

**SY826 软件Demo介绍**

**& 应用注意事项 V3.0**

**电子烟领域专用芯片**

**集成气流检测、负载检测、功率输出、电池充电、高精度ADC、马达驱动、8051内核**

请认真阅读关于Amicro知识产权政策

本文中提及的“珠海一微半导体股份有限公司”(简称“本公司”)对此产品持有知识产权及其对应的法定权益。未经合法授权，使用本公司的MCU或其他相关产品的行为将被视为侵权。对于任何未经授权而侵犯本公司知识产权的实体或个人，本公司有权采取法律手段保护权益，并将对由此造成的损害寻求赔偿。

\*本公司保留对产品规格书中，关于产品设计、功能和可靠性方面的改进作进一步说明的权利，但对于规格内容的使用并不承担责任。文档中所描述的应用案例仅供参考，本公司不保证和不表示这些应用，在没有更深入地更改和修正就能适用。同时，本公司不推荐产品使用在可能会对人身造成危害的场景。本公司的产品未经特别授权，不得用于救生、维生器件或系统中或作为关键器件使用。

\*本公司保留在未经预告的情况下修改其产品的权利。

目录

[1工程目录说明 4](#_Toc145875799)

[1.1文件夹目录 4](#_Toc145875800)

[1.1.1 App-应用层 4](#_Toc145875801)

[1.1.2 Bsp-中间层 5](#_Toc145875802)

[1.1.3 Pmcu\_Lib-API接口库 6](#_Toc145875803)

[1.1.4 Project-Keil工程文件 7](#_Toc145875804)

[1.1.5 Startup-启动文件 7](#_Toc145875805)

[2 工程结构框架 8](#_Toc145875806)

[2.1咪头吸烟和按键检测 9](#_Toc145875807)

[2.2电池电量管理 9](#_Toc145875808)

[2.3充电状态管理 9](#_Toc145875809)

[2.4负载检测管理 10](#_Toc145875810)

[2.5灯效管理 10](#_Toc145875811)

[2.6功率输出管理 11](#_Toc145875812)

[2.7掉电参数保存 12](#_Toc145875813)

[2.8系统参数配置 12](#_Toc145875814)

[3 应用注意事项 14](#_Toc145875815)

[3.1系统内核相关 14](#_Toc145875816)

[3.1.1上电关闭看门狗 14](#_Toc145875817)

[3.1.2上电增加延时，确保可烧录 14](#_Toc145875818)

[3.1.3上电使能系统时钟 14](#_Toc145875819)

[3.1.4 启动文件对SRAM进行清0 15](#_Toc145875820)

[3.1.5 API接口的延时函数 15](#_Toc145875821)

[3.1.6 配置参数保存 15](#_Toc145875822)

[3.1.7 固件调试方式 16](#_Toc145875823)

[3.2充电相关 16](#_Toc145875824)

[3.3.1 600mA充电电流以上应用 16](#_Toc145875825)

[3.3.2 充电使能 17](#_Toc145875826)

[3.3 GPIO相关 17](#_Toc145875827)

[3.3.1 JTAG调试接口复用问题 17](#_Toc145875828)

[3.4 睡眠唤醒相关 17](#_Toc145875829)

[3.4.1 睡眠唤醒源说明 17](#_Toc145875830)

[3.4.2 睡眠静态功耗处理 18](#_Toc145875831)

[3.5 咪头吸烟相关 18](#_Toc145875832)

[3.5.1 MIC参数配置 18](#_Toc145875833)

[3.5.2 电容MIC吸烟阈值导致误触发问题 18](#_Toc145875834)

[3.6 ADC驱动相关 19](#_Toc145875835)

[3.6.1 ADC分压系数和电压转换 19](#_Toc145875836)

[3.6.2 ADC触发采样 19](#_Toc145875837)

[3.6.3 GPIOBx作为ADC输入配置 20](#_Toc145875838)

[3.7 功率输出相关 20](#_Toc145875839)

[3.7.1 双MOS同时驱动 20](#_Toc145875840)

[3.7.2 负载检测 21](#_Toc145875841)

[3.7.3 OCP保护锁存 21](#_Toc145875842)

[3.7.4 功率输出时，VOUT采样问题 21](#_Toc145875843)

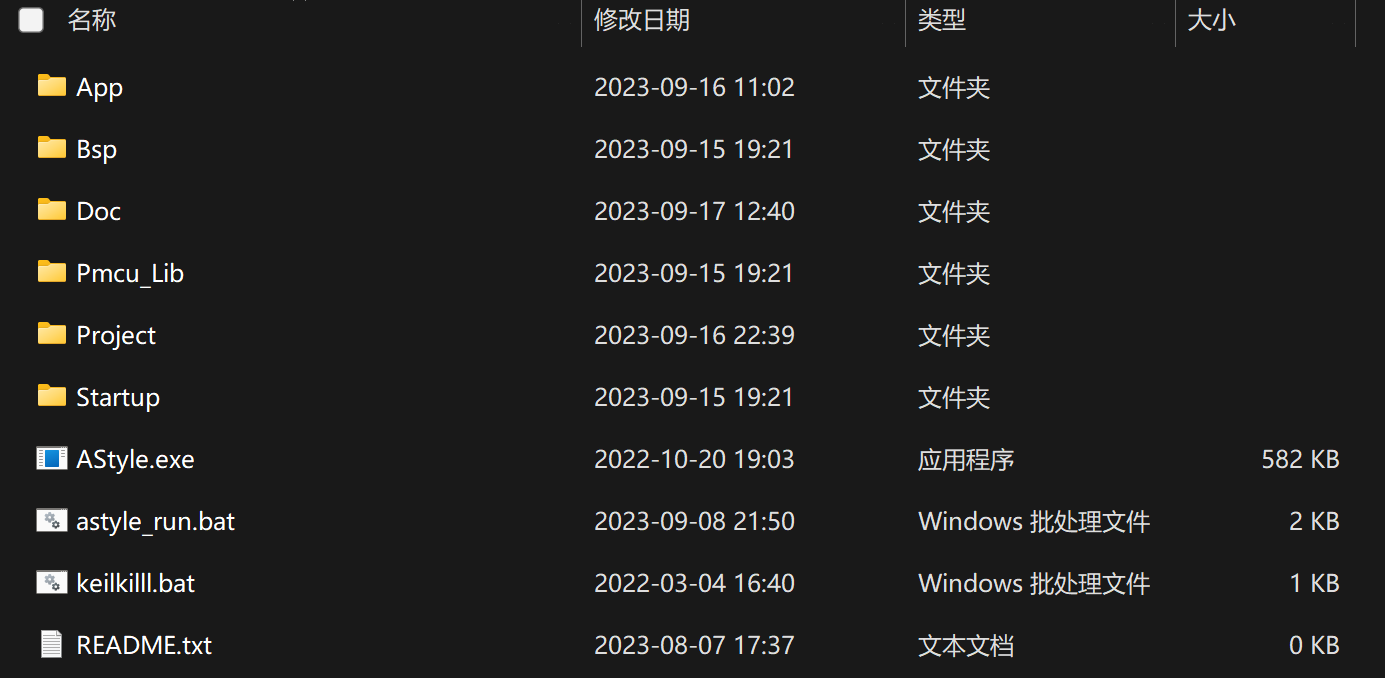
[3.8 LED驱动相关 22](#_Toc145875844)

[3.8.1 LED驱动配置 22](#_Toc145875845)

