Linux Standby

开发指南

版本号: 2.3
发布日期: 2025.06.19

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.08.04	AWA1743	添加初始版本。
1.1	2022.11.26	AWA1691	添加 A133 的适配。
1.2	2023.03.06	AWA1691	添加 A523 的适配。
1.3	2023.04.25	AWA1691	添加 MR527 的适配。
1.4	2023.06.21	AWA1691	添加 AI985 的适配。
1.5	2023.08.01	AWA1691	添加 T527 的适配。
1.6	2024.05.13	AWA0863	添加假关机的说明。
1.7	2024.09.11	AWA2152	T536/MR536 系列适用
1.8	2024.11.13	AWA2152	A733 系列适用
1.9	2025.03.29	AWA0863	A537/A333 系列适用
2.0	2025.04.07	AWA2099	T736 系列适用
2.1	2025.04.21	AWA0863	H723/H726/H727/TV323 系列适用
2.2	2025.06.03	AWA0863	补充 H728 配置说明
2.3	2025.06.19	AWA2152	补充 MR153/T153 配置说明

目　录

1 前言

1.1 文档简介

介绍 Standby 模块配置和调试方法。

1.2 目标读者

Standby 模块开发、维护人员。

1.3 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
T507/T507-H	Linux-5.10	kernel/power/
T517/T517-H	Linux-5.10	kernel/power/
A133	Linux-5.15	kernel/power/
A523	Linux-5.15	kernel/power/
MR527	Linux-5.15	kernel/power/
AI985	Linux-5.15	kernel/power/
T527	Linux-5.15	kernel/power/
H618	Linux-5.15	kernel/power/
H728	Linux-5.15	kernel/power/
MR536	Linux-5.15-origin	kernel/power/
T536	Linux-5.10-origin	kernel/power/
A733	Linux-6.6	kernel/power/
A537	Linux-6.6	kernel/power/
A333	Linux-6.6	kernel/power/
T736	Linux-5.15	kernel/power/
H723	Linux-5.15	kernel/power/
H726	Linux-5.15	kernel/power/
H727	Linux-5.15	kernel/power/
TV323	Linux-5.15	kernel/power/
MR153	Linux-5.15-origin	kernel/power/
T153	Linux-5.10-rt	kernel/power/

2 模块介绍

2.1 模块功能介绍

- 休眠唤醒指系统进入低功耗和退出低功耗模式，一般称之为 Standby。standby 分为 super standby 和 normal standby，区别是 cpu 是否掉电。
- 假关机是类似 standby 的一种低功耗模式。进入假关机，系统会先复位，再进入低功耗模式，等待唤醒源；检测到唤醒源，系统退出假关机，直接从低功耗模式复位重启。适用于 OTT 类产品代替常规的关机，实现红外/蓝牙遥控开机功能。

2.2 相关术语介绍

表 2-1: 术语介绍

术语	说明
Super standby	Vdd_cpu 掉电或 Core 掉电，dram 进入 self refresh 状态。
Normal standby	CPUX WFI，dram 进入自刷新状态。
WFI	Wait for interrupt，低功耗待机模式。
Fake Poweroff	假关机，类似 standby，主要区别是系统退出假关机会重启，而不是唤醒。
SCP/CPUS	全志平台辅助进行电源管理的协处理器。

2.3 模块配置介绍

Standby 模块在 dtsi 中无用户可用配置。

2.3.1 Device Tree 配置说明

2.3.1.1 uboot

1. 假关机是 u-boot 深度绑定功能，需要系统启动时根据 dts 配置检测，所以使用 u-boot dts 配置假关机参数

2. 只有部分平台支持假关机功能；假关机属于方案定制化功能，每个方案存在差异，不同方案支持的假关机的功能也不尽相同。以下仅以 H618 OTT 和 H726 投影方案为例，对假关机参数配置进行说明

