涂鸦智能WIFI-SDK 说明

# 版本记录：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 编写/修订说明 | 修订人 | 修订日期 | 备注 |
| 1.0.0 | 创建文档 | 刘康 | 20160630 |  |
| 1.0.1 | 优化部分流程图 | 刘康 | 20160715 |  |
| 1.0.2 | 修改部分输入参数和返回值描述 | 胡赛 | 20160729 |  |
| 1.0.3 | 1. 增加低配网方式设置 2. 更新底层驱动库和烧录 bin 文件 | 刘康 | 20170330 |  |
| 1.0.4 | 1. 设置 wifi 配置接口说明 2. 增加 SDK 目录结构 3. 增加函数说明 | 刘康 | 20180412 |  |

* **目录结构：**

├── app

│ ├── tuya\_common

│ │ └── include

│ │ └── system

│ └── tuya\_user

├── bin

│ └── upgrade

├── extra\_include

├── include

├── ld

├── lib

└── tools

1. 应用文件目录

app/tuya\_user

1. 引用头文件目录

app/tuya\_common/include 和 app/tuya\_common/include/system

1. 编译生成 bin 文件目录

bin/upgrade

# 函数说明：

1. 设置固件标识名和版本回调函数，用于工厂生产固件校验

VOID set\_firmware\_tp(IN OUT CHAR \*firm\_name, IN OUT CHAR \*firm\_ver)

{

strcpy(firm\_name,APP\_BIN\_NAME); strcpy(firm\_ver,USER\_SW\_VER); return;

}

1. 应用初始化回调，可设置 wifi 配置模式

VOID app\_init(VOID)

{

app\_cfg\_set(WCM\_OLD,NULL);

}

1. gpio 测试回调函数

BOOL gpio\_func\_test(VOID)

{

return TRUE;

}

1. 应用入口函数

OPERATE\_RET device\_init(VOID)

{

//框架初始化

//PSM 扇区注册

//定时器、按键功能创建等

}

# 框架基本接口(接入涂鸦云需使用)

* + 1. **获取 sdk 版本号**

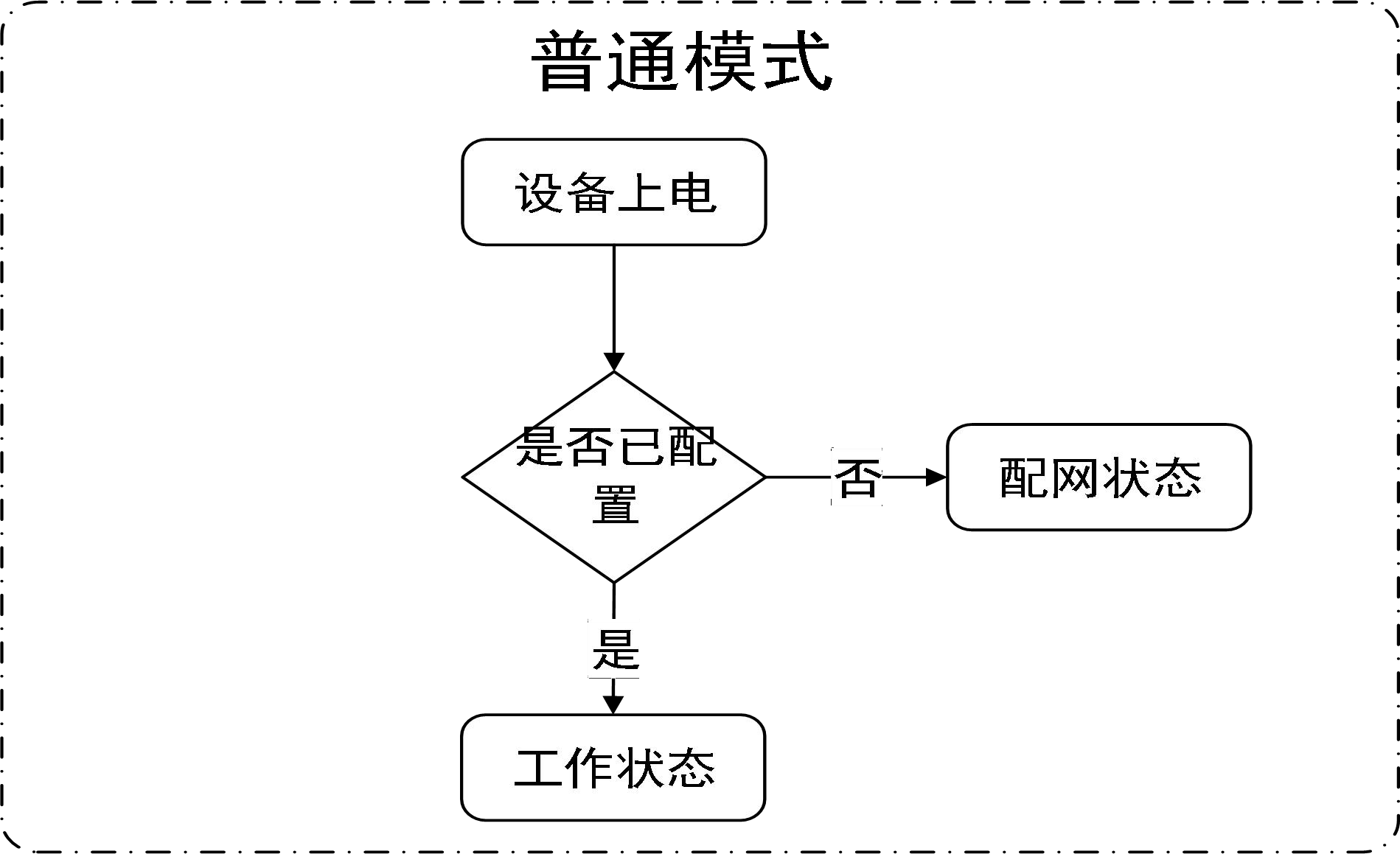
|  |  |
| --- | --- |
| **函数原型** | CHAR \*tuya\_get\_sdk\_ver(VOID) |
| **功能描述** | 获取 sdk 版本号 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | sdk 版本号,如"1.0.0" |
| **备注** | 无 |

