

Rockchip RT-Thread Power Manager

文件标识: RK-KF-YF-104

发布版本: V1.1.0

日期: 2020-05-28

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

本文档按“现状”提供, 福州瑞芯微电子股份有限公司(“本公司”, 下同)不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自拥有者所有。

版权所有© 2020福州瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

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

本文提供基于RT Thread平台功耗控制说明。

产品版本

芯片名称	内核版本
本公司采用RT Thread系统的芯片	

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

版本号	作者	修改日期	修改说明
V1.0.0	谢修鑫	2020-03-04	初始版本
V1.0.1	谢修鑫	2020-05-27	修正格式
V1.1.0	许盛飞	2020-05-28	更新低功耗相关说明

目录

Rockchip RT-Thread Power Manager

- 1 RT-Thread Power Manager 功能支持
- 2 功耗管理方法
 - 2.1 基本概念
 - 2.2 Clock Gating
 - 2.3 Power Gating
 - 2.4 DVFS 控制
 - 2.5 RT-Thread PM MODE控制
 - 2.5.1 基本概念
 - 2.5.2 RT-Thread PM MODE 使用
 - 2.5.3 初始化配置-Power Management Mode 对应的 DVFS
 - 2.5.4 初始化配置-Power Management Mode
 - 2.6 Runtime Power 控制
 - 2.7 低功耗（Sleep Mode）设计及实现
 - 2.7.1 主要操作
 - 2.7.2 具体实现

1 RT-Thread Power Manager 功能支持

- 支持Clock Gating
- 支持Power Gating
- 支持DVFS 动态调频调压

2 功耗管理方法

2.1 基本概念

Clock Gating: 控制CLOCK开关

Power Gating: SOC内部会划分不同的Power Domain，每个Power Domain的供电可以通过SOC内部的控制进行开关，开关操作即Power Gating

DVFS-动态调频调压: 芯片运行一定频率需要对应电压支持，如：运行频率、电压为400M-0.9V，600M-0.95V两组，软件上支持频率从400M切换到600M，对应的电压从0.9V切换到0.95V

动态功耗: 动态功耗和运行电压、运行频率相关，控制方法为DVFS、Clock Gating

静态功耗: 静态功耗和电压、温度相关，控制方法为Power Gating

2.2 Clock Gating

一个模块不需要工作时，需要关闭对应的CLOCK；接口如下：

```
1 | rt_err_t clk_enable(struct clk_gate *gate, int on)
```

涉及参考文档为：CLK/Rockchip_Developer_Guide_RTOS_Clock_CN.md

2.3 Power Gating

一个模块不需要工作时，第一步需要先关闭对应的CLOCK，然后再关闭对应的Power Domain；由于关闭Power Gating的耗时比Clock Gating的耗时长，所以一般在长时间不使用的情况下才进行Power Gating操作；接口如下：

```
1 | rt_err_t pd_power(struct pd *power, int on)
```

涉及参考文档为：CLK/Rockchip_Developer_Guide_RTOS_Clock_CN.md

2.4 DVFS 控制

涉及参考文档为：Rockchip_Developer_Guide_RT-Thread_DVFS_CN.md

2.5 RT-Thread PM MODE控制

RT Thread基于PM MODE机制控制不同场景下的功耗需求,具体实现参数RT-Thread官方文档。实现上面通过和DVFS机制协同实现。

2.5.1 基本概念

系统在pm_cfg.h中定义了run modes、sleep modes 两类模式，用户也可以根据需求进行扩展。

```

1  enum
2  {
3      /* run modes */
4      PM_RUN_MODE_HIGH = 0,
5      PM_RUN_MODE_NORMAL,
6      PM_RUN_MODE_LOW,
7
8      /* sleep modes */
9      PM_SLEEP_MODE_SLEEP,
10     PM_SLEEP_MODE_TIMER,
11     PM_SLEEP_MODE_SHUTDOWN,
12 };

```

2.5.2 RT-Thread PM MODE 使用

1. 通过下面两个函数申请和释放一个模式:

```
void rt_pm_request(rt_ubase_t mode)
```

```
void rt_pm_release(rt_ubase_t mode)
```

参数mode: 对应上面的PM_RUN_MODE_HIGH、PM_RUN_MODE_NORMAL、PM_RUN_MODE_LOW、PM_SLEEP_MODE_SLEEP等模式。

2. 通过命令控制

通过命令: msh>pm_dump查看系统所在状态。

```

1  msh >pm_dump
2  | Power Management Mode | Counter | Timer |
3  +-----+-----+-----+
4  |      Running High Mode |        0 |      0 |
5  |      Running Normal Mode |        1 |      0 |
6  |      Running Low Mode |        0 |      0 |
7  |           Sleep Mode |        1 |      0 |
8  |           Timer Mode |        0 |      0 |
9  |      Shutdown Mode |        1 |      0 |
10 +-----+-----+-----+
11 pm current mode: Running Normal Mode

