

OpenCPU 用户编程手册

0 前言

蜂窝移动通信（Cellular Mobile Communication）是采用蜂窝无线组网方式，在终端和网络设备之间通过无线通道连接起来，进而实现用户在活动中可相互通信。其主要特征是终端的移动性，并具有越区切换和跨本地网自动漫游功能。蜂窝移动通信业务是指经过由基站子系统和移动交换子系统等设备组成蜂窝移动通信网提供的话音、数据、视频图像等业务。

针对当前的蜂窝物联网场景，尽管GSM曾经是，现在仍旧是很多物联网通讯使用的技术，但是这个技术太陈旧，而且很多网络运行商迟早会关掉它。这样未来只能在LTE网络中增加物联网通讯的特功能，3GPP对LTE技术增加了几个增强型的标准，一些做了简化，一些是完全新的，用来实现连接IoT设备。

3GPP制定了一些增强型的标准：

1. LTE Category 0 设备速率可以达到1Mbps；
2. LTE Category 1 设备速率可以达到10Mbps；
3. LTE Category M1 设备速率可以达到1Mbps，并且在降低功耗方面上做了优化；
4. LTE Category M2 也被称为Narrow-Band IoT，设备速率只有几百Kbps，但是在功耗上做了很大的优化，并且可以扩大室内覆盖范围；

上面四个类型设备的共同点是，它们都能和现在部署好的LTE网络通讯，只需要对基站和核心网的软件进行升级即可。基站可以同时处理传统的LTE网络、LTE-Advanced移动宽带网络、以及上述的增强型物联网网络。

本文档主要介绍 TUYA CATX模组OpenCPU方式提供给用户编程的API，协助客户快速开发。

模组型号	固件版本号	应用程序FLASH空间大小
MA510GL		

1 oc_app_entry API

1.1 tuya_cniot API

1.1.1 初始化蜂窝系统管理信息

```
/**
 * @brief 初始化蜂窝系统管理信息
 *
 * @param 无
 *
 * @return 无
 */
```

```
void tuyacn_init(void);
```

1.1.2 启动蜂窝网络功能服务

```
/**
 * @brief 启动蜂窝网络数据服务功能，默认SIM 0
 *
 * @param 无
 *
 * @return
 *      OPRT_OK 成功
 *      其它 失败
 */
```

```
OPERATE_RET tuyacn_start_cellular(void);
```

1.1.3 根据SIM卡ID启动蜂窝网络数据服务功能

```
/**
 * @brief 根据SIM卡ID启动蜂窝网络数据服务功能
 *
 * @param simid 0~1
 *
 * @return
 *      OPRT_OK 成功
 *      其它 失败
 */
```

```
OPERATE_RET tuyacn_adv_start_cellular(UINT8_T simid);
```

1.1.4 启动产测功能

```

/**
 * @brief 启动涂鸦协议产测功能
 *
 * @param 无
 *
 * @return
 *      OPRT_OK 成功
 *      其它 失败
 */

```

```

OPERATE_RET tuya_cmiot_start_mftest(void);

```

1.1.5 启动产测功能

```

/**
 * @brief 启动TUYAOS业务的服务
 *
 * @param 无
 *
 * @return
 *      OPRT_OK 成功
 *      其它 失败
 */

```

```

OPERATE_RET tuya_cmiot_start_tuyaos(void);

```

1.1.6 tuya-iot平台连接通知的回调函数原型

```

/**
 * @brief tuya-iot平台连接通知的回调函数原型
 * @param stat GW_NW_STAT_T类型
 * @return 无
 */

```

```

typedef void(*cloud_status_notify)(GW_NW_STAT_T stat);

```

1.1.7 蜂窝配置设备信息

```

/**
 * @brief tuya_iot_cellular_set_prod_info
 * @desc 设置涂鸦产品信息
 *
 * @param prod_info: tuya-sdk product info
 *
 * @return OPRT_OK: success  Other: fail
 */

```

```

OPERATE_RET tuya_iot_cellular_set_prod_info(IN CONST
CELLULAR_PROD_INFO_S *prod_info);

```

2.1.7 蜂窝设备涂鸦云服务参数初始化

```

/**
 * @brief 启动IoT设备
 * @note 启动IoT设备后，设备将向涂鸦云发起连接，如果设备未与APP绑定，APP可通过
 *       设备支持的配网方式，完成设备配网绑定；绑定成功的设备，将具备IoT设备相关的
 *       功能。
 *
 * @param
 *       cbs 涂鸦云回调函数
 *       pid 设备PID
 *       appv 应用固件版本号，该字段必填，版本号必须是合法版本字符串。
 *       fw_key 产品固件key，用户需传入正确的应用固件KEY。
 *       ext_attch 扩展固件配置，如果没有，传入NULL。通道0和通道10已经被内部使用，请勿设置
 *       ext_attch_count 扩展固件数量
 * @return
 *       OPRT_OK 成功
 *       其它 失败
 */

