Cygnus 功耗管理详细设计

© 2022 Mooresilicon All rights reserved.

本文档版权归TCL摩星半导体所有，受相关法律法规的保护。未经书面许可不得复制传播。

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 作者 | 说明 |
| 0.1 | 2022-4-15 | 张海民 | 起草 |
|  |  |  |  |

目录

[1 引言 5](#_Toc101533342)

[1.1 背景 5](#_Toc101533343)

[1.2 约束 5](#_Toc101533344)

[1.3 术语 5](#_Toc101533345)

[1.4 参考资料 5](#_Toc101533346)

[2 总体设计 6](#_Toc101533347)

[2.1 系统工作状态 6](#_Toc101533348)

[2.2 软件整体架构 6](#_Toc101533349)

[3 模块设计 7](#_Toc101533350)

[3.1 回调管理 7](#_Toc101533351)

[3.1.1 功能 7](#_Toc101533352)

[3.1.2 接口 7](#_Toc101533353)

[3.1.3 模块代码 8](#_Toc101533354)

[3.1.4 模块逻辑框图 8](#_Toc101533355)

[3.1.5 实现逻辑说明 9](#_Toc101533356)

[3.2 睡眠管理 9](#_Toc101533357)

[3.2.1 功能 9](#_Toc101533358)

[3.2.2 接口 9](#_Toc101533359)

[3.2.3 模块代码 10](#_Toc101533360)

[3.2.4 模块逻辑框图 11](#_Toc101533361)

[3.2.5 实现逻辑说明 13](#_Toc101533362)

[3.3 资源锁定管理 14](#_Toc101533363)

[3.3.1 功能 14](#_Toc101533364)

[3.3.2 接口 14](#_Toc101533365)

[3.3.3 模块代码 14](#_Toc101533366)

[3.3.4 模块逻辑框图 15](#_Toc101533367)

[3.3.5 实现逻辑说明 15](#_Toc101533368)

[3.4 电源管理 16](#_Toc101533369)

[3.4.1 功能 16](#_Toc101533370)

[3.4.2 接口 16](#_Toc101533371)

[3.4.3 模块代码 16](#_Toc101533372)

[3.4.4 模块逻辑框图 17](#_Toc101533373)

[3.4.5 实现逻辑说明 17](#_Toc101533374)

[3.5 频率管理 17](#_Toc101533375)

[3.5.1 功能 17](#_Toc101533376)

[3.5.2 接口 18](#_Toc101533377)

[3.5.3 模块代码 18](#_Toc101533378)

[3.5.4 模块逻辑框图 19](#_Toc101533379)

[3.5.5 实现逻辑说明 19](#_Toc101533380)

[4 数据结构设计 20](#_Toc101533381)

[5 异常处理 22](#_Toc101533382)

[6 系统扩展 23](#_Toc101533383)

# 引言

## 背景

在前一阶段Cygnus功耗管理设计概要文档中，已描述了功耗管理模块的整体设计要求以及系统工作状态定义和软件的整体架构等。

在本阶段中将确定如何具体的实现Cygnus功耗管理模块，从而在接下来的编码阶段可以根据本文档的说明转换成具体的代码实现。

本详细设计文档的适用读者为：内部软件开发工程师。

## 约束

SDK平台功耗管理的设计面向当前的MS1008 BLE SOC芯片，需要FreeRTOS的支持。

## 术语

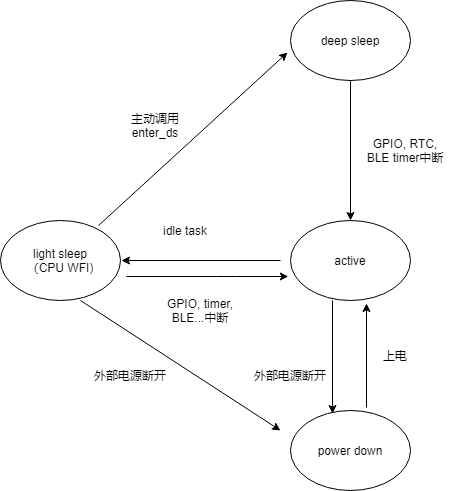
| **缩略语/术语** | **全 称** | **说 明** |
| --- | --- | --- |
| BLE | Bluetooth Low Energy | 低功耗蓝牙 |
| PM | Power Management | 功耗管理 |
|  |  |  |

