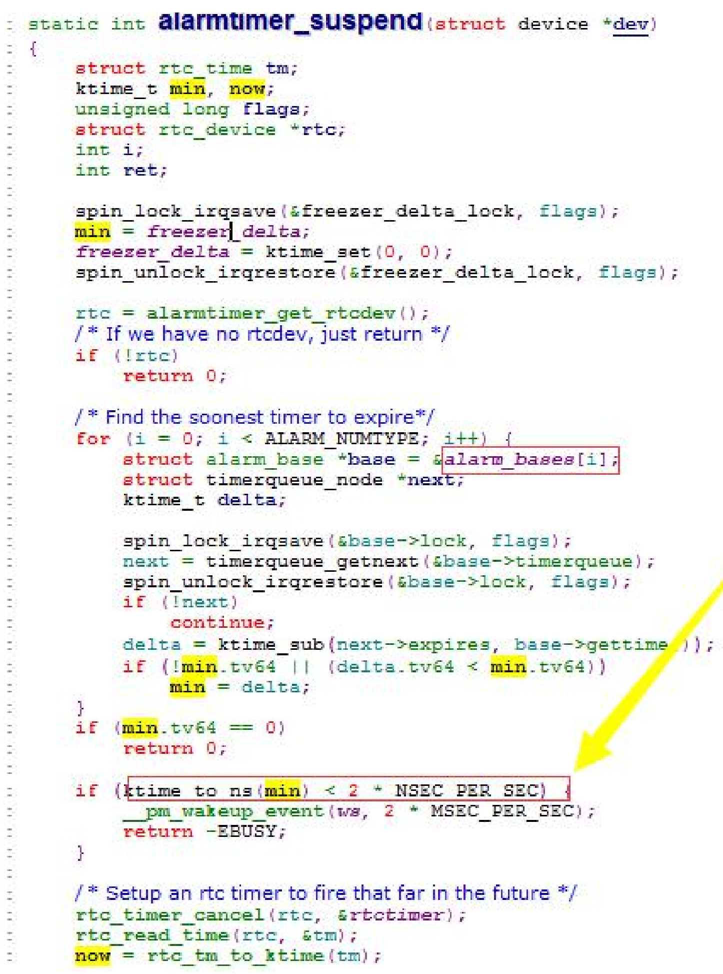
和 cat /sys/kernel/debug/wake_source（有锁都可以看到，即使你通过命令行 echo xxxxx> /sys/power/wake_lock 输入的）



```
rk3399:/ # echo xxxx > /sys/power/wake_lock
rk3399:/ # cat /sys/kernel/debug/wakeup_sources
name                                    active_count    event_count     wakeup_count    expire_count    active_since    total_time      max_time        last_change
XXXX                                    1               1               0               0               1616            1616            1616            70272
NETLINK                                 24              24              0               0               0               5               1               31743
[timerfd]                               2               2               0               0               0               1               1               64068
eventpoll                               26              26              0               0               0               0               0               64068
wlan_wd_wake                            2               2               0               0               0               0               0               2308
wlan_wake                               0               0               0               0               0               0               0               2284
wlan_evt_wake                           0               0               0               0               0               0               0               2284
wlan_ctrl_wake                          0               0               0               0               0               0               0               2284
wlan_rx_wake                            0               0               0               0               0               0               0               2284
rockchip-key                            0               0               0               0               0               0               0               2084
rk29-keypad                             0               0               0               0               0               0               0               2084
rtw_wifi_scan                           0               0               0               0               0               0               0               2039
rtw_wifi_resume                         0               0               0               0               0               0               0               2039
rtw_wifi_traffic                        0               0               0               0               0               0               0               2039
rtw_wifi_rx                             0               0               0               0               0               0               0               2039
rtw_wifi_ext                            0               0               0               0               0               0               0               2039
rtw_wifi                                0               0               0               0               0               0               0               2039
fe900000.dwc3                           0               0               0               0               0               0               0               1720
fe800000.dwc3                           0               0               0               0               0               0               0               1710
gpu                                     3               3               0               0               67861           68364           67861           4027
rockchip_otg                            0               0               0               0               0               0               0               1595
rga                                     0               0               0               0               0               0               0               1592
rfkill_rk_irq_wl                        0               0               0               0               0               0               0               1588
rfkill_wlan_wake                        0               0               0               0               0               0               0               1586
fe330000.sdhci                          0               0               0               0               0               0               0               1553
test_usb                                1               1               0               0               0               0               0               1466
test_battery                            1               1               0               0               0               0               0               1466
test_ac                                 1               1               0               0               0               0               0               1466
rk808-rtc                               0               0               0               0               0               0               0               760
0-001b                                  0               0               0               0               0               0               0               728
vpu                                     1               1               0               0               0               0               0               562
vpu                                     1               1               0               0               0               1               1               560
iep                                     1               1               0               0               0               0               0               558
alarmtimer                              0               0               0               0               0               0               0               526
serial-debug                            1               1               0               0               0               496             496             683
autosleep                               0               0               0               0               0               0               0               113
deleted                                 0               0               0               0               0               0               0               0
rk3399:/ # █
```

