Rockchip RT-Thread 电源配置说明

文件标识：RK-KF-YF-112

发布版本：1.1.1

日期：2020-05-28

文件密级：□绝密 □秘密 □内部资料 ■公开

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

产品版本

芯片名称	版本
PISCES	RT-THREAD&HAL
RK2108	RT-THREAD&HAL
RV1108	RT-THREAD&HAL
RK1808	RT-THREAD&HAL

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

日期	版本	作者	修改说明
2019-07-17	V1.0	Elaine	第一次临时版本发布
2020-03-06	V1.1.0	Tony.xie	增加SOC集成LDO等电源模块支持描述
2020-05-28	V1.1.1	Elaine	修正格式

目录

1 RT-Thread REGULATOR 功能特点

- 支持 I2C 接口的 PMIC 调压、使能输出（如：RK808、RK818、RK809、RK817...）
- 支持 I2C 接口的独立 DCDC 调压、使能输出（如：SYR82X、TCS452X...）
- 支持针对 SOC 集成的 LDO 等电源模块的调压、使能输出（如：RK2108、PISCES...）

2 软件

2.1 代码路径

REGULATOR 接口：

```
struct regulator_desc *regulator_get_desc_by_pwrid(ePWR_ID pwrId);
rt_err_t regulator_set_voltage(struct regulator_desc *desc, int volt);
uint32_t regulator_get_voltage(struct regulator_desc *desc);
rt_err_t regulator_set_suspend_voltage(struct regulator_desc *desc, int
volt);
uint32_t regulator_get_suspend_voltage(struct regulator_desc *desc);
uint32_t regulator_get_real_voltage(struct regulator_desc *desc);
rt_err_t regulator_enable(struct regulator_desc *desc);
rt_err_t regulator_disable(struct regulator_desc *desc);
void regulator_desc_init(struct regulator_desc *descs, uint32_t cnt);
```

外部 PMIC 接口：

```
rt_uint32_t pmic_get_voltage(struct pwr_i2cbus_desc *desc);
rt_err_t pmic_set_voltage(struct pwr_i2cbus_desc *desc,
                          rt_uint32_t voltUv);
rt_uint32_t pmic_get_suspend_voltage(struct pwr_i2cbus_desc *desc);
rt_err_t pmic_set_suspend_voltage(struct pwr_i2cbus_desc *desc,
                                  rt_uint32_t voltUv);
rt_err_t pmic_set_enable(struct pwr_i2cbus_desc *desc, rt_uint32_t enable);
rt_uint32_t pmic_is_enabled(struct pwr_i2cbus_desc *desc);
int pmic_desc_init(struct pwr_i2cbus_desc *descs, uint32_t cnt);
void pmic_desc_deinit(void);
rt_err_t pmic_check_desc_by_pwrId(struct pwr_i2cbus_desc *pdesc, ePWR_ID
pwrId);
```

内部 SOC 集成调压接口：

```
int HAL_PWR_GetEnableState(struct PWR_INTREG_DESC *desc);
uint32_t HAL_PWR_GetVoltage(struct PWR_INTREG_DESC *desc);
uint32_t HAL_PWR_GetVoltageSuspend(struct PWR_INTREG_DESC *desc);
uint32_t HAL_PWR_GetVoltageReal(struct PWR_INTREG_DESC *desc);
HAL_Status HAL_PWR_SetVoltage(struct PWR_INTREG_DESC *desc, uint32_t volt);
HAL_Status HAL_PWR_SetVoltageSuspend(struct PWR_INTREG_DESC *desc, uint32_t
volt);
HAL_Status HAL_PWR_Enable(struct PWR_INTREG_DESC *desc);
HAL_Status HAL_PWR_Disable(struct PWR_INTREG_DESC *desc);
HAL_Check HAL_PWR_CheckDescByPwrId(struct PWR_INTREG_DESC *pdesc,
                                   ePWR_ID pwrId);
```

2.2 配置

2.2.1 打开 REGULATOR 配置

hal_conf.h

```
#define HAL_PWR_MODULE_ENABLED
```

2.2.2 打开 PMIC 配置

```
RT-Thread bsp drivers  --->
    RT-Thread rockchip common drivers  --->
        [*] Enable PMIC
```

```
RT-Thread Components  --->
    Device Drivers  --->
        [*] Using I2C device drivers
```

```
RT-Thread bsp drivers  --->
    RT-Thread rockchip rk1808 drivers  --->
        Enable I2C  --->
            [*] Enable I2C0
```

hal_conf.h

```
#ifdef RT_USING_I2C
#define HAL_I2C_MODULE_ENABLED
#endif

#ifdef RT_USING_PMIC
#define HAL_PWR_I2C8_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#endif
```

2.2.3 打开内部 SOC 调压功能

hal_conf.h

```
#define HAL_PWR_INTBUS_MODULE_ENABLED
```

备注： 根据产品实际硬件进行配置，如果是单外部 PMIC 调压的只要配置本章的 2.2.1 和 2.2.2（如 RK1808 项目），如果芯片内部 SOC 调压的只要配置 2.2.1 和 2.2.3（如 PISCES 项目），如果是外部 PMIC 和内部 SOC 都支持调压就要配置 2.2.1、2.2.2 和 2.2.3。