## 参考资料

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 资料名称 | 版本 | 作者 |
| 1 | Cygnus功耗管理设计文档 | V0.1 | 薛鹏宇 |
|  |  |  |  |

# 总体设计

## 2.1 系统工作状态



工作状态说明：

* Power down:

关机状态，只能通过GPIO唤醒开机。

* Deep sleep:

深睡眠状态，可由应用或者Sleep主动调用pm\_sys\_enter\_dsleep (void)接口进入此状态。通过GPIO，RTC timer，BLE timer和LPUART(UART2)唤醒。

* Light Sleep:

浅睡眠状态，当没有任务执行时，将运行FreeRTOS的Idle task调用pm\_sys\_enter\_sleep(void)接口进入此状态，可被系统中断唤醒。

* Active:

正常工作模式。

## 软件整体架构

功耗管理包含下图浅红色的5个子模块：



# 模块设计

## 3.1 回调管理

### 3.1.1 功能

各个应用和组件如果需要在进入/退出睡眠（浅睡眠或深睡眠）前后做一些特殊的处理，需要调用回调管理模块注册回调函数。

在系统进入或者退出睡眠的时候，由底层驱动触发，睡眠管理模块检查当前的回调注册情况进行处理。

### 3.1.2 接口

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_register\_enter\_sleep\_callback(app\_sleep\_cb func) |
| 功能说明 | 应用/组件注册进入浅睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_deregister\_enter\_sleep\_callback(app\_sleep\_cb func) |
| 功能说明 | 应用/组件注销进入浅睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_register\_exit\_sleep\_callback (app\_sleep\_cb func) |
| 功能说明 | 应用/组件注册退出浅睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_deregiste\_exit\_sleep\_callback (app\_sleep\_cb func) |
| 功能说明 | 应用/组件注销退出浅睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_register\_enter\_dsleep\_callback (app\_sleep\_cb func) |
| 功能说明 | 应用/组件注册进入深睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_deregister\_enter\_dsleep\_callback (app\_sleep\_cb func) |
| 功能说明 | 应用/组件注销进入深睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_register\_exit\_dsleep\_callback (app\_sleep\_cb func) |
| 功能说明 | 应用/组件注册退出深睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

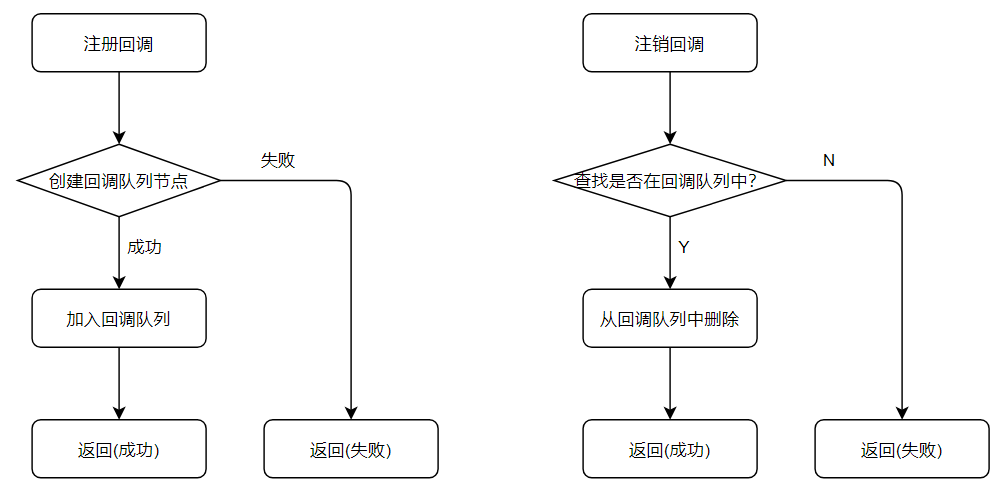
|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_deregiste\_exit\_dsleep\_callback (app\_sleep\_cb func) |
| 功能说明 | 应用/组件注销退出深睡眠回调 |
| 输入参数 | 回调函数指针 |
| 返回值 | 0：成功 -1：失败 |