# 

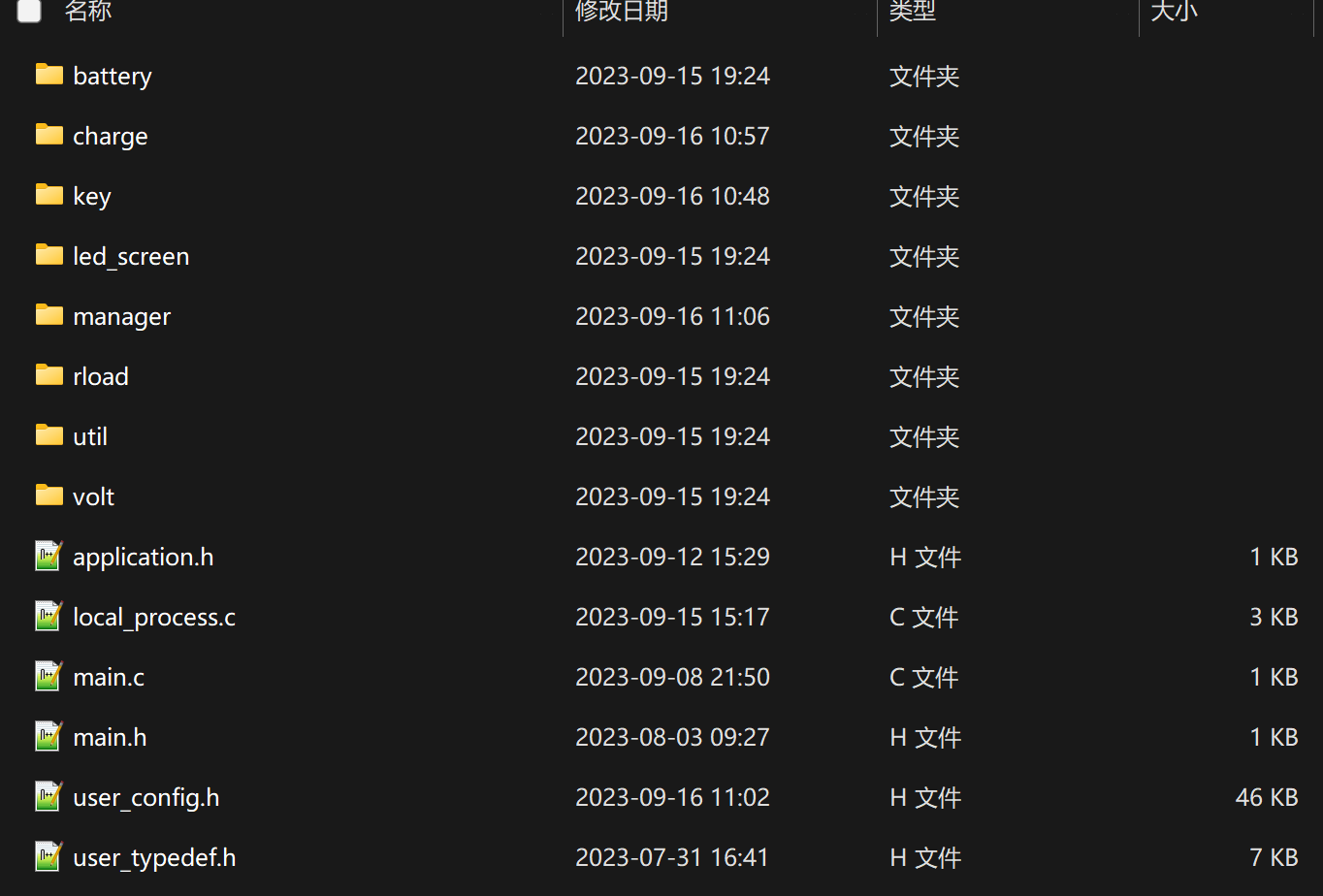
# 1工程目录说明

## 1.1文件夹目录



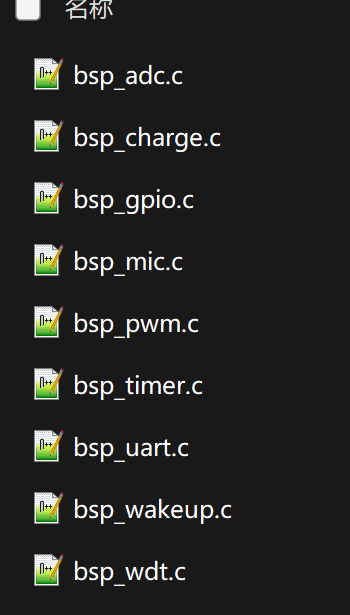
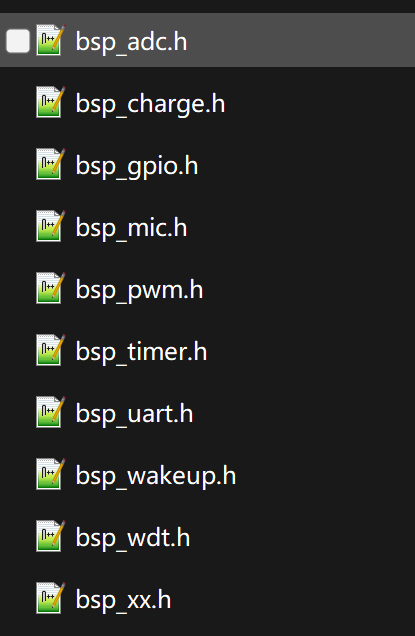
|  |  |
| --- | --- |
| 名称 | 文件内容 |
| App | 应用层源码，用于实现各个模块功能控制逻辑 |
| Bsp | 芯片外设中间层源码，用于实现芯片各个外设模块的初始化和中断控制 |
| Doc | 芯片开发的相关说明文档 |
| Pmcu\_Lib | 芯片外设各模块的API接口库 |
| Project | 代码Keil工程 |
| Startup | 芯片启动文件 |
| keilkill.bat | Window批处理脚本，用于批量删除工程编译的中间文件 |
| AStyle.exe/astyle\_run.bat | Window批处理脚本，用于格式化代码格式 |
| README.txt | 工程说明文件 |

### 1.1.1 App-应用层



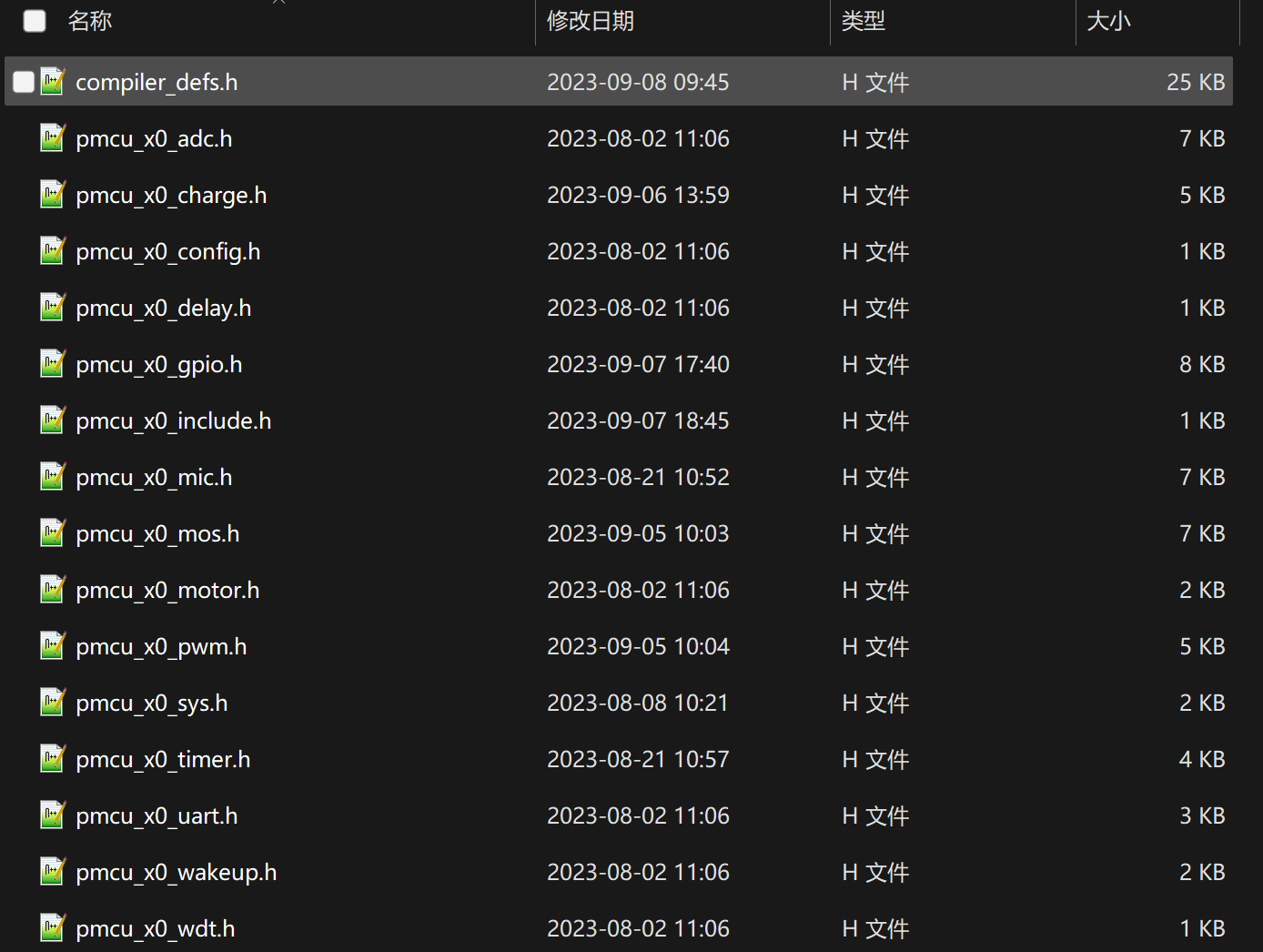
|  |  |
| --- | --- |
| 名称 | 文件内容 |
| battery | 电池管理功能实现，如电池电压获取，电量等级划分等 |
| charge | 充电管理功能实现，如USB拔插、充电流程检测等 |
| key | 按键/咪头检测功能实现，如按键次数、咪头触发吸烟等 |
| led\_screen | LED灯/断码屏功能实现，如LED闪烁、呼吸、常亮或断码屏驱动等 |
| manager | 主控制流程实现，如系统状态机控制、系统事件管理切换等 |
| rload | 电热丝负载管理实现，如负载检测、阻值检测、开短路等 |
| vout | 功率输出实现，如恒压、恒有效电压、恒功率输出等 |
| util | 自行实现的通用接口，类似第三方工具，如；打印函数、滤波函数、队列等 |
| application.h | 应用层总的头文件包含 |
| local\_process.c | 主进程控制文件 |
| main.c/ main.h | 主函数入口 |
| user\_typedef.h | 通用定义头文件 |
| user\_config.h | 用户功能配置头文件，可通过其形成的可视化界面，选配相关功能是否开启 |

