



Tina Linux 功耗管理 开发指南

版本号: 1.1
发布日期: 2021.01.03

版本历史

版本号	日期	制/修订人	内容描述
1.1	2021.01.03	AWA1610	适配新的文档模版



目 录

1 概述	1
1.1 编写目的	1
1.2 适用范围	1
1.3 适用人员	1
2 Tina 功耗管理概述	2
2.1 功能介绍	2
2.2 相关术语	2
3 Tina 休眠唤醒系统简介	4
3.1 Tina 休眠唤醒系统基础	4
3.1.1 Tina 休眠模式	4
3.1.2 唤醒源分类	6
3.1.3 休眠唤醒流程	7
3.2 Tina 休眠唤醒系统使用	9
3.2.1 休眠示例	9
3.2.2 唤醒示例	10
3.2.3 调试节点	10
3.3 Tina 休眠唤醒系统配置	13
3.3.1 使能休眠唤醒功能	13
3.3.1.1 Linux-4.9 平台	13
3.3.1.2 Linux-3.4 平台	14
3.3.2 使能常用唤醒源	15
3.3.2.1 Linux-4.9 平台	15
3.3.2.2 Linux-3.4 平台	17
3.4 常见问题及技巧	18
3.4.1 如何查看唤醒设备的唤醒源	18

插 图

2-1 功耗管理系统分类	2
3-1 Normal Standby 调用结构	5
3-2 Super Standby 调用结构	5
3-3 中断结构	6
3-4 休眠唤醒回调顺序	8
3-5 休眠唤醒配置	14
3-6 休眠唤醒配置	15



1 概述

1.1 编写目的

简要介绍 Tina 平台功耗管理机制，为关注功耗的开发者，维护者和测试者提供参考。

1.2 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	协处理器（功耗）
R328	Linux-4.9	无
R818	Linux-4.9	CPUS
MR813	Linux-4.9	CPUS
R329	Linux-4.9	DSP0

1.3 适用人员

Tina 平台下功耗管理相关的开发、维护及测试相关人员。

2 Tina 功耗管理概述

2.1 功能介绍

Tina 功耗管理系统主要由休眠唤醒（standby、autosleep、runtime pm），调频调压（cpufreq、devfreq、dvfs），开关核（hotplug），cpuidle 等子系统组成。主要用于对系统功耗进行管理和控制，平衡设备功耗和性能。

一般我们可将其分为两类，即静态功耗管理和动态功耗管理。

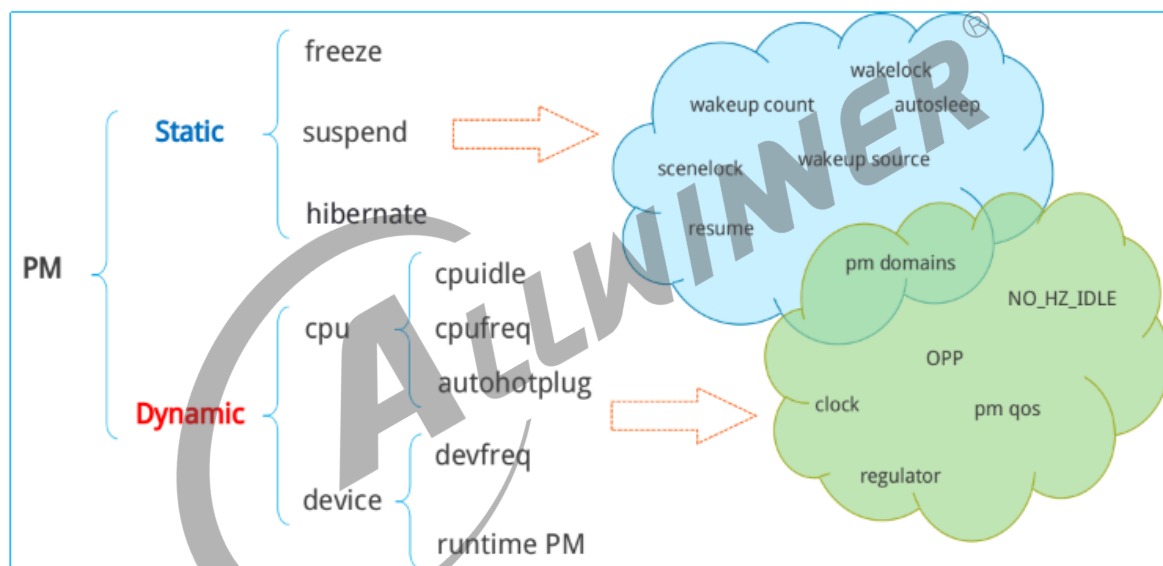


图 2-1: 功耗管理系统分类

一般地，可以动态调整或实时改变系统状态而达到节能目的技术，称为动态功耗管理；相对地，我们把单纯地将系统置为某一种状态，而不实时调整的低功耗技术，称为静态功耗管理。

2.2 相关术语

表 2-1: 术语表

术语	解释
CPUS	一种专用于低功耗的协处理器单元。
CPUX	主处理器单元，ARM CPU 核心。
WFI	ARM 体系中一种指令，可将 CPUX 置于低功耗状态，直到有中断发生。类似于 WFE，详细请参考 ARM 手册 DDI0487A_d_armv8_arm.pdf。
suspend	Linux 休眠框架，可以将系统置为低功耗休眠模式，并根据休眠程度不同，分为 freeze, standby, suspend 等状态。Tina 中仅使用 standby 一种。
Normal standby	Allwinner 内部术语：一种低功耗休眠状态，不能够关闭 CPUX，系统唤醒借助 ARM CPU 的 WFI 指令。
Super standby	Allwinner 内部术语：一种超低功耗休眠状态，可以关闭 CPUX 的电，系统唤醒借助 CPUS 实现，硬件上必须有 CPUS 或 PMU。
Arm Trusted Firmware	ARMv8-A 安全世界软件的一种实现，包含标准接口：PSCI、TBRR、SMCCC 等。
BMU	电池管理芯片。
PMU	电源管理芯片。