### 3.1.3 模块代码

头文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_callback.h

源文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_callback.c

### 3.1.4 模块逻辑框图



### 3.1.5 实现逻辑说明

1） 回调函数采用队列管理，共有4个回调队列，分别为：

enter\_sleep\_callback\_q:进入浅睡眠时的回调函数队列

由pm\_register\_enter\_sleep\_callback ()和pm\_deregister\_enter\_sleep\_callback ()管理。

exit\_sleep\_callback\_q:退出浅睡眠时的回调函数队列

由pm\_register\_exit\_sleep\_callback ()和pm\_deregister\_exit\_sleep\_callback ()管理。

enter\_dsleep\_callback\_q:进入深睡眠时的回调函数队列

由pm\_register\_enter\_dsleep\_callback ()和pm\_deregister\_enter\_dsleep\_callback ()管理。

exit\_dsleep\_callback\_q:退出深睡眠时的回调函数队列

由pm\_register\_exit\_dsleep\_callback ()和pm\_deregister\_exit\_dsleep\_callback ()管理。

2） 回调队列中的节点为动态内存申请，注册回调时申请内存，注销回调时释放内存。

## 睡眠管理

### 3.2.1 功能

本模块实现系统睡眠管理(深睡眠或浅睡眠)的具体逻辑代码。

包括进入和退出睡眠逻辑，获取开机类型，获取唤醒类型等功能。

### 3.2.2 接口

|  |  |
| --- | --- |
| 函数名 | void pm\_sys\_init (void) |
| 功能说明 | 睡眠管理模块初始化 |
| 输入参数 | 无 |
| 返回值 | 无 |

|  |  |
| --- | --- |
| 函数名 | void pm\_sys\_enter\_sleep (void) |
| 功能说明 | 浅睡眠处理函数，由FreeRTOS调用 |
| 输入参数 | 无 |
| 返回值 | 无 |

|  |  |
| --- | --- |
| 函数名 | void pm\_sys\_exit\_sleep (void) |
| 功能说明 | 退出浅睡眠处理函数时，用于系统相关状态。 |
| 输入参数 | 无 |
| 返回值 | 无 |

|  |  |
| --- | --- |
| 函数名 | void pm\_sys\_enter\_dsleep (void) |
| 功能说明 | 进入深睡眠，由FreeRTOS/App调用 |
| 输入参数 | 无 |
| 返回值 | 无 |

|  |  |
| --- | --- |
| 函数名 | void pm\_sys\_exit\_dsleep (void) |
| 功能说明 | 退出深睡眠时，在系统启动之后判断是深睡眠开机时调用此函数，用于恢复BLE以及系统相关状态。 |
| 输入参数 | 无 |
| 返回值 | 无 |

|  |  |
| --- | --- |
| 函数名 | uint32\_t pm\_get\_pre\_mode (void) |
| 功能说明 | 启动之后判断开机类型（上电还是唤醒），以便后续跟据开机类型进行相应的操作。 |
| 输入参数 | 无 |
| 返回值 | 0：关机上电  1：深睡眠唤醒 |

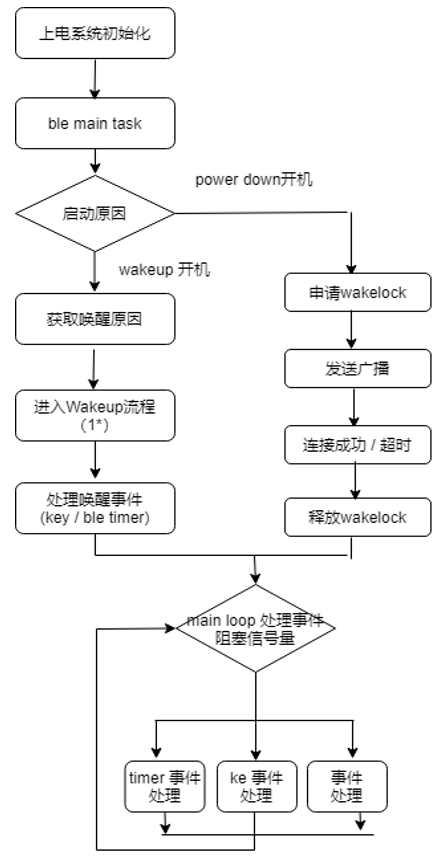
|  |  |
| --- | --- |
| 函数名 | uint32\_t pm\_get\_wakeup\_resource (void) |
| 功能说明 | 启动之后判断唤醒类型（gpio/RTC/ble timer） |
| 输入参数 | 无 |
| 返回值 | 0：未知唤醒类型(默认值)  1：GPIO  2：LPUART (UART2)  3：RTC Timer  4：BLE Timer |

