Rockchip RT-Thread DVFS 使用说明

文件标识：RK-KF-YF-115

发布版本：V1.0.1

日期：2020-05-17

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	**RT-Thread** 版本
全部采用 RT-Thread 的芯片	

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
2019-07-19	V1.0.0	Tony Xie	初始发布
2020-05-27	V1.0.1	Tony Xie	修正格式

目录

1 RT-Thread DVFS 功能特点

- 管理一个 IC 模块对应的频率、电压需求
- 支持多个 IC 模块公用一路 regulator 电源
- 支持不同驱动/应用对同一个 IC 模块频率、电压进行申请。

2 软件

2.1 代码路径

Regulator req 接口：

```
void regulator_req_init(void);
void regulator_req_desc_init(struct req_pwr_desc *desc_arr, uint8_t cnt);
struct req_pwr_desc  *regulator_get_req_volt_id(ePWR_ID pwrid, uint8_t
*req_id);
rt_err_t regulator_req_set_voltage(struct req_pwr_desc *req_pwr, uint8_t
req_id,
                                   uint32_t volt);
uint32_t regulator_req_get_voltage(struct req_pwr_desc *req_pwr);
uint32_t regulator_req_get_max_voltage(struct req_pwr_desc *req_pwr);
uint32_t regulator_req_get_set_voltage(struct req_pwr_desc *req_pwr, uint8_t
req_id);
rt_err_t regulator_req_voltage_release(struct req_pwr_desc *req_pwr, uint8_t
req_id);
rt_err_t regulator_req_release(struct req_pwr_desc *req_pwr, uint8_t
req_id);
```

CLK req 接口：

```
void clk_req_desc_init(struct req_clk_desc *desc_array, uint8_t cnt);
void clk_req_init(void);
struct req_clk_desc  *clk_get_req_rate_id(eCLOCK_Name clk_id, uint8_t
*req_id);
rt_err_t clk_req_set_rate(struct req_clk_desc *req_clk, uint8_t req_id,
uint32_t rate);
uint32_t clk_req_get_rate(struct req_clk_desc *req_clk);
uint32_t clk_req_get_max_rate(struct req_clk_desc *req_clk);
uint32_t clk_req_get_set_rate(struct req_clk_desc *req_clk, uint8_t req_id);
rt_err_t clk_req_rate_release(struct req_clk_desc *req_clk, uint8_t req_id);
rt_err_t clk_req_release(struct req_clk_desc *req_clk, uint8_t req_id);
```

DVFS 接口：

```
rt_err_t dvfs_set_rate(struct rk_dvfs_desc *dvfs_desc, uint8_t
dvfs_clk_req_id, uint32_t rate);
rt_err_t dvfs_set_rate_by_idx(struct rk_dvfs_desc *dvfs_desc,
                              uint8_t tbl_idx, uint8_t dvfs_clk_req_id);
struct rk_dvfs_desc *dvfs_get_by_clk(eCLOCK_Name clk_id, uint8_t
 *dvfs_clk_req_id);
uint32_t dvfs_req_get_rate(struct rk_dvfs_desc *dvfs_desc);
uint32_t dvfs_req_get_max_rate(struct rk_dvfs_desc *dvfs_desc);
uint32_t dvfs_req_get_set_rate(struct rk_dvfs_desc *dvfs_desc, uint8_t
dvfs_clk_req_id);
void rk_dvfs_req_rate_release(struct rk_dvfs_desc *dvfs_desc,
                              uint8_t dvfs_clk_req_id);
void rk_dvfs_req_release(struct rk_dvfs_desc *dvfs_desc,  uint8_t
dvfs_clk_req_id);
void dvfs_desc_init(struct rk_dvfs_desc *dvfs_array, uint32_t  cnt);
void dvfs_init(void);
```

2.2 配置

2.2.1 开 DVFS 配置

```
RT-Thread rockchip common drivers  --->
    RT-Thread rockchip pm drivers  --->
        [*] Enble dvfs
```

2.2.2 开 Regulator req 配置

```
RT-Thread rockchip common drivers  --->
    RT-Thread rockchip pm drivers  --->
        [*] Enable request regulator vol
```

2.2.2 开 CLK req 配置

```
RT-Thread rockchip common drivers  --->
    RT-Thread rockchip pm drivers  --->
        [*] Enable request clk
```

2.2.3 开调测 log

```
RT-Thread rockchip common drivers  --->
    RT-Thread rockchip pm drivers  --->
        [*] Enable request clk
```

2.3 Regulator Req 使用说明

这个功能在多个 IC 模块公用一路电源时使用，功能为从各个模块的电压申请中找出最高的电压进行配置

2.3.1 初始化配置

```
static uint32_t core_pwr_req[2];
static struct req_pwr_desc req_pwr_array[] =
{
    {
        .pwr_id = PWR_ID_CORE,
        .req_ctrl = {
            .info.ttl_req = HAL_ARRAY_SIZE(core_pwr_req), /* for core & shrm
*/
            .req_vals = &core_pwr_req[0],
        }
    }
};

```

1. pwr_id 对应这路电源的 Regulator ID，参考：Rockchip_Developer_Guide_RT_Thread_Power_CN.md
2. core_pwr_req[2]这个数组用于记录各个模块申请的电压值，这里 core 和 shrm 两个模块公用这路电源，所以数组大小为 2.
3. 通过下面代码初始化指定支持 Regulator req 功能的电源

```
void rt_hw_board_init()
{
    regulator_req_desc_init(req_pwr_array, HAL_ARRAY_SIZE(req_pwr_array));
}
```