2.3 初始化设置

board.c 中有一个 desc 的结构体需要填充，主要是描述每路电源的硬件信息（是 I2C 还是内部，I2C 地址、寄存器的相关信息）

```
#ifdef HAL_PWR_MODULE_ENABLED
struct regulator_desc regulators[] =
{
    /****** vdd_npu **********/
    {
        .flag = REGULATOR_FLG_I2C8 | REGULATOR_FLG_LOCK,
```

```
        .desc.i2c_desc = {
            .flag = DESC_FLAG_LINEAR(PWR_CTRL_VOLT_SSPD),
            .info = {
                .pwrId = PWR_ID_DSP_CORE,
            },
            .i2c8.name = "i2c0",
            .i2c8.i2cAddr = 0x1c,
            PWR_DESC_I2C8_SHIFT_RUN(0x10, 0),
            PWR_DESC_I2C8_SHIFT_SSPD(0x11, 0),
            PWR_DESC_I2C8_SHIFT_EN(0x10, 1 << 7),
            .voltMask = 0x7f,
            PWR_DESC_LINEAR_VOLT(600000, 1300000, 6250),
        },
    },
    /****** vdd_log **********/
    {
        .flag = REGULATOR_FLG_I2C8 | REGULATOR_FLG_LOCK,
        .desc.i2c_desc = {
            .flag = DESC_FLAG_LINEAR(PWR_CTRL_VOLT_SSPD | PWR_FLG_ENMASK),
            .info = {
                .pwrId = PWR_ID_LOG,
            },
            .i2c8.name = "i2c0",
            .i2c8.i2cAddr = 0x20,
            PWR_DESC_I2C8_SHIFT_RUN(0xBB, 0),
            PWR_DESC_I2C8_SHIFT_SSPD(0xBC, 0),
            PWR_DESC_I2C8_SHIFT_EN(0xB1, 1 << 0),
            .voltMask = 0x7f,
            PWR_DESC_LINEAR_VOLT(500000, 1300000, 12500),
        },
    },
    /****** vdd_cpu **********/
    {
        .flag = REGULATOR_FLG_I2C8 | REGULATOR_FLG_LOCK,
        .desc.i2c_desc = {
            .flag = DESC_FLAG_LINEAR(PWR_CTRL_VOLT_SSPD | PWR_FLG_ENMASK),
            .info = {
                .pwrId = PWR_ID_CORE,
            },
            .i2c8.name = "i2c0",
            .i2c8.i2cAddr = 0x20,
            PWR_DESC_I2C8_SHIFT_RUN(0xBE, 0),
            PWR_DESC_I2C8_SHIFT_SSPD(0xBF, 0),
            PWR_DESC_I2C8_SHIFT_EN(0xB1, 1 << 1),
            .voltMask = 0x7f,
            PWR_DESC_LINEAR_VOLT(500000, 1300000, 12500),
        },
    },
};

const struct regulator_init regulator_inits[] =
{
    DUMP_REGULATOR("vdd_npu", PWR_ID_DSP_CORE, 875000),
    DUMP_REGULATOR("vdd_log", PWR_ID_LOG, 800000),
    DUMP_REGULATOR("vdd_arm", PWR_ID_CORE, 800000),
};
const rt_uint32_t regulator_init_num = HAL_ARRAY_SIZE(regulator_inits);
#endif
```

1. **desc 参数详解**

- **flag**：支持下面几种配置
  - **REGULATOR_FLG_I2C8**：8 位 I2C 传输的设备
  - **REGULATOR_FLG_INTREG**：内部 SOC 调压设备
  - **REGULATOR_FLG_LOCK**：是否需要锁（I2C 的设备都是需要的，内部调压的需要看场景和应用）
- **desc.i2c_desc**：
  - **flag**：支持下面几种配置
    - **PWR_FLG_FIXED**：固定电压，不支持电压调整
    - **PWR_FLG_ALWAYSON**：常开，不支持关闭输出
    - **PWR_FLG_ENMASK**：使能位是否带有 mask（RK808、RK818 没有 MASK 功能，RK816、RK805、RK817、RK809 都有 MASK 功能）
  - **info**：ePWR_ID，各路regulator对应的pwrId，用于 desc 结构的获取
- **i2c8.name**：i2c0\i2c1...用于 i2c 的 device 的获取
- **i2c8.i2cAddr**：i2c 地址
- **PWR_DESC_I2C8_SHIFT_RUN**：运行电压配置（寄存器，偏移）
- **PWR_DESC_I2C8_SHIFT_SSPD**：休眠电压配置（寄存器，偏移）
- **PWR_DESC_I2C8_SHIFT_EN**：使能输出(寄存器, 偏移)
- **voltMask**：电压 mask
- **PWR_DESC_LINEAR_VOLT**：电压设置步进(最小电压, 最大电压, 步进值)
- **desc.intreg_desc**：(同 i2c_desc 类似)：
  - **PWR_INTREG_SHIFT_RUN**：运行电压配置（寄存器，偏移）
  - **PWR_INTREG_SHIFT_SSPD**：休眠电压配置（寄存器，偏移）

2. **init 参数详解**

```
DUMP_REGULATOR("vdd_npu", PWR_ID_DSP_CORE, 875000),
```

vdd_npu： 电源 name, 只是为了打印 PWR_ID_DSP_CORE： ePWR_ID pwrId，用于 desc 结构的获取
875000： init 电压，如果设置 0，不会在初始化的时候设置此路电压

2.4 示例

vdd_cpu 调压

```
#include "drv_regulator.h"
{
struct regulator_desc *desc;

desc = regulator_get_desc_by_pwrid(PWR_ID_CORE);

regulator_set_voltage(desc, 900000);

regulator_enable(desc);
}
```

2.5 dump 接口

打开 dump config 配置：

```
RT-Thread bsp drivers  --->
    RT-Thread rockchip common drivers  --->
        [*] Enable REGULATOR_DUMP
```

REGULATOR DUMP 只能 DUMP 部分在 regulator_inits[]结构中的电源，如果需要增加时钟请按照 regulator_inits[]结构添加。

REGULATOR DUMP 使用是用 FINSH_FUNCTION_EXPORT，在 shell 命令行，切到 finsh 下，直接敲 regulator_dump()就可以。