### 3.2.3 模块代码

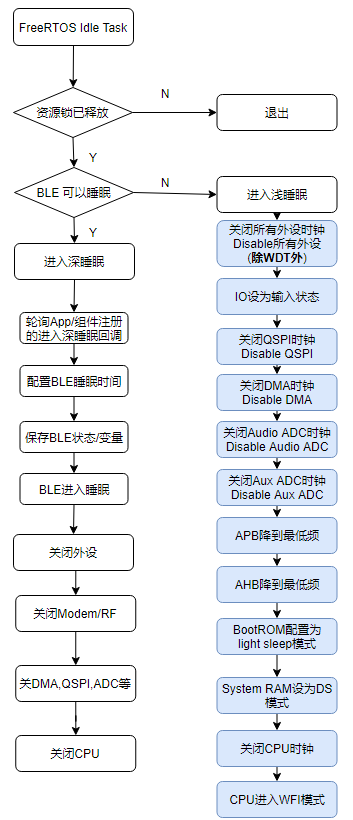
头文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_sleep.h

源文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_sleep.c

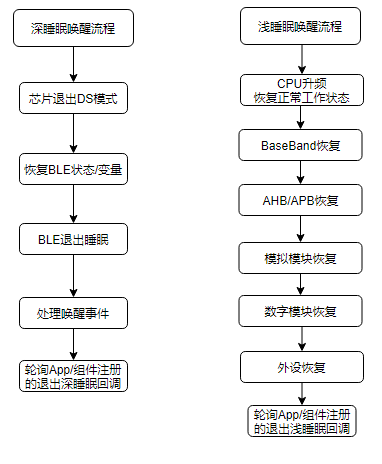
### 3.2.4 模块逻辑框图



进入睡眠逻辑框图：



退出睡眠逻辑框图：



### 3.2.5 实现逻辑说明

1）当FreeRTOS运行空闲任务(idle task)的时候，系统根据运行状态判断进入睡眠的模式：

* 浅睡眠由pm\_sys\_enter\_sleep()函数进入，关闭不需要的模块，进入CPU WFI状态
* 深睡眠由pm\_sys\_enter\_dsleep()函数进入，保存必要的参数到Retention RAM中，然后关闭不需要的模块，芯片电源域进入DS2状态(详见概要设计《Cygnus功耗管理设计文档》说明)。

2）如果有App/组件获取(持有)资源锁，则无法进入睡眠模式；

3） 如深睡眠中，蓝牙需要间隔唤醒和电视保持通信连接，睡眠前需要从蓝牙模块获取本次进入睡眠的事件，设置蓝牙唤醒timer；

4）唤醒之后处理唤醒事件包含了蓝牙协议栈必要的变量恢复以及蓝牙基带必要的寄存器恢复，才能保证能正常发送蓝牙按键或者蓝牙连接。

5）进入/退出睡眠时，在轮询回调函数的时候采用先注册先调用的顺序执行；

6）系统退出深睡眠的时候，通过pm\_get\_wakeup\_resource ()获取唤醒的状态；

## 资源锁定管理

### 3.3.1 功能

资源锁定管理模块为应用或者组件提供资源锁定接口，应用或者组件可以通过本模块接口获取需要保持的资源，以让系统保持工作状态不进入睡眠。

### 3.3.2 接口

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_lock\_create(void) |
| 功能说明 | 创建资源锁 |
| 输入参数 | 无 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_lock\_acquire(PM\_LOCK\_Type locktype) |
| 功能说明 | 应用/组件获取资源锁，不让系统进入睡眠等 |
| 输入参数 | locktype：资源锁类型  0：无用锁  1：CPU锁 |
| 返回值 | 0：成功 -1：失败 |

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_lock\_release(PM\_LOCK\_Type locktype) |
| 功能说明 | 应用/组件释放资源锁 |
| 输入参数 | locktype：资源锁类型  0：无用锁  1：CPU锁 |
| 返回值 | 0：成功 -1：失败 |