3 Tina 休眠唤醒系统简介

3.1 Tina 休眠唤醒系统基础

休眠唤醒框架主要降低设备在待机状态下的功耗，我们称为静态功耗管理技术。它通过在系统处于空闲状态时，关闭非必须模块，或降低模块电压和频率来实现设备的低功耗长时待机。

说明

由于很多设备并不总是处于工作状态，相反他们可能大部分时间处于待机状态，如手机，智能音响等，所以降低待机功耗，可以显著降低其平均功耗。

值得一提的是，降低功耗的前提是保证系统可以在使用场景下正常运行，因此能够关闭哪些模块，降低多少电压和频率，还依赖于系统的硬件配置，性能和使用场景。最终体现出来的休眠功耗，也取决于硬件和使用场景上的差异。

例如：

- 硬件差异：在支持 Super standby 的硬件方案中，休眠时就能够实现关闭 CPUX 的供电，从而达到很低的功耗。
- 场景差异：一些客户场景下，需要使用外设唤醒（如 uart, lradc），在这种场景下，休眠时就不能关闭该外设依赖的时钟和电源，从而导致休眠功耗比不使用该外设时高。

3.1.1 Tina 休眠模式

在 Tina 中，实现了两种主要低功耗休眠模式 Normal standby 和 Super standby，其主要区别在于，CPUX 是否掉电。

- Normal standby 模式

在 Normal standby 模式下，CPUX 完成 Linux 部分休眠流程后，通过 SMC 指令陷入安全世界继续执行平台休眠操作，包括切换到 SRAM 运行，使能唤醒中断，调整总线时钟，关闭部分模块电源，将 DRAM 置位自刷新模式，以及等待唤醒中断等等。

在整个休眠流程中，不论是安全或非安全操作，都是 CPUX 在执行（黄色部分），因此，无法将 CPUX 关闭。如下图所示：

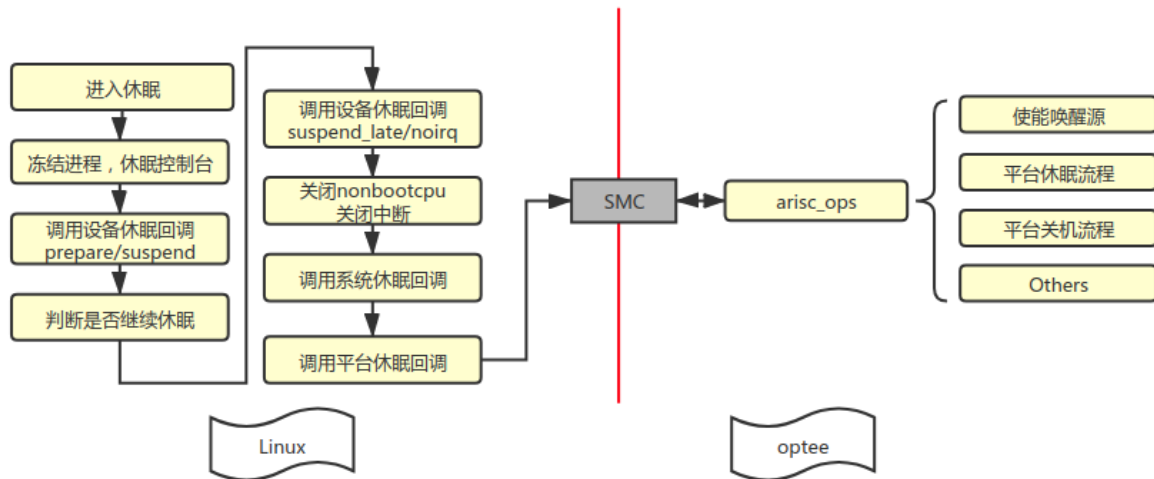


图 3-1: Normal Standby 调用结构

• Super standby 模式

在 Super standby 模式下，CPUX 完成 Linux 部分休眠流程后，通过 SMC 指令陷入安全世界，在安全世界中通过两个硬件模块（spin_lock, message_box）和 CPUS 通信，请求 CPUS 执行一次平台休眠流程。CPUS 接受到平台休眠请求后执行平台休眠流程，将 CPUX 关闭，使能唤醒中断，调整总线时钟，关闭部分模块电源，将 DRAM 置为自刷新模式，以及等待唤醒中断等等。