### 1.1.2 Bsp-中间层

|  |  |
| --- | --- |
| 名称 | 文件内容 |
| bsp\_adc.c/ bsp\_adc.h | ADC模块初始化，ADC通道采样接口 |
| bsp\_charge.c/ bsp\_charge.h | 充电模块初始化 |
| bsp\_gpio.c/ bsp\_gpio.h | GPIO模块初始化 |
| bsp\_mic.c/ bsp\_mic.h | 咪头模块初始化 |
| bsp\_pwm.c/ bsp\_pwm.h | PWM模块初始化 |
| bsp\_timer.c/ bsp\_timer.h | 定时器模块初始化 |
| bsp\_uart.c/ bsp\_uart.h | 串口模块初始化 |
| bsp\_wakeup.c/ bsp\_wakeup.h | 睡眠模块初始化 |
| bsp\_wdt.c/ bsp\_wdt.h | 看门狗模块初始化 |
| bsp\_xx.h | Bsp层头文件汇总 |

### 1.1.3 Pmcu\_Lib-API接口库





pmcu\_x0\_stdperiph\_lib\_um.chm为库接口函数的详细说明文件，主要针对各个接口的使用方式进行说明，如参数、返回值、功能、注意事项等。



### 1.1.4 Project-Keil工程文件



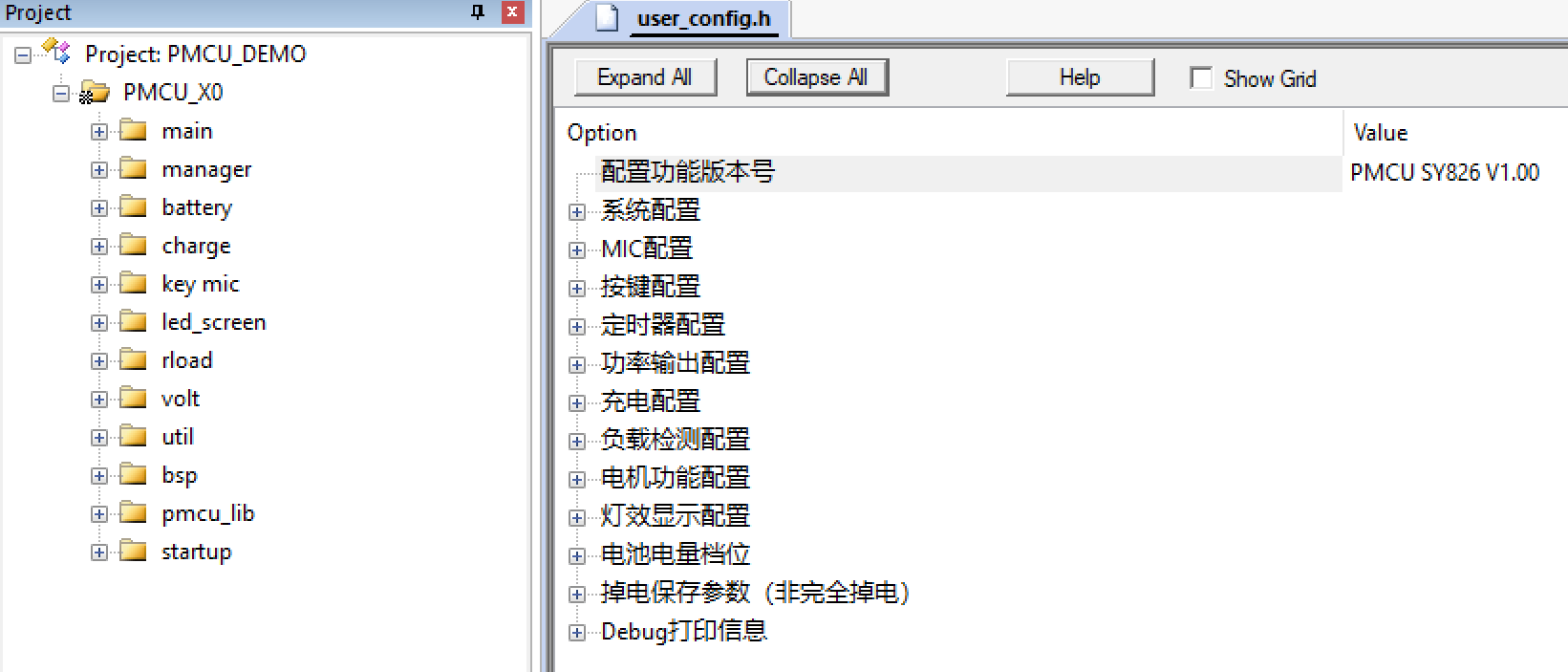
|  |  |
| --- | --- |
| 名称 | 文件内容 |
| Listings | 工程编译中间文件夹 |
| Objects | 工程编译的结果文件夹，编译成功后，生成.hex文件 |
| PMCU\_DEMO.uvproj | 工程文件，双击即可打开Keil工程 |

### 1.1.5 Startup-启动文件



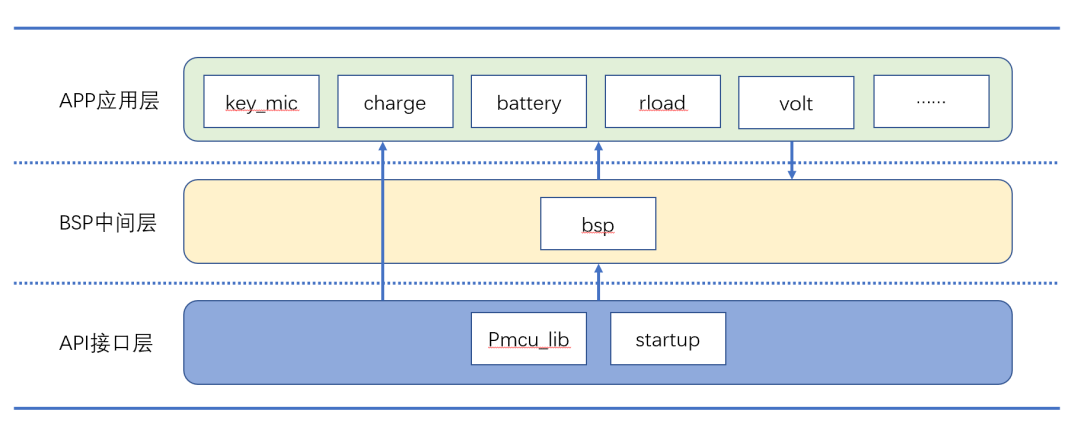
8051启动文件，非必要不需要修改。

# 2 工程结构框架



此代码Demo已实现电子烟方案基本的功能：按键检测、咪头吸烟检测、单发输出、双发输出、恒压输出、恒有效输出、恒功率输出、LED点灯、断码屏功能、单发负载检测、双发负载检测、充电状态管理、电池电量管理、油量计算、各种异常保护、童锁、系统锁机等，基于此框架做部分修改即可适配大部分的市场方案需求。

工程整体的调用关系框架如下：



整个工程应用层，通过一个简单的队列进行消息事件的切换：

涉及的源文件：util\_queue.c/ util\_queue.h

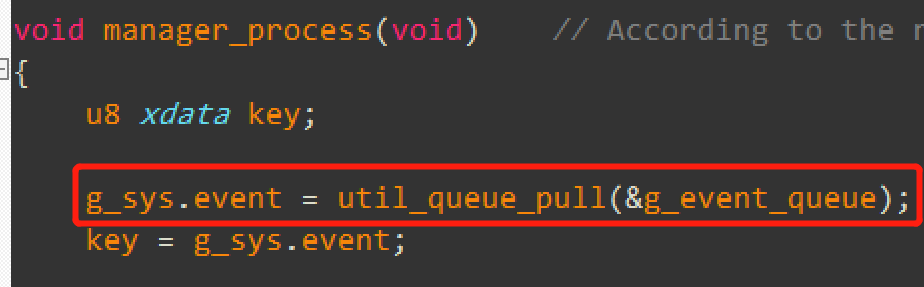
入队：void util\_queue\_push(u8 a)

出队：u8 util\_queue\_pull(struct util\_queue \* queue)

在触发某些特定事件后，使用入队函数，将事件状态更新保存，如按键触发：

util\_queue\_push(EVENT\_KEY\_PRESS);