1.2、休眠的过程中断产生，或者驱动的休眠函数里面有 wake_lock

1.3、判断出 2S 内会有闹钟中断产生

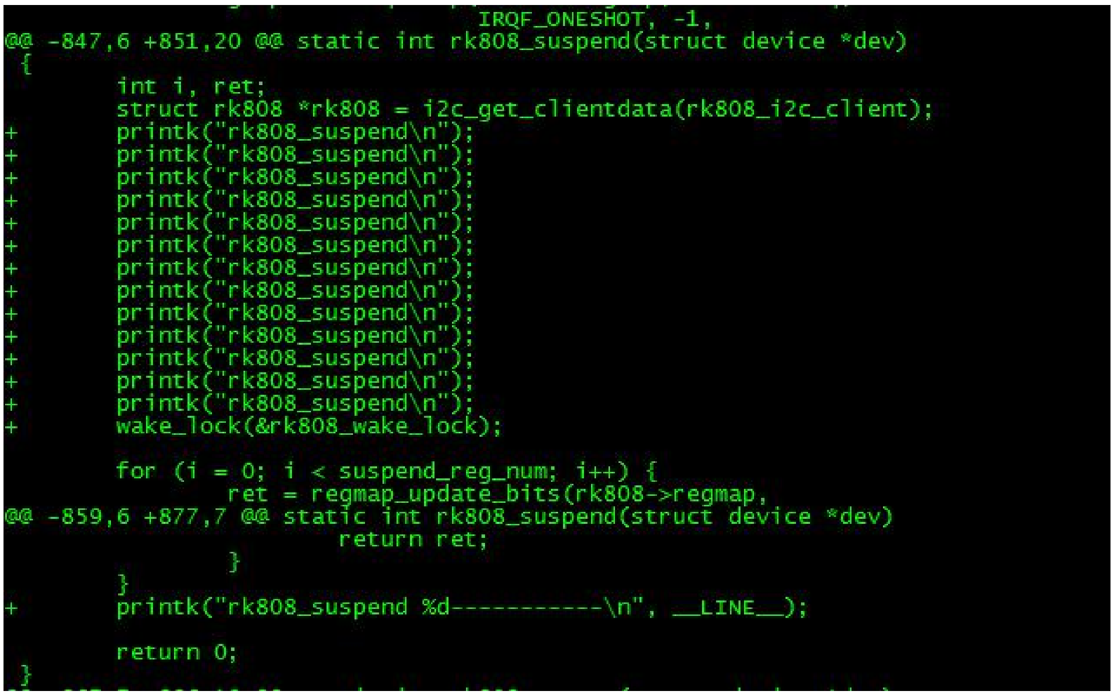


```
static int alarmtimer_suspend(struct device *dev)
{
    struct rtc_time tm;
    ktime_t min, now;
    unsigned long flags;
    struct rtc_device *rtc;
    int i;
    int ret;

    spin_lock_irqsave(&freezer_delta_lock, flags);
    min = freezer_delta;
    freezer_delta = ktime_set(0, 0);
    spin_unlock_irqrestore(&freezer_delta_lock, flags);

    rtc = alarmtimer_get_rtcdev();
    /* If we have no rtcdev, just return */
    if (!rtc)
        return 0;

    /* Find the soonest timer to expire*/
    for (i = 0; i < ALARM_NUMTYPE; i++) {
        struct alarm_base *base = &alarm_bases[i];
        struct timerqueue_node *next;
        ktime_t delta;

        spin_lock_irqsave(&base->lock, flags);
        next = timerqueue_getnext(&base->timerqueue);
        spin_unlock_irqrestore(&base->lock, flags);
        if (!next)
            continue;
        delta = ktime_sub(next->expires, base->gettime());
        if (!min.tv64 || (delta.tv64 < min.tv64))
            min = delta;
    }
    if (min.tv64 == 0)
        return 0;

    if (ktime_to_ns(min) < 2 * NSEC_PER_SEC) {
        __pm_wakeup_event(ws, 2 * MSEC_PER_SEC);
        return -EBUSY;
    }

    /* Setup an rtc timer to fire that far in the future */
    rtc_timer_cancel(rtc, &rtctimer);
    rtc_read_time(rtc, &tm);
    now = rtc_tm_to_ktime(tm);
```