在 uboot 中，主要有以下配置：

- H618/H728 OTT 假关机参数配置

```
box_start_os0 {
    compatible = "allwinner,box_start_os";
    start_type = <0x1>;
    irkey_used = <0x0>;
    pmukey_used = <0x0>;
    pmukey_num = <0x0>;
    led_power = <0x0>;
    led_state = <0x0>;
    pinctrl-0 = <&standby_blue>;
    pinctrl-1 = <&standby_red>;
    pinctrl-2 = <&standby_bt>;
    /*pinctrl-3 = <&standby_bt>;*/
}

...

&pio {
    standby_red: standby@0 {
        allwinner,pins = "PH5";
        allwinner,function = "gpio_out";
        allwinner,muxsel = <1>;
        allwinner,data = <1>;
        allwinner,drive = <0>;
        allwinner,pull = <0>;
    };
    standby_blue: standby@1 {
        allwinner,pins = "PH9";
        allwinner,function = "gpio_out";
        allwinner,muxsel = <1>;
        allwinner,data = <0>;
        allwinner,drive = <2>;
        allwinner,pull = <2>;
    };
    standby_bt: standby@2 {
        allwinner,pins = "PG16";
        allwinner,function = "gpio_in";
        allwinner,muxsel = <0>;
```

```
    allwinner,data = <0>;
    allwinner,drive = <0>;
    allwinner,pull = <0>;
    /*allwinner,eint = <1>;*/
  };
}

...

s_cir0 {
  s_cir0_used = <1>;
  ir_power_key_code0 = <0x40>;
  ir_addr_code0 = <0xfe01>;
  ir_power_key_code1 = <0x1a>;
  ir_addr_code1 = <0xfb04>;
  ir_power_key_code2 = <0xf2>;
  ir_addr_code2 = <0x2992>;
  ir_power_key_code3 = <0x57>;
  ir_addr_code3 = <0x9f00>;
  ir_power_key_code4 = <0xdc>;
  ir_addr_code4 = <0x4cb3>;
  ir_power_key_code5 = <0x18>;
  ir_addr_code5 = <0xff00>;
  ir_power_key_code6 = <0xdc>;
  ir_addr_code6 = <0xdd22>;
  ir_power_key_code7 = <0x0d>;
  ir_addr_code7 = <0xbc00>;
  ir_power_key_code8 = <0x4d>;
  ir_addr_code8 = <0x4040>;
  wakeup-source;
}
```

```
box_start_os0 {
  start_type <u32>
    0x0: 当系统启动时检测到是适配器上电启动，则进入假关机（如H700 TV）；如果是其他方式上电启动，如power按键，
      则不进入假关机。
    0x1: 当系统启动时不做适配器上电启动检测，即适配器上电启动也不进入假关机

  pinctrl-0、pinctrl-1 <u32>
    LED显示控制pin在假关机时对应的pinctrl配置，详见standby_red说明

  pinctrl-2、pinctrl-3 <u32>
    外部pin唤醒源（如蓝牙/wifi/rtc模块）在假关机时对应pinctrl配置，详见standby_bt说明
    pinctrl-2: 高脉冲有效，高脉冲持续时间需要大于200 ms；对于H728等支持“allwinner,eint”属性的平台，由该属性配置
      上升沿/下降沿/高电平/低电平有效
    pinctrl-3: 低脉冲有效，低脉冲持续时间需要大于200 ms；对于H728等支持“allwinner,eint”属性的平台，由该属性配置
      上升沿/下降沿/高电平/低电平有效

  irkey_used、pmukey_num、pmukey_num、led_power、led_state <u32>
    当前代码并未解析该节点，为无用节点，不影响假关机流程
}

&pio {
  standby_red: standby@0 {
    allwinner,pins = "PH5";
    allwinner,function = "gpio_out";
    allwinner,muxsel = <1>;
    allwinner,data = <1>;
    allwinner,drive = <0>;
    allwinner,pull = <0>;
```

```
    };
    假关机时PH5设置为输出高电平，控制红色LED显示

    standby_bt: standby@2 {
        allwinner,pins = "PG16";
        allwinner,function = "gpio_in";
        allwinner,muxsel = <0>;
        allwinner,data = <0>;
        allwinner,drive = <0>;
        allwinner,pull = <0>;
        /*allwinner,eint = <1>;*/
    };
    假关机时PG16设置为输入，检测蓝牙脉冲唤醒
}

s_cir0 {
    ir_power_key_code <u32>
        ir模块特定数据的码值，根据方案需求进行配置

    ir_addr_code <u32>
        ir模块特定地址的码值，根据方案需求进行配置
}
```