在while(1)系统轮询中，通过出队函数查询当前最新状态，以实现对应的功能：

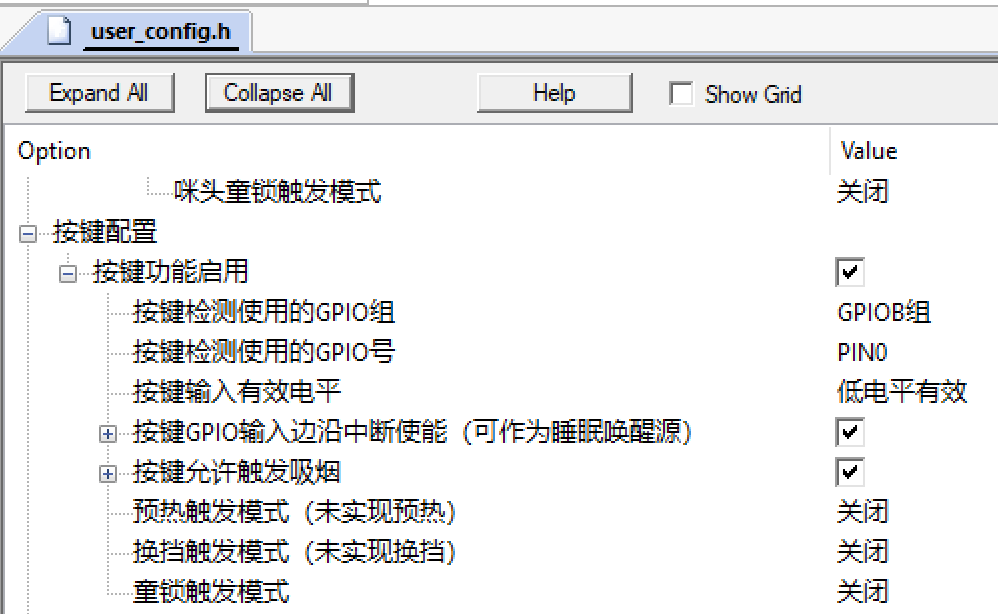
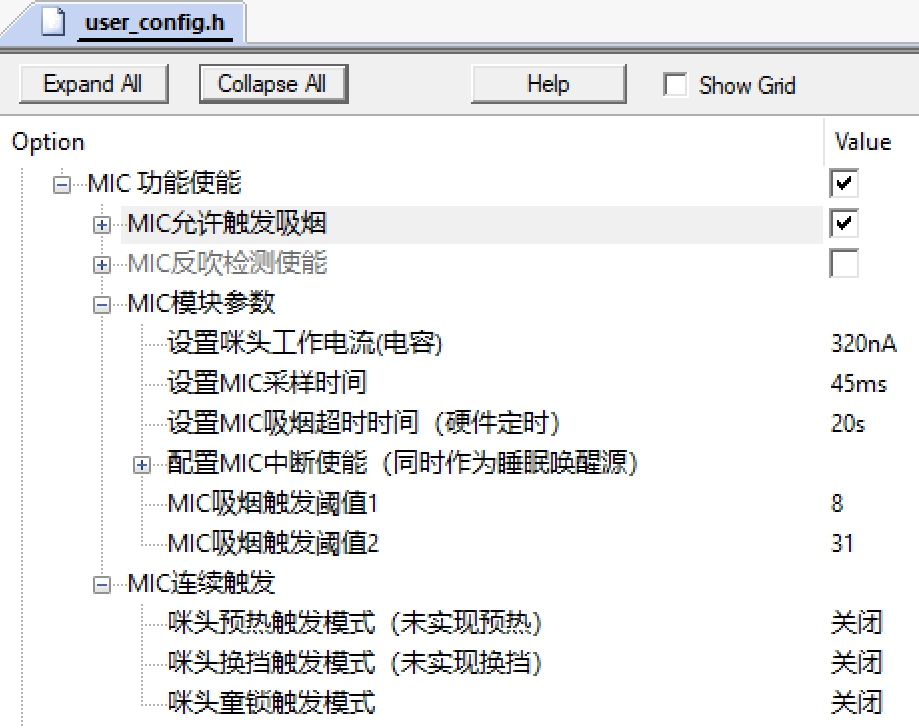


## 2.1咪头吸烟和按键检测

源文件：key\_mic.c/key\_mic.h

已实现按键/咪头单击、双击、连续按3次、连续按4次、连续按5次、长按等功能，可基于此作功能扩展，如3次按键童锁、换挡等；

关键的功能或参数，可通过配置头文件“user\_config.h”配置编译，如下图所示：



## 2.2电池电量管理

源文件：battery.c/battery.h

Demo以实现电池电压的采样，电池电量等级分配，分配为100个等级（0~99），且定义了电量欠压门限、满电门限等，详见配置头文件“user\_config.h”定义：

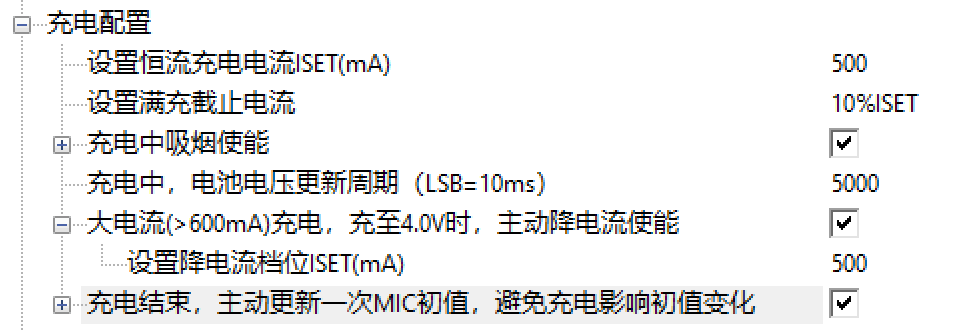


## 2.3充电状态管理

源文件：charger\_usb.c/ charger\_usb.h

由于SY826整个充电过程均由硬件控制，软件只需要查询相应的充电状态，做灯显效果即可，如USB接入、USB拔插、满充、复充等。

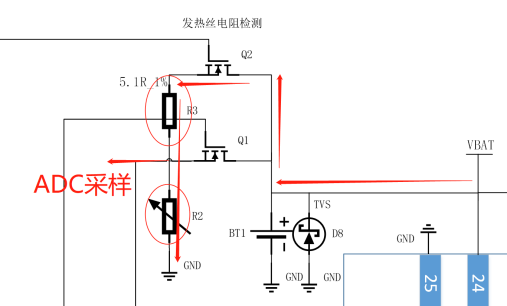
Demo中对整个充电过程的状态，均进行了对应的灯效显示，并对部分特殊功能做了选配编译，如下图：



## 2.4负载检测管理

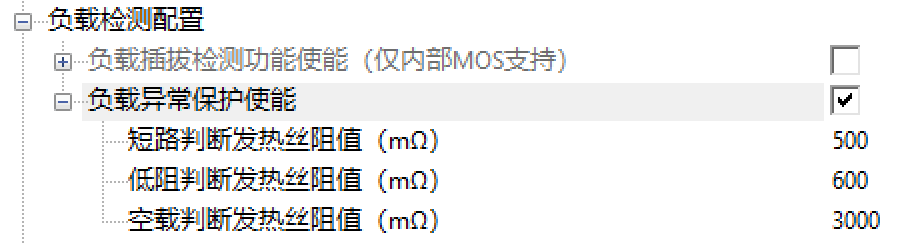
源文件：rl\_det.c/ rl\_det.h

Demo实现了内部MOS的负载接入检测、详细阻值检测、开短路检测；外部MOS的详细阻值检测、开路检测等。其中外部MOS的负载检测原理如下：



通过GPIO控制Q2导通，使VBAT🡪5.1R🡪R2（发热丝）🡪GND，形成导电通路，通过ADC采样R2上的电压和VBAT的电压，通过分压原理换算R2的电阻，即可得出负载的真实电阻。

