D1-H Linux Standby

开发指南

版本号: 1.0
发布日期: 2021.2.4

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.2.4	AWA1556	1.D1-H 初始化版本

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
D1-H	Linux-5.4	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/蓝牙开机功能,D1-H 目前不支持假关机功能。

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 self refresh 状态
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

- 唤醒源配置

以 RTC 模块为例，RTC 驱动支持通过 “wakeup-source” 配置是否作为唤醒源；在 RTC 模块节点下添加 “wakeup-source” 属性，则可以设置为唤醒源。

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

表 2-2: 平台支持唤醒源列表

平台/唤醒源	非CPUS域GPIO	CPUS域GPIO	POWERKEY	RTC	USB	红外	蓝牙/WiFi	MAD
D1-H	super standby	不支持	不支持	super standby	super standby	不支持	super standby	不支持

2.3.1 kernel menuconfig 配置说明

进入配置主界面 (Linux-5.4 内核版本执行：./build.sh menuconfig)，并按以下步骤操作。

- 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
```

```
    Symbol: SUNXI_RISCV_SUSPEND [=y]
    -> Device Drivers
      -> SOC (System On Chip) specific Drivers
      <*> Allwinner sunxi riscv suspend support
```

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。

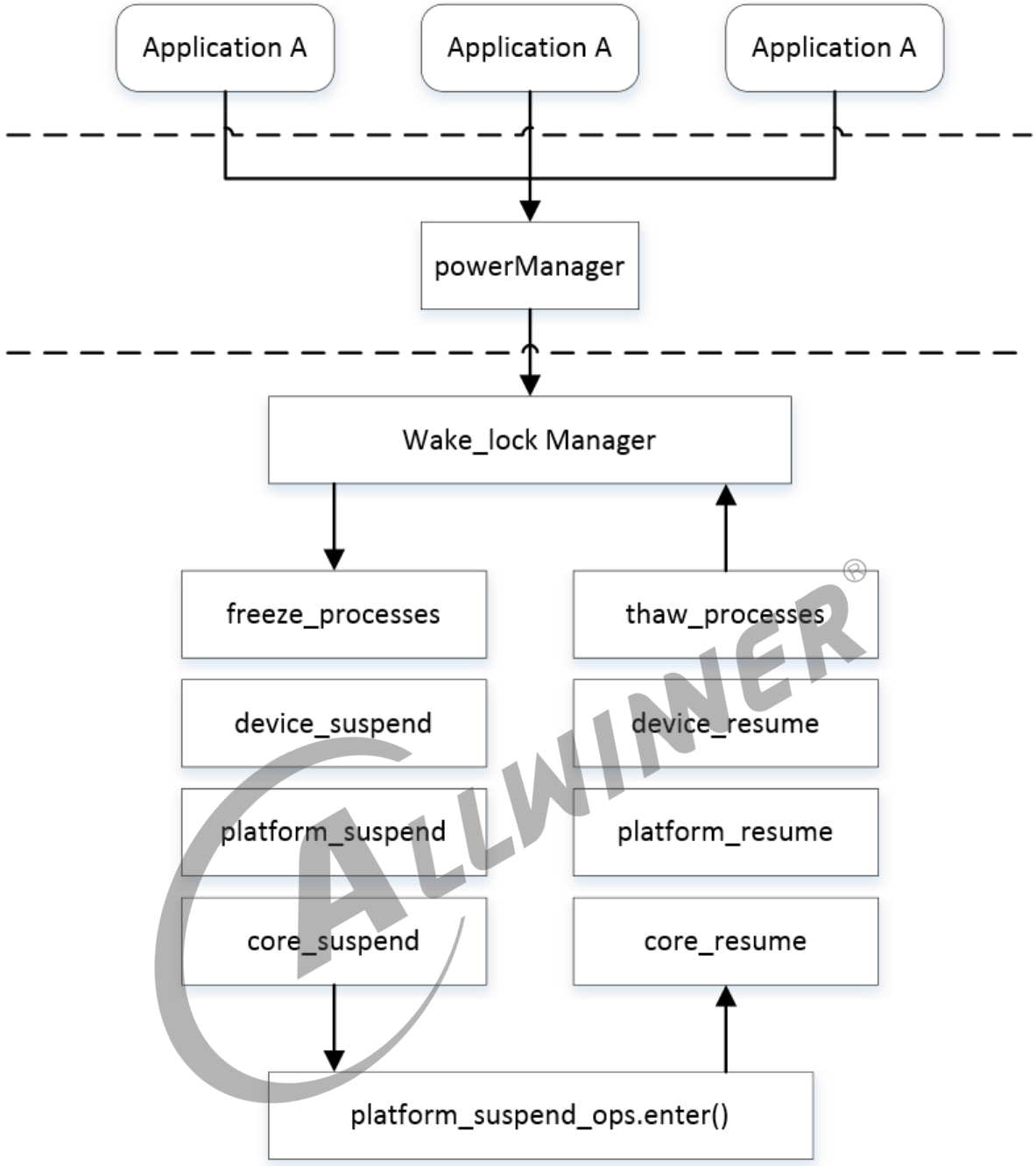

图 2-1：standby 驱动总体结构

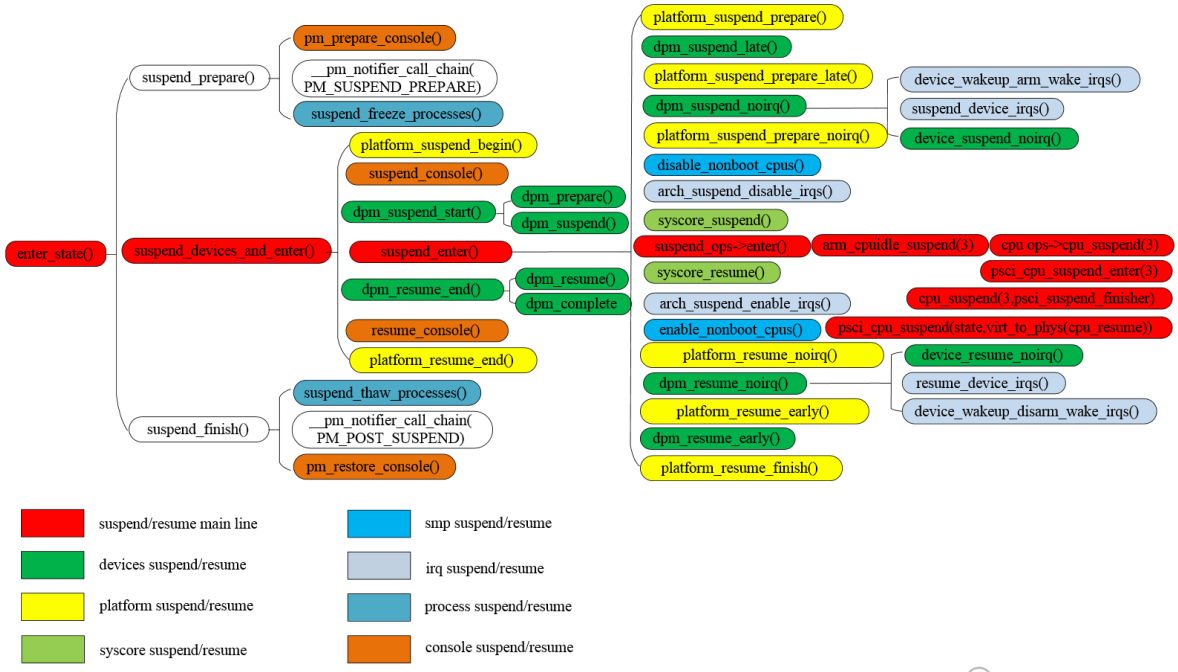

图 2-2: linux standby 流程

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。Eg：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；

- wakeup_sources 节点

该节点可查看系统唤醒源的情况。Eg：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：

```
[ 3466.063570] wake up source:0x80000
[21676.174594] [pm]platform wakeup, standby wakesource is:0x100000