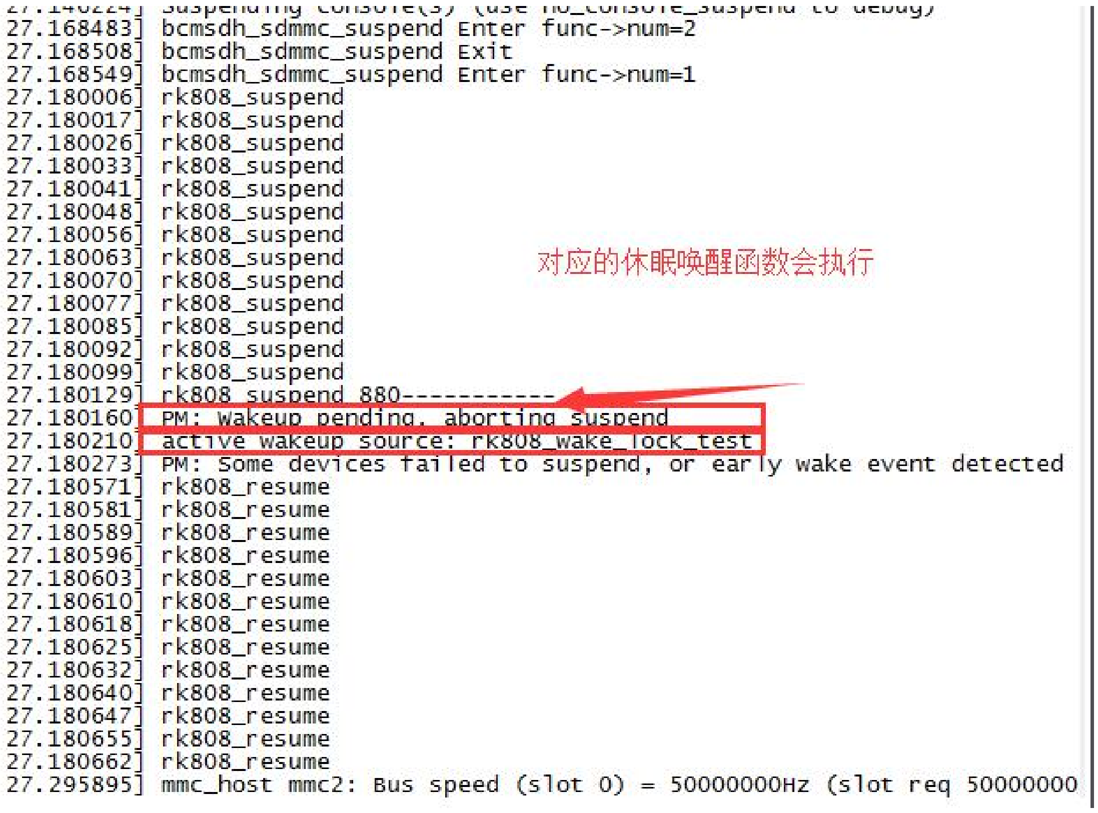
2S内就唤醒就不让睡。

1.4、实验

在驱动代码的休眠函数中加 wake_lock 阻止系统休眠（从对应的 log 可以看到，虽然休眠失败，带他会去执行对应驱动 resume 函数）



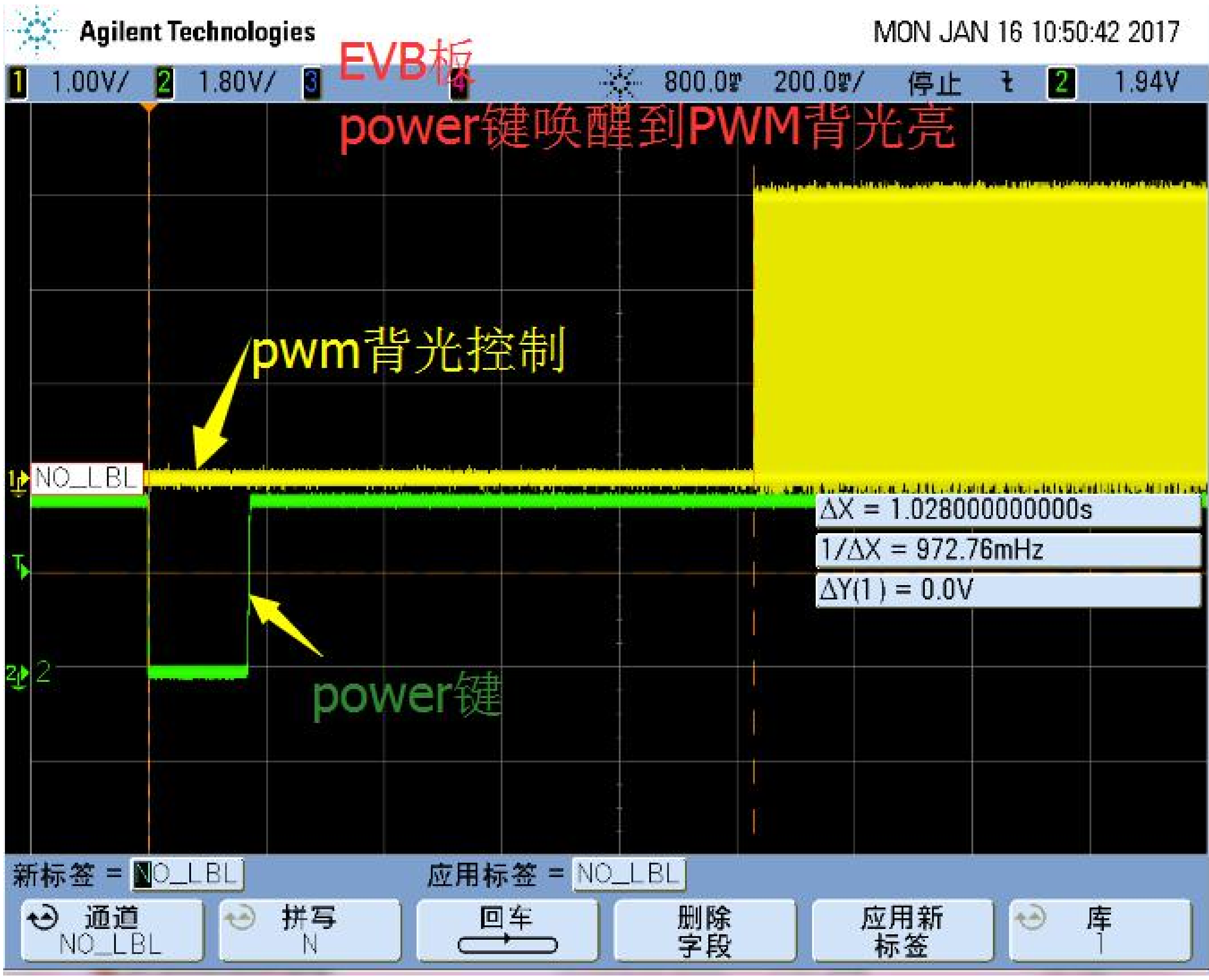
```
                                        IRQF_ONESHOT, -1,
@@ -847,6 +851,20 @@ static int rk808_suspend(struct device *dev)
 {
         int i, ret;
         struct rk808 *rk808 = i2c_get_clientdata(rk808_i2c_client);
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        printk("rk808_suspend\n");
+        wake_lock(&rk808_wake_lock);

         for (i = 0; i < suspend_reg_num; i++) {
                 ret = regmap_update_bits(rk808->regmap,
@@ -859,6 +877,7 @@ static int rk808_suspend(struct device *dev)
                         return ret;
                 }
         }
+        printk("rk808_suspend %d-----------\n", __LINE__);

         return 0;
 }
```



```
27.148224] suspending console(s) (use no_console_suspend to debug)
27.168483] bcmsdh_sdmmc_suspend Enter func->num=2
27.168508] bcmsdh_sdmmc_suspend Exit
27.168549] bcmsdh_sdmmc_suspend Enter func->num=1
27.180006] rk808_suspend
27.180017] rk808_suspend
27.180026] rk808_suspend
27.180033] rk808_suspend
27.180041] rk808_suspend
27.180048] rk808_suspend
27.180056] rk808_suspend
27.180063] rk808_suspend
27.180070] rk808_suspend
27.180077] rk808_suspend
27.180085] rk808_suspend
27.180092] rk808_suspend
27.180099] rk808_suspend
27.180129] rk808_suspend 880-----------
27.180160] PM: wakeup pending, aborting suspend
27.180210] active wakeup source: rk808_wake_lock_test
27.180273] PM: Some devices failed to suspend, or early wake event detected
27.180571] rk808_resume
27.180581] rk808_resume
27.180589] rk808_resume
27.180596] rk808_resume
27.180603] rk808_resume
27.180610] rk808_resume
27.180618] rk808_resume
27.180625] rk808_resume
27.180632] rk808_resume
27.180640] rk808_resume
27.180647] rk808_resume
27.180655] rk808_resume
27.180662] rk808_resume
27.295895] mmc_host mmc2: Bus speed (slot 0) = 50000000Hz (slot req 50000000
```