在整个休眠流程中，黄色部分为 CPUX 执行的部分，绿色部分为 CPUS 执行的部分，CPUX 关闭后休眠不受影响。如下图所示：

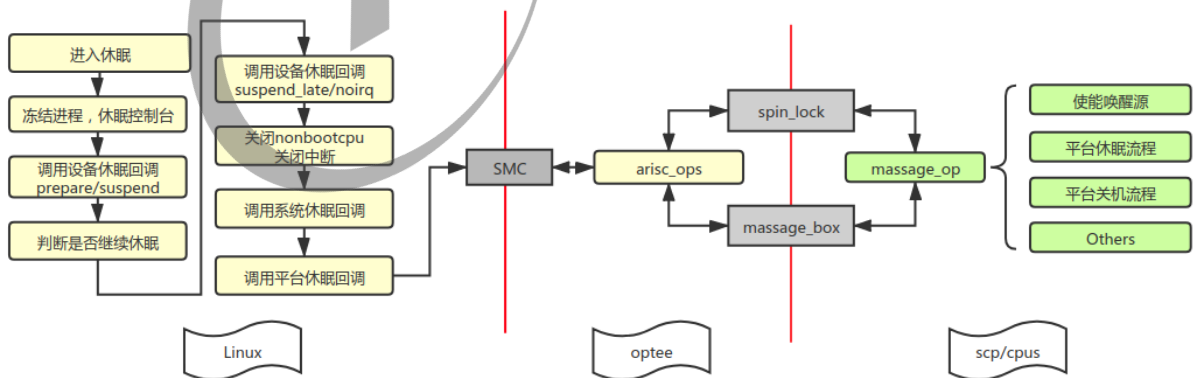


图 3-2: Super Standby 调用结构

⚠ 警告

Super standby 模式下，CPUX 已经关闭，只有 CPUS 上的中断才能够唤醒系统。

原则上，只要最终实现了关闭 CPUX 的电，我们便认为支持 Super standby 休眠模式。

例如：

- 在 R329 中，没有 CPUS，其绿色部分功能集成在 DSP0 的代码中，独立于 CPUX 运行，因此也可以实现 Super Standby。
- 在 R40 中，没有 CPUS，也没有 DSP，但也通过 AXP 延时掉电的方式实现了 CPUX 的掉电操作，因此也属于 Super standby。不过此种情况，属于特例，不作说明。

3.1.2 唤醒源分类

在 Tina 平台上，我们可以按照中断将唤醒源分为两大类，

- 1、内部唤醒源，一般为 IC 内部外设，有自己独立的中断号，如 rtc, uart, radc, usb 等。
- 2、外部唤醒源，这类设备都通过 GPIO 中断实现唤醒功能，如 wifi, bluetooth, gpiokey 等。

如下图，粉色为 irq_chip 【GPIO 也看做是一个 irq_chip】，蓝色为内部唤醒源，紫色为外部唤醒源。

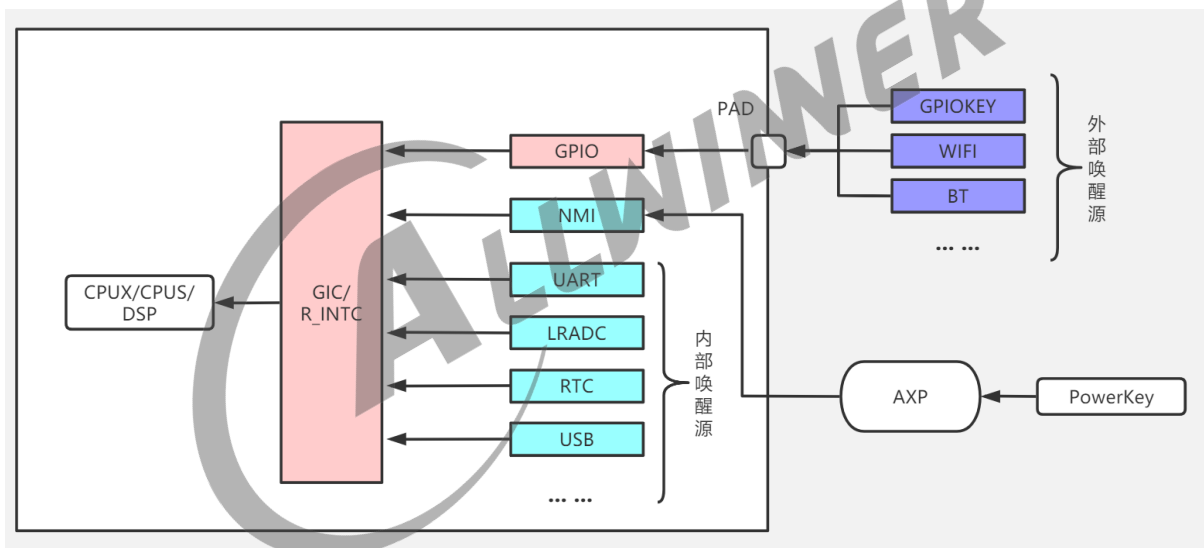


图 3-3: 中断结构

外部唤醒源不同于内部唤醒源，主要有以下不同：

- 1、外部唤醒源依赖于 GPIO 中断，而且 GPIO 中断通常是一个 GPIO Group 共用一个中断号，因此需要借助 irq_chip 框架进行虚拟中断映射。这一点 Tina 已经实现了对应的功能。
- 2、外部唤醒源使能唤醒功能时，还需保证 GPIO 复用功能，时钟，电源，上下拉状态等正常。
- 3、GPIO 中断分为 CPUX 上的 GPIO 和 CPUS 上的 GPIO，以及 PMU 上的 GPIO。不同模块上的 GPIO 在实现上会有一定的差异，但 Tina 基本已屏蔽了这些差异。
- 4、对于常见的内部唤醒源，如 rtc 定时，lradc 唤醒源，已支持通过 dts 配置使能。对于其他的内部唤醒源，由于其时钟、电源依赖关系可能需要更新安全固件或 CPUS 固件来解决，而这部分代码是不开源的，因此使能这部分唤醒源需要联系我们处理。

需要注意的是，不论哪种唤醒源，其正常工作都有以下几个前提：

- 1、休眠后该设备仍可以正常工作【其时钟，电源，工作状态配置正常】，指定事件发生后，可产生唤醒中断；
- 2、该中断处于使能状态，整个通路正常，CPUX 或 CPUS 可以正常接收到该中断。