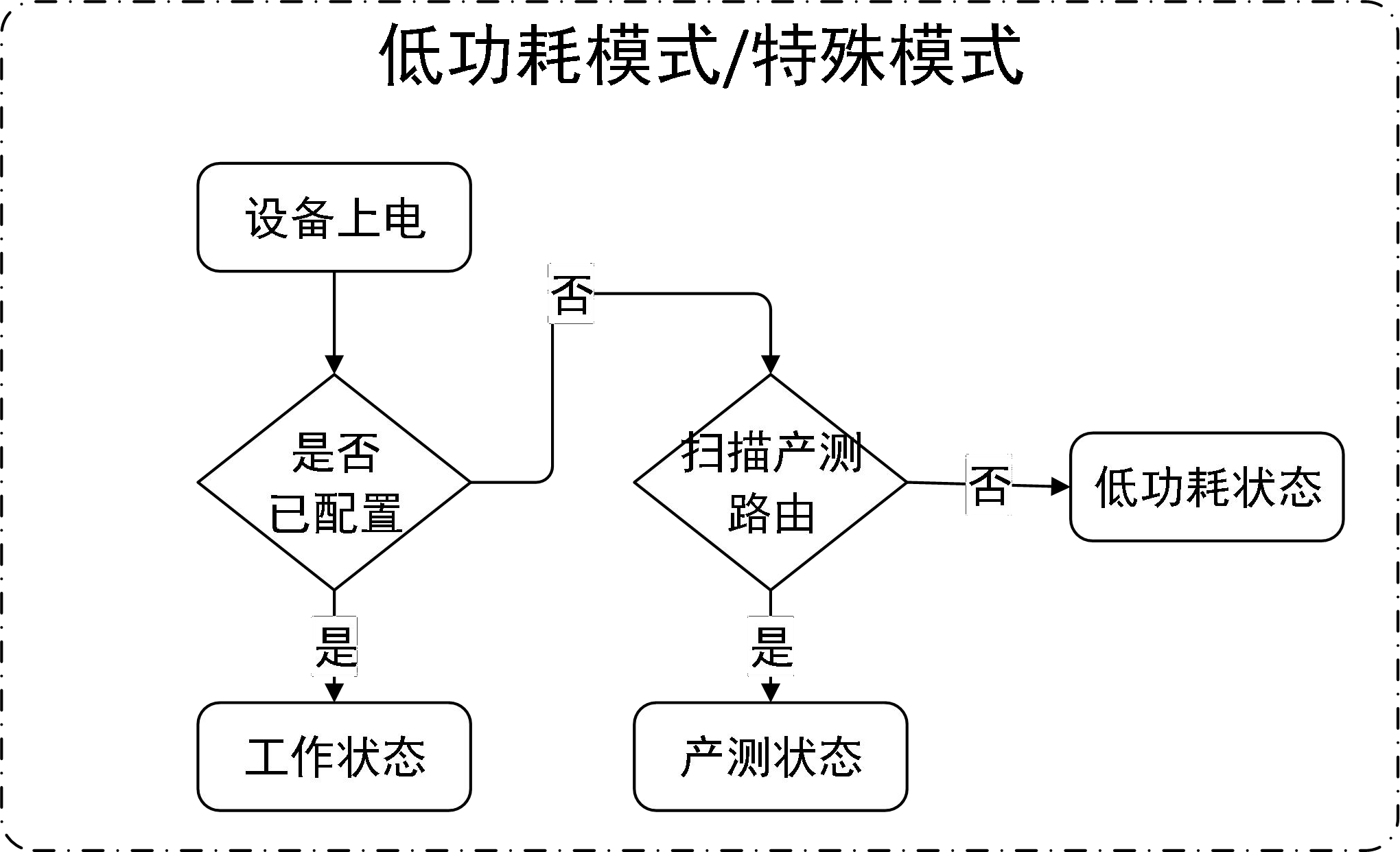
* + 1. **设置 wifi 配置模式**

|  |  |
| --- | --- |
| **函数原型** | VOID app\_cfg\_set(IN CONST WF\_CFG\_MTHD\_SEL mthd, APP\_PROD\_CB callback) |
| **功能描述** | 设置 wifi 配置模式 |
| **输入参数** | mthd  WCM\_OLD 普通模式(do not have low power) WCM\_LOW\_POWER 低 功 耗 模 式 (with low power) WCM\_SPCL\_MODE 特殊配网模式(special with low power) WCM\_OLD\_CPT 兼容模式 |

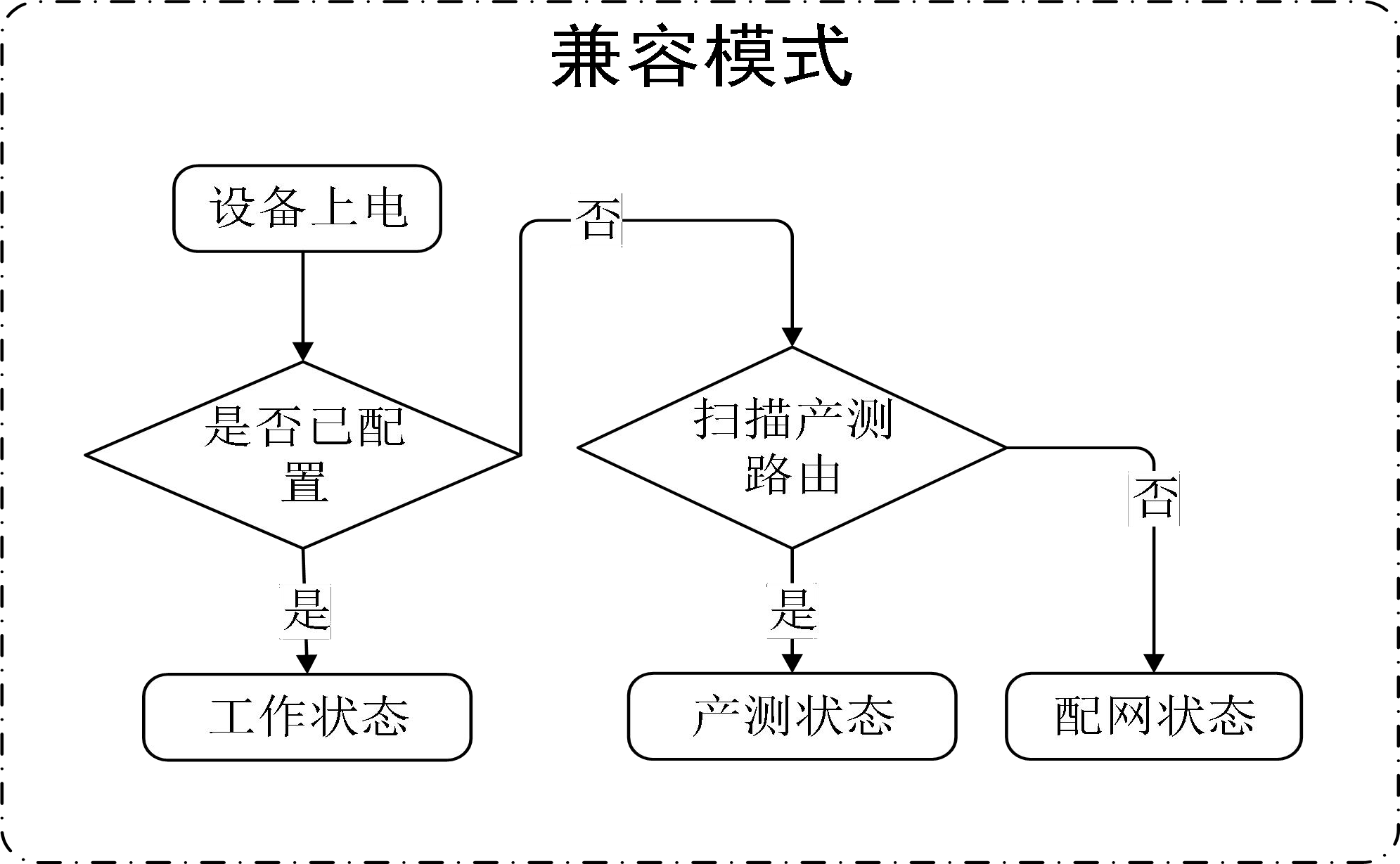
|  |  |
| --- | --- |
|  | callback  产测回调函数，说明：   1. 普通模式，无产测回调，调用方法   app\_cfg\_set(WCM\_OLD, NULL);   1. 低功耗模式，必须设置产测回调   app\_cfg\_set(WCM\_LOW\_POWER, prod\_test);   1. 特殊配网模式，必须设置产测回调   app\_cfg\_set(WCM\_SPCL\_MODE, prod\_test);   1. 兼容模式   app\_cfg\_set(WCM\_OLD\_CPT, prod\_test); |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 默认普通模式,必须在 app\_init 中调用 |

各种工作模式状态切换流程：

1. 普通模式
2. 低功耗模式/ 特殊配网模式



1. 兼容模式



# 设备初始化

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_device\_init(IN CONST CHAR \*product\_id,  IN CONST SMART\_FRAME\_CB cb,CONST CHAR \*app\_ver); |
| **功能描述** | 注册数据处理函数和应用版本号 |
| **输入参数** | product\_id 产品 ID  cb 手机 App 命令回调函数指针,  VOID (\*)(SMART\_CMD\_E cmd,cJSON \*root)  <1> cmd 命令类型   1. 表示局域网下发的命令 2. 表示外网下发的命令   <2> root 命令数据  例，{"1":100,"2":200}，其中"1"和"2"为数据 ID (dpid)编号,100 和 200 为对应 dpid  的值  app\_ver 应用版本号，如"1.0.0" |
| **输出参数** | 无 |
| **返回值** | 详见返回值列表 |
| **备注** | 初始化设备，注册数据处理函数 |

