# 内核开发实战

问题记录‌1：电源管理函数参数误配置

‌**引言**

在进行存储设备（例如UFS）的驱动程序开发时，不可避免地需要进行其交互接口的开发与调试工作。在此过程中，一个核心且至关重要的概念便是电源管理函数（Power Management Functions）的掌握。若在着手驱动开发之前，未能充分理解主机端与设备端在休眠与唤醒机制上的区别，以及未能准确把握电源管理函数的恰当应用方式，那么程序的正确性将无法得到保证，甚至可能引发严重的稳定性问题或功耗异常。

开发者必须深入理解以下关键概念：

* 主机控制器(Host Controller)与存储设备(Device)之间的运行时电源状态机(Runtime PM State Machine)差异
* 电源域(Power Domain)的层级管理机制
* 引用计数(Reference Counting)在pm\_runtime\_get\_sync/pm\_runtime\_put\_sync等核心API中的实现原理

缺乏这些基础认知将导致：

* 违反电源管理状态转换协议(PM State Transition Protocol)
* 引发引用计数失衡(Reference Count Imbalance)
* 造成系统级的不稳定现象(System Instability)
* 产生非预期的功耗峰值(Power Spike)

请注意，电源管理在Linux内核中是一个复杂且关键的领域，它涉及到系统架构、硬件特性以及驱动程序设计的多个方面。由于篇幅有限，本章主要介绍一个由pm\_runtime API误用导致UFS控制器挂起(Hang) 进而无法响应主机侧控制器命令的实际案例（**笔者在真实开发中遇到并解决的一个问题**），来初步阐述 Linux内核中电源管理函数的正确用法。

1. 稳定性问题（-110）

同事在MTK平台上做UFS存储设备的Linux内核驱动开发时，告诉我内核与驱动相关的日志概率遇到错误码为-110的错误打印，并引入panic。具体报错截图不便展示，形如：

ufshci: [name:ufshci\_core]\_\_ufshci\_query\_xxx: \*\*\*\*, err = -110

1. 问题分析思路

错误码-110（对应宏定义ETIMEDOUT）表示‌**操作超时**‌，即系统在预设时间内未收到硬件或协议的预期响应。在UFS驱动中，该错误通常发生于：

* 设备初始化阶段（如UFS控制器等待设备响应超时）。
* 数据传输过程中（例如UTP层命令执行未完成）。
* 硬件握手失败（如M-PHY物理层信号同步延迟）。

发生该错误通常要么硬件层有缺陷（例如**乱调电压**）或者驱动代码有问题（例如**电源管理函数**用错）：

‌ （1）硬件层缺陷‌

* 供电不稳定或电压波动（例如UFS需VCCQ 1.2V/VCC 2.5V。解决方式用万用表测量电压波动范围，明显不对时禁用动态电压调节功能。笔者建议不要为了一点点功耗就调原本稳定的电压）。
* 时钟信号异常（M-PHY时钟未同步，通常不太可能。解决方式用示波器确认19.2MHz时钟抖动<50ps，修改DTS配置，提升时钟驱动强度（如clock-strength = <3>））。
* 物理连接故障（例如接口氧化、线缆损坏，通常不太可能。解决方式换个手机再试试）。

‌ （2）驱动逻辑问题‌

* 电源管理函数用错（未正确让UFS器件保持唤醒）。
* 超时阈值设置过短（未考虑UFS设备响应延迟）。
* 中断处理阻塞（如DMA操作未及时释放资源）。
* 总线竞争冲突（SCSI子系统资源被其他进程占用）。

本文要介绍的便是电源管理函数用错问题，驱动逻辑问题中的后三种问题的解决方式本文暂不介绍，待后续更新。

1. 问题定位

遇到-110错误如何定位呢？首先看日志显示的进程和函数，确定涉及的文件位置，然后搜索电源管理函数名pm\_runtime\_get\_sync；如何不确定是否此处电源函数用错，那么可以把UFS autosuspend（UFSHCD\_CAP\_RPM\_AUTOSUSPEND）给关了，看看问题是否还复现。如果不复现那么就是电源管理函数参数用错了。