- H726 投影假关机参数配置

```
box_start_os0: box_start_os0 {
    compatible = "allwinner,box_start_os";
    start_type = <0x1>;

    power_key = "PL4";
    key_debounce = <50>; /* unit:ms */
    led_red = "PL0";
    led_blue = "PL1";
    bt_powerkey = "PM3";

    vdd-cpu = <0x00000002>;
    vdd-sys = <0x00000004>;
    vcc-pll = <0x00000008>;
    vcc-io  = <0x00000001>;
    vcc18-hdmi = <0x00000010>;
    vcc-dram-gpio = "PL6";
};

...

/* used for cpus */
cir_param: cir_param {
    gpio_group = "PL";
    gpio_pin = <10>;
    gpio_function = <2>;
    count = <15>;
    ir_power_key_code0 = <0x40>;
    ir_addr_code0 = <0xfe01>;
    ir_power_key_code1 = <0x1a>;
    ir_addr_code1 = <0xfb04>;
    ir_power_key_code2 = <0xf2>;
    ir_addr_code2 = <0x2992>;
    ir_power_key_code3 = <0x57>;
    ir_addr_code3 = <0x9f00>;
```

```
    ir_power_key_code4 = <0xdc>;
    ir_addr_code4 = <0x4cb3>;
    ir_power_key_code5 = <0x18>;
    ir_addr_code5 = <0xff00>;
    ir_power_key_code6 = <0xdc>;
    ir_addr_code6 = <0xdd22>;
    ir_power_key_code7 = <0x0d>;
    ir_addr_code7 = <0xbc00>;
    ir_power_key_code8 = <0x4d>;
    ir_addr_code8 = <0x4040>;
    ir_power_key_code9 = <0x08>;
    ir_addr_code9 = <0xfb04>;
    ir_power_key_code10 = <0x00>;
    ir_addr_code10 = <0xfc03>;
    ir_power_key_code11 = <0x00>;
    ir_addr_code11 = <0xbf00>;
    ir_power_key_code12 = <0xea>;
    ir_addr_code12 = <0xfb04>;
    ir_power_key_code13 = <0x42>;
    ir_addr_code13 = <0xbf00>;
    ir_power_key_code14 = <0x0014>;
    ir_addr_code14 = <0xFF00>;
};
```

```
box_start_os0 {
    start_type = <0x1>;
        同H618 OTT，不再赘叙

    power_key = "PL4";
        PL4配置为powerkey，检测到低电平唤醒
    key_debounce = <50>; /* unit:ms */
        powerkey去抖配置为50ms
    led_red = "PL0";
        当前代码并未解析该节点，为无用节点，不影响假关机流程
    led_blue = "PL1";
        当前代码并未解析该节点，为无用节点，不影响假关机流程
    bt_powerkey = "PM3";
        PM3配置为bt powerekey，检测到高电平唤醒

    vdd-cpu = <0x00000002>;
    vdd-sys = <0x00000004>;
    vcc-pll = <0x00000008>;
    vcc-io  = <0x00000001>;
    vcc18-hdmi = <0x00000010>;
        配置同standby参数配置，不再赘叙
    vcc-dram-gpio = "PL6";
        PL6控制vcc-dram供电，假关机时输出低电平，关闭vcc-dram
}

cir_param: cir_param {
        同H618 OTT，不再赘叙
}
```

2.3.1.2 kernel

在内核中，主要有以下配置：

1. aliases 配置
2. standby 参数配置
3. 唤醒源配置
4. 假关机参数配置，同 u-boot，不再赘叙

- aliases 配置

aliases 节点的别名信息。

“pmu0 = &pmu0;”，用于引用 pmu0 节点，解析 pmu 相关参数。

“standby_param = &standby_param;”，用于引用 standby_param 节点，解析 standby 相关参数。

```
aliases {
    ...
    pmu0 = &pmu0;
    standby_param = &standby_param;
    ...
};
```

- standby 参数配置

描述系统资源的相关信息：

```
standby_param: standby_param {
    vdd-cpu = <0x00000006>;
    vdd-sys = <0x00000008>;
    vcc-pll = <0x00000100>;
    osc24m-on = <0x1>;
    ...
};
```

```
vdd-xxx <u32> 或 <u64>
    表示系统休眠后vdd-xxx是否关闭，各个bit对应PMU的各路供电。
    以vdd-cpu = <0x00000006>为例，表示系统休眠后会关闭PMU第二、三路供电，即DCDC2、DCDC3。具体是关闭第几路电
        由具体PMU和硬件方案决定。
    0x0：该bit对应的PMU该路供电不关闭
    0x1：该bit对应的PMU该路供电关闭

osc24m-on <u32>
    表示系统休眠后osc24m是否关闭，默认不关闭。
    在休眠后需要使用WiFi的场景下，osc24m不关闭。
    0x0：可能关闭，取决于是否有其他模块使用
    0x1：不关闭
```

- 唤醒源配置

一般模块在配置上wakeup-source后即可设置成唤醒源，以 RTC 模块为例：

```
rtc: rtc@07000000 {
    compatible = "allwinner,sunxi-rtc";
    device_type = "rtc";
    wakeup-source;
    ...
};
```

```
wakeup-source <bool>
  配置wakeup-source后，该模块可以设置为唤醒源。
```