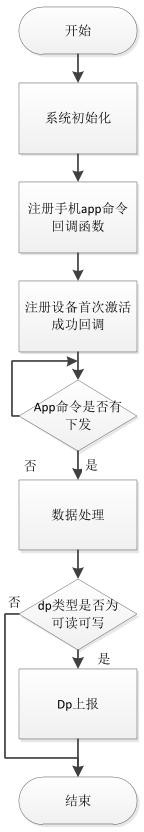


图 1-1 设备初始化流程

# 设备首次激活成功回调

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_active\_reg(IN CONST SYN\_DATA\_CB callback) |

|  |  |
| --- | --- |
| **功能描述** | 注册产品功能 |
| **输入参数** | callback 回调函数(用于同步设备状态)  回调函数定义 typedef VOID(\*SYN\_DATA\_CB)(VOID); |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 设备首次激活会调用 callback |

* + 1. **获取设备 ID**

|  |  |
| --- | --- |
| **函数原型** | CHAR \*tuya\_get\_devid(VOID) |
| **功能描述** | 获取设备 ID |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | 设备 ID |
| **备注** | 无 |

* + 1. **数据上报**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_obj\_dp\_report(IN CONST CHAR \*data) |
| **功能描述** | 数据上报 |
| **输入参数** | data 上报的数据,例{"1":100,"2":200} |
| **输出参数** | 无 |
| **返回值** | 参照返回值列表 |
| **备注** | 调用此接口固件会保存各 DP 的数据状态，如再次上传的 DP 数据与保存的状态  相同则忽略上传，推荐使用该函数，可使 APP、云端、固件三方性能最佳 |

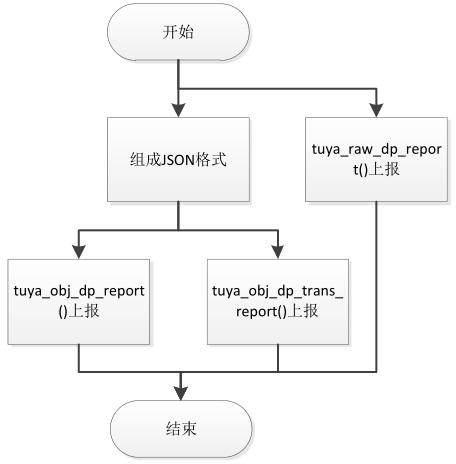


图 1-2 数据上报流程

# 数据上报(透传)

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_obj\_dp\_trans\_report(IN CONST CHAR \*data) |
| **功能描述** | 数据上报(透传) |
| **输入参数** | data 上报的数据,例{"1":100,"2":200} |
| **输出参数** | 无 |
| **返回值** | 详见返回值列表 |
| **备注** | 数据透传到服务器，固件内部不做状态处理 |

* + 1. **RAW 数据上报**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_raw\_dp\_report(IN CONST BYTE dpid,IN CONST BYTE \*data, IN  CONST UINT len) |
| **功能描述** | RAW 数据上报 |
| **输入参数** | dpid 功能点 dp 序号  data 原始二进制数据  len 数据长度 |
| **输出参数** | 无 |

|  |  |
| --- | --- |
| **返回值** | 详见返回值列表 |
| **备注** | 数据透传到服务器，固件内部不做状态处理 |

* + 1. **恢复出厂设置**

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_dev\_reset\_factory(VOID) |
| **功能描述** | 恢复出厂设置 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 清除配网和设备信息  <1> 设备已激活，调用会将设备重置成 smartconfig 配网状态并清除激活信息  <2> 设备未激活，重复调用该函数会导致设备在 smartconfig、ap 配网状态来回切换 |

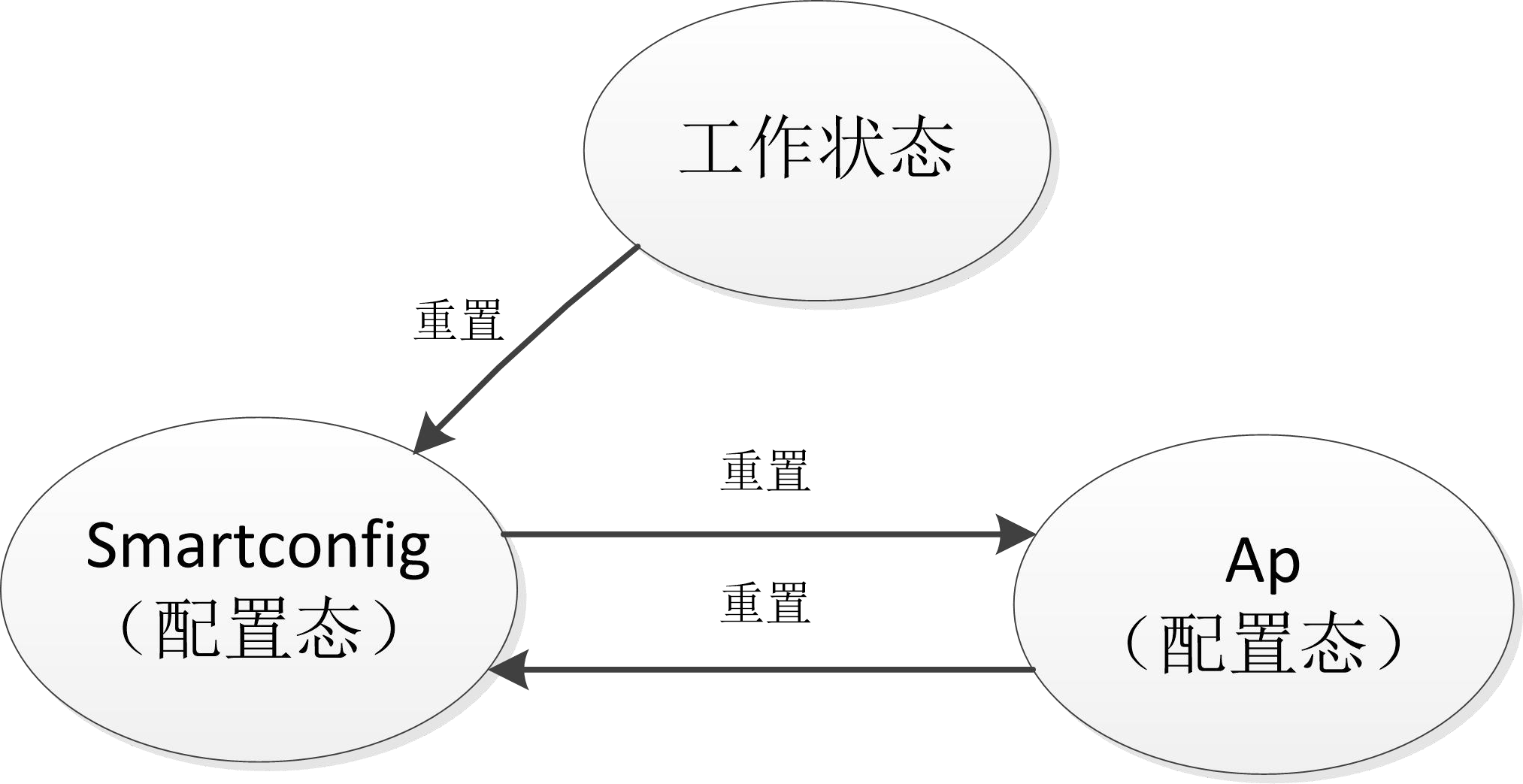


图 1-3 WiFi 状态切换图

# 恢复出厂设置并切换到指定状态

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_dev\_reset\_select(NW\_CFG\_MODE\_E mode) |
| **功能描述** | 恢复出厂设置并切换到指定状态 |
| **输入参数** | mode 配置状态  <1> NW\_SMART\_CFG EZ 状态  <2> NW\_AP\_CFG AP 状态 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 清除配网和设备信息 |