根据相关进程用户态打印信息，找到用户态进程文件用了ioctl函数自定义命令字与内核交互。进一步找到内核对该命令字处理的注册的ioctl回调函数，发现在使用该ioctl回调函数与器件交互的函数接口前后使用了电源管理函数。由于具体报错截图不便展示，以下代码仅示意：

pm\_runtime\_get\_sync(hba->dev);

err=ufs\_xxx\_query\_ioctl(hba, X, buffer);

pm\_runtime\_put\_sync(hba->dev);

至此问题已明晰，历史产品上之前的同事电源管理函数的参数用错了，原因是：

* hba->dev：代表主机总线适配器（HBA）本身的设备
* hba->ufs\_device\_wlun->sdev\_gendev：通过HBA连接的特定UFS设备或逻辑单元（LUN）

因此，pm\_runtime\_get\_sync(hba->dev)只能使主机侧controller保持唤醒，并不能唤醒device。所以在UFS autosuspend（UFSHCD\_CAP\_RPM\_AUTOSUSPEND）开的情况下，UFS器件可能会在空闲一段时间进入休眠/低功耗模式。此时，主机侧给器件发命令后，由于器件处于休眠状态不响应主机命令，因此最终主机侧会受到-110错误。因此，在做驱动开发时，需要弄清楚所使用的函数作用对象是主机还是设备，并使用正确的电源管理函数和/或参数。

ufshcd\_rpm\_get\_sync(hba)是社区已封装好的与UFS器件交互的电源管理函数，查询社区源码（内核6.1以上版本）可以发现，该函数即是对pm\_runtime\_get\_sync(&hba->ufs\_device\_wlun->sdev\_gendev)的封装。

\_static inline int ufshcd\_rpm\_get\_sync(struct ufs\_hba \*hba)

{

return pm\_runtime\_get\_sync(&hba->ufs\_device\_wlun->sdev\_gendev);

}

1. 正确用法

pm\_runtime\_get\_sync的正确用法：

（1）设置HBA时：

系统启动时需要对HBA进行初始化，包括配置HBA的寄存器、初始化数据结构等。这时需要确保HBA设备本身处于活跃状态，使用pm\_runtime\_get\_sync(hba->dev)

pm\_runtime\_get\_sync(hba->dev);

ufshcd\_hold(hba, false);

ahit = ufshcd\_readl(hba, REG\_AUTO\_HIBERNATE\_IDLE\_TIMER);

ufshcd\_release(hba);

pm\_runtime\_put\_sync(hba->dev);

（2）与UFS设备交互：

当需要对特定的UFS设备或LUN进行操作，如读写数据、查询设备状态等，这些操作是针对特定设备或LUN的，而不是HBA本身。在这种情况下，需要确保该特定UFS设备或LUN处于活跃状态，使用pm\_runtime\_get\_sync(hba-ufs\_device\_wlun->sdev\_gendev)

ufshcd\_rpm\_get\_sync(hba);

ret = ufshcd\_query\_descriptor\_retray(hba, UPIU\_QUERY\_OPCODE\_READ\_DESC, QUERY\_DESC\_IDN\_UNIT, lun, 0, desc\_buf, &size)

ufshcd\_rpm\_put\_sync(hba);

1. **本章总结**

本章聚焦于电源管理函数误用所引发的诸多问题中的一个具体类别——参数误配置。避免此类问题的核心在于深刻洞察并理解主机端与设备端在休眠与唤醒机制上的根本差异，以及精确把握存储于结构体hba中的各项参数所承载的具体含义与作用。通过深化对此类细节的认知，开发者能够更有效地规避参数误用的风险，从而提升系统的整体稳定性和可靠性。