



# **RTOS PMIC 开发指南**

**版本号: 0.1**  
**发布日期: 2020-10-22**

## 版本历史

| 版本号 | 日期        | 制/修订人     | 内容描述  |
|-----|-----------|-----------|-------|
| 0.1 | 2020.8.21 | Allwinner | 1. 初版 |



# 目 录

|                 |          |
|-----------------|----------|
| <b>1 前言</b>     | <b>1</b> |
| 1.1 文档简介        | 1        |
| 1.2 目标读者        | 1        |
| 1.3 适用范围        | 1        |
| <b>2 模块介绍</b>   | <b>2</b> |
| 2.1 模块功能介绍      | 2        |
| 2.2 相关术语介绍      | 2        |
| 2.3 模块配置介绍      | 2        |
| 2.4 模块源码结构      | 2        |
| <b>3 模块接口说明</b> | <b>3</b> |
| 3.1 接口列表        | 3        |
| 3.2 接口使用说明      | 3        |
| 3.2.1 电源初始化接口   | 3        |
| 3.2.2 电源使能接口    | 3        |
| 3.2.3 电源关闭接口    | 4        |
| 3.2.4 电源获取电压接口  | 4        |
| 3.2.5 电源设置电压接口  | 4        |
| <b>4 模块使用范例</b> | <b>5</b> |
| <b>5 FAQ</b>    | <b>7</b> |

# 1 前言

## 1.1 文档简介

介绍 RTOS 中 PMIC 驱动接口及使用方法，为 PMIC 使用者提供参考。

## 1.2 目标读者

PMIC 驱动层/应用层开发/使用/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  |
|------|-------|
| V459 | Melis |

## 2 模块介绍

### 2.1 模块功能介绍

该模块主要提供供电使能关闭、电压调节功能。

### 2.2 相关术语介绍

| 术语    | 解释说明                       |
|-------|----------------------------|
| Sunxi | 指 Allwinner 的一系列 SOC 硬件平台  |
| PMIC  | Power Management IC，电源管理芯片 |

### 2.3 模块配置介绍

Drivers Setup  
Melis Source Support →  
[\*] Support REGULATOR

### 2.4 模块源码结构

PMIC 模块源码结构如下所示：

```
rtos-hal/  
|--hal/source/regulator/axp2101.c  
|--hal/source/regulator/axp.c  
|--hal/source/regulator/axp_twi.c  
|--hal/source/regulator/ffs.h  
|--hal/source/regulator/type.h  
|--hal/source/regulator/sun8iw19p1/core.c  
--include/hal/sunxi_hal_regulator.h
```

## 3 模块接口说明

### 3.1 接口列表

PMIC 提供的接口列表如下：

```
int hal_regulator_get_voltage(struct regulator_dev *rdev, int *vol_uV);
int hal_regulator_set_voltage(struct regulator_dev *rdev, int target_uV);
int hal_regulator_enable(struct regulator_dev *rdev);
int hal_regulator_disable(struct regulator_dev *rdev);
int hal_regulator_get(unsigned int request_flag, struct regulator_dev *rdev);
```

### 3.2 接口使用说明

#### 3.2.1 电源初始化接口

- `int hal_regulator_get(unsigned int request_flag, struct regulator_dev *rdev)`
- 功能：根初始化 regulator
- 参数：
  - `request_flag` 指定要获取电源类型
  - `rdev` 为指向 `regulator_dev` 实例的非空指针
- 返回值：
  - 0, 成功

#### 3.2.2 电源使能接口

- `int hal_regulator_enable(struct regulator_dev *rdev)`
- 功能：使能电源
- 参数：
  - `rdev` 为要使能的电源
- 返回值：
  - 0, 成功

### 3.2.3 电源关闭接口

- `int hal_regulator_disable(struct regulator_dev *rdev)`
- 功能：关闭电源
- 参数：
  - `rdev` 为要关闭的电源
- 返回值：
  - 0, 成功

### 3.2.4 电源获取电压接口

- `int hal_regulator_get_voltage(struct regulator_dev rdev, int vol_uV)`
- 功能：获取电源电压值
- 参数：
  - `rdev` 为指定的电源
  - `vol_uV` 该电源对应的电压值
- 返回值：
  - 0, 成功

### 3.2.5 电源设置电压接口

- `int hal_regulator_set_voltage(struct regulator_dev *rdev, int target_uV)`
- 功能：设置电源电压
- 参数：
  - `rdev` 为指定的电源
  - `target_uV` 为要设置的电压值
- 返回值：
  - 0, 成功

## 4 模块使用范例

可参考测试程序。

```
static struct regulator_dev regulator;  
  
#define REGULATOR_ENABLE 1  
#define REGULATOR_DISABLE 2  
#define REGULATOR_SETVOL 3  
#define REGULATOR_GETVOL 4  
  
static int cmd_regulator(int argc, const char **argv)  
{  
    int cur_vol = 0, tar_vol = 0;  
    char regulator_type, regulator_id;  
    int c, state = REGULATOR_GETVOL;  
  
    if (argc < 4)  
    {  
        printk("Usage:\n");  
        printk("\tregulator [regulator type] [regulator id] -e\n");  
        printk("\t\t\t\t\t-d\n");  
        printk("\t\t\t\t\t-s [vol]\n");  
        printk("\t\t\t\t\t-g\n");  
        return -1;  
    }  
  
    regulator_type = strtol(argv[1], NULL, 0);  
    regulator_id = strtol(argv[2], NULL, 0);  
  
    while ((c = getopt(argc, (char *const *)argv, "degS")) != -1)  
    {  
        switch (c)  
        {  
            case 'e':  
                state = REGULATOR_ENABLE;  
                break;  
            case 's':  
                state = REGULATOR_SETVOL;  
                tar_vol = strtol(argv[4], NULL, 0);  
                break;  
            case 'd':  
                state = REGULATOR_DISABLE;  
                break;  
            case 'g':  
                state = REGULATOR_GETVOL;  
                break;  
            default:  
                state = REGULATOR_GETVOL;  
                break;  
        }  
    }  
}
```



```
hal_regulator_get(REGULATOR_GET(regulator_type, regulator_id), &regulator);

switch (state) {
case REGULATOR_ENABLE:
    hal_regulator_enable(&regulator);
    break;
case REGULATOR_DISABLE:
    hal_regulator_disable(&regulator);
    break;
case REGULATOR_GETVOL:
    hal_regulator_get_voltage(&regulator, &cur_vol);
    printk("cur_vol:%d\n", cur_vol);
    break;
case REGULATOR_SETVOL:
    hal_regulator_set_voltage(&regulator, tar_vol);
    break;
default:
    break;
}

return 0;
}
```



## 5 FAQ

---






## 著作权声明

版权所有 © 2020 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。