* + 1. **获取设备状态**

|  |  |
| --- | --- |
| **函数原型** | GW\_STAT\_E tuya\_get\_gw\_status(VOID) |
| **功能描述** | 获取设备状态 |
| **输入参数** | 无 |

|  |  |
| --- | --- |
| **输出参数** | 无 |
| **返回值** | typedef enum {  UN\_INIT = 0, // 未初始化，比如生产信息未写入  PROD\_TEST, // 产品产测模式UN\_ACTIVE, // 未激活ACTIVE\_RD, // 激活就绪态STAT\_WORK, // 正常工作态  }GW\_STAT\_E; |
| **备注** | 无 |

* + 1. **获取 WIFI 工作状态**

|  |  |
| --- | --- |
| **函数原型** | GW\_WIFI\_STAT\_E tuya\_get\_wf\_status(VOID) |
| **功能描述** | 获取 WIFI 工作状态 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | typedef enum {  STAT\_LOW\_POWER = 0, //低功耗状态(射频处于关闭状态) STAT\_UNPROVISION = 0, //EZ 状态(未配置)  STAT\_AP\_STA\_UNCONN, //AP 状 态 ( 未 配 置 ) STAT\_AP\_STA\_CFG\_UNC, //AP 和 STA 混合,STA 未连接状态(保留) STAT\_AP\_STA\_CONN, //AP 和 STA 混合,STA 已连接状态(保留) STAT\_STA\_UNCONN, //STA 未连接状态  STAT\_STA\_CONN, //STA 已连接状态  }GW\_WIFI\_STAT\_E; |
| **备注** | 无 |

* + 1. **获取 WIFI 信号强度**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_get\_wf\_rssi(IN UCHAR \*ssid, OUT CHAR \*rssi) |
| **功能描述** | 获取 WIFI 信号强度 |
| **输入参数** | ssid 热点名称 |
| **输出参数** | rssi 信号强度(单位 dbm) |
| **返回值** | 详见返回值列表 |
| **备注** | 扫描过程会阻塞，请勿在初始化函数中使用 |

* + 1. **获取云连接状态**

|  |  |
| --- | --- |
| **函数原型** | BOOL tuya\_get\_cloud\_stat(VOID) |

|  |  |
| --- | --- |
| **功能描述** | 获取云连接状态 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | TRUE 已连接 FALSE 未连接 |
| **备注** | 无 |

* + 1. **获取设备升级状态**

|  |  |
| --- | --- |
| **函数原型** | BOOL tuya\_get\_ug\_stat(VOID) |
| **功能描述** | 获取设备升级状态 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | TRUE 升级中 FALSE 未升级 |
| **备注** | 无 |

* + 1. **获取本地时间**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_get\_local\_time(OUT struct tm \*st\_time) |
| **功能描述** | 获取本地时间 |
| **输入参数** | 无 |
| **输出参数** | st\_time 时间结构体 |
| **返回值** | 参照返回值列表 |
| **备注** | 设备激活后才能使用 |

* + 1. **注册 PSM 模块名称和扇区名称**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_psm\_register\_module(IN CONST CHAR \*module\_name,  IN CONST CHAR \*partition\_key) |
| **功能描述** | 注册 PSM 模块名称和扇区名称 |
| **输入参数** | module\_name 模块名称  partition\_key 扇区名称 |
| **输出参数** | 无 |
| **返回值** | OPRT\_PSM\_E\_EXIST 扇区已注册, OPRT\_OK 成功，其它为失败 |
| **备注** | 每个扇区最多保存 4K 数据 |

* + 1. **写入 PSM 数据**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_psm\_set\_single(IN CONST CHAR \*module,IN CONST CHAR  \*variable,IN CONST CHAR \*value) |
| **功能描述** | 写入 PSM 数据 |
| **输入参数** | module\_name 模块名称  variable 变量名  value 变量值 |
| **输出参数** | 无 |
| **返回值** | 参照返回值列表 |
| **备注** | 只支持字符串类型 |

* + 1. **读取 PSM 数据**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_psm\_get\_single(IN CONST CHAR \*module,  IN CONST CHAR \*variable,OUT CHAR \*value,IN CONST unsigned max\_len) |
| **功能描述** | 读取 PSM 数据 |
| **输入参数** | module\_name 模块名称  variable 变量名  max\_len 缓冲区大小 |
| **输出参数** | value 变量值 |
| **返回值** | 参照返回值列表 |
| **备注** | 只支持字符串类型 |

1. **按键接口**

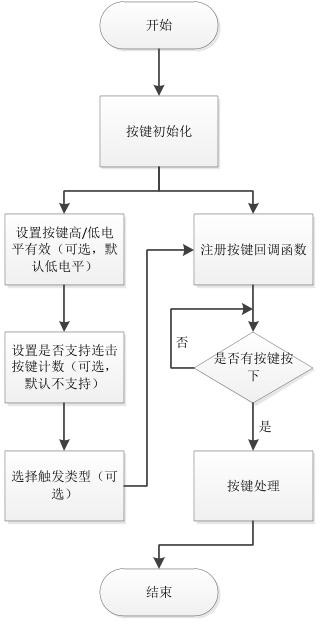


图 2-1 按键使用流程

# 按键初始化

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_kb\_init(VOID) |
| **功能描述** | 按键初始化 |
| **输入参数** | 无 |

|  |  |
| --- | --- |
| **输出参数** | 无 |
| **返回值** | 参照返回值列表 |
| **备注** | 无 |

* + 1. **设置是否支持连击按键计数(即 SEQ\_KEY 类型)**

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_set\_kb\_seq\_enable(IN BOOL enable) |
| **功能描述** | 设置是否支持连击按键计数即 SEQ\_KEY 类型 |
| **输入参数** | enable  <1> TRUE 支持连击按键计数事件，可触发 SEQ\_KEY 类型事件  <2> FALSE 不支持连击按键计数，仅支持触发 NORMAL\_KEY、LONG\_KEY 事件 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 如果不调用该接口则默认所有的按键处理均支持连击按键计数功能 |

* + 1. **设置按键高电平有效**

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_set\_kb\_detect\_high\_valid(BOOL is\_high) |
| **功能描述** | 设置按键高电平有效 |
| **输入参数** | is\_high  <1> TRUE 高电平有效  <2> FALSE 低电平有效 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 无 |

* + 1. **设置按键触发类型**

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_set\_kb\_trig\_type(IN CONST INT gpio\_no,  IN CONST KEY\_TRIGGER\_TP\_E trig\_ty,IN CONST BOOL down\_trig\_cont) |
| **功能描述** | 按键初始化 |
| **输入参数** | gpio\_no:0-16 分别对应 IO0-IO16 trig\_ty:  <1> KEY\_UP\_TRIG 弹起触发  <2> KEY\_DOWN\_TRIG 按下触发，仅支持 NORMAL\_KEY 以及 LONG\_KEY down\_trig\_cont:  <1> 仅当 trig\_ty == KEY\_DOWN\_TRIG 时有效  <2> TURE 如果用户一直按着不放，则间隔 400ms 会再次出发 NORMAL\_KEY 事件  <3> FALSE 用户一直按着不放，仅触发一次 |

|  |  |
| --- | --- |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 默认如果不调用此函数，则按键事件均为弹起触发类型 |