而 GPIO 为唤醒源配置略微不同，需要额外配置 gpio，以 wlan 驱动为例:

```
wlan {
    ......
    wlan_hostwake = <&r_pio PM 0 GPIO_ACTIVE_HIGH>;  (1)
    wakeup-source;                                   (2)
    ......
};
```

```
wlan_hostwake:
  配置GPIO中断，具体参数含义如下：
  wlan_hostwake = <&r_pio PM 0 GPIO_ACTIVE_HIGH>;
                  | | |
                  | | `-gpio active时状态，如果需要上下拉，还可以或上GPIO_PULL_UP、GPIO_PULL_DOWN标志
                  | `-----哪个bank
                  `---------指向哪个pio，属于cpus要用&r_pio

wakeup-source <bool>
  配置wakeup-source后，该模块可以设置为唤醒源。
```

表 2-2: 平台支持唤醒源列表

平台/唤醒源	GPIO	GPIO-cpus	电源按键	RTC	USB	红外	蓝牙/WiFi	MAD
T507/T507-H	否	否	super standby	super standby	super standby	否	super standby	否
T517/T517-H	否	否	super standby	super standby	super standby	否	super standby	否
A133	否	super standby	super standby	super standby	super standby	否	super standby	否
A523	否	super standby	super standby	super standby	super standby	否	super standby	否
MR527	否	super standby	super standby	super standby	super standby	否	super standby	否
AI985	否	super standby	super standby	super standby	super standby	否	super standby	否
T527	否	super standby	super standby	super standby	super standby	否	super standby	否

平台/唤醒源	GPIO	GPIO-cpus	电源按键	RTC	USB	红外	蓝牙/WiFi	MAD
H618	normal standby	否	normal standby	normal standby	normal standby	normal standby	normal standby	否
H728	否	super standby	super standby	super standby	super standby	super standby	super standby	否
MR536	否	super standby	super standby	super standby	super standby	否	super standby	否
T536	否	super standby	super standby	super standby	super standby	否	super standby	否
A733	否	super standby	super standby	super standby	super standby	否	super standby	否
T736	否	super standby	super standby	super standby	super standby	否	super standby	否
A537	否	super standby	super standby	super standby	super standby	否	super standby	否
A333	否	super standby	super standby	super standby	super standby	否	super standby	否
H723	否	super standby	super standby	super standby	否	super standby	super standby	否
H726	否	super standby	super standby	super standby	否	super standby	super standby	否
H727	否	super standby	super standby	super standby	否	super standby	super standby	否
TV323	否	super standby	super standby	super standby	否	super standby	super standby	否
MR153	否	super standby	super standby	super standby	super standby	否	super standby	否
T153	否	super standby	super standby	super standby	super standby	否	super standby	否

表 2-3: 平台支持假关机开机源列表

平台/唤醒源	GPIO	GPIO-cpus	电源按键	RTC	USB	红外	蓝牙/WiFi	MAD
H618	normal standby	否	normal standby	normal standby	normal standby	normal standby	normal standby	否
H728	否	super standby	super standby	super standby	super standby	super standby	super standby	否
H723	否	super standby	super standby	super standby	否	super standby	super standby	否

平台/唤醒源	GPIO	GPIO-cpus	电源按键	RTC	USB	红外	蓝牙/WiFi	MAD
H726	否	super standby	super standby	super standby	否	super standby	super standby	否
H727	否	super standby	super standby	super standby	否	super standby	super standby	否
TV323	否	super standby	super standby	super standby	否	super standby	super standby	否

2.3.2 kernel menuconfig 配置说明

在命令行中进入 longan 目录，执行./build.sh menuconfig，并按以下步骤操作。

- 内核 PSCI 选项

```
Kernel Features  --->
  [*] Support for the ARM Power State Coordination Interface (PSCI)
```

- 内核 CPUIDLE 相关选项（可选）

```
CPU Power Management  --->
  CPU Idle  --->
    [*] CPU idle PM support
    ARM CPU Idle Drivers  --->
      [*] Generic ARM/ARM64 CPU idle Driver
```

- 内核 POWER 相关选项

```
Power management options  --->
  [*] Suspend to RAM and standby
  [ ] Opportunistic sleep
  [*] User space wakeup sources interface
  (100) Maximum number of user space wakeup sources (0 = no limit)
  -*- Device power management core functionality
  [*]   Power Management Debug Support
  [*]     Extra PM attributes in sysfs for low-level debugging/testing
```

- 内核假关机 FAKE_POWEROFF 相关选项

```
Device Drivers  --->
  [*] Real Time Clock  --->
    [*]    support ir fake poweroff
```

2.3.3 u-boot defconfig 配置说明

在 uboot 中，主要有以下配置：

- uboot-2018 假关机 FAKE_POWEROFF 相关配置

在平台的 defconfig 中，将 CONFIG_ATF_BOX_STANDBY 或 CONFIG_SUNXI_BOX_STANDBY_NG 配置为 Y 后即可打开假关机配置

```
CONFIG_ATF_BOX_STANDBY = Y;
```