```

```

OPERATE_RET tuya_cniot_start_iot_device(TY_IOT_CBS_S* cbs,
CONST CHAR_T* pid, CONST CHAR_T* appv, CONST CHAR_T*
fw_key,GW_ATTACH_ATTR_T *ext_attch,UCHAR_T
ext_attch_count);

```

1.1.8 蜂窝设备心跳参数初始化

```

/**
 * @brief 启动IoT设备
 * @note 启动IoT设备后，设备将向涂鸦云发起连接，如果设备未与APP绑

```

定，APP可通过

```
*      设备支持的配网方式，完成设备配网绑定；绑定成功的设备，将具备IoT
设备相关的
*      功能。
*
* @param
*      cbs 涂鸦云回调函数
*      pid 设备PID
*      appv 应用固件版本号，该字段必填，版本号必须是合法版本字符串。
*      fw_key 产品固件key，用户需传入正确的应用固件KEY。
*      ext_attch 扩展固件配置，如果x没有，传入NULL。通道0和通道10已
经被内部使用，请勿设置
*      ext_attch_count 扩展固件数量
*      mq_keepalive MQTT 心跳间隔时间，单位秒
* @return
*      OPRT_OK 成功
*      其它 失败
*/
```

```
#define TUYA_CN_IOT_START_IOT_DEVICE_MQTT_KEEPALIVE(cbs,
pid, appv, fw_key, ext_attch, ext_attch_count, mq_keepalive)
```

1.1.9 解除和tuya iot平台的激活状态

```
/**
 * @brief 解除和tuya iot平台的激活状态
 * @param
 * @return OPRT_OK: success Other: fail
 */
```

```
OPERATE_RET TUYA_IOT_CELLULAR_DEV_UNACTIVE(VOID);
```

1.1.10 获取当前设备的连iot平台状态

```
/**
 * @brief 获取当前设备的连iot平台状态
 * @param
 * @return GW_NW_STAT_T
 */
```

```
GW_NW_STAT_T TUYA_IOT_CELLULAR_DEV_GET_CLOUD_STATUS(VOID);
```

1.1.11 注册当前设备状态通知

```
/**
 * @brief 注册当前设备了解涂鸦iot平台的状态通知
 * @param fun 状态变化通知函数
 * @return 0 成功 -1 失败
 */
```

```
OPERATE_RET tuya_iot_reg_get_cellular_stat_cb(IN CONST
cloud_status_notify nw_stat_cb);
```

1.1.12 查询设备是否烧录授权信息

```
/**
 * @brief 查询设备是否烧录授权信息
 * @param 无
 * @return TRUE 已烧录 FALSE 未烧录
 */
```

```
BOOL_T tuya_iot_cellular_dev_is_auth_info_restored(VOID);
```

1.1.16 蜂窝设备扩展固件升级通知回调函数定义

```
/**
 * @brief 扩展固件开始升级通知函数原型
 *
 * @param 参考FW_UG_S
 * @return 无
 */
```

```
typedef int (*opencpu_ext_fota_inform_hook)(FW_UG_S *fw);
```

1.1.17 蜂窝设备扩展固件升级过程回调函数定义

```
/**
 * @brief 扩展固件数据接收处理函数原型
 * @note 升级开始后，将通过涂鸦云传输升级固件，固件数据传输过程中，将
调用
 * 该函数，接收升级固件。
 *
 * @param offset 数据起始字节在固件中的偏移
 * @param data 固件数据
 * @param len 固件数据长度
 * @return 无
 */
```

```
typedef int (*opencpu_ext_fota_process_hook)
(GW_PERMIT_DEV_TP_T tp, UINT_T offset, const BYTE_T* data,
UINT_T len);
```

1.1.15 蜂窝设备扩展固件升级扩展固件传输完成通知回调函数定义

```
/**
 * @brief 扩展固件传输完成通知函数原型
 * @note 该函数在固件升级数据接收完成后调用，调用该函数只代表固件从云端传输至
 *       设备完成，并不意味着升级成功，用户需要在函数中实现将固件写入MCU Flash。
 *
 * @param result 固件传输结果，OPRT_OK表示传输成功，其它表示传输失败。
 * @return 无
 */
```

```
typedef int (*opencpu_ext_fota_notify_hook)
(GW_PERMIT_DEV_TP_T tp, INT_T result)
```

1.1.16 注册扩展更加升级通知回调函数

```
/**
 * @brief 注册扩展固件升级开始通知函数
 * @note 1、升级过程中不允许调用该函数，否则可能引起空指针异常。
 *       2、注册传入空指针，可以取消原注册函数。
 *       3、MCU固件开始升级时，将调用注册的通知函数。
 *
 * @param hook 扩展升级开始通知函数
 * @return 0 注册成功
 */
```

```
int
tuya_cniot_register_ext_fota_inform_hook(opencpu_ext_fota_inform_hook hook);
```