二、休眠唤醒流程时间

以 rk3399 为例，从按下 power 键到背光 pwm 有波形输出的时间在 972mS。

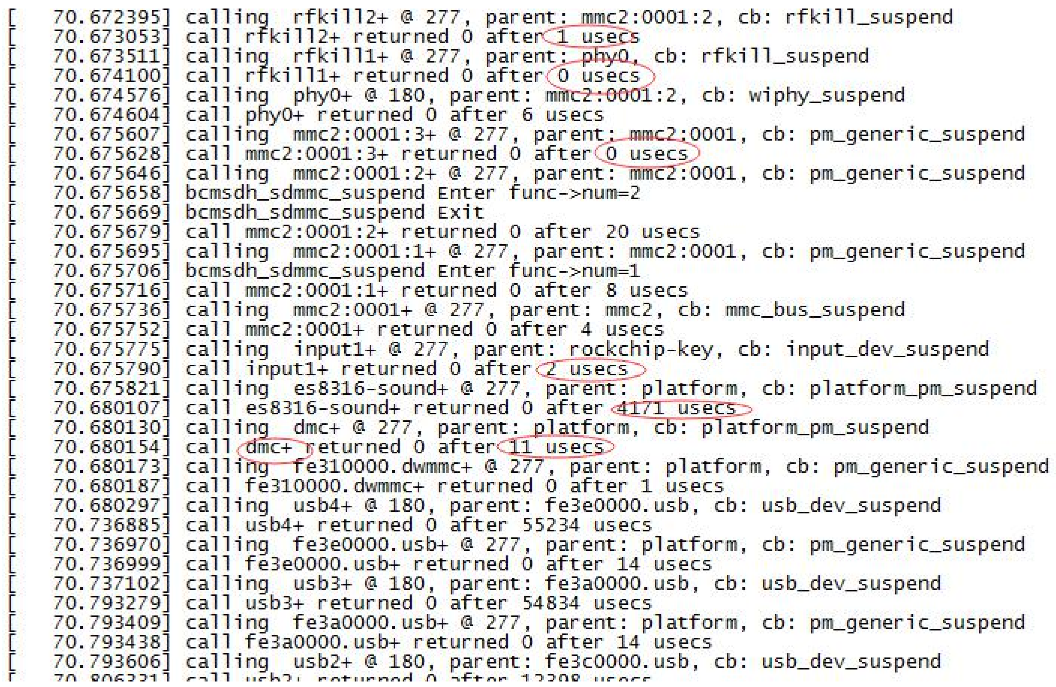


若发现休眠唤醒时间太长可以通过以下方法定位：

2.1、命令行形式定位

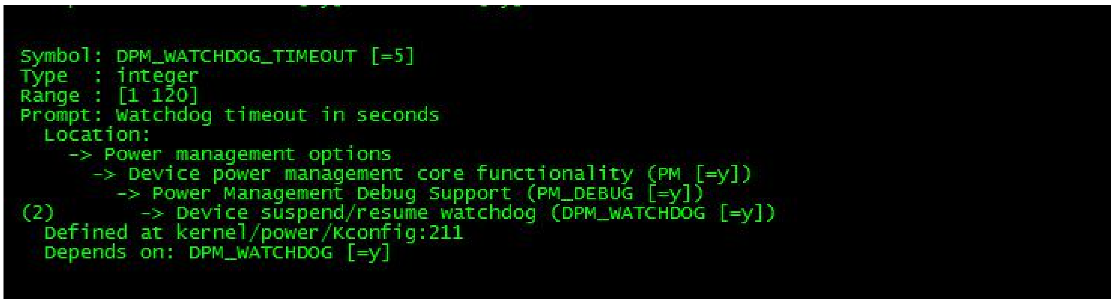
```
echo N > /sys/module/printk/parameters/console_suspend
echo 1 > /sys/power/pm_print_times
```

打印出每个设备休眠的耗时



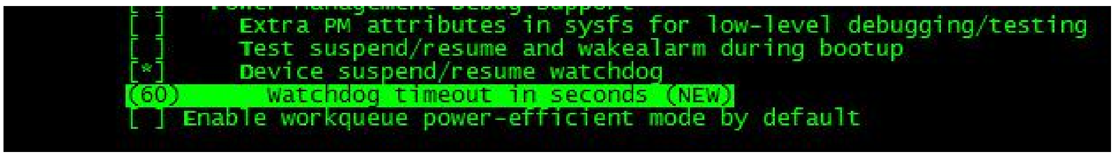
```
[   70.672395] calling  rfkill2+ @ 277, parent: mmc2:0001:2, cb: rfkill_suspend
[   70.673053] call rfkill2+ returned 0 after 1 usecs
[   70.673511] calling  rfkill1+ @ 277, parent: phy0, cb: rfkill_suspend
[   70.674100] call rfkill1+ returned 0 after 0 usecs
[   70.674576] calling  phy0+ @ 180, parent: mmc2:0001:2, cb: wiphy_suspend
[   70.674604] call phy0+ returned 0 after 6 usecs
[   70.675607] calling  mmc2:0001:3+ @ 277, parent: mmc2:0001, cb: pm_generic_suspend
[   70.675628] call mmc2:0001:3+ returned 0 after 0 usecs
[   70.675646] calling  mmc2:0001:2+ @ 277, parent: mmc2:0001, cb: pm_generic_suspend
[   70.675658] bcmsdh_sdmmc_suspend Enter func->num=2
[   70.675669] bcmsdh_sdmmc_suspend Exit
[   70.675679] call mmc2:0001:2+ returned 0 after 20 usecs
[   70.675695] calling  mmc2:0001:1+ @ 277, parent: mmc2:0001, cb: pm_generic_suspend
[   70.675706] bcmsdh_sdmmc_suspend Enter func->num=1
[   70.675716] call mmc2:0001:1+ returned 0 after 8 usecs
[   70.675736] calling  mmc2:0001+ @ 277, parent: mmc2, cb: mmc_bus_suspend
[   70.675752] call mmc2:0001+ returned 0 after 4 usecs
[   70.675775] calling  input1+ @ 277, parent: rockchip-key, cb: input_dev_suspend
[   70.675790] call input1+ returned 0 after 2 usecs
[   70.675821] calling  es8316-sound+ @ 277, parent: platform, cb: platform_pm_suspend
[   70.680107] call es8316-sound+ returned 0 after 4171 usecs
[   70.680130] calling  dmc+ @ 277, parent: platform, cb: platform_pm_suspend
[   70.680154] call dmc+ returned 0 after 11 usecs
[   70.680173] calling  fe310000.dwmmc+ @ 277, parent: platform, cb: pm_generic_suspend
[   70.680187] call fe310000.dwmmc+ returned 0 after 1 usecs
[   70.680297] calling  usb4+ @ 180, parent: fe3e0000.usb, cb: usb_dev_suspend
[   70.736885] call usb4+ returned 0 after 55234 usecs
[   70.736970] calling  fe3e0000.usb+ @ 277, parent: platform, cb: pm_generic_suspend
[   70.736999] call fe3e0000.usb+ returned 0 after 14 usecs
[   70.737102] calling  usb3+ @ 180, parent: fe3a0000.usb, cb: usb_dev_suspend
[   70.793279] call usb3+ returned 0 after 54834 usecs
[   70.793409] calling  fe3a0000.usb+ @ 277, parent: platform, cb: pm_generic_suspend
[   70.793438] call fe3a0000.usb+ returned 0 after 14 usecs
[   70.793606] calling  usb2+ @ 180, parent: fe3c0000.usb, cb: usb_dev_suspend
[   70.806331] call usb2+ returned 0 after 12398 usecs
```

2.2、打开 DPM_WATCHDOG_TIMEROUT

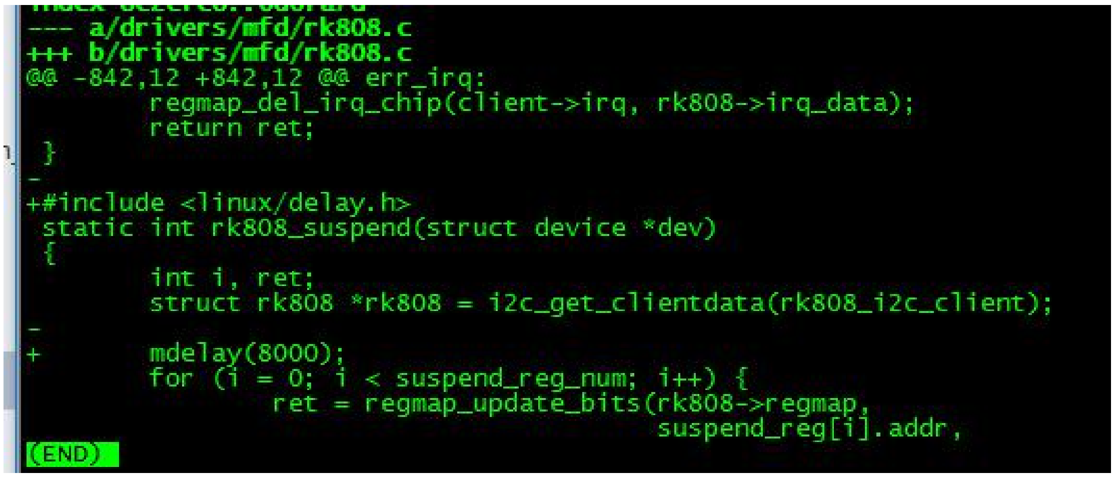


```
Symbol: DPM_WATCHDOG_TIMEOUT [=5]
Type  : integer
Range : [1 120]
Prompt: Watchdog timeout in seconds
  Location:
    -> Power management options
      -> Device power management core functionality (PM [=y])
        -> Power Management Debug Support (PM_DEBUG [=y])
(2)       -> Device suspend/resume watchdog (DPM_WATCHDOG [=y])
  Defined at kernel/power/Kconfig:211
  Depends on: DPM_WATCHDOG [=y]
```