2.4 源码结构介绍

Standby 的源代码位于内核kernel/power/目录下：

```
kernel/power/
├── autosleep.c
├── console.c
├── hibernate.c
├── Kconfig
├── main.c
├── Makefile
├── modules.builtin
├── modules.order
├── power.h
├── poweroff.c
├── process.c
├── qos.c
├── snapshot.c
├── suspend.c
├── suspend_test.c
├── swap.c
├── user.c
├── wakelock.c
└── wakeup_reason.c
```

2.5 驱动框架介绍

休眠唤醒指系统进入低功耗和退出低功耗模式，一般称之为 Standby。休眠过程由应用发起，经由内核的电源管理框架来进行休眠唤醒管理工作，如果存在 CPUS（一颗集成在 IC 内部的对电源

进行管理的 openrisc 核，是 SoC 内置的超低功耗硬件管理模块），最终会传递到到 CPUS。因此休眠唤醒类出现问题的可能为应用层、内核层、CPUS 层，如果不存在 CPUS，则 CPU 进入 WFI。休眠唤醒流程图如下，虚线部分为部分内核实现。

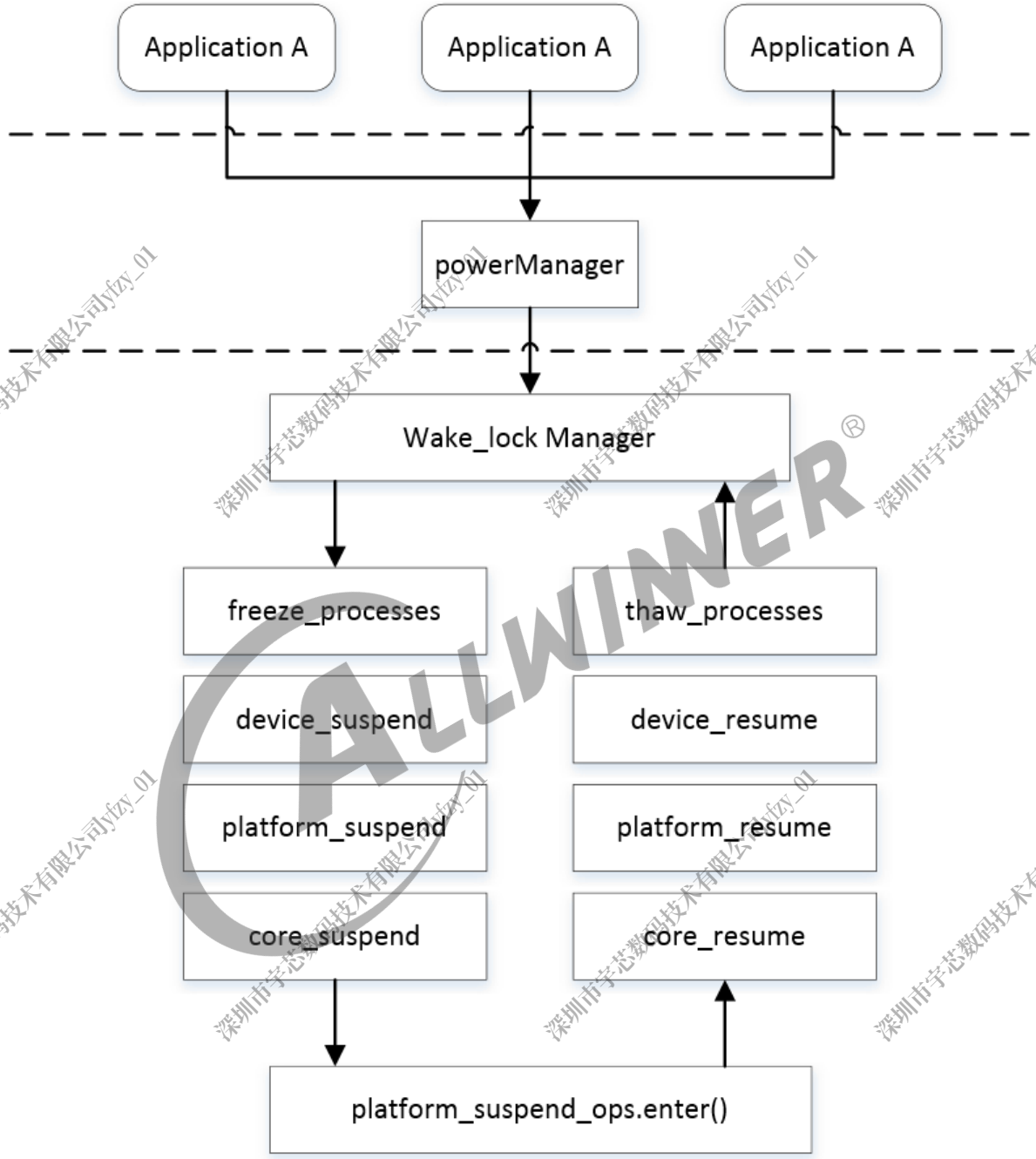

图 2-1: standby 驱动总体结构

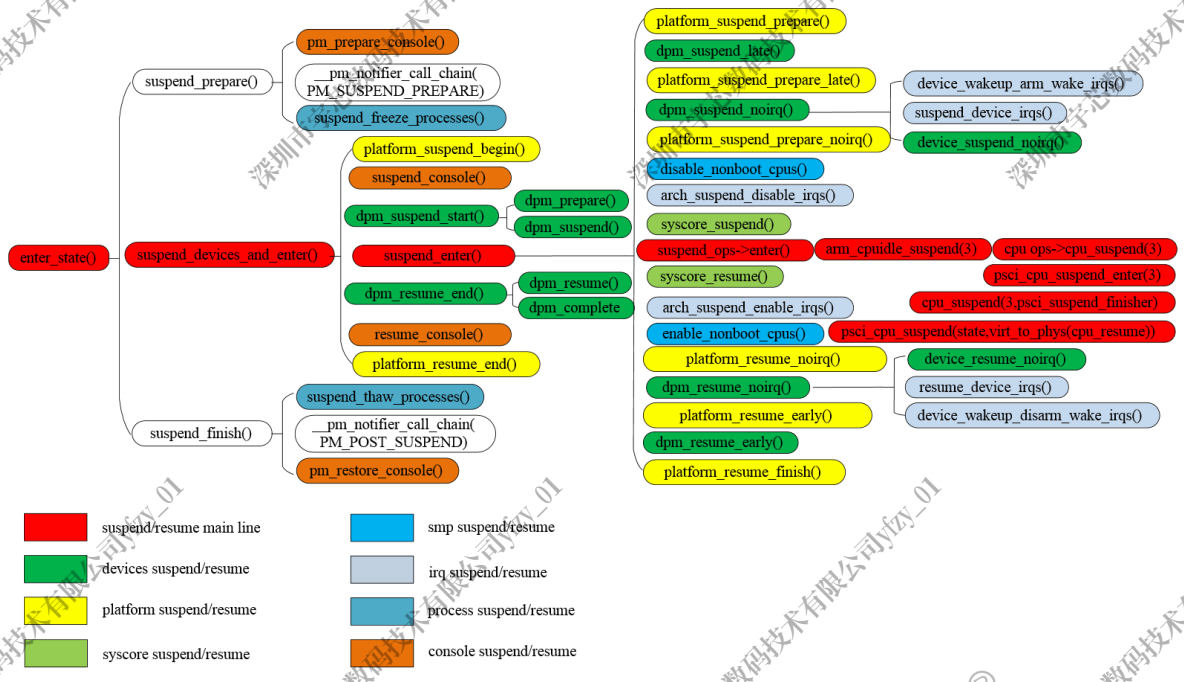

图 2-2: linux standby 流程

3 FAQ

3.1 调试方法

3.1.1 调试节点

- pm_test 节点

该节点可用于测试 linux 部分休眠唤醒功能。

