

H3 项目

standby 使用说明书 V1.0

文档履历

版本号	日期	制/修订人	内容描述
V1.0	2015-01-14		正式版本

confidential

目 录

1. 概述	4
1.1. 编写目的	4
1.2. 适用范围	4
1.3. 相关人员	4
2. 模块介绍	5
2.1. 模块功能介绍	5
2.2. 模块配置介绍	5
2.2.1. Menuconfig 配置	5
2.2.2. 开启 super standby 支持	6
2.2.3. 唤醒源配置	6
2.2.4. Extended_standby 下对唤醒源的配置	6
2.2.5. 调试打印信息配置	7
2.3. 内核对 standby 支持	7
2.4. 普通模块对 super standby 的支持	7
2.5. late_resume 部分的特别处理	8
2.6. 源码结构介绍	8
3. 接口描述	9
3.1. 用户与内核交互接口	9
3.1.1. Linux	9
3.1.2. Android	9
Declaration	10

1. 概述

1. 1. 编写目的

该文档的目的在于，简要介绍，H3 sdk 所支持的 standby 行为，以及为了支持 standby，各驱动模块所需工作及其注意的事项。

1. 2. 适用范围

硬件平台：H3

软件平台：H3 homlet44-dev sdk v1.0 及其后续版本。

1. 3. 相关人员

H3 平台下的 bsp 开发/维护人员。

Confidential

2. 模块介绍

2.1. 模块功能介绍

对于嵌入式设备尤其是移动设备来说，功耗是系统的重要指标，系统设计的重要目标之一就是要尽可能地降低功耗。

standby 模块，在特定策略控制下，通过与 CPU，内存、显示屏，gpu 等相协调，支持对一系列电压和频率的快速调节，从而降低嵌入式系统的功耗。

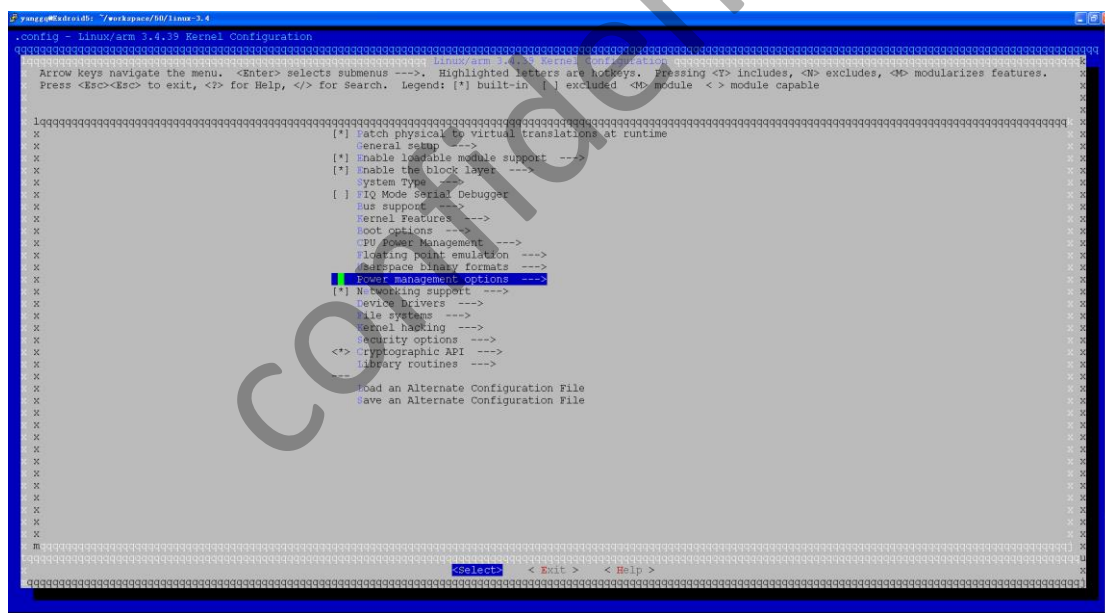
standby 模块，具有以下特点：

1. 支持 super standby + extended_standby;
2. 支持 standby 模式下：唤醒源的增加；
3. 在 extended_standby 的支持下，支持在特定场景下，进入 standby 的可能（如：usb_standby, normal standby, misc_standby）。