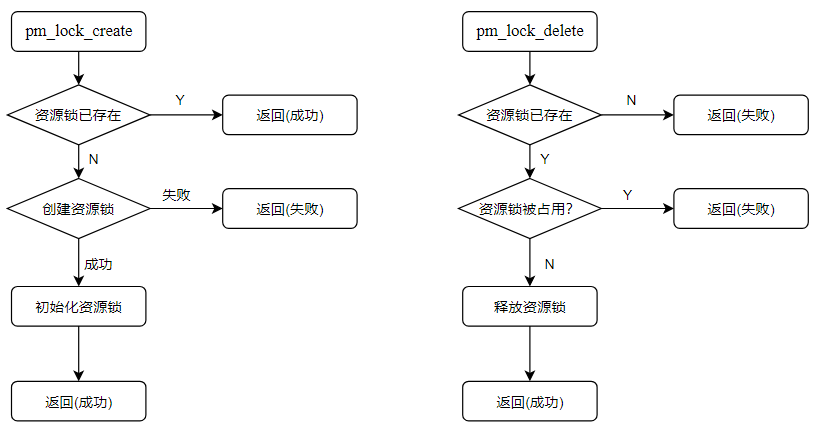
|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_lock\_delete(void) |
| 功能说明 | 删除创建的资源锁 |
| 输入参数 | 无 |
| 返回值 | 0：成功 -1：失败 |

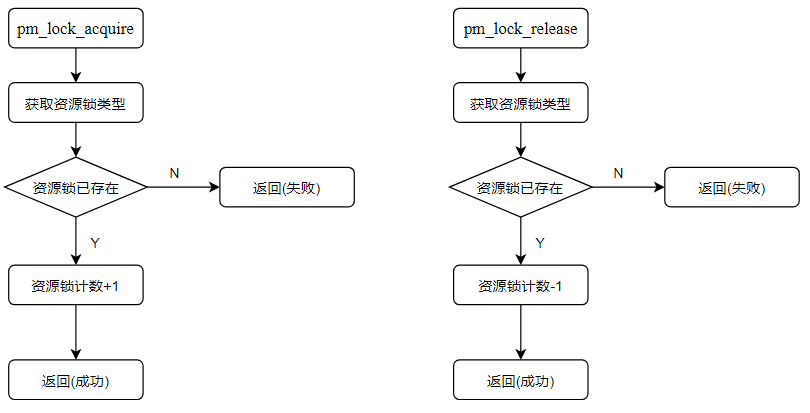
### 3.3.3 模块代码

头文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_lock.h

源文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_lock.c

### 3.3.4 模块逻辑框图





### 3.3.5 实现逻辑说明

1）资源锁有多个类型用于锁定不同的工作状态。比如可以锁定CPU不进入睡眠，或者锁定某个外设保持特定的工作状态。第一阶段先支持锁定CPU不进入睡眠功能，后续根据应用需求可扩展；

2）多个应用/组件可以用同一个资源锁， 同一个资源锁每被获取一次内部计数加1，每被释放一次内部计数减1；

3）删除资源锁的时候，如果资源锁仍被占用(内部计数不等于0)则不能被删除；

4）使用资源锁的时候必须保证获取与释放操作匹配；

## 电源管理

### 3.4.1 功能

本模块用于获取当前电池电量，给BLE battery profile提供支撑。

### 3.4.2 接口

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_get\_batterylevel (PM\_Battery\_Type type) |
| 功能说明 | 应用/组件获取当前的电池电量(百分比/实际电压值) |
| 输入参数 | Type：获取当前的电池电量类型  0：电量百分比  1：实际电压值 |
| 返回值 | 0 ~ 100：电池电量百分比  说明：  电压3V以上为100% ，2V 及以下为0% ，精度为0.01V（1% ） |