```
echo x > /sys/power/pm_test。
```

Freezer：表明，任务冻结后，等待 5s，即返回，执行唤醒动作。

Devices: 表明，设备冻结后，等待 5s, 即返回，执行唤醒动作。

Platform：在a1x, a2x, a3x上，与 devices 相同；

Processors: 冻结 non-boot cpu 后，等待 5s, 即返回，执行唤醒动作。

Core: 冻结 timer 等系统资源后，等待 5s, 即返回，执行唤醒动作。

None: 表明，整个休眠流程全部走完，等待唤醒源唤醒。

- wake_lock 节点

该节点可查看安卓系统 wake lock 状态，安卓系统在持锁时不会进入深度睡眠流程 (Suspend-to-mem)。

```
cat /sys/power/wake_lock。
```

- wakeup_sources 节点

该节点可查看系统唤醒源的情况。

```
cat /sys/kernel/debug/wakeup_sources。
```

3.2 常见问题

3.2.1 系统被错误唤醒

3.2.1.1 系统被定时器唤醒

问题现象

休眠后，自动被唤醒，过会自动进入休眠，屏幕黑屏，串口有输出。

问题分析

系统休眠后自动被唤醒，原因可能是，某些应用或者后台进程，通过设置闹钟的方式，定时唤醒系统。

当出现如下打印，表示 Linux 已经休眠完成，准备进入 CPUS 休眠阶段：

```
[ 3465.885063] PM: noirq suspend of devices complete after 16.487 msecs
[ 3465.892225] Disabling non-boot CPUs ...

......
```