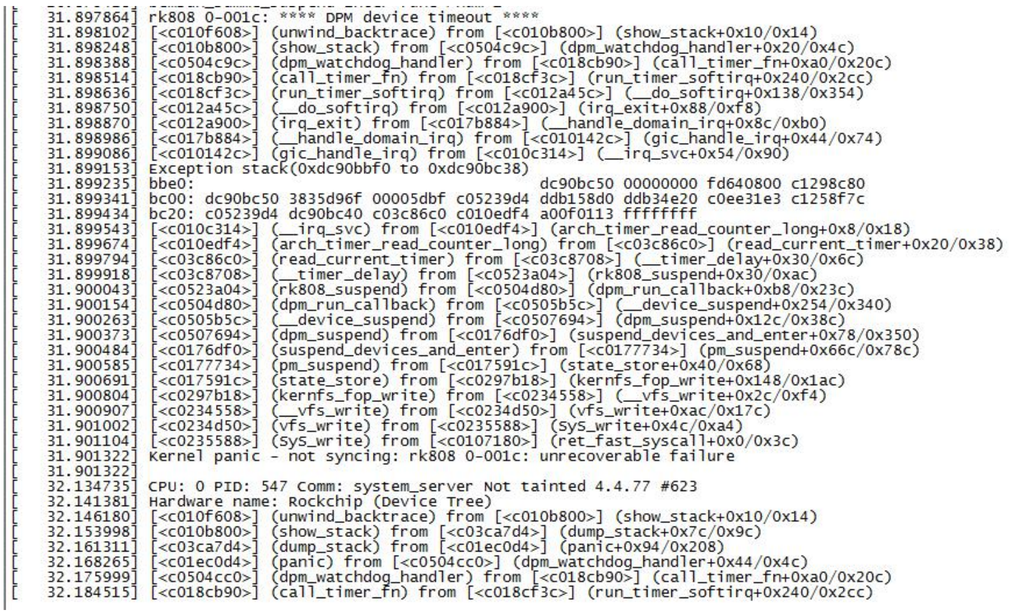
超时时间可配置，但只能精确到秒



```
        [ ]     Extra PM attributes in sysfs for low-level debugging/testing
        [ ]     Test suspend/resume and wakealarm during bootup
        [*]     Device suspend/resume watchdog
        (60)      Watchdog timeout in seconds (NEW)
        [ ] Enable workqueue power-efficient mode by default
```



```
--- a/drivers/mfd/rk808.c
+++ b/drivers/mfd/rk808.c
@@ -842,12 +842,12 @@ err_irq:
        regmap_del_irq_chip(client->irq, rk808->irq_data);
        return ret;
 }
-
+#include <linux/delay.h>
 static int rk808_suspend(struct device *dev)
 {
        int i, ret;
        struct rk808 *rk808 = i2c_get_clientdata(rk808_i2c_client);
-
+       mdelay(8000);
        for (i = 0; i < suspend_reg_num; i++) {
                ret = regmap_update_bits(rk808->regmap,
                                         suspend_reg[i].addr,
(END)
```



```
[   31.897864] rk808 0-001c: **** DPM device timeout ****
[   31.898102] [<c010f608>] (unwind_backtrace) from [<c010b800>] (show_stack+0x10/0x14)
[   31.898248] [<c010b800>] (show_stack) from [<c0504c9c>] (dpm_watchdog_handler+0x20/0x4c)
[   31.898388] [<c0504c9c>] (dpm_watchdog_handler) from [<c018cb90>] (call_timer_fn+0xa0/0x20c)
[   31.898514] [<c018cb90>] (call_timer_fn) from [<c018cf3c>] (run_timer_softirq+0x240/0x2cc)
[   31.898636] [<c018cf3c>] (run_timer_softirq) from [<c012a45c>] (__do_softirq+0x138/0x354)
[   31.898750] [<c012a45c>] (__do_softirq) from [<c012a900>] (irq_exit+0x88/0xf8)
[   31.898870] [<c012a900>] (irq_exit) from [<c017b884>] (__handle_domain_irq+0x8c/0xb0)
[   31.898986] [<c017b884>] (__handle_domain_irq) from [<c010142c>] (gic_handle_irq+0x44/0x74)
[   31.899086] [<c010142c>] (gic_handle_irq) from [<c010c314>] (__irq_svc+0x54/0x90)
[   31.899153] Exception stack(0xdc90bbf0 to 0xdc90bc38)
[   31.899235] bbe0:                                     dc90bc50 00000000 fd640800 c1298c80
[   31.899341] bc00: dc90bc50 3835d96f 00005dbf c05239d4 ddb158d0 ddb34e20 c0ee31e3 c1258f7c
[   31.899434] bc20: c05239d4 dc90bc40 c03c86c0 c010edf4 a00f0113 ffffffff
[   31.899543] [<c010c314>] (__irq_svc) from [<c010edf4>] (arch_timer_read_counter_long+0x8/0x18)
[   31.899674] [<c010edf4>] (arch_timer_read_counter_long) from [<c03c86c0>] (read_current_timer+0x20/0x38)
[   31.899794] [<c03c86c0>] (read_current_timer) from [<c03c8708>] (__timer_delay+0x30/0x6c)
[   31.899918] [<c03c8708>] (__timer_delay) from [<c0523a04>] (rk808_suspend+0x30/0xac)
[   31.900043] [<c0523a04>] (rk808_suspend) from [<c0504d80>] (dpm_run_callback+0xb8/0x23c)
[   31.900154] [<c0504d80>] (dpm_run_callback) from [<c0505b5c>] (__device_suspend+0x254/0x340)
[   31.900263] [<c0505b5c>] (__device_suspend) from [<c0507694>] (dpm_suspend+0x12c/0x38c)
[   31.900373] [<c0507694>] (dpm_suspend) from [<c0176df0>] (suspend_devices_and_enter+0x78/0x350)
[   31.900484] [<c0176df0>] (suspend_devices_and_enter) from [<c0177734>] (pm_suspend+0x66c/0x78c)
[   31.900585] [<c0177734>] (pm_suspend) from [<c017591c>] (state_store+0x40/0x68)
[   31.900691] [<c017591c>] (state_store) from [<c0297b18>] (kernfs_fop_write+0x148/0x1ac)
[   31.900804] [<c0297b18>] (kernfs_fop_write) from [<c0234558>] (__vfs_write+0x2c/0xf4)
[   31.900907] [<c0234558>] (__vfs_write) from [<c0234d50>] (vfs_write+0xac/0x17c)
[   31.901002] [<c0234d50>] (vfs_write) from [<c0235588>] (SyS_write+0x4c/0xa4)
[   31.901104] [<c0235588>] (SyS_write) from [<c0107180>] (ret_fast_syscall+0x0/0x3c)
[   31.901322] Kernel panic - not syncing: rk808 0-001c: unrecoverable failure
[   31.901322]
[   32.134735] CPU: 0 PID: 547 Comm: system_server Not tainted 4.4.77 #623
[   32.141381] Hardware name: Rockchip (Device Tree)
[   32.146180] [<c010f608>] (unwind_backtrace) from [<c010b800>] (show_stack+0x10/0x14)
[   32.153998] [<c010b800>] (show_stack) from [<c03ca7d4>] (dump_stack+0x7c/0x9c)
[   32.161311] [<c03ca7d4>] (dump_stack) from [<c01ec0d4>] (panic+0x94/0x208)
[   32.168265] [<c01ec0d4>] (panic) from [<c0504cc0>] (dpm_watchdog_handler+0x44/0x4c)
[   32.175999] [<c0504cc0>] (dpm_watchdog_handler) from [<c018cb90>] (call_timer_fn+0xa0/0x20c)
[   32.184515] [<c018cb90>] (call_timer_fn) from [<c018cf3c>] (run_timer_softirq+0x240/0x2cc)
```

