**涂鸦智能WIFI-SDK说明**

(Realtek平台)

## 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 | 王秋萍 | 20181227 |  |

1. **框架基本接口(接入涂鸦云需使用)**
   * 1. **系统初始化，必须最先调用**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_init(IN CONST CHAR\_T \*fs\_storge\_path); |
| **功能描述** | 用于tuya iot sdk系统的初始化，必须最先调用 |
| **输入参数** | fs\_storge\_path->filesystem read write storge path  (if os have no fs,then fs\_storge\_path is should be NULL) |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | 无 |

**使用举例：**

**OPRATE\_RET ret = tuya\_iot\_init(NULL);**

**If(OPRT\_OK != ret ) {**

**return OPRT\_COM\_ERR;**

**}**

* + 1. **获取sdk版本信息**

|  |  |
| --- | --- |
| **函数原型** | CHAR\_T \*tuya\_iot\_get\_sdk\_info(VOID); |
| **功能描述** | 获取tuya iot sdk版本信息 |
| **输入参数** | VOID |
| **输出参数** | 无 |
| **返回值** | sdk版本信息,如"< TUYA\_IOT\_SDK V1.0.2 FOR rt871x  BUILD AT 2018\_11\_29\_15\_49\_56 D:1 S:0 W:1 V:11.02\_2.2\_3.2 >" |
| **备注** | 无 |

**使用举例：**

* + 1. **联网模块固件升级处理接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_upgrade\_gw(IN CONST FW\_UG\_S \*fw,\  IN CONST GET\_FILE\_DATA\_CB get\_file\_cb,\  IN CONST UPGRADE\_NOTIFY\_CB upgrd\_nofity\_cb,IN CONST PVOID\_T pri\_data); |
| **功能描述** | 联网模块固件升级处理接口 |
| **输入参数** | fw->fireware upgrade;  get\_file\_cb->get file callback;  upgrd\_nofity\_cb->upgrade notify callback;  pri\_data->private data; |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | 无 |

* + 1. **联网模块固件升级处理接口(优先级高)**

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_iot\_force\_reg\_gw\_ug\_cb(IN CONST GW\_UG\_INFORM\_CB ug\_cb); |
| **功能描述** | 联网模块固件升级处理接口（优先级较1.1.3高） |
| **输入参数** | ug\_cb |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** | force register gw\_ug\_cb,Once set,take it as the main Highest priority |

* + 1. **设备固件升级接口库，包含网关上的主控设备以及各终端子设备的升级**

|  |  |
| --- | --- |
| **函数原型** | PERATE\_RET tuya\_iot\_upgrade\_dev(IN CONST CHAR\_T \*devid,\  IN CONST FW\_UG\_S \*fw, IN CONST GET\_FILE\_DATA\_CB get\_file\_cb,\  IN CONST UPGRADE\_NOTIFY\_CB upgrd\_nofity\_cb,\  IN CONST PVOID\_T pri\_data); |
| **功能描述** | 设备固件升级接口库，包含网关上的主控设备以及各终端子设备的升级 |
| **输入参数** | dev\_id->设备ID  如果ID为空表示升级GW物理上在一起的其他模块  （比如：zigbee、ble、单品mcu(DEV\_NM\_NOT\_ATH\_SNGL)等）  不为空：表示升级与GW物理上分离的模块  fw->fireware upgrade  get\_file\_cb->get file callback  upgrd\_nofity\_cb->upgrade notify callback  pri\_data->private data |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **获取某个设备特定功能点的描述信息**

|  |  |
| --- | --- |
| **函数原型** | DP\_DESC\_IF\_S \*tuya\_iot\_get\_dp\_desc(IN CONST CHAR\_T \*dev\_id,IN CONST BYTE\_T dpid); |
| **功能描述** | 获取某个设备特定功能点的描述信息 |
| **输入参数** | dev\_id dpid |
| **输出参数** | 无 |
| **返回值** | (DP\_DESC\_IF\_S \*) fail(NULL) |
| **备注** |  |