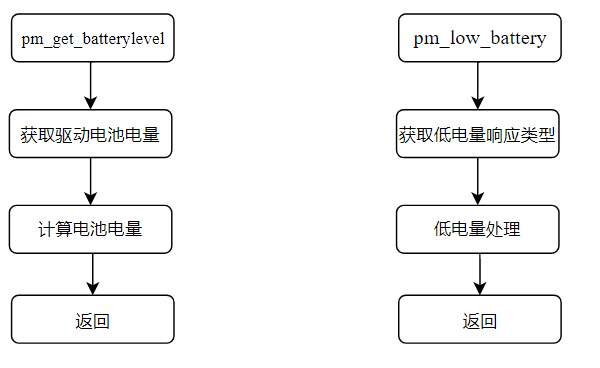
|  |  |
| --- | --- |
| 函数名 | void pm\_low\_battery(uint32\_t type) |
| 功能说明 | 低电量处理（闪led灯，发送警告信息） |
| 输入参数 | Type：低电量处理类型  0：不需处理  1：发送警告信息  2：闪LED灯 |
| 返回值 | 无 |

### 3.4.3 模块代码

头文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_battery.h

源文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_battery.c

### 3.4.4 模块逻辑框图



### 3.4.5 实现逻辑说明

1） 电池电量计算：电压3V以上为100% ，2V 及以下为0% ，精度为0.01V（1% ）；

2） 低电压处理可分为：

* 通过BLE battery profile发送报警指令（CTRL + ALT + B）给电视，在电视端警告
* 通过遥控器本身的指示灯警告

3）本模块功能后续可根据应用需要扩展充电相关功能；

## 频率管理

### 3.5.1 功能

设置CPU运行的频率，降低功耗

### 3.5.2 接口

|  |  |
| --- | --- |
| 函数名 | int32\_t pm\_set\_frequency(PM\_FREQ\_Type type, uint32\_t freq) |
| 功能说明 | 应用/组件设置当前的系统频率 |
| 输入参数 | Type：频率类型  0：CPU  1：AHB - (TBD)  2：APB - (TBD)  Freq：频率值 |
| 返回值 | 0：成功 -1：失败 |

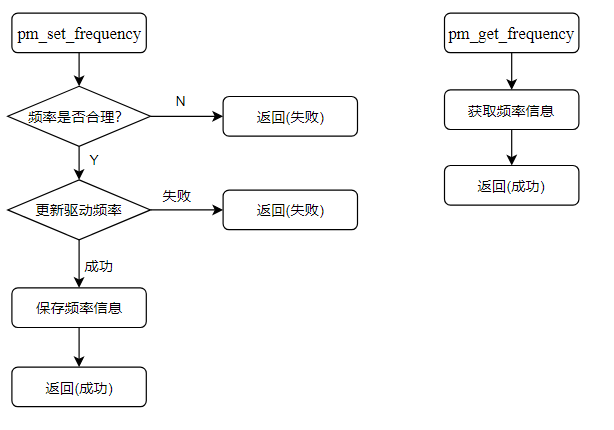
|  |  |
| --- | --- |
| 函数名 | uint32\_t pm\_get\_frequency(PM\_FREQ\_Type type) |
| 功能说明 | 应用/组件获取当前的系统频率 |
| 输入参数 | Type：频率类型  0：CPU  1：AHB - (TBD)  2：APB - (TBD) |
| 返回值 | 频率值 |