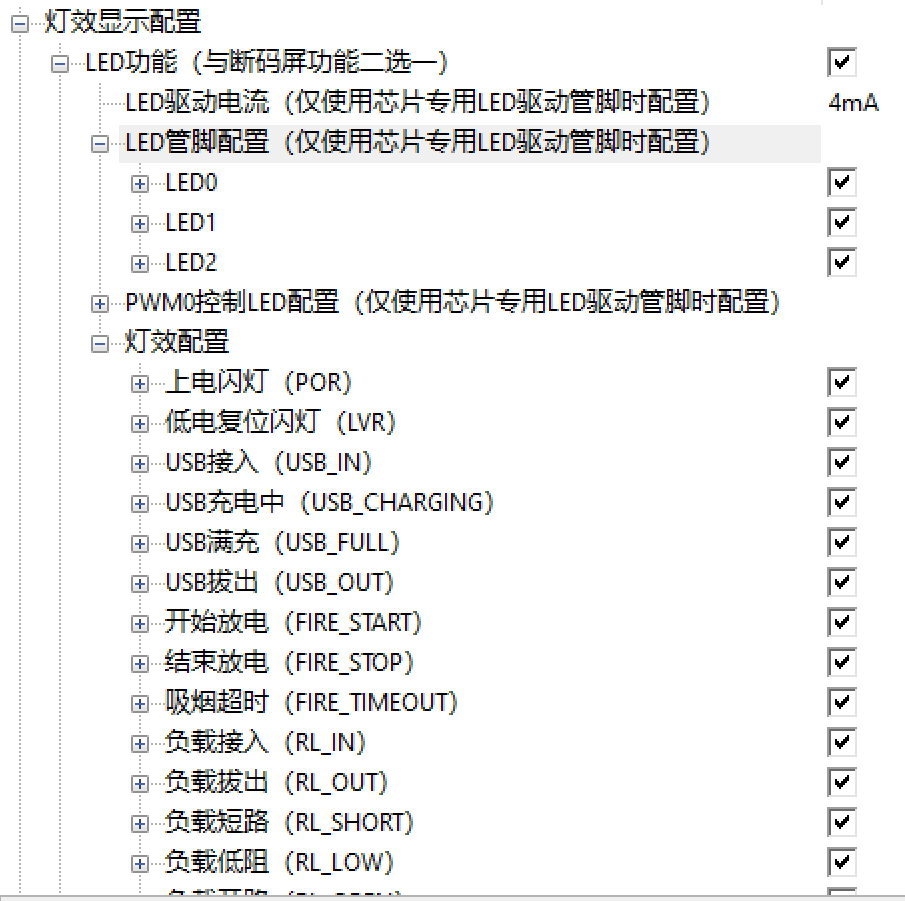
负载检测的相应参数在配置头文件“user\_config.h”中定义如下：



## 2.5灯效管理

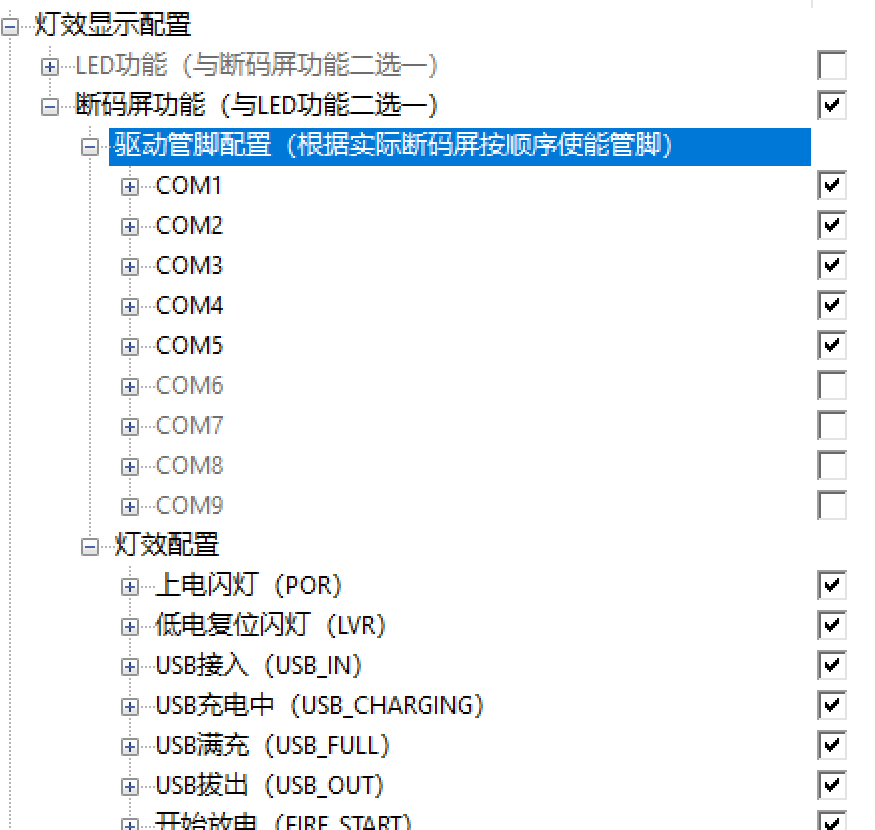
LED控制源文件：

|  |  |
| --- | --- |
| 名称 | 文件内容 |
| led\_ctrl.c/led\_ctrl.h | Led的灯效控制，如闪烁、呼吸、渐亮、渐灭等 |
| led\_show.c/led\_show.h | 系统事件灯效宏定义，如吸烟灯、充电灯、短路灯等 |
| led\_screen\_common.c/ led\_screen\_common.h | Led事件宏定义 |



断码屏控制源文件：

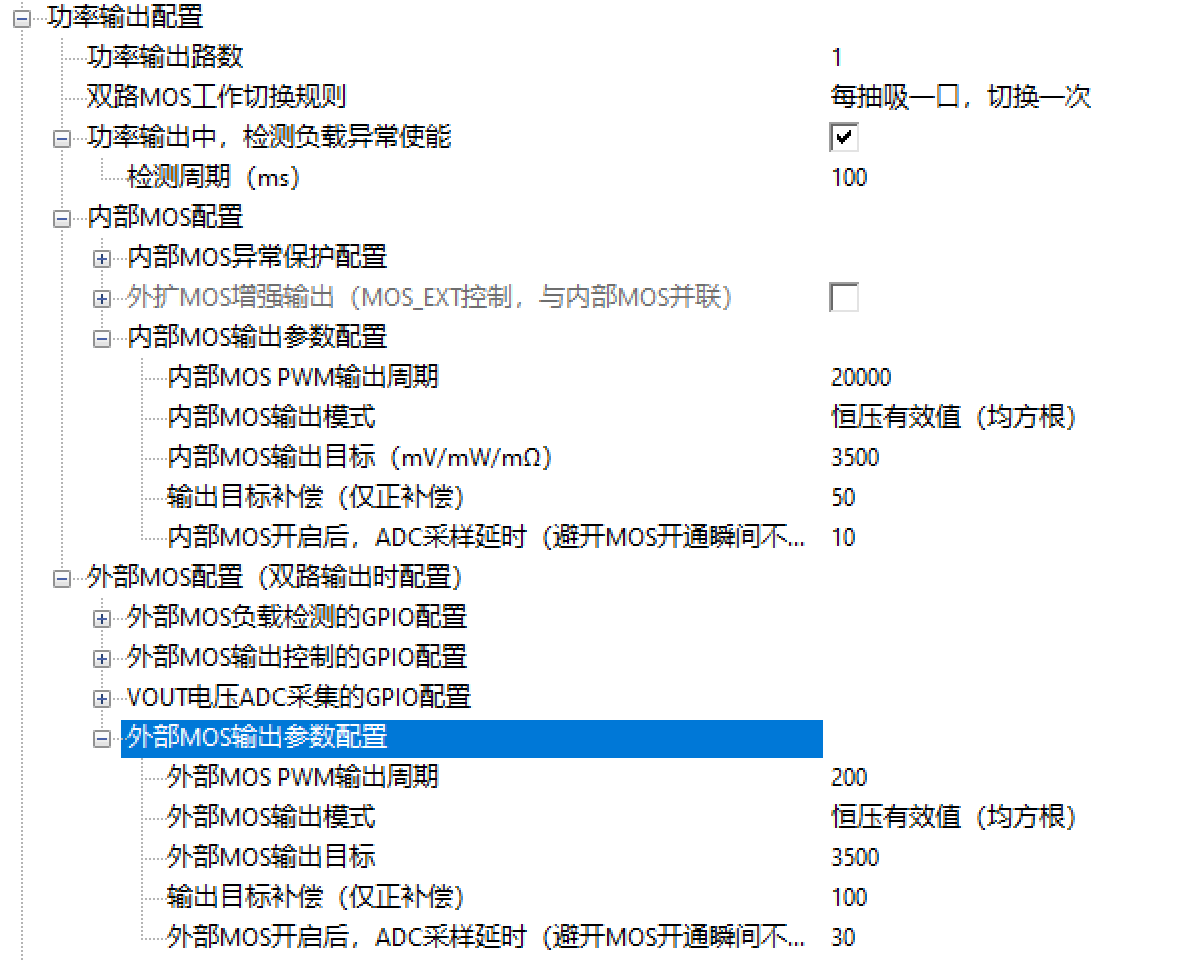
|  |  |
| --- | --- |
| 名称 | 文件内容 |
| screen\_x1\_driver.c/ screen\_x1\_driver.h | 某款特定断码屏点灯驱动，方案中需根据实际断码屏修改驱动 |
| screen\_ctrl\_mode.c/ screen\_ctrl\_mode.h | 断码屏灯效控制，如吸烟灯、充电灯、短路灯、跑马灯等 |
| screen\_show.c/screen\_show.h | 系统事件灯效宏定义，如吸烟灯、充电灯、短路灯等 |



## 2.6功率输出管理

源文件：volt\_out.c/ volt\_out.h

Demo实现了单发/双发MOS的功率输出控制，内部MOS支持恒压、恒有效、恒功率输出；外部MOS支持恒压、恒有效输出，且每路MOS输出电压支持单独配置，其配置参数见配置头文件“user\_config.h”定义：



## 2.7掉电参数保存

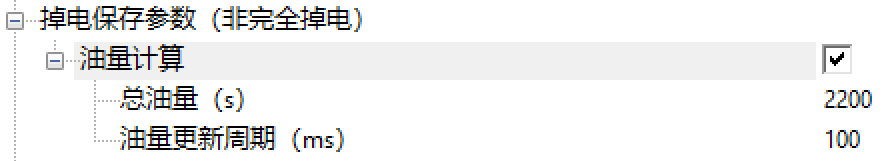
源文件：userdatas.c/ userdatas.h

Demo实现的掉电参数保存，包含了油量计算保存和欠压复位状态保存。

油量计算原则为：已知总油量x秒，通过累计吸烟时的时间来反映剩余油量。

欠压状态为吸烟或其他原因导致芯片欠压复位，系统会保存复位状态，使芯片在复位启动后知道当前启动时正常上电启动，还是欠压导致的复位启动。

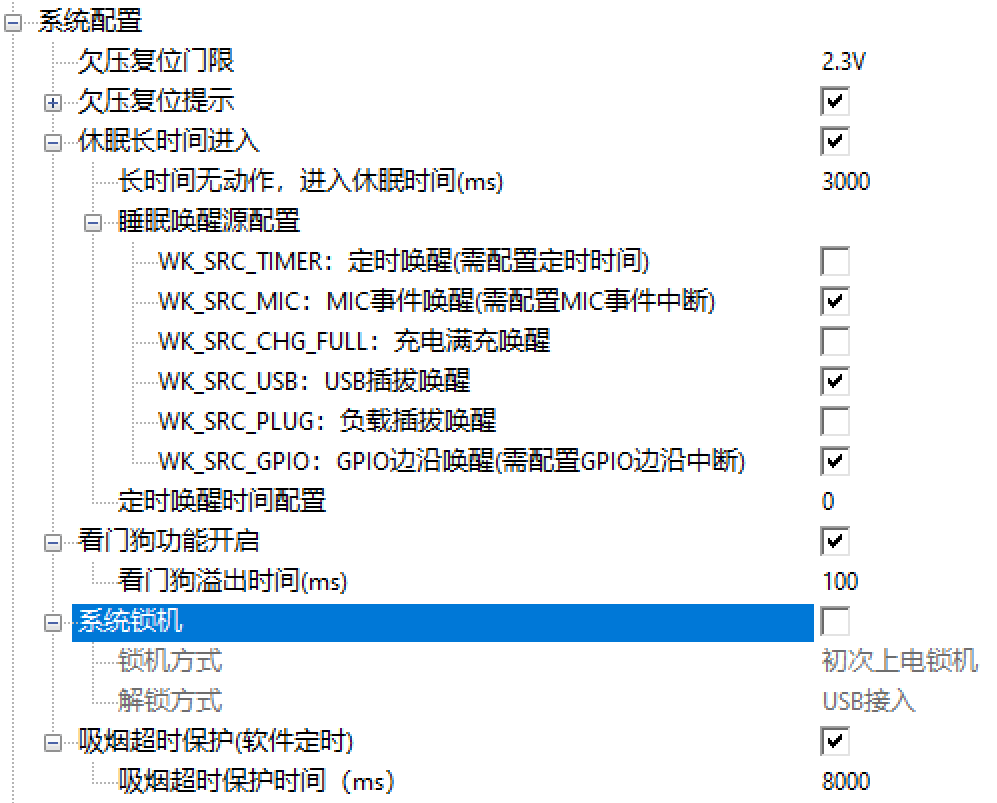
保存参数的原理参考“3.1.6配置参数保存”