当出现以上打印后自动唤醒，则查看如下打印：

```
PM: PM: Pending Wakeup Sources: 7090000.rtc

后面的字符代表唤醒源名字，rtc指的是闹钟
```

使用范围

常见场景：android 某些应用或者后台进程，会通过设置闹钟的方式，定时唤醒系统

问题解决

确认是某些应用或者后台进程设置闹钟定时唤醒系统，方案开发人员可以自行解决。

3.2.1.2 系统被其他唤醒源唤醒

问题现象

休眠后，被异常唤醒。

问题分析

系统休眠后被异常唤醒，原因可能是，被其他非预期的唤醒源唤醒。

查看关键打印：

```
PM: PM: Pending Wakeup Sources: NETLINK
```

后面的字符代表唤醒源名字，可以跟据名字排查对应驱动

问题解决

确认是其他非预期的唤醒源唤醒系统，方案开发人员可以自行解决。

3.2.2 系统不能被唤醒

3.2.2.1 休眠后无法唤醒

问题现象

系统休眠后无法唤醒。

问题分析

系统休眠后无法唤醒，原因可能有：

- 模块休眠失败。

查看是否模块休眠失败，输入以下命令，确认是否在内核休眠唤醒模块出异常。

由于在 GKI 默认配置配置下部分节点不可用

建议调试时打开CONFIG_PM_ADVANCED_DEBUG、 CONFIG_PM_DEBUG的配置

```
CONFIG_PM_DEBUG=y
CONFIG_PM_ADVANCED_DEBUG=y
```

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

- 若上述无异常打印，则认为是 Linux 后的阶段出现异常。

不断电重启系统，将启动时候的 RTC 寄存器的信息发给休眠模块负责人，根据 RTC 寄存器信息判断。

```
[2341]HELLO! pmu_init stub called!
[2645]set pll start
[2648]set pll end
[2649]try to probe rtc region
[2652]rtc[0] value = 0x00000000
[2655]rtc[1] value = 0x000000e0
[2658]rtc[2] value = 0xf1f18000
[2661]rtc[3] value = 0x0000000f
[2663]rtc[4] value = 0x00000000
[2666]rtc[5] value = 0x00000000
```

问题解决

- 模块休眠失败。确认是模块休眠失败，方案开发人员可以自行解决。
- Linux 后，即 cpus 的阶段中出现异常。将复位重启时的 RTC 寄存器信息发给相关负责人。

3.2.2.2 唤醒源不支持唤醒

问题现象

休眠后，唤醒源无法唤醒系统，串口没有输出。

问题分析

休眠后，唤醒源无法唤醒，可能是唤醒源不支持。

- cpus 休眠后异常。

当出现如下打印时表示 Linux 休眠已经完毕，此时唤醒不了，则可能是 cpus 退出休眠失败，或者唤醒源不对。

```
[ 3465.885063] PM: noirq suspend of devices complete after 16.487 msecs
[ 3465.892225] Disabling non-boot CPUs ...
.....
```

通过以下手段可以判断 cpus 休眠后是否正常运行，以下命令表示休眠后 cpus 过一定时间软件自动唤醒。

```
echo 1000 > /sys/module/suspend/parameters/time_to_wakeup //休眠1000ms后自动唤醒
```

如果串口能正常打印，wake up source 为 0x400000，则表示 cpus 是正常运行的，这时应该排查一下系统是否支持相应的唤醒源。

- 唤醒源不支持。

确认唤醒源不支持的情况。

问题解决

- cpus 休眠后异常。

将复位重启时的 RTC 寄存器信息发给相关负责人。

- 唤醒源不支持。

将唤醒源的情况发给相关负责人。

3.2.2.3 红外遥控器不能唤醒系统

问题现象

红外遥控不能唤醒系统。

问题分析

红外遥控唤醒需要配置唤醒源。

问题解决

红外遥控器默认支持 NEC 红外协议遥控器唤醒，也支持 RC5 红外协议遥控器唤醒，但是需要在 sys_config.fex 进行配置，配置如下：

```
;----------------------------------------------------------------------------
;ir --- infra remote configuration
;ir_protocol_used : 0 = NEC / 1 = RC5
;----------------------------------------------------------------------------
[s_cir0]
s_cir0_used = 1
ir_used = 1
ir_protocol_used = 0 //置为0 支持NEC 红外协议遥控唤醒，配置为0 支持RC5 红外协议遥控唤醒
```