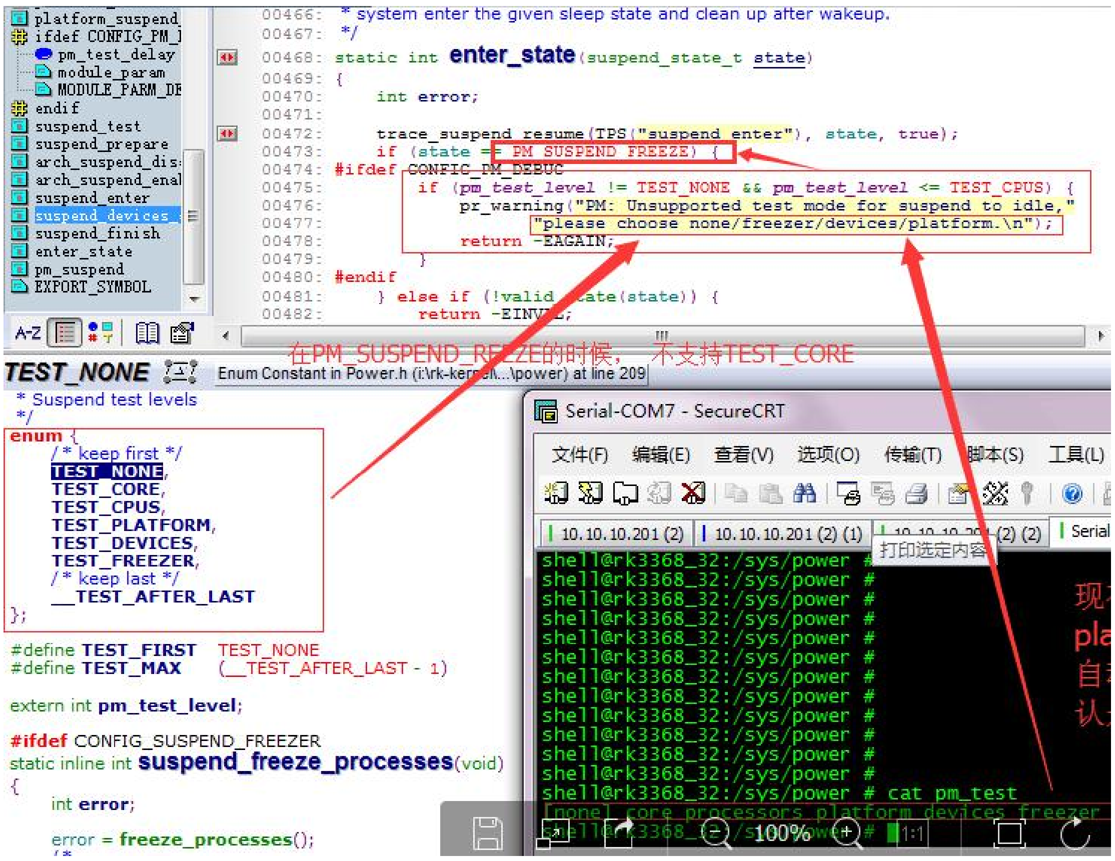
三、Pm-test 使用说明：

3.1、/sys/power/pm_test

可以设置 pm-test 唤醒的间隔时间，默认是 5s

```
rk3399:/ # cat /sys/power/pm_test
[none] core processors platform devices freezer

# echo 30 > /sys/module/suspend/parameters/pm_test_delay
# echo core > /sys/power/pm_test
# echo mem   > /sys/power/state
```

```
...
[    17.583625] suspend debug: Waiting for 30 second(s).
...
real          0m30.381s
user          0m0.017s
sys           0m0.080s
```



```
 * system enter the given sleep state and clean up after wakeup.
 */
static int enter_state(suspend_state_t state)
{
	int error;

	trace_suspend_resume(TPS("suspend_enter"), state, true);
	if (state == PM_SUSPEND_FREEZE) {
#ifdef CONFIG_PM_DEBUG
		if (pm_test_level != TEST_NONE && pm_test_level <= TEST_CPUS) {
			pr_warning("PM: Unsupported test mode for suspend to idle,"
				   "please choose none/freezer/devices/platform.\n");
			return -EAGAIN;
		}
#endif
	} else if (!valid_state(state)) {
		return -EINVAL;
```