```

通过命令: msh>pm_request 1申请进入1的模式 (PM_RUN_MODE_NORMAL)

通过命令: msh>pm_release 1释放1的模式 (PM_RUN_MODE_NORMAL)。

2.5.3 初始化配置-Power Management Mode 对应的 DVFS

```

1  const static struct dvfs_table dvfs_core_table[] =
2  {
3      {
4          .freq = 297000000,
5          .volt = 800000,
6      },
7      {
8          .freq = 396000000,
9          .volt = 900000,
10     },
11 };
12
13 const static struct dvfs_table dvfs_shrm_table[] =

```

```

14  {
15      {
16          .freq = 297000000,
17          .volt = 800000,
18      },
19      {
20          .freq = 396000000,
21          .volt = 850000,
22      },
23  };
24
25  struct rk_dvfs_desc dvfs_data[] =
26  {
27      {
28          .clk_id = SCLK_SHRM,
29          .pwr_id = PWR_ID_CORE,
30          .tbl_idx = 1,
31          .table = &dvfs_shrm_table[0],
32          .tbl_cnt = HAL_ARRAY_SIZE(dvfs_shrm_table),
33      },
34      {
35          .clk_id = HCLK_M4,
36          .pwr_id = PWR_ID_CORE,
37          .tbl_idx = 1,
38          .table = &dvfs_core_table[0],
39          .tbl_cnt = HAL_ARRAY_SIZE(dvfs_core_table),
40      },
41  };

```

该代码段为配置DVFS table，参考：Rockchip_Developer_Guide_RT-Thread_DVFS_CN.md

2.5.4 初始化配置-Power Management Mode

```

1  static struct pm_mode_dvfs pm_mode_data[] =
2  {
3      {
4          .clk_id = HCLK_M4,
5          .run_tbl_idx = {1, 1, 0},
6          .sleep_tbl_idx = 0,
7      },
8      {
9          .clk_id = SCLK_SHRM,
10         .run_tbl_idx = {1, 1, 0},
11         .sleep_tbl_idx = 0,
12     },
13 };

```

通过struct pm_mode_dvfs结构管理每一个clk（clk_id指定）在各个PM MODE下的运行频率，指定方式为：

1.run_tbl_idx = {1, 1, 0}

这里的1, 1, 0指定PM_RUN_MODE_HIGH、PM_RUN_MODE_NORMAL、PM_RUN_MODE_LOW三种运行模式对应的频率为DVFS table (以core为例) 的dvfs_core_table[1]、dvfs_core_table[1]、dvfs_core_table[0]。

2.sleep_tbl_idx = 0 表示SLEEP 模式对应的频率电压为dvfs_data[0]定义。

3.系统通过下面函数初始化

```
1 void rkpm_register_dvfs_info(struct pm_mode_dvfs *dvfs, int cnt, void
  (*pm_func)(uint32_t))`
```

参数dvfs: 即上面 pm_mode_data[]首地址

参数pm_func: 用于不同芯片对于功耗的个性控制, 暂时没有使用。

2.6 Runtime Power 控制

对于在运行状态的系统, 系统会根据各个模块的状态进行相关功耗控制, 所以各个模块的驱动中需要通过下面两个函数申请、释放自己的运行状态。

```
1 void pm_runtime_request(ePM_RUNTIME_ID runTimeId)
2
3 void pm_runtime_release(ePM_RUNTIME_ID runTimeId)
```

2.7 低功耗 (Sleep Mode) 设计及实现

2.7.1 主要操作

低功耗的主要操作如下:

1. 配置休眠模式
2. 判断是否需要将系统休眠时钟切换到32K
3. 可选择配置GPIO, TIMER, TIMEOUT中断作为休眠时唤醒系统的信号源

2.7.2 具体实现

1. 配置休眠模式, 根据进入休眠时PD_DSP, PD_AUDIO的状态, 决定休眠时, 是否关闭对应的PD,VD及系统休眠时钟是否切换到32k, 在下面的函数中实现对应的操作。

```
1 static void SOC_SleepModeInit(struct PMU_REG *pPmu)
```

2. 配置唤醒信号源, 使能的唤醒源产生的中断能在系统休眠时唤醒系统。支持GPIO, TIMER, TIMEOUT等唤醒源的使能配置, 默认使能GPIO, TIMER唤醒源, TIMEOUT主要用于开发时调试验证使用。

更改默认唤醒源, 如disable GPIO中断唤醒功能, 只需要将下面的函数中value变量对应的唤醒源宏去掉。

```
1 static void SOC_WakeupSourceConfig(struct PMU_REG *pPmu)
```