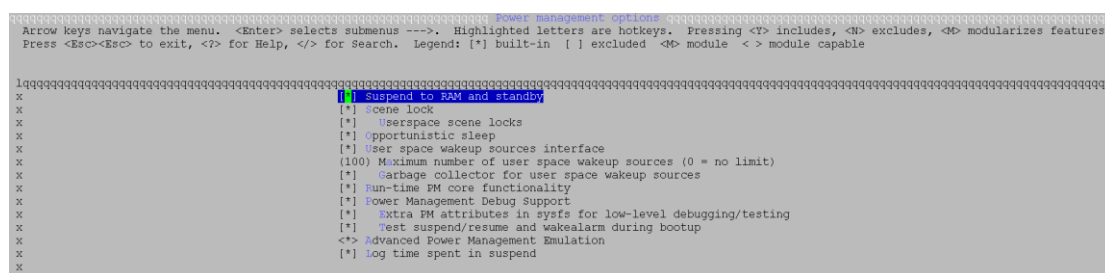
2.2. 模块配置介绍

2.2.1. Menuconfig 配置

1. 运行 make ARCH=arm menuconfig，见下图界面；



2. 选择 Power Management options，按下图所示配置即可：



2.2.2. 开启 super standby 支持

旧平台通过 standby_mode 的值区分 normal standby 和 super standby，但是在 H3 平台上，对其进行了统一，不再区分 normal / super。此处只保留作为兼容，**必须配置为 1**。

```
[pm_para]
standby_mode      = 1
```

2.2.3. 唤醒源配置

H3 支持 gpio 唤醒源，休眠状态下，指定 gpio 产生电平变化是，可以触发系统进入唤醒流程，唤醒源的配置如下：

```
-----
;
; 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.
;
;-----
[wakeup_src_para]
wakeup_src_wl      = port:PL04<6><default><default><0>
wakeup_src_bt      = port:PL03<6><default><default><0>
```

注：

1. wakeup_src_para 是对 standby 状态下所需要支持的唤醒源的描述；
2. 默认情况下，对于唤醒源为 PL, PM, AXP_PORT 的情况，以上配置是可直接使用；若要对 cpu、dram 的配置有特殊需求，请咨询相关人员后再做修改；
3. 本配置中，对于 wakeup_src_xx 这一子键，命名上没有限制，可根据需要配置为脚本解析系统支持的任意名字；
4. 本配置中，关心这一唤醒源的模块，需自行处理该唤醒源对应的 io 口的配置（触发类型：上升沿/下降沿/高电平/低电平）；pm 管理模块只是在进入 standby 时，对配置的 pin 进行 enable 中断，无其他多余操作。

2.2.4. Extended_standby 下对唤醒源的配置

standby 总是和唤醒源联系在一起的，唤醒源描述的是使得系统退出 standby 状态的条件。要支持某些唤醒源，总是与 standby 状态下，器件的工作状态联系在一起的。

extended_standby 将参与动态运行时，各种场景下的功耗管理。在这些场景下的供电依赖关系与唤醒源之间需要保持一致性，才能满足系统的工作需求。

Extended_standby 的配置，是支持动态更改的，因而，对其唤醒源的配置，也是支持动态更改的。因而，extended_standby 的配置包含两个方面：

1. 供电依赖关系（能够支持对唤醒源的检测）；
2. 唤醒源；

2.2.5. 调试打印信息配置

通过以下命令：`echo 0xff > /sys/module/pm_tmp/parameters/debug_mask`

更改 `debug_mask` 的配置，可控制 `standby` 的打印。

每一个 `bit` 控制不同内容的打印，如下：

```
typedef enum{
    PM_STANDBY_PRINT_STANDBY           = (1U << 0),
    PM_STANDBY_PRINT_RESUME             = (1U << 1),
    PM_STANDBY_PRINT_IO_STATUS          = (1U << 2),
    PM_STANDBY_PRINT_CACHE_TLB_MISS     = (1U << 3),
    PM_STANDBY_PRINT_CCU_STATUS         = (1U << 4),
    PM_STANDBY_PRINT_CPUS_IO_STATUS     = (1U << 5),
    PM_STANDBY_PRINT_RESUME_IO_STATUS   = (1U << 6),
    PM_STANDBY_ENABLE_JTAG              = (1U << 7),
    PM_STANDBY_PRINT_CCI400_REG         = (1U << 8),
    PM_STANDBY_PRINT_GTBUS_REG          = (1U << 9),
    PM_STANDBY_TEST                     = (1U << 10)
}debug_mask_flag;
```