### 3.5.3 模块代码

头文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_frequency.h

源文件：ssh://git@10.127.8.16/remote/depot/git/cygnus/components/pm/pm\_frequency.c

### 3.5.4 模块逻辑框图



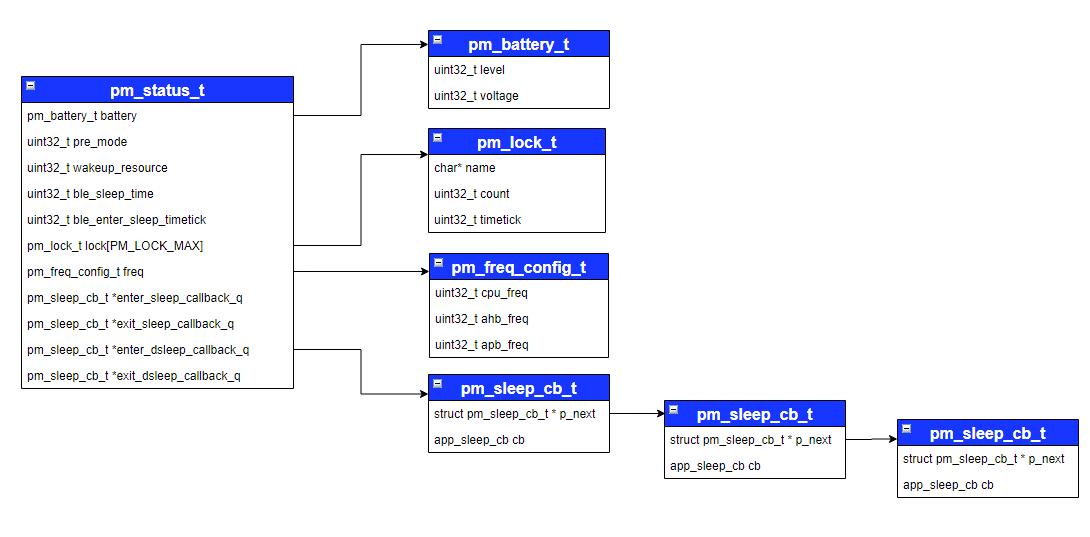
### 3.5.5 实现逻辑说明

1）因为频点是非连续性的，所以设置频率参数的时候须检查参数的有效性，确保频点正确；

2）本模块接口支持设置多个系统模块频率(CPU/AHB/APB等)，第一阶段先实现CPU频率配置功能，后续根据实际应用需求再扩展。

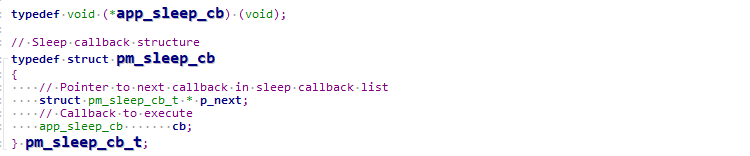
# 4 数据结构设计

功耗管理整体数据结构如下：

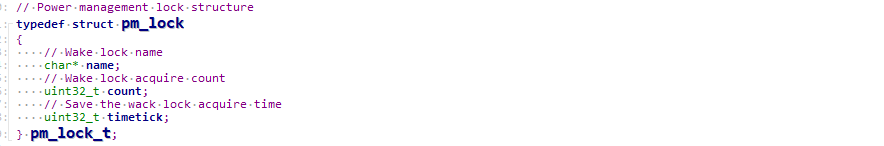


结构体定义：

|  |  |  |
| --- | --- | --- |
| 结构体名字 | pm\_sleep\_cb\_t | |
| 变量成员 | struct pm\_sleep\_cb\_t \* p\_next | 链表指针，指向回调函数队列的下一个成员 |
|  | app\_sleep\_cb cb | 回调函数指针 |
| 说明：回调函数类型app\_sleep\_cb定义为：typedef void (\*app\_sleep\_cb) (void); | | |



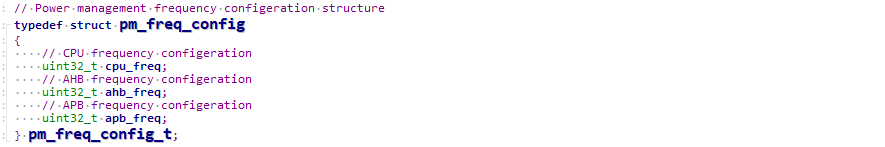
|  |  |  |
| --- | --- | --- |
| 结构体名字 | pm\_lock\_t | |
| 变量成员 | char\* name | 资源锁名字 |
|  | uint32\_t count | 资源锁当被已经获取次数 |
|  | uint32\_t timetick | 资源锁被获取时间 |
| 说明：同一个资源锁每被获取一次内部计数加1，每被释放一次内部计数减1；  资源锁被获取时间timetick主要用于异常判断的超时处理； | | |



|  |  |  |
| --- | --- | --- |
| 结构体名字 | pm\_battery\_t | |
| 变量成员 | uint32\_t level | 电池电量百分比 |
|  | uint32\_t voltage | 电池实际电压值 |
| 说明：无 | | |



|  |  |  |
| --- | --- | --- |
| 结构体名字 | pm\_freq\_config\_t | |
| 变量成员 | uint32\_t cpu\_freq | CPU工作频率 |
|  | uint32\_t ahb\_freq | AHB工作频率 |
|  | uint32\_t apb\_freq | APB工作频率 |
| 说明：无 | | |