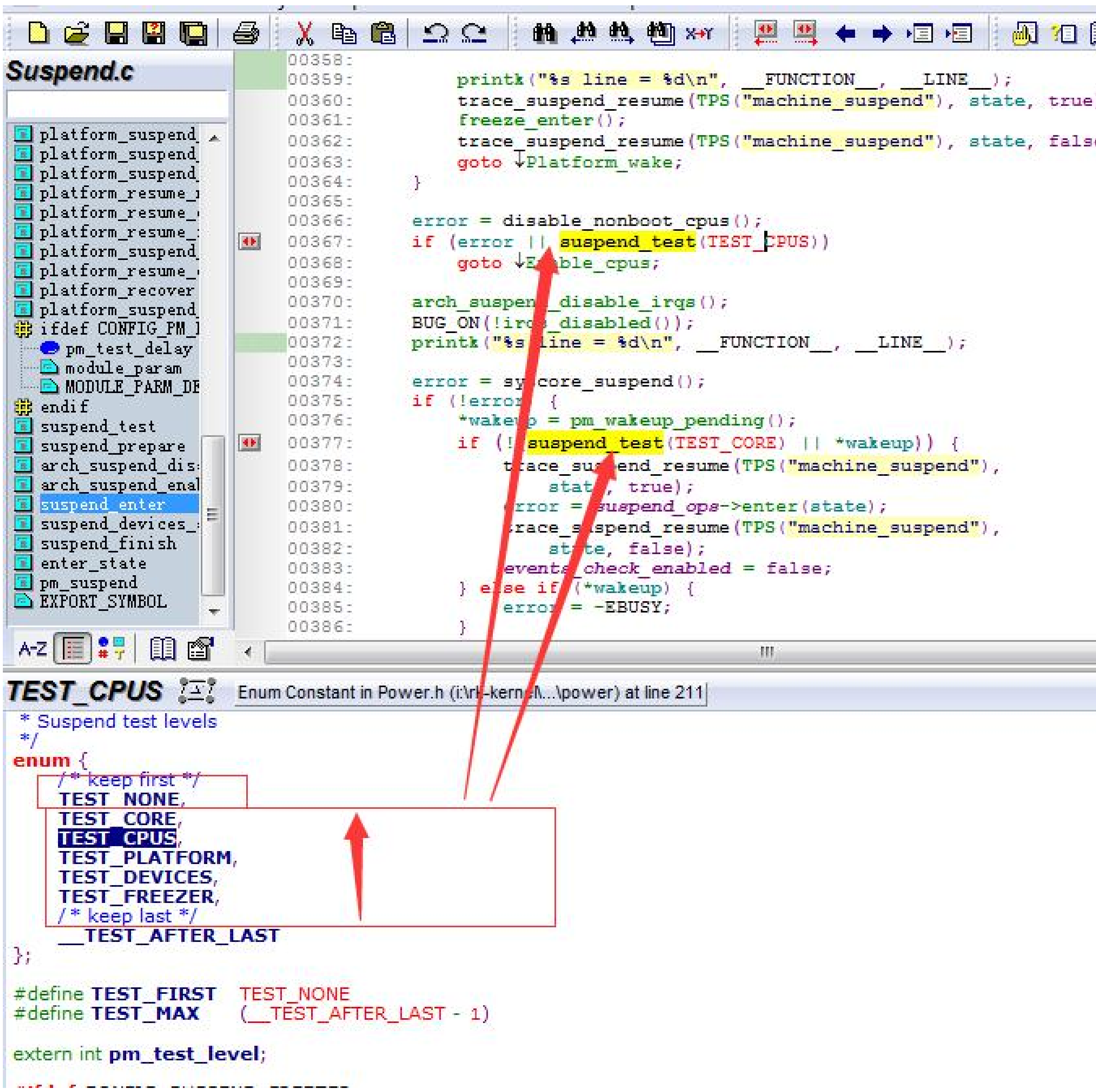
TEST_NONE   Enum Constant in Power.h (i:\rk-kernel\...\power) at line 209

```
 * Suspend test levels
 */
enum {
	/* keep first */
	TEST_NONE,
	TEST_CORE,
	TEST_CPUS,
	TEST_PLATFORM,
	TEST_DEVICES,
	TEST_FREEZER,
	/* keep last */
	__TEST_AFTER_LAST
};

#define TEST_FIRST	TEST_NONE
#define TEST_MAX	(__TEST_AFTER_LAST - 1)

extern int pm_test_level;

#ifdef CONFIG_SUSPEND_FREEZER
static inline int suspend_freeze_processes(void)
{
	int error;

	error = freeze_processes();
```

Serial-COM7 - SecureCRT

```
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power #
shell@rk3368_32:/sys/power # cat pm_test
[none] core processors platform devices freezer
shell@rk3368_32:/sys/power #
```

注意 PM_TEST 中的 TEST_FREEZER 与 PM_SUSPEND_FREEZE 这个事不一样的。


Suspend.c

```
platform_suspend
platform_suspend
platform_suspend
platform_resume_
platform_resume_
platform_resume_
platform_suspend
platform_resume_
platform_recover
platform_suspend
ifdef CONFIG_PM_I
    pm_test_delay
    module_param
    MODULE_PARM_DE
endif
suspend_test
suspend_prepare
arch_suspend_dis
arch_suspend_enal
suspend_enter
suspend_devices_
suspend_finish
enter_state
pm_suspend
EXPORT_SYMBOL
```

```

        printk("%s line = %d\n", __FUNCTION__, __LINE__);
        trace_suspend_resume(TPS("machine_suspend"), state, true
        freeze_enter();
        trace_suspend_resume(TPS("machine_suspend"), state, fals
        goto ↓Platform_wake;
    }

    error = disable_nonboot_cpus();
    if (error || suspend_test(TEST_CPUS))
        goto ↓Enable_cpus;

    arch_suspend_disable_irqs();
    BUG_ON(!irqs_disabled());
    printk("%s line = %d\n", __FUNCTION__, __LINE__);

    error = syscore_suspend();
    if (!error) {
        *wakeup = pm_wakeup_pending();
        if (!(suspend_test(TEST_CORE) || *wakeup)) {
            trace_suspend_resume(TPS("machine_suspend"),
                state, true);
            error = suspend_ops->enter(state);
            trace_suspend_resume(TPS("machine_suspend"),
                state, false);
            events_check_enabled = false;
        } else if (*wakeup) {
            error = -EBUSY;
        }
```

TEST_CPUS   Enum Constant in Power.h (i:\rk-kernel\...\power) at line 211

```
 * Suspend test levels
 */
enum {
    /* keep first */
    TEST_NONE,
    TEST_CORE,
    TEST_CPUS,
    TEST_PLATFORM,
    TEST_DEVICES,
    TEST_FREEZER,
    /* keep last */
    __TEST_AFTER_LAST
};

#define TEST_FIRST   TEST_NONE
#define TEST_MAX     (__TEST_AFTER_LAST - 1)

extern int pm_test_level;
```

3.2、Suspend_test 说明

kernel/power/suspend_test.c
注意确保 RTC 的驱动已经加载了。

在 4.4 的 kernel 里面有自带一个 suspend_test.c 是利用 rtc 定时唤醒系统（自测稳定性），

1、需要再 menuconfig 里面配置这个

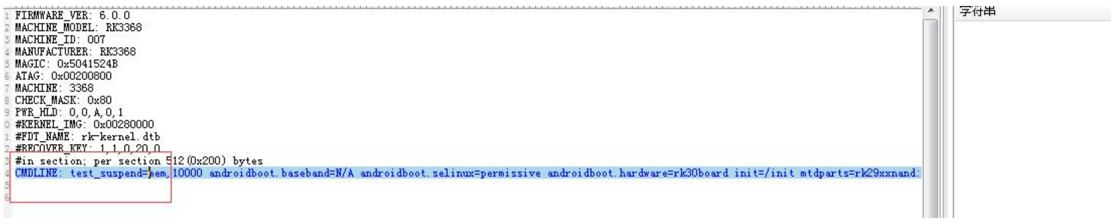
```
    -> Power management
options

                                 |
    |            -> Device power management core functionality (PM
[=y])
                                                             |
    | (1)        -> Power Management Debug Support (PM_DEBUG [=y])
   [*] Test suspend/resume and wakealarm during bootup
```