后面的数字代表唤醒源，根据数字定位唤醒源，定位唤醒源后再判断为何被唤醒。
WAKEUP_SRC is as follow:
CPUS_WAKEUP_LOWBATT bit 0x1000
CPUS_WAKEUP_USB bit 0x2000
CPUS_WAKEUP_AC bit 0x4000
CPUS_WAKEUP_ASCEND bit 0x8000
CPUS_WAKEUP_DESCEND bit 0x10000
CPUS_WAKEUP_IR bit 0x80000
CPUS_WAKEUP_ALM0 bit 0x100000
CPUS_WAKEUP_HDMI_CEC bit 0x100000
```

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，当判断唤醒源为 0x100000 时，大多数为该原因导致。

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。

查看唤醒源。对应的代码路径在：lichee/linux4.9/include/linux/power/aw_pm.h

```
1  /* the wakeup source of assistant cpu: cpus */
2  #define CPUS_WAKEUP_HDMI_CEC (1<<11)
3  #define CPUS_WAKEUP_LOWBATT (1<<12)
4  #define CPUS_WAKEUP_USB (1<<13)
5  #define CPUS_WAKEUP_AC (1<<14)
6  #define CPUS_WAKEUP_ASCEND (1<<15)
7  #define CPUS_WAKEUP_DESCEND (1<<16)
8  #define CPUS_WAKEUP_SHORT_KEY (1<<17)
9  #define CPUS_WAKEUP_LONG_KEY (1<<18)
10 #define CPUS_WAKEUP_IR (1<<19)
11 #define CPUS_WAKEUP_ALM0 (1<<20)
```

```
#define CPUS_WAKEUP_ALM1 (1<<21)
#define CPUS_WAKEUP_TIMEOUT (1<<22)
#define CPUS_WAKEUP_GPIO (1<<23)
#define CPUS_WAKEUP_USBMOUSE (1<<24)
#define CPUS_WAKEUP_LRADC (1<<25)
#define CPUS_WAKEUP_WLAN (1<<26)
#define CPUS_WAKEUP_CODEC (1<<27)
#define CPUS_WAKEUP_BAT_TEMP (1<<28)
#define CPUS_WAKEUP_FULLBATT (1<<29)
#define CPUS_WAKEUP_HMIC (1<<30)
#define CPUS_WAKEUP_POWER_EXP (1<<31)
#define CPUS_WAKEUP_KEY (CPUS_WAKEUP_SHORT_KEY | CPUS_WAKEUP_LONG_KEY)
```

查看关键打印：

```
platform wakeup, standby wakesource is:0x800000
```

此时对应的是 gpio 唤醒。

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。

```
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 后的阶段出现异常。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 sys_config.fex 配置 usbc* 节点。需要注意 [usbc0] usb_wakeup_suspend = 1 //1 表示支持 USB0 唤醒，0 表示屏蔽该唤醒源。

3.2.2.5 hdmi_cec 不能唤醒系统

问题现象

HDMI 不能唤醒系统。

问题分析

HDMI 需要设置为唤醒源。

问题解决

需要在 sys_config.fex 配置 hdmi 节点。

```
[hdmi]
hdmi_cec_support = 1
hdmi_cec_super_standby = 1
```

3.2.2.6 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；

- Linux devices driver suspend 失败

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；如果是 androidN ，screen off timeout 还取决于 Sleep timeout，Sleep timeout 的值比 Screen off timeout 小时，系统取 Sleep timeout 当做定时休眠的时间，当 Sleep timeout 为-1 时，也表示系统无法定时休眠。如果是以上情况可咨询 android 系统的同事，修改相应的属性。

```
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
```

查看具体失败在哪个阶段

1. echo freezer > /sys/power/pm_test 查看 freezer 阶段是否 ok
2. echo devices > /sys/power/pm_test 查看 freezer/devices 阶段是否 ok
3. echo platform > /sys/power/pm_test 查看 freezer/devices/platform 阶段是否 ok
4. echo processors > /sys/power/pm_test 查看 freezer/devices/platform/processors 阶段是否 ok
5. echo core > /sys/power/pm_test 查看 freezer/devices/platform/processors/core 阶段是否 ok
6. 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

商标声明

免责声明