3.1.3 休眠唤醒流程

休眠唤醒流程基本上都是由内核框架完成，各家厂商差异不大。具体差异在于设备，系统，平台注册的回调函数，各厂商可通过修改这些回调，来适配各个平台，实现差异化。

主要休眠流程：

- 1、冻结用户进程和线程；
- 2、休眠控制台，同步文件系统；
- 3、休眠设备，调用设备休眠回调（prepare,suspend,suspend_late,suspend_noirq），内核根据唤醒源配置使能和关闭中断；
- 4、关闭非引导CPU，关闭全局中断；
- 5、调用syscore休眠回调，休眠系统服务，如kernel time等；
- 6、调用平台休眠回调（suspend_ops->enter），进入最终的休眠状态。在此阶段可关闭不必要的时钟，电源，并进入等待唤醒模式。

唤醒流程：

- 1、检测到唤醒中断后开始平台唤醒，从平台休眠回调（suspend_ops->enter）中退出，并使能休眠时关闭的时钟，电源；
- 2、调用syscore唤醒回调，恢复系统服务；
- 3、使能全局中断，使能关闭的CPU；
- 4、恢复设备，调用设备唤醒回调（resume_noirq,resume_early,resume,complete），内核在此阶段还原中断配置；
- 5、恢复控制台；
- 6、恢复用户进程和线程，还原到休眠前的状态。

在整个休眠流程中，调用回调函数的顺序，如下图所示：

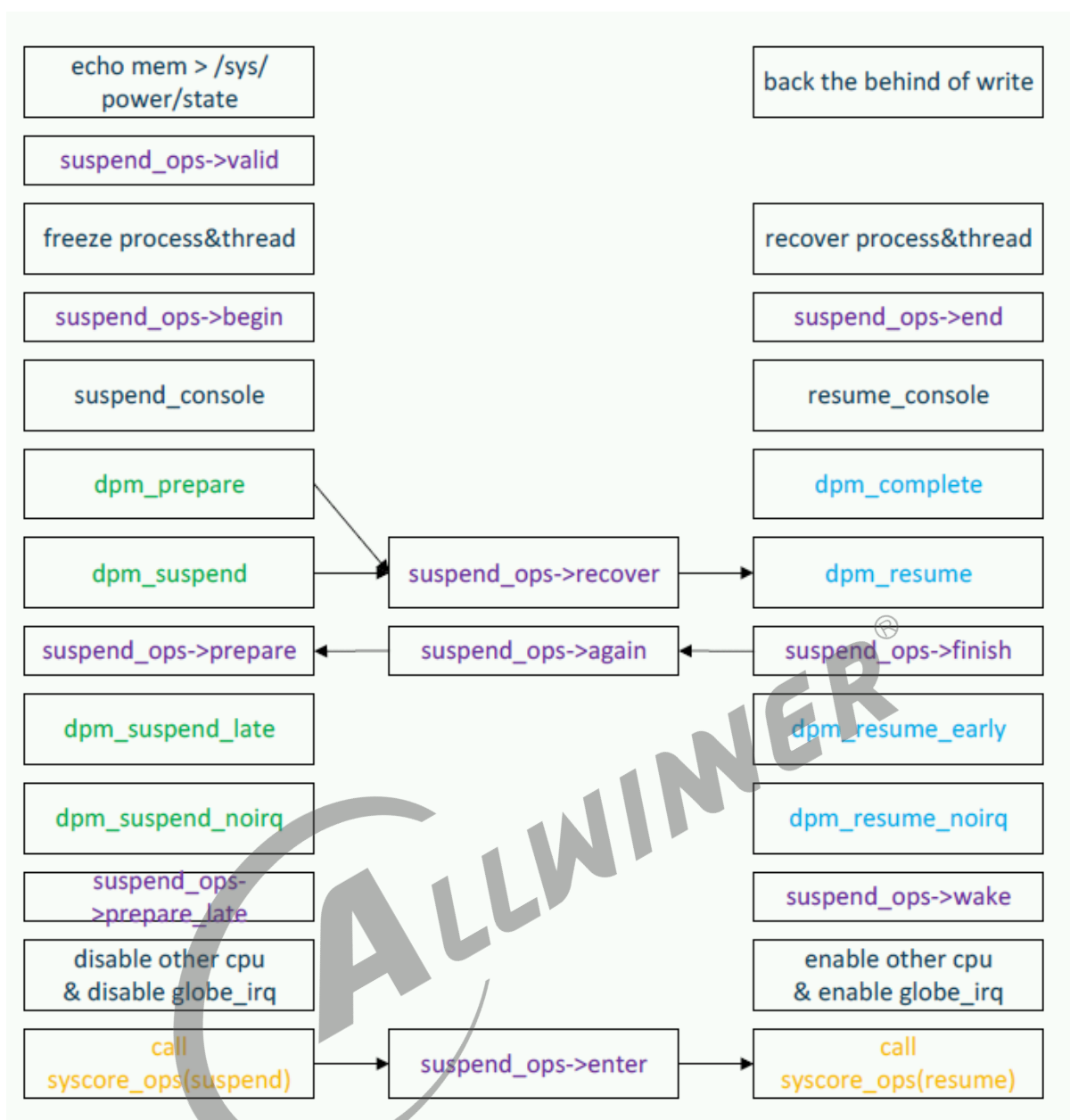


图 3-4: 休眠唤醒回调顺序

说明

在本文中，无特殊说明，有如下约定：

绿色和蓝色方框部分：称为设备休眠唤醒回调，由设备驱动注册；每个驱动可注册一份或留空不注册，调用时，为每个设备都调用一次。

橙黄色方框部分：称为系统休眠唤醒回调，由内核模块或驱动（如 *pinctrl*）注册；由平台厂商提供，可不了解。

紫色方框部分：称为平台休眠唤醒回调，由平台厂商实现并注册，实现平台休眠逻辑，必须实现 *valid* 和 *enter* 函数。

3.2 Tina 休眠唤醒系统使用

3.2.1 休眠示例

1、首先读出当前系统的 cnt

若读取时阻塞，说明系统存在 wakeup event 正在处理，此时不能休眠。

```
root@TinaLinux:/# cat /sys/power/wakeup_count
8
```