* + 1. **获取某个设备特定功能点的缓存数据**

|  |  |
| --- | --- |
| **函数原型** | DP\_PROP\_VALUE\_U \*tuya\_iot\_get\_dp\_prop\_value(IN CONST CHAR\_T \*dev\_id,IN CONST BYTE\_T dpid); |
| **功能描述** | 获取某个设备特定功能点的缓存数据 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | DP\_PROP\_VALUE\_U \* |
| **备注** |  |

* + 1. **获取设备信息接口**

|  |  |
| --- | --- |
| **函数原型** | DP\_PROP\_VALUE\_U \*tuya\_iot\_get\_dp\_prop\_value(IN CONST CHAR\_T \*dev\_id,IN CONST BYTE\_T dpid); |
| **功能描述** | 获取设备信息 |
| **输入参数** | dev\_id->device id,Note:if(dev\_id == NULL) then means the gateway device |
| **输出参数** | 无 |
| **返回值** | (DEV\_DESC\_IF\_S \*) fail(NULL) |
| **备注** |  |

* + 1. **网关绑定子设备接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_gw\_bind\_dev(IN CONST GW\_PERMIT\_DEV\_TP\_T tp,\  IN CONST USER\_DEV\_DTL\_DEF\_T uddd,IN CONST CHAR\_T \*id,\  IN CONST CHAR\_T \*pk,IN CONST CHAR\_T \*ver); |
| **功能描述** | 网关绑定子设备接口，当涂鸦app使能网关添加子设备时，调用此接口将发现的子设备绑定。 |
| **输入参数** | tp uddd id pk ver |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **网关解绑子设备接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_gw\_unbind\_dev(IN CONST CHAR\_T \*id); |
| **功能描述** | 网关解绑子设备接口 |
| **输入参数** | id |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **网关更新子设备在线/离线状态**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_dev\_online\_stat\_update(IN CONST CHAR\_T \*dev\_id,IN CONST BOOL\_T online); |
| **功能描述** | 网关更新子设备在线/离线状态 |
| **输入参数** | dev\_id online |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **子设备结构化数据异步上报接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET dev\_report\_dp\_json\_async(IN CONST CHAR\_T \*dev\_id,IN CONST TY\_OBJ\_DP\_S \*dp\_data,IN CONST UINT\_T cnt); |
| **功能描述** | 子设备结构化数据异步上报接口，后台保障数据上传的可靠性 |
| **输入参数** | dev\_id->device id,Note:if(dev\_id == NULL) then means  the gateway device report data  dp\_data->dp data,note "time\_stamp" is useless.The data  reporting time is based on the actual arrival of the server  cnt->count |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **子设备透传数据同步上报接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET dev\_report\_dp\_raw\_sync(IN CONST CHAR\_T \*dev\_id,\  IN CONST BYTE\_T dpid,IN CONST BYTE\_T \*data,IN CONST UINT\_T len,\  IN CONST UINT\_T timeout); |
| **功能描述** | 子设备透传数据同步上报接口，由调用者保障数据上报的可靠性 |
| **输入参数** | dev\_id->device id,Note:if(dev\_id == NULL) then means  the gateway device report data  data->raw data  len->raw data len  timeout |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **子设备结构化数据同步上报接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET dev\_report\_dp\_raw\_sync(IN CONST CHAR\_T \*dev\_id,\  IN CONST BYTE\_T dpid,IN CONST BYTE\_T \*data,IN CONST UINT\_T len,\  IN CONST UINT\_T timeout); |
| **功能描述** | 子设备结构化数据同步上报接口，由调用者保障数据上报的可靠性，通常用于统计数据的上报 |
| **输入参数** | dev\_id->device id,Note:if(dev\_id == NULL) then means  the gateway device report data  data->raw data  len->raw data len  timeout |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **启动子设备心跳管理能力**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_sys\_mag\_hb\_init(IN CONST DEV\_HEARTBEAT\_SEND\_CB hb\_send\_cb); |
| **功能描述** | 启动子设备心跳管理能力 |
| **输入参数** | hb\_send\_cb  if(NULL == hb\_send\_cb)  passive heartbeat pattern  else  active heartbeat pattern |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **子设备心跳超时时间设置**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_set\_dev\_hb\_timeout(IN CONST CHAR\_T \*dev\_id,IN CONST TIME\_S hb\_timeout\_time); |
| **功能描述** | 子设备心跳超时时间设置，若网关再超过设置时间内未收到子设备心跳，则网关会将子设备设置为离线。 |
| **输入参数** | dev\_id  hb\_timeout\_time->if(0xffffffff == hb\_timeout\_time)  then device online forever |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **子设备刷新超时离线时间**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_fresh\_dev\_hb(IN CONST CHAR\_T \*dev\_id); |
| **功能描述** | 子设备刷新超时离线时间 |
| **输入参数** | dev\_id |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **子设备固件版本更新**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_gw\_subdevice\_update(IN CONST CHAR\_T \*id,IN CONST CHAR\_T \*ver); |
| **功能描述** | 子设备固件版本更新 |
| **输入参数** | id ver |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **子设备遍历，通过此接口可以遍历网关下所有的子设备**