对于同一协议的红外遥控器，能支持唤醒的个数也是有限的，具体在 sys_config.fex 配置 s_cir0 节点。

```
ir_power_key_code0 = 0x57 //遥控POWER 值
ir_addr_code0 = 0x9f00 //遥控设备码
ir_power_key_code1 = 0x1a
ir_addr_code1 = 0xfb04
```

3.2.2.4 USB 设备不能唤醒系统

问题现象

USB 不能唤醒系统。

问题分析

USB 需要设置为唤醒源。

问题解决

USB 设备唤醒需要系统支持 USB_STANDBY，需要在 board.dts 配置

```
&usbc0 {
……
   usb_wakeup_suspend = <1>;
   wakeup-source;
……
};
```

```
usb_wakeup_suspend <u32>
  1: 支持USB0唤醒
  0: 屏蔽该唤醒源
```

3.2.2.5 cpus 退出休眠失败

问题现象

休眠后，无法唤醒，串口没有输出。

问题分析

可能是 cpus 退出休眠失败。

如果通过写 time_to_wakeup 命令，系统没法正常唤醒，则考虑是 cpus 退出休眠失败的了，这时需要短接 reset 脚重启系统（注意不是完全断电，完全断电将无法保留 RTC 值），然后将 boot 阶段打印的 RTC 码值发送给休眠唤醒负责人，定位问题。

```
[2341]HELLO! pmu_init stub called!
[2645]set pll start
[2648]set pll end
[2649]try to probe rtc region
[2652]rtc[0] value = 0x00000000
[2655]rtc[1] value = 0x000000e0
[2658]rtc[2] value = 0xf1f18000
[2661]rtc[3] value = 0x0000000f
[2663]rtc[4] value = 0x00000000
[2666]rtc[5] value = 0x00000000
```

问题解决

- 确认是否 dram 错误。
- 确认是否上下电时序错误。
- 将复位重启时的 RTC 寄存器信息发给相关负责人。

3.2.3 系统无法休眠

3.2.3.1 系统持锁无法休眠

问题现象

系统持锁，suspend 失败。

问题分析

suspend 失败，可能是系统持锁阻止休眠。

问题解决

- 安卓查看是否有持锁相关信息：

```
dumpsys power | grep PART
```

- 内核中是否有相关持锁信息：

```
cat /sys/kernel/debug/wakeup_sources
```

查看 active_since 项，若对应模块不为 0，则该模块一直阻止系统进入休眠，查看该模块是否异常；

```
cat /sys/power/wake_lock
```

查看是否有安卓申请的锁。

3.2.3.2 Android 系统持锁无法休眠

问题现象

定时休眠到时后，屏幕亮屏，串口可以输入，系统无法休眠。

问题分析

系统无法休眠，确认 Android 是否支持，是否禁止定时休眠。

串口输入 dumpsys power，查看如下打印。如果发现 mStayOn=true，则系统是不支持定时休眠功能，可以将该属性设置成 false；

另外 screen off timeout 表示休眠时间，可通过该值判断你设置的定时时间是否正确；

```
Power Manager State:
......
mStayOn=true //true 保持常亮，false 可以支持定时休眠
Sleep timeout: -1 ms //只有androidN 平台无该该值，定时休眠时间
Screen off timeout: 1800000 ms //定时灭屏时间
Screen dim duration: 7000 ms //屏保时间
```

常见场景：eng 固件开发阶段为了测试长时间老化的测试项，会禁止系统定时进入休眠。

问题解决

确认是 Android 系统持锁阻止休眠，方案开发人员可以自行解决。

3.2.4 休眠唤醒过程中挂掉

3.2.4.1 分阶段过程挂掉

问题现象

在 Linux 某个阶段出现的休眠或者唤醒失败。

问题分析

打开内核选项。

```
CONFIG_PM_DEBUG=y
CONFIG_PM_ADVANCED_DEBUG=y
```

查看具体失败在哪个阶段：

```
echo freezer > /sys/power/pm_test
```

查看 freezer 阶段是否正常；

```
echo devices > /sys/power/pm_test
```

查看freezer/devices 阶段是否正常；

```
echo platform > /sys/power/pm_test
```

查看freezer/devices/platform 阶段是否正常；

```
echo processors > /sys/power/pm_test
```

查看freezer/devices/platform/processors 阶段是否正常；

```
echo core > /sys/power/pm_test
```

查看freezer/devices/platform/processors/core 阶段是否正常；

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

测试分阶段休眠唤醒。

问题解决

确认是哪个阶段出现的休眠或者唤醒失败，方案开发人员可以自行解决。

著作权声明

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明





（不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。