* + 1. **注册按键回调函数**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_kb\_reg\_proc(IN CONST INT gpio\_no,  IN CONST INT long\_key\_time,IN CONST KEY\_CALLBACK call\_back); |
| **功能描述** | 注册按键回调函数 |
| **输入参数** | gpio\_no:0-16 分别对应 IO0-IO16  long\_key\_time:长按键触发时间(ms)，如设置为 0 则屏蔽长按键  call\_back 按键回调函数,  VOID(\*)(INT gpio\_no,PUSH\_KEY\_TYPE\_E type,INT cnt) gpio\_no GPIO 序号  type 按键类型  <1> NORMAL\_KEY 普通按键  <2> SEQ\_KEY  说明:连击按键,仅对弹起触发的按键类型有效，比如快速按下两次回调参数 type  == SEQ\_KEY,cnt == 2(cnt 表示连续点击次数)  <3> LONG\_KEY 长按按键  cnt 按键次数 |
| **输出参数** | 无 |
| **返回值** | 参照返回值列表 |
| **备注** | 无 |

1. **LED 接口**

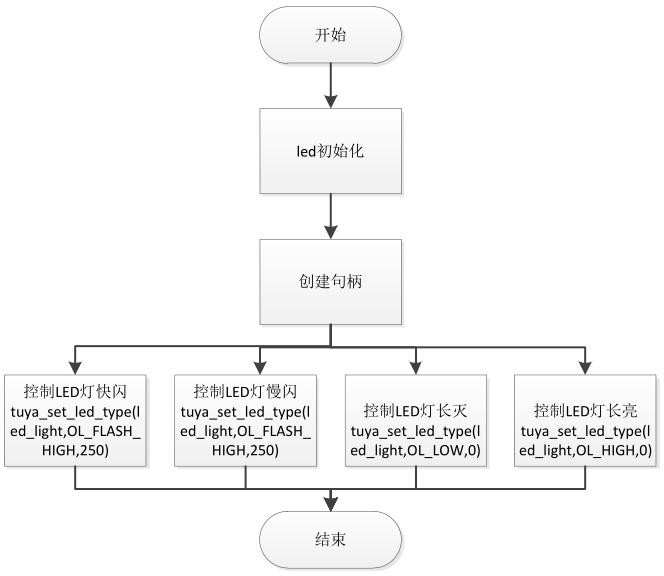


图 3-1 LED 使用流程

# 创建 LED 控制句柄

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_create\_led\_handle(IN CONST INT gpio\_no,  OUT LED\_HANDLE \*handle) |
| **功能描述** | 创建 LED 控制句柄 |
| **输入参数** | gpio\_no GPIO 序号 |
| **输出参数** | handle LED 句柄 |
| **返回值** | 参照返回值列表 |
| **备注** | 无 |

* + 1. **LED 控制**

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_set\_led\_type(IN CONST LED\_HANDLE handle,IN CONST LED\_LT\_E type,  IN CONST USHORT flh\_mstime) |
| **功能描述** | 按键初始化 |
| **输入参数** | handle LED 句柄 |

|  |  |
| --- | --- |
|  | type 控制类型  <1> OL\_LOW LED 低电平  <2> OL\_HIGH LED 高电平  <3> OL\_FLASH\_LOW LED 低电平闪烁  <4> OL\_FLASH\_HIGH LED 高电平闪烁  flh\_mstime 闪烁间隔时间 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 无 |

1. **UART 接口**



图 4-1 串口使用流程

# 打印串口号设置

|  |  |
| --- | --- |
| **函数原型** | void print\_port\_init(UART\_Port uart\_no) |
| **功能描述** | 打印串口号设置 |
| **输入参数** | uart\_no 串口号 |
| **输出参数** | 无 |

|  |  |
| --- | --- |
| **返回值** | 无 |
| **备注** | 默认打印波特率 74880  如 uart\_io==UART1，则打印信息由 IO2 端口输出，tysdk 默认所有的打印信息由  IO2 端口输出，用户在编写应用代码时可通过该接口重配置 |

* + 1. **打印串口参数设置**

|  |  |
| --- | --- |
| **函数原型** | void print\_port\_full\_init(UART\_Port uart\_no, UART\_BautRate bit\_rate,  UART\_WordLength data\_bits, UART\_ParityMode parity,UART\_StopBits stop\_bits); |
| **功能描述** | 打印串口参数设置 |
| **输入参数** | uart\_no 串口号  bit\_rate 波特率(300-3686400) data\_bits 数据位  parity 奇偶校验位  stop\_bits 停止位 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 如 uart\_io==UART1，则打印信息由 IO2 端口输出，tysdk 默认所有的打印信息由  IO2 端口输出，用户在编写应用代码时可通过该接口重配置 |

* + 1. **读取 GPIO 电平**

|  |  |
| --- | --- |
| **函数原型** | INT tuya\_read\_gpio\_level(USHORT gpio\_no) |
| **功能描述** | 读取 GPIO 电平 |
| **输入参数** | gpio\_no:0-16 分别对应 IO0-IO16 |
| **输出参数** | 无 |
| **返回值** | 0 低电平 1 高电平 |
| **备注** | 无 |

* + 1. **UART0 串口初始化配置**

|  |  |
| --- | --- |
| **函数原型** | void user\_uart\_full\_init(UART\_BautRate bit\_rate,UART\_WordLength data\_bits,\  UART\_ParityMode parity,UART\_StopBits stop\_bits); |
| **功能描述** | UART0 串口初始化配置 |
| **输入参数** | bit\_rate 波特率(300-3686400) data\_bits 数据位  parity 奇偶校验位  stop\_bits 停止位 |
| **输出参数** | 无 |
| **返回值** | 无 |

|  |  |
| --- | --- |
| **备注** | 如果采用该接口初始化 UART0，I015 电路需做特殊处理，具体可见乐鑫原厂文档。采用该端口进行串口通讯的好处是，可以避免 8266 启动后默认信息输出对  用户控制板的数据干扰 |

* + 1. **UART0 串口初始化配置(默认 UART0)**

|  |  |
| --- | --- |
| **函数原型** | void user\_uart\_raw\_full\_init(UART\_BautRate bit\_rate,UART\_WordLength data\_bits,\  UART\_ParityMode parity,UART\_StopBits stop\_bits); |
| **功能描述** | UART0 串口初始化配置(默认 UART0) |
| **输入参数** | bit\_rate 波特率(300-3686400) data\_bits 数据位  parity 奇偶校验位  stop\_bits 停止位 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | esp8266 默认 UART0 的收发 IO 端口 |

* + 1. **UART0 串口初始化（esp8266 的收发端口设置为 I015:TX IO13:RX）**

|  |  |
| --- | --- |
| **函数原型** | void user\_uart\_init(UART\_BautRate bit\_rate) |
| **功能描述** | UART0 串口初始化（esp8266 的收发端口设置为 I015:TX IO13:RX） |
| **输入参数** | bit\_rate 波特率(300-3686400) |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 默认 data\_bits==8,parity==无,stop\_bits==0  如果采用该接口初始化 UART0，I015 电路需做特殊处理，具体可见乐鑫原厂文档。采用该端口进行串口通讯的好处是，可以避免 8266 启动后默认信息输出对用户控制板的数据干扰。 |

* + 1. **UART0 串口初始化(esp8266 的收发端口设置为 IO3:RX IO1:TX)**

|  |  |
| --- | --- |
| **函数原型** | void user\_uart\_raw\_init(UART\_BautRate bit\_rate) |
| **功能描述** | UART0 串口初始化(esp8266 的收发端口设置为 IO3:RX IO1:TX) |
| **输入参数** | bit\_rate 波特率(300-3686400) |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 默认 data\_bits==8,parity==无,stop\_bits==0, esp8266 默认 UART0 的收发 IO 端口 |

* + 1. **读串口缓冲数据大小**

|  |  |
| --- | --- |
| **函数原型** | uint16 user\_uart\_read\_size(void) |
| **功能描述** | 读串口缓冲数据大小 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | 读到的缓冲数据长度 |
| **备注** | 无 |