|  |  |
| --- | --- |
| **函数原型** | DEV\_DESC\_IF\_S \*tuya\_iot\_dev\_traversal(INOUT VOID \*\*iterator); |
| **功能描述** | 子设备遍历，通过此接口可以遍历网关下所有的子设备 |
| **输入参数** | iterator |
| **输出参数** | 无 |
| **返回值** | (DEV\_DESC\_IF\_S \*) (NULL means end) |
| **备注** |  |

1. **Wifi配置类接口**

**详见”tuya\_iot\_wifi\_api.h”**

* + 1. **设置配置类wifi设备的授权信息**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_set\_wf\_gw\_prod\_info(IN CONST WF\_GW\_PROD\_INFO\_S \*wf\_prod\_info); |
| **功能描述** | 设置配置类wifi设备的授权信息，授权信息需通过涂鸦后台获取，否则设备无法正常使用 |
| **输入参数** | wf\_prod\_info |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **wifi mcu 设备初始化接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_wf\_mcu\_dev\_init(IN CONST GW\_WF\_CFG\_MTHD\_SEL cfg,\  IN CONST GW\_WF\_START\_MODE start\_mode,\  IN CONST TY\_IOT\_CBS\_S \*cbs,IN CONST CHAR\_T \*product\_key,\  IN CONST CHAR\_T \*wf\_sw\_ver,IN CONST CHAR\_T \*mcu\_sw\_ver); |
| **功能描述** | wifi mcu 设备初始化接口 |
| **输入参数** | cfg  start\_mode  cbs->tuya wifi sdk user callbacks  product\_key->product key/proudct id,get from tuya open platform  wf\_sw\_ver->wifi module software version format:xx.xx.xx (0<=x<=9)  mcu\_sw\_ver->mcu software version format:xx.xx.xx (0<=x<=9) |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | typedef BYTE\_T GW\_WF\_START\_MODE;  #define WF\_START\_AP\_ONLY 0 // only have ap-cfg mode  #define WF\_START\_SMART\_ONLY 1 // only have smart-cfg mode  #define WF\_START\_AP\_FIRST 2 // default is ap-cfg mode  #define WF\_START\_SMART\_FIRST 3 // default is smart-cfg mode |

* + 1. **wifi soc 设备初始化接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_wf\_soc\_dev\_init(IN CONST GW\_WF\_CFG\_MTHD\_SEL cfg,\  IN CONST GW\_WF\_START\_MODE start\_mode, \  IN CONST TY\_IOT\_CBS\_S \*cbs,IN CONST CHAR\_T \*product\_key,\  IN CONST CHAR\_T \*wf\_sw\_ver); |
| **功能描述** | wifi soc 设备初始化接口 |
| **输入参数** | Cfg  start\_mode  cbs->tuya wifi sdk user callbacks,note cbs->dev\_ug\_cb is useless  product\_key->product key/proudct id,get from tuya open platform  wf\_sw\_ver->wifi module software version format:xx.xx.xx (0<=x<=9) |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | typedef BYTE\_T GW\_WF\_START\_MODE;  #define WF\_START\_AP\_ONLY 0 // only have ap-cfg mode  #define WF\_START\_SMART\_ONLY 1 // only have smart-cfg mode  #define WF\_START\_AP\_FIRST 2 // default is ap-cfg mode  #define WF\_START\_SMART\_FIRST 3 // default is smart-cfg mode |