## 2.8系统参数配置

源文件：多文件、多模块均涉及

Demo针对系统运行的关键参数作独立配置，便于开发修改，如配置头文件“user\_config.h”定义：

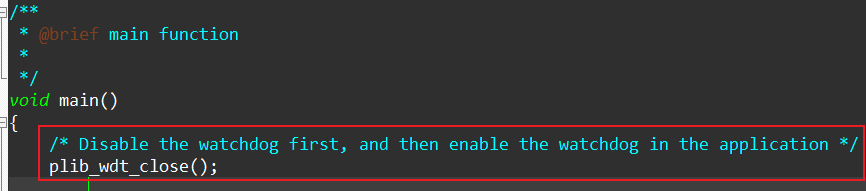


# 3 应用注意事项

## 3.1系统内核相关

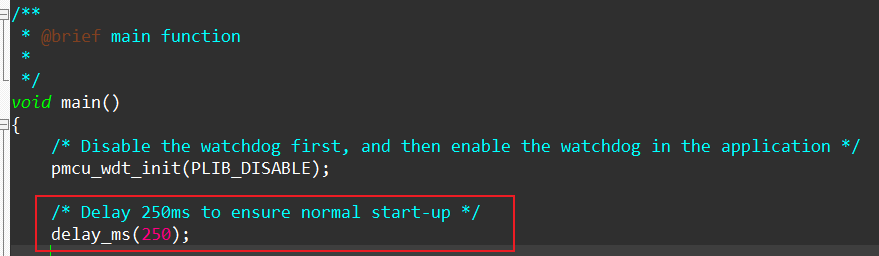
### 3.1.1上电关闭看门狗

为实现后续方案bin固件兼容，系统上电运行需要先关闭看门狗，在初始化时配置看门狗参数后再开启。



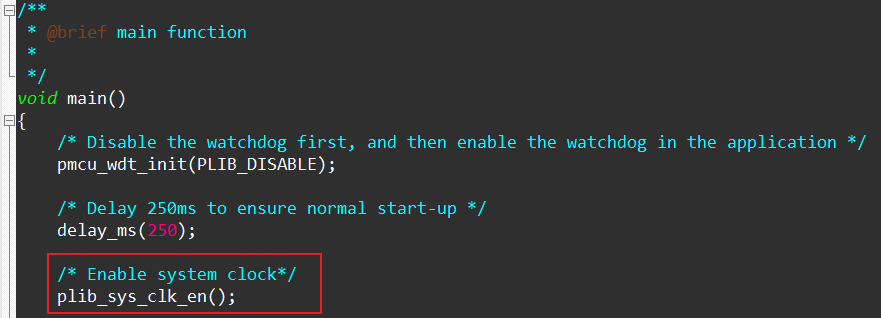
### 3.1.2上电增加延时，确保可烧录

由于芯片GPIO复用关系，芯片烧录需要使用GPIOC0和GPIOC1接口；为避免由于程序中占用了升级接口，导致升级失败，需要在系统上电初始化前增加至少250ms的延时。



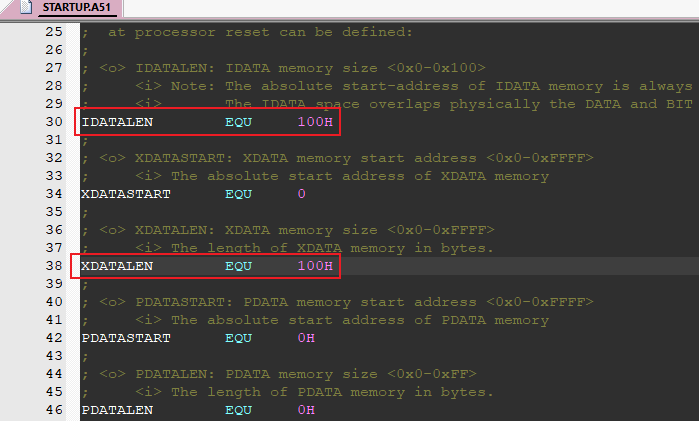
### 3.1.3上电使能系统时钟

为实现后续方案bin固件兼容，系统上电运行需要使能芯片主时钟，以确保系统正常运行。



### 3.1.4 启动文件对SRAM进行清0

由于SRAM具备上电数据随机的特性，可能会影响系统运行中时未初始化的变量，需要在启动脚本对SRAM进行统一清0，包含IRAM和XRAM。其中IRAM空间256Bytes，XRAM空间256Bytes，根据应用需要选择清零的长度。



### 3.1.5 API接口的延时函数

API库实现了两个延时函数：

（1）delay\_ms()：ms级延时函数

（2）delay10us()：10us级延时函数

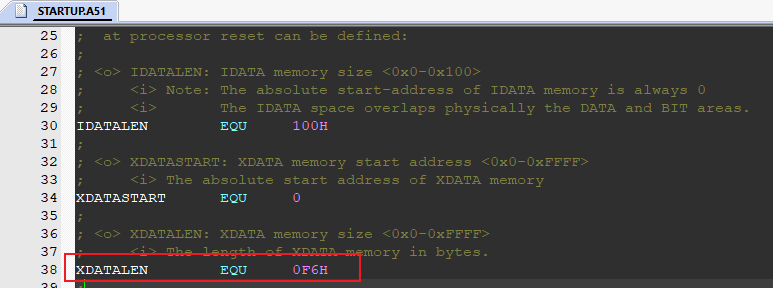
两个延时均为软件延时，存在偏差，对延时精度要求高的，请使用定时器TIM0和TIM1实现。

特别注意：两个延时函数均未对看门狗进行处理，使用这两个延时函数时，需要注意看门狗的溢出时间，避免在延时时导致看门狗溢出。

### 3.1.6 配置参数保存

若方案上有保存配置参数的需求，可利用SRAM的上电随机性和非完全掉电（>1.0V）数据不丢失的特性，即使用SRAM作参数保存，实现方法如下：

（1）在系统启动文件对保存数据的SRAM地址不清0，如使用10Bytes XRAM保存数据，则只对246Bytes进行清零。



（2）通过“xdata”和“\_at\_”关键字定义保存数据的变量，并指定保存地址。



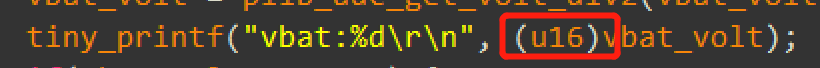
（3）在需要保存参数的位置，对变量进行赋值即可。

特别注意：此方案仅在VBAT电压>1.0V以上有效，完全掉电的场景无法实现。

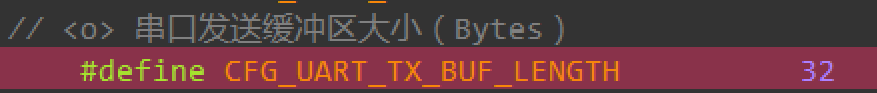
### 3.1.7 固件调试方式

固件开发使用串口打印的形式调试，打印函数的实现在文件：util\_printf.c/util\_printf.h

特别注意：打印函数实现占用约2K Bytes空间，量产时可关闭打印调试功能。打印8bit或16bit数据时，使用“u16”类型转换；打印32bit数据时，使用“u32”类型转换。



由于打印接口使用xram定义打印缓冲空间，最大长度为32，故打印字符长度不能大于32。



否则，可能出现其他xram参数被覆盖的问题，如下：



## 3.2充电相关

### 3.3.1 600mA充电电流以上应用

当充电电流使用600mA以上档位，USB适配器电压小于4.6V（进入欠压自适应，还未到达欠压保护），且VBAT电压充至4.0V时，充电电流可能无法满足±50mA的充电精度要求，充电电流会上升，最大不超过1100mA，但依旧在芯片充电裕量内，具体表现为：

（1）600mA充电，最大充电电流<680mA；

（2）650mA充电，最大充电电流<850mA；

（3）700mA充电，最大充电电流<1100mA；

（4）750mA充电，最大充电电流<1100mA；

若对此条件下的充电电流精度要求较高，对于600mA以上充电电流的应用，可通过软件检测VBAT电压，当VBAT电压高于4.0V时，软件使用API接口：plib\_chg\_set\_cc\_current()降低充电电流档位。