* + 1. **读取串口数据**

|  |  |
| --- | --- |
| **函数原型** | uint16 user\_uart\_read\_data(uint8 \*out,uint16 out\_len) |
| **功能描述** | 读取串口数据 |
| **输入参数** | out\_len 缓冲大小 |
| **输出参数** | out 读取的数据 |
| **返回值** | 读取的数据长度 |
| **备注** | 无 |

* + 1. **写串口数据**

|  |  |
| --- | --- |
| **函数原型** | void user\_uart\_write\_data(uint8 \*in,uint16 in\_len) |
| **功能描述** | 写串口数据 |
| **输入参数** | in 要写的数据  in\_len 数据长度 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 无 |

1. **定时器接口**
   * 1. **添加一个系统定时器**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET sys\_add\_timer(IN CONST P\_TIMER\_FUNC pTimerFunc,  IN CONST PVOID pTimerArg, OUT TIMER\_ID \*p\_timerID) |
| **功能描述** | 添加一个系统定时器 |
| **输入参数** | pTimerFunc: 定时器处理函数  pTimerArg: 定时器处理参数 |
| **输出参数** | p\_timerID：定时器 ID 编号 |
| **返回值** | 参照返回值列表 |

|  |  |
| --- | --- |
| **备注** | 无 |

* + 1. **停止一个定时器**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET sys\_stop\_timer(IN CONST TIMER\_ID timerID) |
| **功能描述** | 停止一个定时器 |
| **输入参数** | timerID：定时器 ID 编号 |
| **输出参数** | 无 |
| **返回值** | 参照返回值列表 |
| **备注** | 无 |

* + 1. **定时器是否运行**

|  |  |
| --- | --- |
| **函数原型** | BOOL IsThisSysTimerRun(IN CONST TIMER\_ID timerID) |
| **功能描述** | 定时器是否运行 |
| **输入参数** | timerID：定时器 ID 编号 |
| **输出参数** | 无 |
| **返回值** | TRUE 正在运行 FALSE 未运行 |
| **备注** | 无 |

* + 1. **启动一个定时器**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET sys\_start\_timer(IN CONST TIMER\_ID timerID,  IN CONST TIME\_MS timeCycle, IN CONST TIMER\_TYPE timer\_type) |
| **功能描述** | 启动一个定时器 |
| **输入参数** | timerID：定时器 ID 编号timeCycle: 定时周期(单位毫秒) timer\_type: 定时器类型  <1> TIMER\_ONCE 单次执行  <2> TIMER\_CYCLE 循环执行 |
| **输出参数** | 无 |
| **返回值** | 参照返回值列表 |
| **备注** | 无 |

1. **系统层接口**
   * 1. **系统休眠**

|  |  |
| --- | --- |
| **函数原型** | VOID SystemSleep(IN CONST TIME\_MS msTime) |
| **功能描述** | 系统休眠 |
| **输入参数** | msTime: 休眠时间(单位毫秒) |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 无 |

* + 1. **系统重启**

|  |  |
| --- | --- |
| **函数原型** | VOID SystemReset(VOID) |
| **功能描述** | 系统重启 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | 无 |

* + 1. **比较两个字符串是否相等**

|  |  |
| --- | --- |
| **函数原型** | int strcasecmp(const char \*s1, const char \*s2) |
| **功能描述** | 比较两个字符串是否相等 |
| **输入参数** | s1: 字符串 s1  s2: 字符串 s2 |
| **输出参数** | 无 |
| **返回值** | 0 字符串相等 非 0 字符串不等 |
| **备注** | 无 |

* + 1. **ASCII 码转 HEX**

|  |  |
| --- | --- |
| **函数原型** | void ascs2hex(unsigned char \*hex,unsigned char \*ascs,int srclen) |
| **功能描述** | ASCII 码转 HEX |
| **输入参数** | ascs: ASCII 码  srclen: ASCII 码长度 |

|  |  |
| --- | --- |
| **输出参数** | hex: 转换后的 hex 数据 |
| **返回值** | 无 |
| **备注** | 无 |

1. **任务接口**
   * 1. **任务创建**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET CreateAndStart(OUT THRD\_HANDLE \*pThrdHandle,\  IN CONST P\_THRD\_FUNC pThrdFunc,\ IN CONST PVOID pThrdFuncArg,\  IN CONST STACK\_SIZE stack\_size,\ IN CONST TRD\_PRI pri,\  IN CONST CHAR \*thrd\_name); |
| **功能描述** | 任务创建 |
| **输入参数** | pThrdHandle 任务句柄pThrdFunc 任务处理函数pThrdFuncArg 任务参数stack\_size 指定任务堆栈大小pri 任务优先级  thrd\_name 任务名称 |
| **输出参数** | 无 |
| **返回值** | 参照返回值列表 |
| **备注** | 无 |

* + 1. **任务删除**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET ThrdJoin(IN CONST THRD\_HANDLE thrdHandle,  OUT VOID \*\*ppThrdRet); |
| **功能描述** | 任务删除 |
| **输入参数** | thrdHandle 任务句柄 |
| **输出参数** | ppThrdRet 任务退出码 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

1. **信号量接口**
   * 1. **信号量创建**

|  |  |
| --- | --- |
| **函数原型** | SEM\_HANDLE CreateSemaphore(VOID) |
| **功能描述** | 信号量创建 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | 信号量句柄 |
| **备注** | 无 |

* + 1. **信号量初始化**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET InitSemaphore(IN CONST SEM\_HANDLE semHandle,  IN CONST UINT semCnt, IN CONST UINT sem\_max) |
| **功能描述** | 信号量初始化 |
| **输入参数** | semHandle 信号量句柄  semCnt 信号量初始值  sem\_max 信号量最大值 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **信号量发送**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET PostSemaphore(IN CONST SEM\_HANDLE semHandle) |
| **功能描述** | 信号量发送 |
| **输入参数** | semHandle 信号量句柄 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **信号量接收**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET WaitSemaphore(IN CONST SEM\_HANDLE semHandle) |
| **功能描述** | 信号量接收 |

|  |  |
| --- | --- |
| **输入参数** | semHandle 信号量句柄 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

1. **互斥锁接口**
   * 1. **互斥锁创建**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET CreateMutexAndInit(OUT MUTEX\_HANDLE \*pMutexHandle) |
| **功能描述** | 互斥锁创建 |
| **输入参数** | 无 |
| **输出参数** | pMutexHandle 互斥锁句柄 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **互斥锁锁定**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET MutexLock(IN CONST MUTEX\_HANDLE mutexHandle) |
| **功能描述** | 互斥锁锁定 |
| **输入参数** | pMutexHandle 互斥锁句柄 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **互斥锁解锁**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET MutexUnLock(IN CONST MUTEX\_HANDLE mutexHandle) |
| **功能描述** | 互斥锁解锁 |
| **输入参数** | pMutexHandle 互斥锁句柄 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

1. **消息队列接口**
   * 1. **消息队列创建**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET CreateMsgQueAndInit(OUT MSG\_QUE\_HANDLE \*pMsgQueHandle) |
| **功能描述** | 消息队列创建 |
| **输入参数** | 无 |
| **输出参数** | pMsgQueHandle 消息管理结构句柄 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **消息添加**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET AddMsgNodeToQueue(IN CONST MSG\_QUE\_HANDLE  msgQueHandle, IN CONST MSG\_ID msgID, IN CONST P\_MSG\_DATA pMsgData, IN CONST MSG\_DATA\_LEN msgDataLen, IN CONST MSG\_TYPE msgType) |
| **功能描述** | 消息添加 |
| **输入参数** | pMsgQueHandle 消息管理结构句柄  msgID 消 息 ID pMsgData 消息数据  msgDataLen 消息数据长度  msgType 消息类型 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 先进先出方式，系统顺序执行消息 |