* + 1. **wifi网关初始化接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_wf\_gw\_init(IN CONST GW\_WF\_CFG\_MTHD\_SEL cfg,\  IN CONST TY\_IOT\_CBS\_S \*cbs,IN CONST TY\_IOT\_GW\_CBS\_S \*gw\_cbs,\  IN CONST CHAR\_T \*product\_key,IN CONST CHAR\_T \*wf\_sw\_ver,\  IN CONST GW\_ATTACH\_ATTR\_T \*attr,IN CONST UINT\_T attr\_num); |
| **功能描述** | wifi网关初始化接口 |
| **输入参数** | cfg  cbs->tuya wifi sdk user callbacks  gw\_cbs->tuya gw user callbacks  product\_key->product key/proudct id,get from tuya open platform  wf\_sw\_ver->wifi module software version format:xx.xx.xx (0<=x<=9)  attr->gw attr |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **wifi网关加设备初始化接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_wf\_gw\_dev\_init(IN CONST GW\_WF\_CFG\_MTHD\_SEL cfg,\  IN CONST TY\_IOT\_CBS\_S \*cbs,IN CONST TY\_IOT\_GW\_CBS\_S \*gw\_cbs,\  IN CONST CHAR\_T \*product\_key,IN CONST CHAR\_T \*wf\_sw\_ver,\  IN CONST GW\_ATTACH\_ATTR\_T \*attr,IN CONST UINT\_T attr\_num); |
| **功能描述** | wifi网关加设备初始化接口 |
| **输入参数** | cfg  cbs->tuya wifi sdk user callbacks  gw\_cbs->tuya gw user callbacks  product\_key->product key/proudct id,get from tuya open platform  wf\_sw\_ver->wifi module software version format:xx.xx.xx (0<=x<=9)  attr->gw attr  attr\_num |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **获取wifi状态接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_reg\_get\_wf\_nw\_stat\_cb(IN CONST GET\_WF\_NW\_STAT\_CB wf\_nw\_stat\_cb); |
| **功能描述** | 获取wifi状态接口 |
| **输入参数** | wf\_nw\_stat\_cb |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **wifi设备重置接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_wf\_gw\_unactive(VOID); |
| **功能描述** | wifi设备重置接口 |
| **输入参数** | VOID |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **设置wifi设备ap配网时默认的ssid和passwd**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_set\_user\_def\_ap\_if(IN CONST CHAR\_T \*ssid,IN CONST CHAR\_T \*passwd); |
| **功能描述** | 设置wifi设备ap配网时默认的ssid和passwd。如不设置，设备处于ap状态时，默认热点名称为：SmartLife-xxxx（xxxx为mac地址后四位）。 |
| **输入参数** | ssid passwd |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | need call before tuya\_iot\_wf\_xxx\_init |

* + 1. **其他配网方式调用接口(非ap和smart config)**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_iot\_gw\_wf\_user\_cfg(IN CONST CHAR\_T \*ssid,IN CONST CHAR\_T \*passwd,IN CONST CHAR\_T \*token); |
| **功能描述** | 当通过其他模式配网时（非ap 非smartconfig），比如摄像头的二维码配网，声波配网，调用此接口处理。 |
| **输入参数** | ssid passwd token |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | use to TY\_APP\_CFG\_WF mode |

1. **应用层的入口函数介绍**
   * 1. **tuya\_main.c**

|  |  |
| --- | --- |
| **函数原型** | tuya\_main.c : user\_main() : |
| **功能描述** | tuya\_iot的入口函数,tuya\_iot的各模块初始化（定时器，事件，flash相关）、厂测流程、网关升级注册、最后启动用户代码入口函数device\_init()。 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** |  |

* + 1. **tuya\_device.c**

|  |  |
| --- | --- |
| **函数原型** | tuya\_device.c : device\_init()： |
| **功能描述** | 启动设备初始化流程比如soc设备tuya\_iot\_wf\_soc\_dev\_init、启动wifi联网状态监测定时器以及项目相关功能。 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | 无 |
| **备注** |  |