### 3.3.2 充电使能

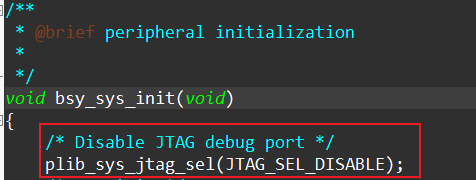
充电功能支持软件关闭充电，API接口：plib\_chg\_en()，软件关闭后，充电立即停止。

特别注意：若此时重新拔插USB，且满足充电条件，硬件会自动开启充电，且把充电使能位复位。

## 3.3 GPIO相关

### 3.3.1 JTAG调试接口复用问题

由于JTAG调试接口和GPIOA/GPIOB组存在复用关系，且默认复用在GPIOB组，为避免JTAG接口占用GPIO，导致GPIO功能受限，需要在系统初始化中，关闭JTAG接口功能，以释放对GPIO的限制。



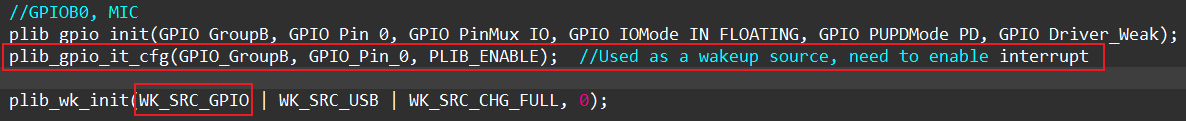
## 3.4 睡眠唤醒相关

### 3.4.1 睡眠唤醒源说明

SY826支持6个唤醒源唤醒，

1. 负载插拔，使能即可；
2. USB插拔，使能即可；
3. 充电满充，使能即可；
4. 定时唤醒，需要配置定时时间，与TIM0和TIM1无关；
5. GPIO双边沿唤醒，需要通过配置GPIO的双边沿中断使能（API接口：plib\_gpio\_it\_cfg()），来选择具体唤醒的GPIO；

如使用GPIOB0作MIC吸烟检测，故需要把GPIOB0作为唤醒源，配置如下：



1. MIC吸烟事件唤醒，需要配置MIC吸烟事件中断使能（API接口：plib\_mic\_it\_cfg()），来选择具体的唤醒事件。

### 3.4.2 睡眠静态功耗处理

芯片静态功耗小于15uA，为了避免系统进入睡眠时，无关模块使能或GPIO配置不对等导致系统静态功耗过高的问题，在系统进入睡眠之前，需要先关闭无关模块（仅保留唤醒源模块）和根据硬件应用配置GPIO的模式。

如，外部无上拉的GPIO，可配置为：输入弱下拉。

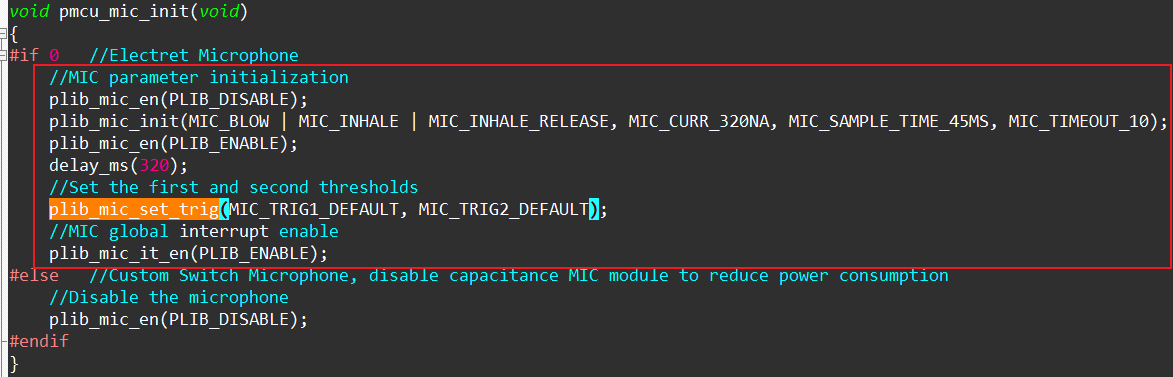
## 3.5 咪头吸烟相关

### 3.5.1 MIC参数配置

咪头模块，系统上电默认使能，且上电至少300ms后，才开始工作。调用API接口：plib\_mic\_init()配置MIC参数时，需要先关闭MIC使能，配置完再开启MIC使能，且需要延时至少300ms；

咪头阈值配置，API接口：plib\_mic\_set\_trig()需要在MIC开始工作后配置才生效（即使能，延时300ms后）。

参考配置如下：



### 3.5.2 电容MIC吸烟阈值导致误触发问题

SY826支持两级阈值配置。

第一阈值：0/1024~31/1024，1/1024/step，默认值为 8/1024

第二阈值：0/1024~255/1024，1/1024/step，默认值为 31/1024

其中第一阈值用于区分吸/吹气微变，即吸气超过第一阈值且小于第二阈值，不会触发吸烟，但维持时间超过4秒时，会更新MIC当前实时值为初值；第二阈值为触发吸烟的阈值门限。

如未吸烟时，MIC的实时计数值和吸烟初值均为1000，当MIC有吸气触发动作时，MIC的实时计数值变为970，则变化比例为：

Pre =（吸烟时实时值-吸烟前实时值）/初值 =（970-1000）/1000 = -3%

可正常触发吸烟事件。

吸烟释放后，MIC的实时计数值从970恢复到原来的值1000时，但未触发初值更新，故吸烟初值依旧为1000，变化比例：

Pre =（释放时实时值-释放前实时值）/初值 = （1000-970）/1000=+3%

可正常触发吸烟释放事件。

**应用中可能存在以下几种异常情况：**

（1）部分MIC吸烟后，释放恢复较慢，如吸烟释放后，MIC的实时计数值从970恢复到原来的值990时，无法及时恢复到1000，变化比例：

Pre =（释放时实时值-释放前实时值）/初值 = （990-970）/1000=+2%

无法触发吸烟释放事件，导致出现持续吸烟直至超时现象。

**针对此情况的解决办法：**

a.选用稳定性更高的MIC，如硅咪；

b.设置合适的吸烟触发阈值；

（2）USB充电时，可能会出现MIC实时值被抬高/拉低的问题，如抬高至1020，超过第一阈值，小于第二阈值，若维持时间超过4秒，MIC的初值更新为1020；此时，在充电过程中，触发吸气动作，MIC的实时计数值变为970，则变化比例为：