* + 1. **获取指定 ID 的消息节点**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET GetMsgNodeFromQueue(IN CONST MSG\_QUE\_HANDLE  msgQueHandle, IN CONST MSG\_ID msgID, OUT P\_MSG\_LIST \*ppMsgListNode) |
| **功能描述** | 获取指定 ID 的消息节点 |
| **输入参数** | pMsgQueHandle 消息管理结构句柄  msgID 消息 ID |
| **输出参数** | pMsgListNode 消息节点 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **获取首先入链的消息节点**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET GetFirstMsgFromQueue(IN CONST MSG\_QUE\_HANDLE  msgQueHandle, OUT P\_MSG\_LIST \*ppMsgListNode) |
| **功能描述** | 获取首先入链的消息节点 |
| **输入参数** | pMsgQueHandle 消息管理结构句柄 |
| **输出参数** | pMsgListNode 消息节点 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **获取队列中的消息节点总数**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET GetMsgNodeNum(IN CONST MSG\_QUE\_HANDLE msgQueHandle,  OUT PINT pMsgNodeNum) |
| **功能描述** | 获取队列中的消息节点总数 |
| **输入参数** | pMsgQueHandle 消息管理结构句柄 |
| **输出参数** | pMsgNodeNum 消息节点总数 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **从链中删除消息节点并释放消息节点内存**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET DelAndFreeMsgNodeFromQueue(IN CONST MSG\_QUE\_HANDLE  msgQueHandle, IN CONST P\_MSG\_LIST pMsgListNode) |
| **功能描述** | 从链中删除消息节点并释放消息节点内存 |
| **输入参数** | pMsgQueHandle 消息管理结构句柄  pMsgListNode 删除的消息节点 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **释放消息队列所占用内存**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET ReleaseMsgQue(IN CONST MSG\_QUE\_HANDLE msgQueHandle) |
| **功能描述** | 释放消息队列所占用内存 |
| **输入参数** | pMsgQueHandle 消息管理结构句柄 |
| **输出参数** | 无 |

|  |  |
| --- | --- |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **投递消息**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET PostMessage(IN CONST MSG\_QUE\_HANDLE msgQueHandle, IN CONST MSG\_ID msgID, IN CONST P\_MSG\_DATA pMsgData,  IN CONST MSG\_DATA\_LEN msgDataLen) |
| **功能描述** | 投递消息 |
| **输入参数** | msgQueHandle 消息处理句柄  msgID 消 息 ID pMsgData 消息数据  msgDataLen 消息数据长度 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 递送一个消息至模块(消息先进先执行) |

* + 1. **投递紧急消息**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET PostInstancyMsg(IN CONST MSG\_QUE\_HANDLE msgQueHandle,\ IN CONST MSG\_ID msgID, IN CONST P\_MSG\_DATA pMsgData,\  IN CONST MSG\_DATA\_LEN msgDataLen) |
| **功能描述** | 投递紧急消息 |
| **输入参数** | msgQueHandle 消息处理句柄  msgID 消 息 ID pMsgData 消息数据  msgDataLen 消息数据长度 |
| **输出参数** | 无 |
| **返回值** | 参照返回值名称 |
| **备注** | 无 |

* + 1. **等待消息**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET WaitMessage(IN CONST MSG\_QUE\_HANDLE msgQueHandle,  OUT P\_MSG\_LIST \*ppMsgListNode) |
| **功能描述** | 等待消息 |
| **输入参数** | msgQueHandle 消息处理句柄 |
| **输出参数** | ppMsgListNode 消息节点 |

|  |  |
| --- | --- |
| **返回值** | 参照返回值名称 |
| **备注** | WaitMessage 成功需调用，消息处理完后需调用  DelAndFreeMsgNodeFromQueue 释放消息 |

**附录**

1. **返回值列表**

|  |  |  |
| --- | --- | --- |
| 返回值宏定义 | 返回值 | 描述 |
| OPRT\_OK | 0 | 执行成功 |
| OPRT\_COM\_ERROR | 1 | 通用错误 |
| OPRT\_INVALID\_PARM | 2 | 无效的入参 |
| OPRT\_MALLOC\_FAILED | 3 | 内存分配失败 |
| OPRT\_INIT\_MUTEX\_ATTR\_FAILED | 4 | 初始化同步属性失败 |
| OPRT\_SET\_MUTEX\_ATTR\_FAILED | 5 | 设置同步属性失败 |
| OPRT\_DESTROY\_MUTEX\_ATTR\_FAILED | 6 | 销毁同步属性失败 |
| OPRT\_INIT\_MUTEX\_FAILED | 7 | 初始化互斥量失败 |
| OPRT\_MUTEX\_LOCK\_FAILED | 8 | 互斥量加锁失败 |
| OPRT\_MUTEX\_TRYLOCK\_FAILED | 9 | 互斥量尝试加锁失败 |
| OPRT\_MUTEX\_LOCK\_BUSY | 10 | 互斥量忙 |
| OPRT\_MUTEX\_UNLOCK\_FAILED | 11 | 互斥量解锁失败 |
| OPRT\_MUTEX\_RELEASE\_FAILED | 12 | 互斥量释放失败 |
| OPRT\_INIT\_SEM\_FAILED | 13 | 初始化信号量失败 |
| OPRT\_WAIT\_SEM\_FAILED | 14 | 等待信号量失败 |
| OPRT\_POST\_SEM\_FAILED | 15 | 释放信号量失败 |
| OPRT\_THRD\_STA\_UNVALID | 16 | 线程状态非法 |
| OPRT\_THRD\_CR\_FAILED | 17 | 线程创建失败 |
| OPRT\_THRD\_JOIN\_FAILED | 18 | 线程JOIN 函数调用失败 |
| OPRT\_THRD\_SELF\_CAN\_NOT\_JOIN | 19 | 自身线程不能调用JOIN 函数 |
| OPRT\_TIMERID\_EXIST | 20 | 定时器ID 已存在 |
| OPRT\_TIMERID\_NOT\_FOUND | 21 | 未找到指定定时器ID |
| OPRT\_TIMERID\_UNVALID | 22 | 定时器ID 非法 |
| OPRT\_GET\_IDLE\_TIMERID\_ERROR | 23 | 获取空闲定时器ID 错误 |
| OPRT\_MSG\_NOT\_FOUND | 24 | 未找到指定消息 |
| OPRT\_MSG\_LIST\_EMPTY | 25 | 消息链空 |
| OPRT\_PSM\_FLH\_RET\_ERR | 26 | PSM 擦除失败 |
| OPRT\_PSM\_FLH\_TM\_ERR | 27 | PSM 擦除超时 |
| OPRT\_PSM\_E\_INVAL | 28 |  |
| OPRT\_PSM\_E\_IO | 29 |  |