1. **flash读写接口**

flash读写接口头文件 tuya\_ws\_db.h

* + 1. **flash读接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET wd\_common\_read(IN CONST CHAR\_T \*key, OUT BYTE\_T \*\*value, OUT UINT\_T \*p\_len); |
| **功能描述** | 读取flash相关内容 |
| **输入参数** | key |
| **输出参数** | value p\_len |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **flash写接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET wd\_common\_write(IN CONST CHAR\_T \*key, IN CONST BYTE\_T \*value, IN CONST UINT\_T len); |
| **功能描述** | 对flash进行写操作 |
| **输入参数** | key value len |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

1. **oem固件的配置信息读写接口**
   * 1. **oem固件配置信息读接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET ws\_db\_user\_param\_read(OUT BYTE\_T \*\*buf,OUT UINT\_T \*len); |
| **功能描述** | oem固件的配置信息读接口 |
| **输入参数** | buf len |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

* + 1. **oem固件配置信息写接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET ws\_db\_user\_param\_write(IN CONST BYTE\_T \*data,IN CONST UINT\_T len); |
| **功能描述** | oem固件的配置信息写接口 |
| **输入参数** | data len |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |

1. **日志打印相关**
   * 1. **日志等级相关字段说明**

|  |  |
| --- | --- |
| **相关描述** | #define LOG\_LEVEL\_ERR 0 // 错误信息，程序正常运行不应发生的信息  #define LOG\_LEVEL\_WARN 1 // 警告信息  #define LOG\_LEVEL\_NOTICE 2 // 需要注意的信息  #define LOG\_LEVEL\_INFO 3 // 通知信息  #define LOG\_LEVEL\_DEBUG 4 // 程序运行调试信息，RELEASE版本中删除  #define LOG\_LEVEL\_TRACE 5 // 程序运行路径信息，RELEASE版本中删除 |

* + 1. **设置打印等级的接口**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET SetLogManageAttr(IN CONST LOG\_LEVEL curLogLevel); |
| **功能描述** | 设置打印等级 |
| **输入参数** | curLogLevel  // 日志输出等级  typedef INT\_T LOG\_LEVEL;  #define LOG\_LEVEL\_ERR 0 // 错误信息，程序正常运行不应发生的信息  #define LOG\_LEVEL\_WARN 1 // 警告信息  #define LOG\_LEVEL\_NOTICE 2 // 需要注意的信息  #define LOG\_LEVEL\_INFO 3 // 通知信息  #define LOG\_LEVEL\_DEBUG 4 // 程序运行调试信息，RELEASE版本中删除  #define LOG\_LEVEL\_TRACE 5 // 程序运行路径信息，RELEASE版本中删除 |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** |  |
| **使用举例** | SetLogManageAttr(LOG\_LEVEL\_INFO)，那么LOG\_LEVEL\_INFO以下等级的日志信息（LOG\_LEVEL\_DEBUG 和 LOG\_LEVEL\_TRACE）就不会打印。 |

1. **gpio 测试**

gpio测试只需要在tuya\_device.c的gpio\_test接口调用gpio\_test\_cb

* + 1. **gpio测试接口**

|  |  |
| --- | --- |
| **函数原型** | BOOL\_T gpio\_test\_cb(BOARD\_TYPE type)； |
| **功能描述** | gpio测试接口 |
| **输入参数** | 无 |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | gpio测试只需要在tuya\_device.c的gpio\_test接口调用gpio\_test\_cb |
| **使用举例** | BOOL gpio\_test(VOID) {  return gpio\_test\_cb(RTL\_BOARD\_WR3) || gpio\_test\_cb(RTL\_BOARD\_WR1) || \  gpio\_test\_cb(RTL\_BOARD\_WR2)||gpio\_test\_cb(RTL\_BOARD\_WR4)|| \  gpio\_test\_cb(RTL\_BOARD\_WR7);  } |