2、将读出的 cnt 写回 wakeup_count

需要判断返回值，若写入成功返回值为零，说明系统可以休眠。若写入失败，返回值不为零，说明在本次读写过程中产生了 wakeup event，应该重复步骤 1~2，直到写入成功。

```
root@TinaLinux:/# echo 8 > /sys/power/wakeup_count
```

3、尝试休眠

若休眠过程中未产生 wakeup event，系统成功休眠。若产生了 wakeup event，系统会终止休眠，回退到正常状态，应用程序可等待一段时间后，重复 1~3 步，再次尝试。

```
root@TinaLinux:/# echo mem > /sys/power/state
```

休眠脚本示例：

```
#!/bin/ash

function suspend()
{
    while true; do
        if [ -f '/sys/power/wakeup_count' ]; then
            cnt=$(cat /sys/power/wakeup_count)
            echo "Read wakeup_count: $cnt"
            echo $cnt > /sys/power/wakeup_count

            if [ $? -eq 0 ]; then
                echo mem > /sys/power/state
                break;
            else
                echo "Error: write wakeup_count($cnt)"
                sleep 1;
                continue;
            fi
        else
            echo "Error: File wakeup_count not exist"
            break;
        fi
    done
}
```

```
echo "try to mem..."
suspend
```

💡 技巧

休眠时不应连接 *usb*，在 *usb* 连接状态下，*usb driver* 会上报 *wake event*，导致读取 *wakeup_count* 阻塞。若出现执行阻塞的情况，拔掉 *USB* 即可。

⚠️ 警告

未通过 *wakeup_count* 节点判断系统当前状态是否可以休眠，而直接使用 `echo mem > /sys/power/state` 命令强制系统进入休眠的这种操作，是不规范的。

强制休眠操作会使休眠唤醒流程忽略对 *inpr* 和 *cnt* 变量检测，可能会导致一些同步问题。如休眠过程中，*WIFI* 唤醒中断不能导致休眠流程终止，而出现系统强制休眠，无法唤醒的异常。

3.2.2 唤醒示例

- 按键唤醒（LRADC, POWERKEY）

休眠后，短按对应的键，即可唤醒。

- WLAN 唤醒的使用方法

使用同一网络内的其他设备给休眠设备发 *ping* 包，即可唤醒。

- 闹钟唤醒的使用方法

设置 5 秒后闹钟唤醒（注意定时时间从执行此条命令时开始计算）。

```
/*R818/MR813*/
echo +5 > /sys/class/rtc/rtc0/wakealarm
```

3.2.3 调试节点

state

路径：/sys/power/state

Linux 标准节点，系统休眠状态配置节点。通过写入不同的状态级别（freeze, standby, mem）可使系统进入到不同级别的休眠状态。

Tina 一般仅支持 freeze, mem 两种休眠状态，standby 在 Tina 平台与 mem 等效。

freeze 状态为 Linux 系统自身支持的一种休眠状态，与平台无耦合，不调用到平台回调接口，无底层总线，时钟，电源控制，但会在调用设备休眠回调后进入 cpuidle 状态。

mem 状态为 Tina 支持的一种休眠状态，Normal standby 和 Super standby 都是这种状态的一种表现。

```
# 强制进入休眠 @ 这种方式仅用于调试，系统休眠请参考上文： 休眠示例一节
root@TinaLinux:/# echo mem > /sys/power/state
```

wakeup_count

路径：/sys/power/wakeup_count

Linux 标准节点，将 wakeup count 模块维护的计数开放到用户空间，为应用程序提供一个判断系统是否可以休眠的接口。

具体使用参考上文：休眠示例一节。

wake_[un]lock

路径：/sys/power/wake_lock、/sys/power/wake_unlock

Linux 标准节点，wake lock 模块开放到用户空间的接口。

应用程序可以通过 wake_lock 节点申请一个 wakelock，并通过 wake_unlock 节点释放对应的 wakelock，任一应用程序持有 wakelock，系统都不休眠。

```
# 申请一个NativePower.Display.lock
root@TinaLinux:/# echo NativePower.Display.lock > /sys/power/wake_lock
# 可以查看有系统中存在哪些wakelock
root@TinaLinux:/# cat /sys/power/wake_lock
NativePower.Display.lock

# 释放 NativePower.Display.lock
root@TinaLinux:/# echo NativePower.Display.lock > /sys/power/wake_unlock
# 可以查看那些wakelock被释放
root@TinaLinux:/# cat /sys/power/wake_unlock
NativePower.Display.lock
```

console_suspend

路径：/sys/module/printk/parameters/console_suspend

Linux 标准节点，该节点标记在系统进入休眠时，是否休眠控制台。

这个节点默认值为 Y，即默认会休眠控制台。

将其设置为 N 后，系统休眠时将不休眠控制台，这样可以将休眠后期（控制台休眠阶段后）的日志实时打印到控制台，便于调试。

```
# 禁用控制台休眠
root@TinaLinux:/# echo N > /sys/module/printk/parameters/console_suspend
```

ignore_loglevel

路径：/sys/module/printk/parameters/ignore_loglevel

Linux 标准节点，忽略打印级别控制。

这个节点默认值为 N，即不忽略打印级别，仅输出可打印级别的日志。可打印级别由/proc/sys/kernel/printk 节点控制。

将其设置为 Y 后，任何级别的系统日志都可以输出到控制台。这不仅仅在休眠唤醒过程中有效，在系统正常工作时也有效。

```
# 忽略系统日志打印级别
root@TinaLinux:/# echo Y > /sys/module/printk/parameters/ignore_loglevel
```

initcall_debug

路径：/sys/module/kernel/parameters/initcall_debug

Linux 标准节点，该节点标记是否开启内核早期日志，在内核启动早期先初始化控制台，输出内核启动早期日志信息。在休眠唤醒流程中，会影响到唤醒早期部分日志的打印。

该节点默认值由内核参数确定，一般为 N，即不使能早期打印。

将其设置为 Y 后，在 Linu-3.4 中休眠唤醒中，会多打印 device 休眠唤醒调用信息和 syscore_ops 调用信息。但在 Linux-4.9 中，仅影响 syscore_ops 调用信息。因为在 Linux-4.9 中，device 休眠调用信息由 pm_print_times 节点控制。