2.3. 内核对 standby 支持

内核导出了两个符号：`standby_type`, `standby_level`, 同时提供了一个场景接口，以利于各模块实现对 `standby` 的支持：

根据目标：区分 `normal standby` 和 `super standby`。

根据掉电情况：区别对待设备；

根据场景：区分系统当前处在的场景，关心该场景的模块，需要对其进行特殊处理；

变量介绍：

`standby_type`: 表目标，支持 `NORMAL_STANDBY`, `SUPER_STANDBY`;

`standby_level`: 表结果，支持 `STANDBY_WITH_POWER_OFF`, `STANDBY_WITH_POWER`;

`check_scene_locked`: 获取系统当前处在的场景，`talking`, `usb`, `bootfast`, or etc...

2.4. 普通模块对 super standby 的支持

各模块需要实现 `suspend+resume` 接口，以支持超级 `standby`，确保系统唤醒后，能正常稳定的工作；流程如下：

1. 包含头文件 `linux/pm.h`;
2. 判断 `standby` 类型,并进行相应处理;
3. 关心场景的模块，还需判断场景，进行处理。

建议：

`super standby` 和 `normal standby` 使用相同的代码，不对 `normal standby` 进行特殊处理，可减少代码的维护量，且对系统性能影响不大。

示例：

```

if (check_scene_locked(SCENE_XXXX_STANDBY) == 0) {
    // process for the scene you care.
} else {
    // process for super standby and normal standby.
}

```

2.5. late_resume 部分的特别处理

early_suspend + late_resume 部分的特别处理：

对于处于 early_suspend 部分的模块，在唤醒时，模块可能处于两种状态：

1. 系统并没有进入 super standby 状态，就因为某种原因唤醒了；
2. 系统进入了 super standby 状态，由用户或其他信号唤醒。

取决于系统是否进入了 super standby 状态，模块在唤醒时就有两种可能的状态：带电或掉电；为了让模块清楚的知道，模块唤醒时，系统是否真正进入 super standby，从而影响了模块的带电状态，内核导出了另一个全局变量：standby_level；若模块认为有需要对带电和不带电两种状态进行区分处理，从而加速唤醒过程，可借助此变量的帮助。

示例：

```

if(NORMAL_STANDBY== standby_type){
    //process for normal standby
}else if(SUPER_STANDBY == standby_type){
    //process for super standby
    if(STANDBY_WITH_POWER_OFF == standby_level){
        //处理模块经过掉电后的恢复
    }else if(STANDBY_WITH_POWER == standby_level){
        //处理模块带电状态的恢复
    }
}
}

```

2.6. 源码结构介绍

■ 内核提供的公用代码：

```

kernel_src_dir\kernel\power\
kernel_src_dir\drivers\base\power\
kernel_src_dir\drivers\base\

```

■ 平台相关代码：

```

kernel_src_dir\arch\arm\mach-sunxi\power;

```


3. 接口描述

3.1. 用户与内核交互接口

通过 sys 文件系统的节点：/sys/power/state 与上层应用交互；

3.1.1. Linux

通过 echo mem > /sys/power/state，控制系统进入 super standby 状态；

通过 echo standby > /sys/power/state，控制系统进入 normal standby 状态；

3.1.2. Android

1. 按 power 键，执行 early_suspend 部分；
2. 若无用户申请 wake_lock，则进入 kernel 的 suspend 流程；
3. 平台实现的 standby 处理，直至掉电。

Confidential

Declaration

This document is the original work and copyrighted property of Allwinner Technology (“Allwinner”). Reproduction in whole or in part must obtain the written approval of Allwinner and give clear acknowledgement to the copyright owner.

The information furnished by Allwinner is believed to be accurate and reliable. Allwinner reserves the right to make changes in circuit design and/or specifications at any time without notice. Allwinner does not assume any responsibility and liability for its use. Nor for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Allwinner. This datasheet neither states nor implies warranty of any kind, including fitness for any particular application.

Confidential