1. **按键接口**
   * 1. **按键初始化**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET key\_init(IN CONST KEY\_USER\_DEF\_S \*p\_tbl,IN CONST INT\_T cnt,\  IN CONST INT\_T timer\_space) |
| **功能描述** | 按键初始化接口 |
| **输入参数** | p\_tbl cnt  timer\_space->if timer (space == 0) then use default value(20ms) |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | 在按键注册之前调用 |
| **使用举例** | OPERATE\_RET op\_ret = OPRT\_OK;  op\_ret = key\_init(NULL,0,KEY\_TIMER\_MS);  if(op\_ret != OPRT\_OK) {  PR\_ERR("key\_init err:%d",op\_ret);  return op\_ret;  } |

* + 1. **注册按键回调函数**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET reg\_proc\_key(IN CONST KEY\_USER\_DEF\_S \*key\_ud); |
| **功能描述** | 注册按键回调函数 |
| **输入参数** | KEY\_USER\_DEF\_S \*key\_ud  typedef struct { // user define  TY\_GPIO\_PORT\_E port; //io口  BOOL\_T low\_level\_detect; // 低电平检测使能true：低有效 false：高有效  KEY\_LONG\_PRESS\_TP\_E lp\_tp; //触发类型设置  USHORT\_T long\_key\_time; // 长按时间 ms (lp\_tp == LP\_ONCE\_TRIG then valid and must >= 1000ms)  USHORT\_T seq\_key\_detect\_time; //按键禁止连击 ms 0:disable default:400ms  KEY\_CALLBACK call\_back; //按键动作回调函数  }KEY\_USER\_DEF\_S; |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | 使用按键之前先要注册按键回调函数 |
| **使用举例** | OPERATE\_RET op\_ret = OPRT\_OK;  op\_ret = reg\_proc\_key(&hw\_table->rst\_button);//初始化reset按键  if( op\_ret != OPRT\_OK){  PR\_ERR("reg\_proc\_key err:%d",op\_ret);  return;  } |

注：在低功耗模式，只有四个gpio可作为按键使用：GPIO5、GPIO18、GPIO22、GPIO23。

1. **LED接口**
   * 1. **创建LED控制句柄**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_create\_led\_handle(IN CONST TY\_GPIO\_PORT\_E port,IN CONST BOOL\_T high,OUT LED\_HANDLE \*handle); |
| **功能描述** | 创建LED控制句柄 |
| **输入参数** | TY\_GPIO\_PORT\_E port // io口  IN CONST BOOL\_T high //高有效使能 true: 高有效  OUT LED\_HANDLE \*handle //LED handle  //LED创建成功设置可选  typedef PVOID\_T LED\_HANDLE; // led handle  typedef enum {  OL\_LOW = 0, // output level low  OL\_HIGH, // output level high  OL\_FLASH\_LOW, // when led flash,the level output low first.  // when led flash end,the the level output low.  OL\_FLASH\_HIGH, // when led flash,the level output high first.  // when led flash end,the the level output high.  }LED\_LT\_E; |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | 使用继电器以及指示灯之前要创建对应的句柄 |
| **使用举例** | OPERATE\_RET op\_ret = OPRT\_OK;  UCHAR\_T i;  if(hw\_table->wifi\_stat\_led.type == IO\_DRIVE\_LEVEL\_HIGH) { //初始化 点亮led  op\_ret = tuya\_create\_led\_handle(hw\_table->wifi\_stat\_led.pin,FALSE,\  &hw\_table->wifi\_stat\_led\_handle);  }else if(hw\_table->wifi\_stat\_led.type == IO\_DRIVE\_LEVEL\_LOW) {  op\_ret = tuya\_create\_led\_handle(hw\_table->wifi\_stat\_led.pin,\  TRUE,&hw\_table->wifi\_stat\_led\_handle);  } |