|  |  |  |
| --- | --- | --- |
| OPRT\_PSM\_E\_EXIST | 30 |  |
| OPRT\_PSM\_E\_NOENT | 31 |  |
| OPRT\_PSM\_FAIL | 32 |  |
| OPRT\_PSM\_E\_NOSPC | 33 |  |
| OPRT\_PSM\_E\_METADATA\_CRC | 34 |  |
| OPRT\_PSM\_E\_CRC | 35 |  |
| OPRT\_WIFI\_SCAN\_FAIL | 36 |  |
| OPRT\_WF\_MAC\_SET\_FAIL | 37 |  |
| OPRT\_WF\_CONN\_FAIL | 38 |  |
| OPRT\_WF\_NW\_CFG\_FAIL | 39 |  |
| OPRT\_SET\_SOCK\_ERR | 41 |  |
| OPRT\_SOCK\_CONN\_ERR | 42 |  |
| OPRT\_CR\_MUTEX\_ERR | 43 |  |
| OPRT\_CR\_TIMER\_ERR | 44 |  |
| OPRT\_CR\_THREAD\_ERR | 45 |  |
| OPRT\_BUF\_NOT\_ENOUGH | 46 |  |
| OPRT\_URL\_PARAM\_OUT\_LIMIT | 47 |  |
| OPRT\_HTTP\_OS\_ERROR | 48 |  |
| OPRT\_HTTP\_PR\_REQ\_ERROR | 49 |  |
| OPRT\_HTTP\_SD\_REQ\_ERROR | 50 |  |
| OPRT\_HTTP\_RD\_ERROR | 51 |  |
| OPRT\_HTTP\_AD\_HD\_ERROR | 52 |  |
| OPRT\_HTTP\_GET\_RESP\_ERROR | 53 |  |
| OPRT\_HTTP\_AES\_INIT\_ERR | 54 |  |
| OPRT\_HTTP\_AES\_OPEN\_ERR | 55 |  |
| OPRT\_HTTP\_AES\_SET\_KEY\_ERR | 56 |  |
| OPRT\_HTTP\_AES\_ENCRYPT\_ERR | 57 |  |
| OPRT\_TY\_WS\_PART\_ERR | 58 |  |
| OPRT\_CR\_CJSON\_ERR | 59 |  |
| OPRT\_PSM\_SET\_ERROR | 60 |  |
| OPRT\_PSM\_GET\_ERROR | 61 |  |
| OPRT\_CJSON\_PARSE\_ERR | 62 |  |
| OPRT\_CJSON\_GET\_ERR | 63 |  |
| OPRT\_CR\_HTTP\_URL\_H\_ERR | 64 |  |
| OPRT\_HTTPS\_HANDLE\_FAIL | 65 |  |
| OPRT\_HTTPS\_RESP\_UNVALID | 66 |  |
| OPRT\_MEM\_PARTITION\_EMPTY | 67 |  |
| OPRT\_MEM\_PARTITION\_FULL | 68 |  |
| OPRT\_MEM\_PARTITION\_NOT\_FOUND | 69 |  |
| OPRT\_CR\_QUE\_ERR | 70 |  |
| OPRT\_SND\_QUE\_ERR | 71 |  |
| OPRT\_NOT\_FOUND\_DEV | 72 |  |
| OPRT\_NOT\_FOUND\_DEV\_DP | 73 |  |

|  |  |  |
| --- | --- | --- |
| OPRT\_DP\_ATTR\_ILLEGAL | 74 |  |
| OPRT\_DP\_TYPE\_PROP\_ILLEGAL | 75 |  |
| OPRT\_DP\_REPORT\_CLOUD\_ERR | 76 |  |
| OPRT\_NO\_NEED\_SET\_PRODINFO | 77 |  |
| OPRT\_NW\_INVALID | 78 |  |
| OPRT\_SELECT\_ERR | 79 |  |
| OPRT\_SELECT\_TM | 80 |  |
| OPRT\_SEND\_ERR | 81 |  |
| OPRT\_DEV\_NOT\_BIND | 82 |  |
| OPRT\_FW\_UG\_FAILED | 83 |  |
| OPRT\_VER\_FMT\_ERR | 84 |  |
| OPRT\_FW\_NOT\_EXIST | 85 |  |
| OPRT\_SEM\_CR\_ERR | 86 |  |
| OPRT\_SELECT\_TIMEOUT | 87 |  |
| OPRT\_GW\_MQ\_OFFLILNE | 88 |  |
| OPRT\_NOT\_SPT\_CLX\_DP | 89 |  |
| OPRT\_RECV\_ERR | 90 |  |
| OPRT\_UG\_PKG\_NOT\_ENOUGH | 91 |  |
| OPRT\_SCMA\_INVALID | 92 |  |
| OPRT\_PRODECT\_KEY\_NULL | 93 |  |
| OPRT\_DEVICE\_VER\_NULL | 94 |  |
| OPRT\_MSG\_OUT\_OF\_LMT | 95 |  |
| OPRT\_NOT\_FOUND\_AUTH\_SSID | 96 |  |
| OPRT\_SOCKET\_FAULT | 97 |  |
| OPRT\_MQ\_PUBLISH\_TIMEOUT | 98 |  |
| OPRT\_GW\_NOT\_EXIST | 99 |  |
| OPRT\_GW\_SCHEMA\_SIZE\_LMT\_OUT | 100 |  |
| OPRT\_DEV\_DP\_CNT\_INVALID | 101 |  |
| OPRT\_TOKEN\_OVERTIME OPRT\_CR\_TIMER\_FAILED  OPRT\_PUB\_NO\_PERMISSION | 102  103  104 |  |

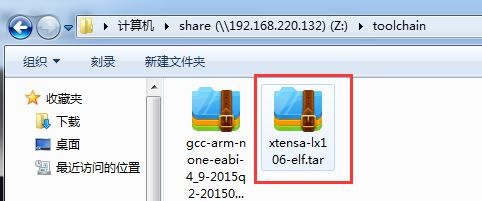
1. **编译环境搭建**
2. **交叉编译器下载**

可先在 windows 下下载完交叉编译器(也可在 ununtu 下直接下载)，下载地址为：

<http://bbs.espressif.com/viewtopic.php?f=57&t=2>

# 交叉编译器安装

将在 windows 下下载完成后的交叉编译器 xtensa-lx106-elf.tar.bz2 复制至共享目录下,我是专门新建了一个 toolchain 目录放置一系列交叉编译器安装文件



# 解压交叉编译器

在 linux 终端下切换至/home/share/samba/toolchain 目录;解压 xtensa-lx106-elf.tar.bz2 至

/usr/bin 目录下,输入命令:

sudo tar -jxvf xtensa-lx106-elf.tar.bz2 -C /usr/bin

# 修改目录属性

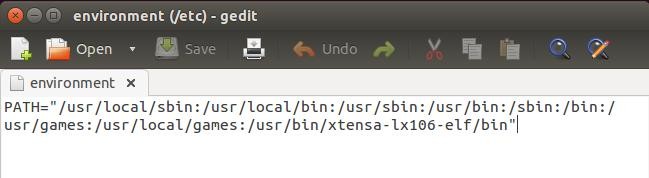
这一步非常重要,不然编译的时候会提示无法编译

sudo chmod 777 -R /usr/bin/xtensa-lx106-elf

# 编辑环境变量

打开/etc/environment 文件.输入命令:

sudo gedit /etc/environment

修 改 /etc/environment 文 件 ( 我 习 惯 编 辑 这 个 文 件 ), 在 文 件 末 尾 添加:/usr/bin/xtensa-lx106-elf/bin 至当前环境变量下

# 执行 source /etc/environment

使环境变量生效.生效后,可以 echo $PATH 查看是否设置成功

# 应用编译方法

1. 进入 tysdk\_for\_esp8266/app 目录
2. 编译有日志输出的固件，例如 sample\_pk，版本号 1.0.0 sh build\_app.sh sample\_pk 1.0.0
3. 编译无日志输出的固件，例如 sample\_pk，版本号 1.0.0 sh build\_app\_release.sh sample\_pk 1.0.0
4. 编译生成 bin 文件说明

编译生成 bin 文件在/bin/upgrade 目录中

1. sample\_pk(1)\_1.0.0.bin 为可下载的固件文件，从 0x1000 地址烧录
2. sample\_pk\_ug\_1.0.0.bin 为升级文件，用作固件升级使用
3. 固件烧录说明(1)文件烧录地址

|  |  |  |
| --- | --- | --- |
| BOOT 文件 | boot\_v1.4(b1).bin | 0x0000 |
| 应用文件 | sample\_pk(1)\_1.0.0.bin | 0x1000 |
| 系统参数文件 | esp\_init\_data\_default.bin | 0xfc000 |
| 填充文件 | blank.bin | 0xfe000 |

(2)建议文件打包成 1M 的 target 文件，适用 1M/2M/4M Byte 的 flash，只烧录 1M 大小即可