2、在 parameter 增加这个属性 test_suspend=mem,10

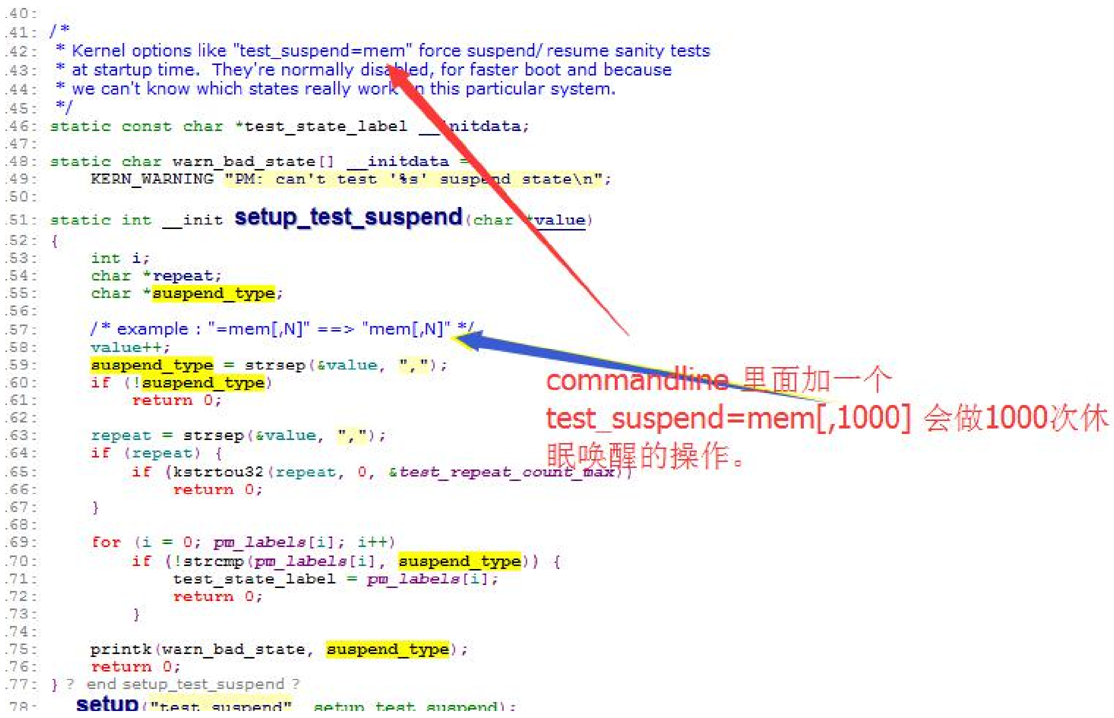
test_suspend=mem,10

这个 10 是拷机的次数，你可以换成 100,1000，

这个功能在平台开发前期会用到（系统没有跑进 Android）



```
FIRMWARE_VER: 6.0.0
MACHINE_MODEL: RK3368
MACHINE_ID: 007
MANUFACTURER: RK3368
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 3368
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
#KERNEL_IMG: 0x00280000
#FDT_NAME: rk-kernel.dtb
#RECOVER_KEY: 1,1,0,20,0
#in section; per section 512(0x200) bytes
CMDLINE: test_suspend=mem,10000 androidboot.baseband=N/A androidboot.selinux=permissive androidboot.hardware=rk30board init=/init mtdparts=rk29xxnand:
```



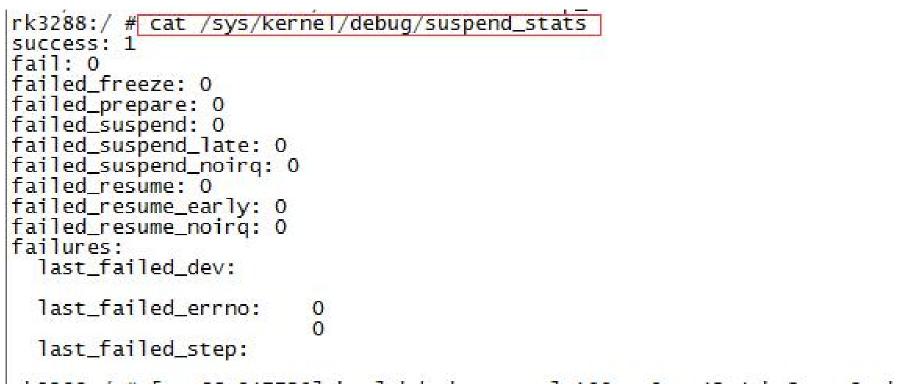
```
/*
 * Kernel options like "test_suspend=mem" force suspend/ resume sanity tests
 * at startup time.  They're normally disabled, for faster boot and because
 * we can't know which states really work on this particular system.
 */
static const char *test_state_label __initdata;

static char warn_bad_state[] __initdata =
    KERN_WARNING "PM: can't test '%s' suspend state\n";

static int __init setup_test_suspend(char *value)
{
    int i;
    char *repeat;
    char *suspend_type;

    /* example : "=mem[,N]" ==> "mem[,N]" */
    value++;
    suspend_type = strsep(&value, ",");
    if (!suspend_type)
        return 0;

    repeat = strsep(&value, ",");
    if (repeat) {
        if (kstrtou32(repeat, 0, &test_repeat_count_max))
            return 0;
    }

    for (i = 0; pm_labels[i]; i++)
        if (!strcmp(pm_labels[i], suspend_type)) {
            test_state_label = pm_labels[i];
            return 0;
        }

    printk(warn_bad_state, suspend_type);
    return 0;
} ? end setup_test_suspend ?
   setup("test_suspend", setup_test_suspend);
```

commandline 里面加一个
test_suspend=mem[,1000] 会做1000次休眠唤醒的操作。

四、重要的节点

4.1、节点/sys/kernel/debug/suspend_stats

（dev_pm_ops 相关的步骤）
查看之前休眠的状态，会总结休眠唤醒成功多少次，失败多少次及在哪一步失败；



```
rk3288:/ # cat /sys/kernel/debug/suspend_stats
success: 1
fail: 0
failed_freeze: 0
failed_prepare: 0
failed_suspend: 0
failed_suspend_late: 0
failed_suspend_noirq: 0
failed_resume: 0
failed_resume_early: 0
failed_resume_noirq: 0
failures:
  last_failed_dev:

  last_failed_errno:    0
                        0
  last_failed_step:
```

4.2、节点/sys/power/state

```
rk3399:/ # cat /sys/power/state
freeze mem
```

我们支持 freeze 和 mem 两种休眠方式

可以通过 echo mem > /sys/power/state 可以强制进入休眠；

4.3、节点/sys/power/pm_wakeup_irq

获取最近一次唤醒系统的中断号。

4.4、节点/sys/power/pm_print_times

Echo 1 > /sys/power/pm_print_times 打印每个设备休眠唤醒所用的时间，一般与/sys/module/printk/parameters/console_suspend 一起操作。

用于调试：

确认对应的休眠函数有没有被调用；

确认休眠和唤醒所用的时间；

4.5、节点/sys/kernel/debug/wake_source

打印出系统的锁，并可以看到其当前所处的状态，包含驱动和 Android 中的锁；

4.6、节点/sys/module/printk/parameters/console_suspend

echo N > /sys/module/printk/parameters/console_suspend 休眠的时候 console 不进休眠状态，可以打印更多 log。