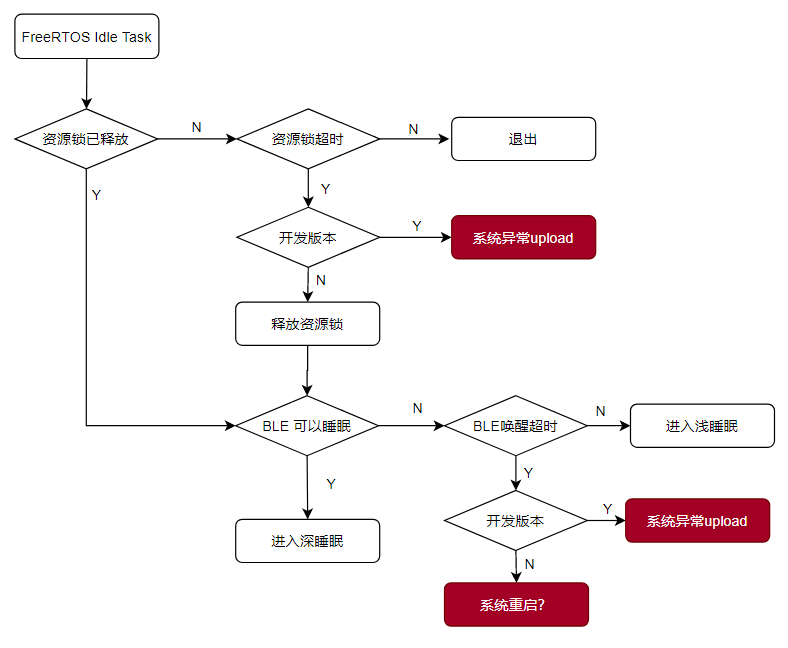
|  |  |  |
| --- | --- | --- |
| 结构体名字 | pm\_status\_t | |
| 变量成员 | pm\_battery\_t battery | 电池电量百分比和电压值 |
|  | uint32\_t pre\_mode | 开机类型 |
|  | uint32\_t wakeup\_resource | 系统唤醒类型 |
|  | uint32\_t ble\_sleep\_time | BLE睡眠时间 |
|  | uint32\_t ble\_enter\_sleep\_timetick | BLE进入睡眠时间 |
|  | pm\_lock\_t lock[PM\_LOCK\_MAX] | 资源锁 |
|  | pm\_freq\_config\_t freq | 系统(CPU/AHB/APB)频率配置 |
|  | pm\_sleep\_cb\_t \*enter\_sleep\_callback\_q | 进入浅睡眠时的回调函数队列 |
|  | pm\_sleep\_cb\_t \*exit\_sleep\_callback\_q | 退出浅睡眠时的回调函数队列 |
|  | pm\_sleep\_cb\_t \*enter\_dsleep\_callback\_q | 进入深睡眠时的回调函数队列 |
|  | pm\_sleep\_cb\_t \*exit\_dsleep\_callback\_q | 退出深睡眠时的回调函数队列 |
| 说明：BLE进入睡眠时间ble\_enter\_sleep\_timetick主要用于异常判断的超时处理； | | |



# 5 异常处理

|  |  |
| --- | --- |
| **异常类型** | **处理方案** |
| 异常持锁导致无法进入睡眠 | * 开发版本系统触发异常upload * 商用版本强制释放资源锁 |
| BLE异常无法进入睡眠 | * 开发版本系统触发异常upload * 商用版本强制系统重启 |
| 深睡眠后无法唤醒 | 是否有硬件处理方案？(长按Power键强制硬件重启) |
|  |  |
|  |  |

异常处理流程：



# 6 系统扩展

1. 电源管理

第一阶段支持电池电量计算，后续根据应用需求扩展充电管理功能。

1. 资源锁定

第一阶段先支持锁定CPU不进入睡眠功能，后续根据应用需求可扩展。

1. 频率管理

第一阶段先实现CPU/AHB/APB频率配置功能，后续根据应用需求再扩展。

1. 动态调整深睡眠时间

学习用户习惯，动态调整深睡眠时间，或者直接关机后定时启动。

1. 每个task的 CPU使用率统计

在开发版本中用于调试分析，商用版本关闭此功能。

1. …