使能该节点后，会休眠唤醒过程中打印各个设备休眠唤醒回调的调用顺序及返回值，通过这些打印信息，可以判断出是哪个设备休眠唤醒回调出了问题，方便调试。

```
# 使能早期打印
root@TinaLinux:/# echo Y > /sys/module/kernel/parameters/initcall_debug
```

pm_print_times

路径：/sys/power/pm_print_times

Linux 标准节点，在 Linux-3.4 中不支持该节点，其功能合并到 initcall_debug 节点中。该节点标志是否在休眠唤醒流程中，打印 device 休眠唤醒调用信息。

该节点默认值为 0，即不打印设备调用信息。

```
# 使能设备回调信息输出
root@TinaLinux:/# echo 1 > /sys/power/pm_print_times
```

pm_wakeup_irq

路径：/sys/power/pm_wakeup_irq

Linux 标准节点，只读。用于查看上一次唤醒系统的唤醒中断号。

说明

该节点对于 *cpus* 上 *gpio* 口的唤醒中断号无法正常显示。这个是由于 *pinctrl* 驱动中，为 *cpus gpio* 设置了 *IRQF_NO_SUSPEND* 标志，由于影响模块较多，无法处理。

```
# 使能设备回调信息输出
root@TinaLinux:/# cat /sys/power/pm_wakeup_irq
```

pm_test

路径：/sys/power/pm_test

Linux 标准节点。由内核实现的一种休眠唤醒调试机制。

读该节点会打印其支持的调试点，如下：

```
# linux 默认支持的调试点
root@TinaLinux:/# cat /sys/power/pm_test
[none] core processors platform devices freezer
```

对该节点写入其支持的调试点，会在休眠过程中，执行到该调试点时，等待几秒后返回。

```
root@TinaLinux:/# echo core > /sys/power/pm_test
```

说明

Freezer: 任务冻结后，等待 5s，即返回；
Devices: 执行设备回调 *prepare*, *suspend* 后，等待 5s，即返回；
Platform: 执行设备回调 *suspend_late*、*suspend_noirq* 后，等待 5s，即返回；
Processors: 关闭非引导 *cpu* 后，等待 5s，即返回；
Core: 冻结系统服务，如内核时间服务后，等待 5s，即返回；
None: 整个休眠流程全部走完，需触发唤醒源唤醒；