2.3.2 使用说明

1. 通过 regulator 的 id 申请一个 struct req_pwr_desc 的描述指针和一个 req_id,其中 req_id 用于管理是那个模块申请的电压，函数如下：

```
struct req_pwr_desc  *regulator_get_req_volt_id(ePWR_ID pwrid, uint8_t
*req_id)
```

2. 设置电压时通过 struct regulator_desc 的描述指针和对应 req_id 进行配置，函数如下：

```
rt_err_t regulator_req_set_voltage(struct req_pwr_desc *req_pwr, uint8_t
req_id,
                                   uint32_t volt)
```

2.4 CLK Req 使用说明

该功能在多个引用或模块申请某一个模块性能时使用，如 MCU 300M 时申请 SRAM 运行 300M，VOP 模块申请 SRAM 运行 200M，通过这个功能会选择 300M 作为 SRAM 的运行频率

2.4.1 初始化配置

```
static uint32_t clk_shrm_req[2];
static struct req_clk_desc req_clk_array[] =
{
    {
        .clk_id = SCLK_SHRM,
        .req_ctrl = {
            .info.ttl_req = HAL_ARRAY_SIZE(clk_shrm_req),
            .req_vals = &clk_shrm_req[0],
        }
    }
};

```

1. clk_id 对应一个 clk id，参考：Rockchip-Clock-Developer-Guide-RTOS-CN.md
2. clk_shrm_req[2]这个数组用于记录各个模块申请的 CLK 频率，这里 core 和 vop 两个模块有 sram 的需求，所以数组大小为 2.
3. 通过下面代码初指定支持 CLK req 功能的 CLK 模块

```
void rt_hw_board_init()
{
    clk_req_desc_init(req_clk_array, HAL_ARRAY_SIZE(req_clk_array));
}
```

2.4.2 使用说明

1. 通过 clk id 申请一个 struct req_clk_desc 的描述指针和一个 req_id,其中 req_id 用于管理是那个模块申请的频率，函数如下：

```
struct req_clk_desc  *clk_get_req_rate_id(eCLOCK_Name clk_id, uint8_t
*req_id);
```

2. 设置频率时通过 struct req_clk_desc 的描述指针和对应 req_id 进行配置，函数如下：

```
rt_err_t clk_req_set_rate(struct req_clk_desc *req_clk, uint8_t req_id,
uint32_t rate);
```

2.5 dvfs 使用

通过 clk id 配置一个模块的频率同时根据预先配置的频率电压表，配置对应的电压。

2.5.1 初始化配置

```
static struct dvfs_table dvfs_core_table[] =
{
    {
        .freq = 200000000,
        .volt = 950000,
    },
    {
        .freq = 300000000,
        .volt = 950000,
    },
};

```

```
static struct dvfs_table dvfs_shrm_table[] =
{
    {
        .freq = 200000000,
        .volt = 950000,
    },
    {
        .freq = 300000000,
        .volt = 950000,
    },
};

struct rk_dvfs_desc dvfs_data[] =
{
    {
        .clk_id = SCLK_SHRM,
        .pwr_id = PWR_ID_CORE,
        .tbl_idx = 1,
        .table = &dvfs_shrm_table[0],
        .tbl_cnt = HAL_ARRAY_SIZE(dvfs_shrm_table),
    },
    {
        .clk_id = HCLK_M4,
        .pwr_id = PWR_ID_CORE,
        .tbl_idx = 1,
        .table = &dvfs_core_table[0],
        .tbl_cnt = HAL_ARRAY_SIZE(dvfs_core_table),
    },
};
```

1. dvfs_data[]指定两个需要 dvfs 控制的 clk，分别为 HCLK_M4、SCLK_SHRM（clk_id 指定）。
2. 每一个 clk 对应的电源模块为 PWR_ID_CORE（pwr_id 指定）。
3. table，每个 CLK 对应的 dvfs 表格，如：dvfs_shrm_table
4. tbl_idx 表示以 dvfs 表格中第几个表项初始化频率、电压
5. 通过下面函数指定需要 dvfs 控制的 clk

```
    dvfs_desc_init(&dvfs_data, HAL_ARRAY_SIZE(dvfs_data));
```

2.5.2 使用说明

1. 通过 clk id 申请一个 struct rk_dvfs_desc 的描述指针和一个针对 clk 的 req_id（dvfs_clk_req_id）。函数如下：

```
struct rk_dvfs_desc *dvfs_get_by_clk(eCLOCK_Name clk_id, uint8_t
 *dvfs_clk_req_id);
```

2. dvfs_clk_req_id 作用为：req_id 记录这个 dvfs 节点对应的 CLK 被各个模块引用的申请信息，同上面 CLK req 中的 req_id 相同
3. 直接设置频率值，通过 struct rk_dvfs_desc 的描述指针和对应 dvfs_clk_req_id 进行配置，函数如下：

```
rt_err_t dvfs_set_rate(struct rk_dvfs_desc *dvfs_desc, uint8_t
dvfs_clk_req_id, uint32_t rate);
```

4. 可以通过 dvfs 节点对应的频率电压表的表项索引设置对应的电压，函数如下，参数 tbl_idx 指定要配置的频率为 struct dvfs_table dvfs_core_table[]中的第几项。

```
rt_err_t dvfs_set_rate_by_idx(struct rk_dvfs_desc *dvfs_desc,
                              uint8_t tbl_idx, uint8_t dvfs_clk_req_id);
```