注:一般指示灯和继电器均调用此接口进行注册。

* + 1. **LED控制**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET tuya\_create\_led\_handle(IN CONST TY\_GPIO\_PORT\_E port,IN CONST BOOL\_T high,OUT LED\_HANDLE \*handle); |
| **功能描述** | 创建LED控制句柄 |
| **输入参数** | TY\_GPIO\_PORT\_E port // io口  IN CONST BOOL\_T high //高有效使能 true: 高有效  OUT LED\_HANDLE \*handle //LED handle  //LED创建成功设置可选  typedef PVOID\_T LED\_HANDLE; // led handle  typedef enum {  OL\_LOW = 0, // output level low  OL\_HIGH, // output level high  OL\_FLASH\_LOW, // when led flash,the level output low first.  // when led flash end,the the level output low.  OL\_FLASH\_HIGH, // when led flash,the level output high first.  // when led flash end,the the level output high.  }LED\_LT\_E; |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | 使用继电器以及指示灯之前要创建对应的句柄 |
| **使用举例** | OPERATE\_RET op\_ret = OPRT\_OK;  UCHAR\_T i;  if(hw\_table->wifi\_stat\_led.type == IO\_DRIVE\_LEVEL\_HIGH) { //初始化 点亮led  op\_ret = tuya\_create\_led\_handle(hw\_table->wifi\_stat\_led.pin,FALSE,\  &hw\_table->wifi\_stat\_led\_handle);  }else if(hw\_table->wifi\_stat\_led.type == IO\_DRIVE\_LEVEL\_LOW) {  op\_ret = tuya\_create\_led\_handle(hw\_table->wifi\_stat\_led.pin,\  TRUE,&hw\_table->wifi\_stat\_led\_handle);  } |

1. **UART接口**
   * 1. **串口初始化**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET ty\_uart\_init(IN CONST TY\_UART\_PORT\_E port,\  IN CONST TY\_UART\_BAUD\_E badu, IN CONST TY\_DATA\_BIT\_E bits,\  IN CONST TY\_PARITY\_E parity,IN CONST TY\_STOPBITS\_E stop,\  IN CONST UINT\_T bufsz,IN CONST BOOL\_T unblock) |
| **功能描述** | 串口初始化设置 |
| **输入参数** | IN CONST TY\_UART\_PORT\_E port \\ 设置串口号  IN CONST TY\_UART\_BAUD\_E badu \\ 设置通信波特率  IN CONST TY\_DATA\_BIT\_E bits \\ 数据位  IN CONST TY\_PARITY\_E parity \\ 校验位  IN CONST TY\_STOPBITS\_E stop \\ 停止位  IN CONST UINT\_T bufsz \\ 缓冲区大小  IN CONST BOOL\_T unblock \\ read uart是否用非阻塞的方法 1：unblock，0：block |
| **输出参数** | 无 |
| **返回值** | OPERATE\_RET |
| **备注** | 需要再使用串口前设置调用 |
| **使用举例** | OPERATE\_RET op\_ret = OPRT\_OK;  op\_ret = ty\_uart\_init(USER\_UART, baud, TYWL\_8B, TYP\_NONE, TYS\_STOPBIT1, DATA\_BUF\_SZ,TRUE); |

1. **低功耗固件**

**11.1.1设置低功耗模式**

|  |  |
| --- | --- |
| **函数原型** | VOID tuya\_set\_lp\_mode(BOOL\_T lp\_enable) |
| **功能描述** | 设置低功耗模式 |
| **输入参数** | BOOL\_T lp\_enable \\ [1-low power](\\\\1-enable) mode； 0-normal mode |
| **输出参数** | 无 |
| **返回值** | VOID |
| **备注** | 在device\_init之前调用，建议在pre\_device\_init里调用。 |
| **使用举例** |  |

**11.1.2 打开低功耗**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET wf\_lowpower\_enable(VOID) |
| **功能描述** | 打开低功耗 |
| **输入参数** |  |
| **输出参数** | 无 |
| **返回值** | VOID |
| **备注** | 在低功耗模式下，有些功能需要关掉低功耗才能正常运行 |
| **使用举例** |  |

**11.1.2 关闭低功耗**

|  |  |
| --- | --- |
| **函数原型** | OPERATE\_RET wf\_lowpower\_disable(VOID) |
| **功能描述** | 关闭低功耗 |
| **输入参数** |  |
| **输出参数** | 无 |
| **返回值** | VOID |
| **备注** | 在低功耗模式下，有些功能需要关掉低功耗才能正常运行 |
| **使用举例** |  |