3.3 Tina 休眠唤醒系统配置

3.3.1 使能休眠唤醒功能

3.3.1.1 Linux-4.9 平台

在 Tina 源码根目录，执行 `make kernel_menuconfig`，进入内核配置菜单。

如下图所示，进入 Power management options 配置项：

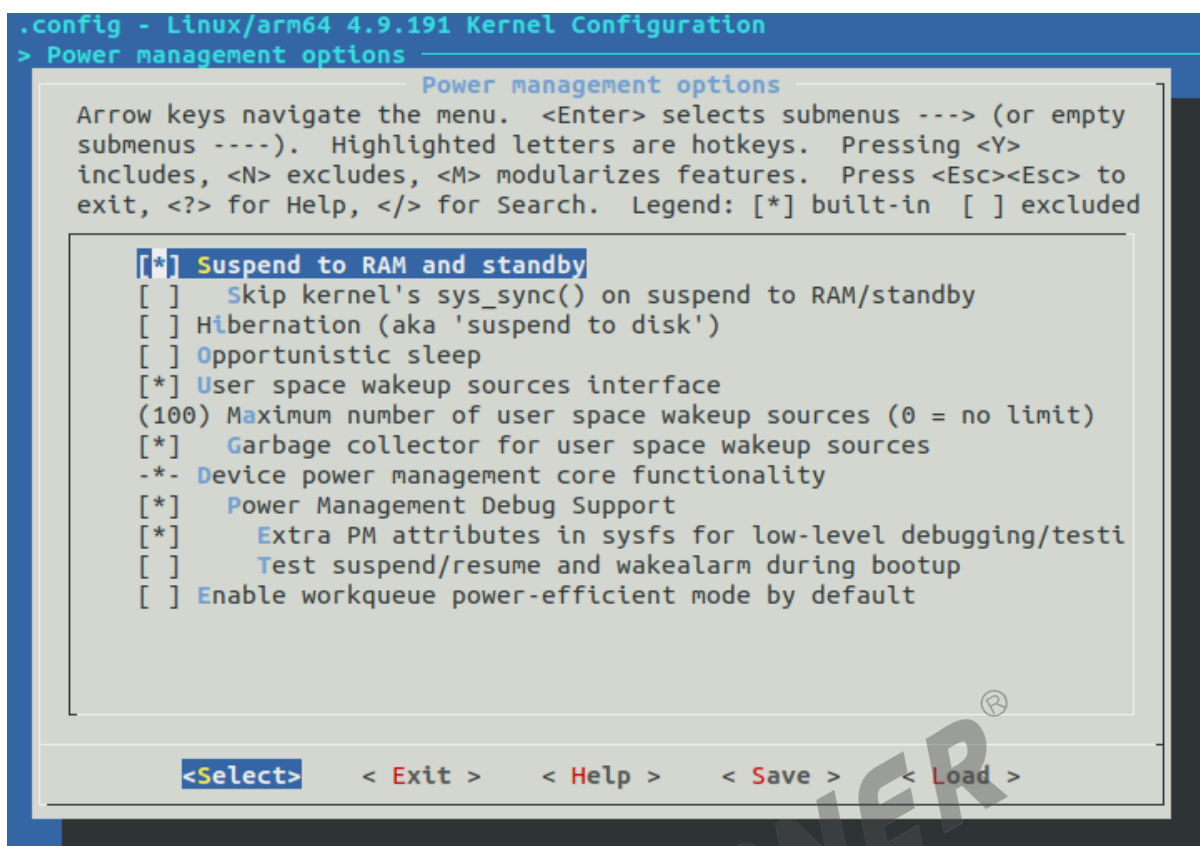


图 3-5: 休眠唤醒配置

选中以下配置项：

- [*] Suspend to RAM and standby //使能休眠唤醒框架，默认选中
- [*] Power Management Debug Support //使能休眠唤醒调节点，默认选中

3.3.1.2 Linux-3.4 平台

在 Tina 源码根目录，执行 make kernel_menuconfig，进入内核配置菜单。

如下图所示，进入 Power management options 配置项：

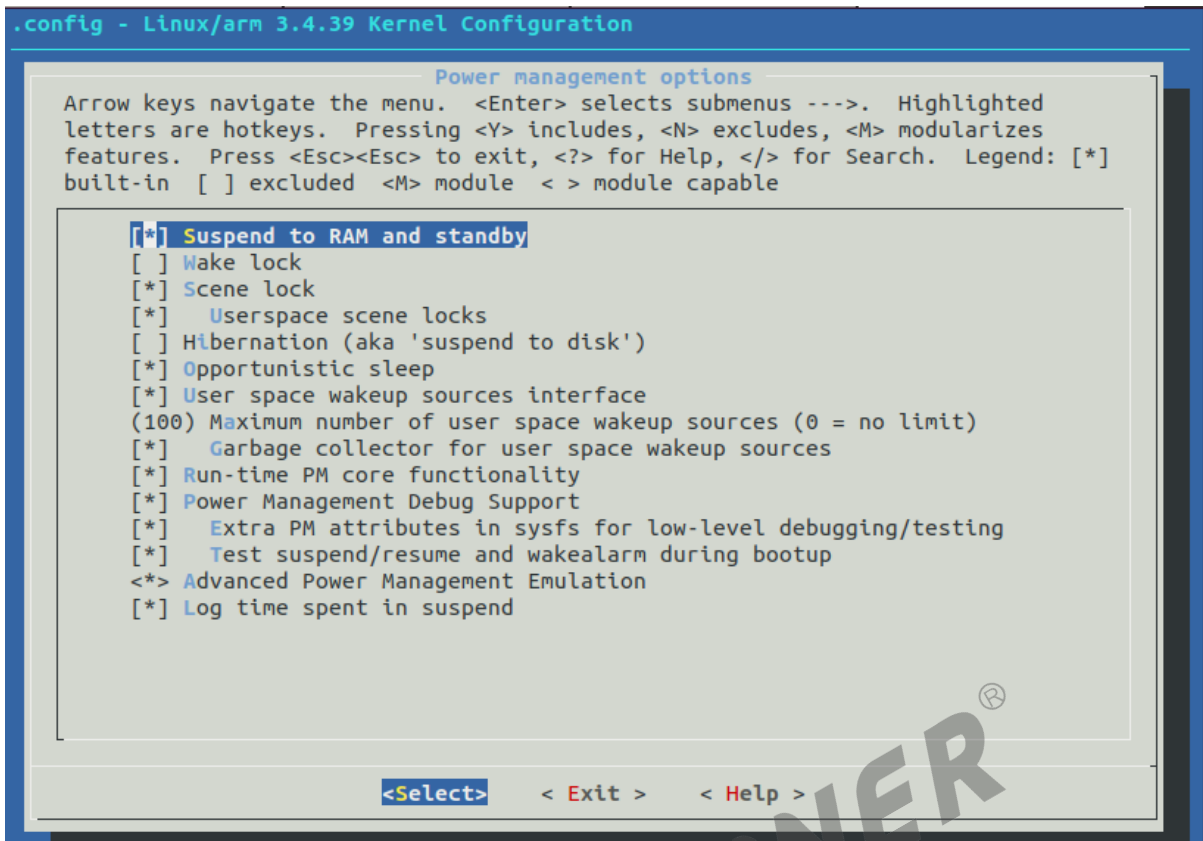


图 3-6: 休眠唤醒配置

选中以下配置项：

- [*] Suspend to RAM and standby //使能休眠唤醒框架，默认选中
- [*] Scene lock //Tina实现的一种休眠场景选择机制，默认选中
- [*] Userspace scene locks
- [*] Power Management Debug Support //使能休眠唤醒调节点，默认选中

3.3.2 使能常用唤醒源

📖 说明

理论上，任何中断都可配置为唤醒源，如下列配置不能满足您的需求或您有特殊需求，请联系我们。

3.3.2.1 Linux-4.9 平台

修改 dts 配置，在对应的设备节点中，增加 wakeup-source 属性即可。

📖 说明

这种配置方式依赖设备驱动支持，详见 **Linux-4.9 DTS 唤醒源支持表**。

平台 dts 配置文件路径：

TinaTop/lichee/linux-4.9/arch/arm/boot/dts/{CHIP}.dtsi
或者 TinaTop/lichee/linux-4.9/arch/arm64/boot/dts/sunxi/{CHIP}.dtsi