Pre =（吸烟时实时值-吸烟前实时值）/初值 =（970-1020）/1020 = -4.9%

可正常触发吸烟事件，且同时关闭充电使能（非边充边放）。

吸烟释放后，MIC的实时计数值从970恢复到原来的值1000时，吸烟初值为1020，变化比例：

Pre =（释放时实时值-释放前实时值）/初值 = （1000-970）/1020=+3.9%

无法触发吸烟释放事件，导致出现持续吸烟直至超时现象。

**针对此情况的解决办法：**

a.设置合适的吸烟触发阈值；

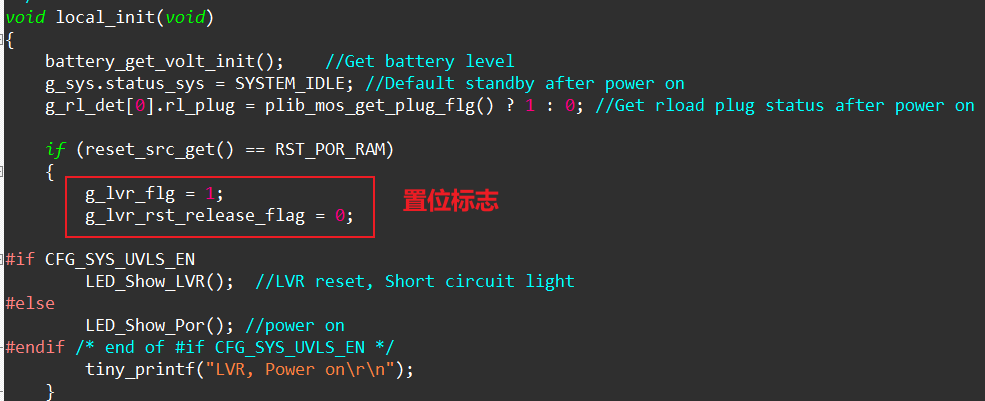
b.在充电结束（满充和USB拔出），无吸烟时，更新一次初值。

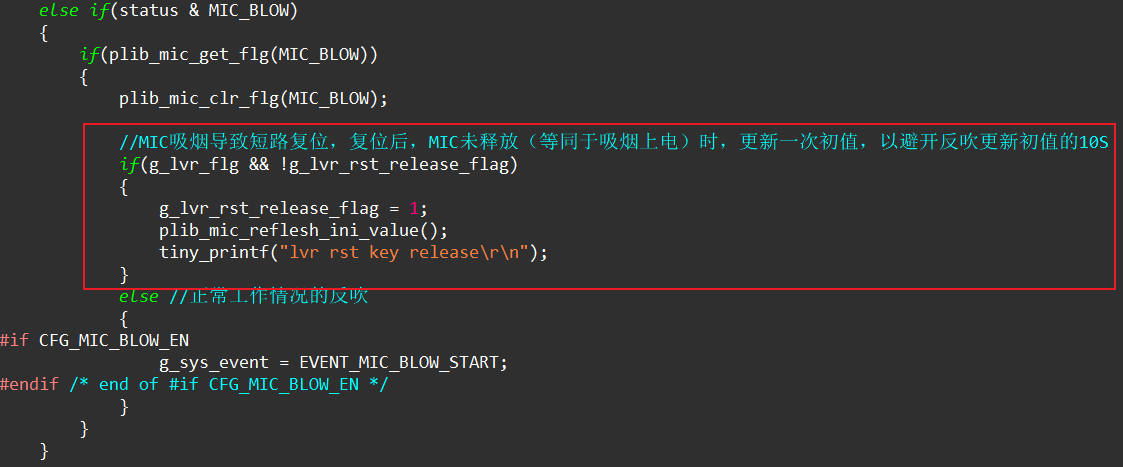
### 3.5.3 吸烟中短路复位，MIC处于吸烟状态，导致芯片复位后

吸烟时，短路电热丝，如果触发芯片复位，等效于：吸着烟，上电，此时MIC的初值=吸烟时的实时值，即初值较低。

如果此时，释放咪头，咪头实时值变大，基本都会超过第2阈值，即触发反吹。触发反吹后，需要超时10S，初值才恢复正常，也就是说这10S内，无法触发吸烟。

故，需要在短路复位后，检测到MIC反吹状态时，更新一次MIC初值。





## 3.6 ADC驱动相关

### 3.6.1 ADC分压系数和电压转换

SY826支持6个ADC通道，分别为通道0（PGA\_OUT）、通道1(VBAT)、通道2(AVCC)；外部占用3个，和GPIO复用，分别为通道3(GPIOB0)、通道4(GPIOB1)、通道5(GPIOB2)。

除了0通道（PGA\_OUT），其他通道ADC分压系数均为2，即作电压转换时，使用API接口：plib\_adc\_get\_volt\_div2()即可。

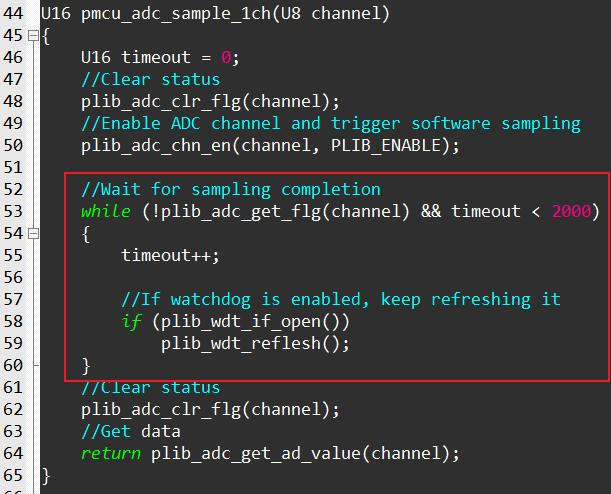
0通道，在采集负载阻抗电压时，分压系数为2，作电压转换时，使用API接口：plib\_adc\_get\_volt\_div2()；在采集VOUT功率输出电压时，分压系数为4，作电压转换时，使用API接口：plib\_adc\_get\_volt\_div4()。

### 3.6.2 ADC触发采样

ADC采样存在两种触发方式：

1. 软件触发，API接口：plib\_adc\_chn\_en()，每使能一次，触发一次采样，多次采样，需要重复使能通道。

使能通道采样后，到ADC采样完成需要一定的时间，不同通道时间不同；故使能后，立即读取ADC结果，有可能读取的结果为上一次采样的结果，为避免此种情况，建议使能采样后，等待ADC的采样完成信号，再读取ADC结果，如bsp\_adc.c文件内的处理方法：



1. 硬件触发，API接口：plib\_adc\_chn\_en()，使能一次采样通道后，开启API接口：plib\_adc\_pwm\_trig\_en()，在每个MOS\_PWM的上升沿均会触发ADC采样，触发ADC采样的延时时间可通过接口：plib\_adc\_init()在初始化时设定。

特别注意：开启硬件触发后，软件触发无效，若需要使用软件触发，需要先关闭硬件触发。

### 3.6.3 GPIOBx作为ADC输入配置

GPIOB0/GPIOB1/GPIOB2均支持复用为ADC输入，参考配置如下：



## 3.7 功率输出相关

### 3.7.1 双MOS同时驱动

SY826支持同时驱动内/外MOS，以增加搭载能力，外扩MOS要求使用NMOS，使用芯片管脚PIN21-MOS\_EXT驱动，驱动信号同内部MOS驱动一致。

特别注意：芯片的保护机制，如OCP/OTP保护，仅针对内部MOS使用，外部MOS需要由外围电路和软件自行保证。

为确保此应用场景工作时，外部MOS先开启，内部MOS后开启；内部MOS先关闭，外部MOS后关闭；必须使用接口plib\_exmos\_init()配置内/外MOS开关同步使能，参考配置如下：



### 3.7.2 负载检测

SY826支持CCO 50mA恒流源和PGA检测负载阻抗，PGA放大倍数支持16倍和32倍配置，

负载检测计算公式为：

VCCO = 50mA \* Rload \* PGA倍数 \* 2（ADC系数）

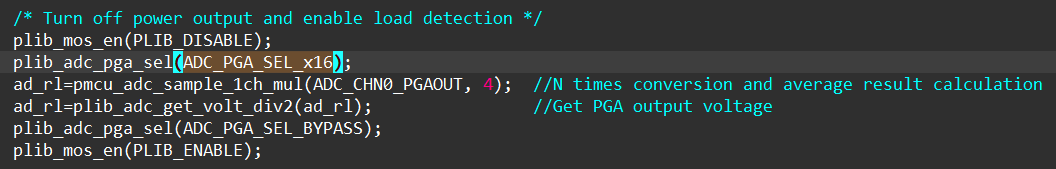
如：负载1欧姆，放大倍数为16倍，则：

VCCO = 50mA \* 1ohm \* 16 \* 2 = 1600mV

负载检测最大检测电压VCCO=4.4V，故PGA 16倍时，支持最大检测负载为3.75欧姆；PGA 32倍时，支持最大检测负载为1.375欧姆。

特别注意：由于负载检测和功率输出均通过VOUT管脚实现。在负载检测时，需要使用API接口：plib\_adc\_pga\_sel()，参数ADC\_PGA\_SEL\_x16或ADC\_PGA\_SEL\_x32开启负载检测；在功率输出时，需要使用API接口：plib\_adc\_pga\_sel()，参数ADC\_PGA\_SEL\_BYPASS开启功率输出电压检测。

负载检测参考配置流程如下：



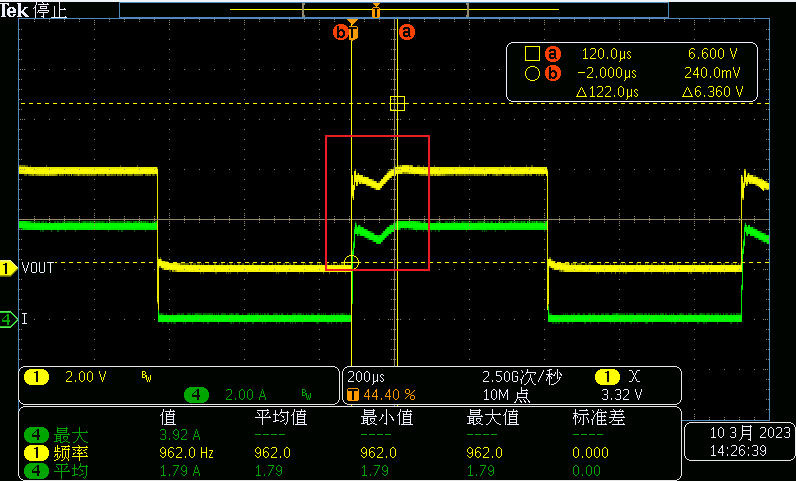
### 3.7.3 OCP保护锁存

功率输出时，负载太小或短路触发OCP过流保护后，硬件会自动关闭功率MOS的输出，且把OCP状态进行锁存。可通过API接口：plib\_mos\_get\_ocp\_lock()获取锁存状态；

软件处理上，在出现OCP异常后，通过API接口：plib\_mos\_get\_abn\_flg()获取OCP状态，且通过接口：plib\_mos\_clr\_abn\_flg()清除OCP状态；确保OCP异常撤销后，至少延时1ms，再通过API接口：plib\_mos\_ocp\_unlock()解锁OCP锁存状态，下次MOS PWM才能正常输出。

### 3.7.4 功率输出时，VOUT采样问题

MOS PWM开启后，MOS导通稳定需要一定的时间，在做输出控制时，如恒压输出，需要软件上处理：在MOS 导通延时一定时间后再启动VOUT的ADC采样，获取输出电压。



## 3.8 LED驱动相关

### 3.8.1 LED驱动配置

SY826支持3路LED专用驱动，驱动电流4/8/21/29mA可选，分别复用在GPIOA1/GPIOA2/GPOA3上，作LED输出时，需要通过GPIO API接口plib\_gpio\_init()将GPIO复用为LED模式，且通过API接口plib\_gpio\_led\_output\_en()使能LED模块。

特别注意：GPIOA1(LED0)使用PWM0的0通道控制亮度调节；GPIOA2(LED1)使用PWM0的1通道控制亮度调节；GPIOA3(LED2)使用PWM0的2通道控制亮度调节。