方案 dts 配置文件路径：

TinaTop/device/config/chips/{IC}/configs/{BOARD}/board.dts

示例：（配置支持 RTC 闹钟唤醒）

```
rtc: rtc@07090000 {
    compatible = "allwinner,sunxi-rtc";
    device_type = "rtc";
    reg = <0x0 0x07090000 0x0 0x400>;
    interrupts = <GIC_SPI 106 IRQ_TYPE_LEVEL_HIGH>;
    gpr_offset = <0x100>;
    gpr_len = <8>;
    gpr_cur_pos = <6>;
    clocks = <&clk_rtc>;
    + wakeup-source;
};
```

支持 DTS 配置的平台及唤醒源：

表 3-1: Linux-4.9 DTS 唤醒源支持表

	PowerKey	LRADC	RTC 闹钟	WIFI	BT
R818	支持	不支持	支持	支持	支持 (XR829)
MR813	支持	不支持	支持	支持	支持 (XR829)
R329	支持 (BMU2585)	支持	支持	支持	支持 (XR829)
R328	无	支持	无	支持	-

其他说明

- R328 支持 MAD 能量唤醒

使用方法可参考《Tina_Linux_音频_开发指南.pdf》。

- R328 支持定时器唤醒（默认生效）

设置定时 5s 后唤醒：

```
echo 5000 > /sys/module/pm/parameters/time_to_wakeup_ms
```

警告

该节点写入值小于 1000ms 无效，写入值为 0 时，关闭定时唤醒功能。

- R329 支持 MAD 能量唤醒

需要配合应用程序使用，使用方法可参考《Tina_Linux_音频_开发指南.pdf》。

说明

在 R329 平台设计中，按照语音唤醒的三个执行阶段，定义了多级唤醒模型：

一级唤醒检测：CPUX 关闭、ddr 进自刷新、wifi/bt 关闭、dsp 休眠/关闭，vad 模块检测能量输入；

二级唤醒检测：CPUX 关闭、ddr 进自刷新、wifi/bt 关闭、dsp 语音小模型工作，监听唤醒词输入；

三级唤醒检测：CPUX/ddr 运行在量产频率，wifi/bt 使能，dsp 运行大语音算法模型。

3.3.2.2 Linux-3.4 平台

在 Linux-3.4 中，通过 sysconfig.fex 提供一个配置外部唤醒源的属性节点 wakeup_src_para，该节点将会设置相应事件的唤醒源标记，并在系统休眠过程中配置这些事件为唤醒源。

sysconfig.fex 路径：

```
TinaTop/device/config/chips/{IC}/configs/{BOARD}/sys_config.fex
```

示例：（配置 WIFI、BT 的唤醒引脚）

```
-----  
; wakeup_src_para:  
; sometimes, u would like to add more wakeup src in standby mode,  
; these para will be help;  
; u need to make sure the standby mode support the wakeup src.  
; Also, some hw condition must be guaranteed.  
; including:  
; cpu_en: power on or off.  
; 1: mean power on  
; 0: mean power off  
; cpu_freq: indicating lowest freq. unit is Mhz;  
; dram selffresh_en: selffresh or not.  
; 1: enable enter selffresh  
; 0: disable enter selffresh  
; dram_pll: if not enter selffresh, indicating lowest freq. unit is Mhz;  
; wakeup_src: to make the scenario work, the wakeup src is needed.  
-----  
[wakeup_src_para]  
cpu_en      = 0  
cpu_freq    = 48  
; (cpu:apb:ahb)  
pll_ratio   = 0x111  
dram_selffresh_en= 1  
dram_freq   = 36  
wakeup_src_wl   = port:PL07<4><default><default><0>  
wakeup_src_bt   = port:PL09<4><default><default><0>  
bb_wake_ap      = port:PL02<4><default><default><0>
```

说明

需要注意的是，由于软件设计问题，该方法配置外部唤醒源时，仅支持 PL/PM 两组 GPIO。

3.4 常见问题及技巧

3.4.1 如何查看唤醒设备的唤醒源

在 R818/MR813/R329 中, 查看唤醒源通过 `cat /sys/power/pm_wakeup_irq` 节点, 读出的值为唤醒源中断号。

示例:

```
root@TinaLinux:/# cat /sys/power/pm_wakeup_irq
390
```

然后, 通过 `cat /proc/interrupts` 可查看 390 号中断对应的是 nmi 中断, 可能是 PowerKey 唤醒或电源异常唤醒。

```
root@TinaLinux:/# cat /proc/interrupts
              CPU0           CPU1           CPU2           CPU3
/*省略部份信息*/
383:             0             0             0             0   wakeupgen 22 Edge   sunxikbd
390:              2             0             0             0   sunxi-8i-nmi  0 Level   axp806
404:             0             0             1             0   axp806 13 Edge   axp20x-pek-dbf
```

在其他平台中, 可查看唤醒时日志来确定唤醒事件代码, 然后通过内核代码中, 对唤醒事件的定义宏确认唤醒源:

```
/* the wakeup source of assistant cpu: cpus */
#define CPUS_WAKEUP_HDMI_CEC (1<<11)
#define CPUS_WAKEUP_LOWBATT (1<<12)
#define CPUS_WAKEUP_USB (1<<13)
#define CPUS_WAKEUP_AC (1<<14)
#define CPUS_WAKEUP_ASCEND (1<<15)
#define CPUS_WAKEUP_DESCEND (1<<16)
#define CPUS_WAKEUP_SHORT_KEY (1<<17)
#define CPUS_WAKEUP_LONG_KEY (1<<18)
#define CPUS_WAKEUP_IR (1<<19)
#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)
```

示例: 对应的是 GPIO 中断唤醒, 可能是 WIFI, BT 或 GPIO 按键。

```
platform wakeup, standby wakesource is:0x8000
```

著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

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

、 全志科技、（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。