1.1.17 注册扩展固件升级数据接收函数

```

/**
 * @brief 注册扩展固件升级数据接收函数
 * @note 1、升级过程中不允许调用该函数，否则可能引起空指针异常。
 *        2、注册传入空指针，可以取消原注册函数。
 *        3、开始接收MCU固件升级数据时，将调用注册的接收处理函数。
 *
 * @param hook 扩展固件升级数据接收函数
 * @return 0 注册成功
 */

```

```

int
tuya_cnlot_register_ext_fota_process_hook(opencpu_ext_fota_
process_hook hook);

```

1.1.18 注册扩展固件升级结束函数

```

/**
 * @brief 注册扩展固件升级结束函数
 * @note 1、升级过程中不允许调用该函数，否则可能引起空指针异常。
 *        2、注册传入空指针，可以取消原注册函数。
 *        3、扩展固件传输完成后，将调用注册的通知函数。
 *
 * @param hook 扩展固件传输完成通知函数
 * @return 0 注册成功
 */

```

```

int
tuya_cnlot_register_ext_fota_notify_hook(opencpu_ext_fota_n
otify_hook hook);

```

1.1.19 将新的扩展版本号更新到IoT平台

```

/**
 * @brief 将新的扩展版本号更新到IoT平台。
 * @note 扩展升级成功后，用户调用该函数将新的扩展版本号发送至IoT平
台，使手机APP能
 *        显示新的版本号。
 * @param tp 更新后的扩展固件的通道号
 * @param version 更新后的扩展固件版本号
 * @note 版本号为字符串形式，以'\0'结束，字符串的长度不超过10个字
符。
 *
 * @return
 *        0 更新成功
 *       -1 版本号错误

```



```

*      -2 通道号错误
*      -3 更新失败
*      其他参考: tuyu_error_code.h的错误号
*/

```

```

int tuyu_cniot_update_ex_fw_ver(GW_PERMIT_DEV_TP_T tp,
CHAR_T ver[11]);

```

1.1.20 获取系统固件信息

```

/**
 * @brief 获取系统固件信息
 *
 * @param
 *      fw_name 固件名称 (字符串形式, 长度不超过63个有效字符)
 *      fw_ver 固件版本 (字符串形式, 长度不超过10个有效字符)
 *
 * @return
 */

```

```

void tuyu_cniot_get_sysfw_info(char fw_name[64], char
fw_ver[11]);

```

1.1.21 用户产测接口原型定义

```

typedef struct {
    MF_USER_CALLBACK user_enter_mf_callback; /*< 进入产测状态
回调接口 */
    MF_PRE_GPIO_TEST_CB user_pre_gpio_test; /*< 执行GPIO产
测前回调接口 */
    MF_USER_PRODUCT_TEST_CB user_product_test; /*< 用户产测项
处理回调接口 */
    MF_USER_CALLBACK user_callback; /*< 产测配置写入回调接口
*/
} USER_MF_TEST_INTF_S;

```

1.1.22 注册用户实现的产测函数

```

/**
 * @brief 注册用户实现的产测函数
 *
 * @param user_mf_intf 用户产测函数接口
 *
 * @return 0 注册成功 其它 注册失败
 */

```

```

int tuyacnmiot_user_mftest_register(USER_MF_TEST_INTF_S*
user_mf_intf);

```

1.1.23 用户注册产测固件信息

```

/**
 * @brief 注册产测固件信息
 * @note 产测阶段，固件版本检测项会获取固件版本，如果信息与期望不一致，产测将
 * 失败。当用户程序实现了自己的产品产测功能时，需要调用该函数将用户产
 * 测程序
 * 版本信息提供给核心产测系统。
 *
 * @param
 * fw_name 应用固件名称（字符串形式，长度不超过63个有效字符）
 * fw_ver 应用固件版本（字符串形式，长度不超过10个有效字符）
 *
 * @return
 * OPRT_OK 成功
 * 其它 失败
 */

```

```

OPERATE_RET tuyacnmiot_set_mftest_fw_info(char* fw_name,
char* fw_ver);

```

1.1.24 启动IoT设备，可选择BLE的蓝牙配网和蓝牙控制是否启动功能（推荐使用）

```

/**
 * @brief 启动IoT设备，可选择BLE的蓝牙配网和蓝牙控制是否启动功能。
 * @note 启动IoT设备后，设备将向涂鸦云发起连接，如果设备未与APP绑定，APP可通过
 * 设备支持的配网方式，完成设备配网绑定；绑定成功的设备，将具备IoT
 * 设备相关的
 * 功能，支持MQTT心跳间隔设置，MQTT心跳间隔最小值10S。
 * 当蓝牙配网和蓝牙DP控制都未启用，则自动关闭内部的蓝牙服务。
 *

```

```

* @param
*   cbs 涂鸦云回调函数
*   pid 设备PID
*   appv 应用固件版本号，该字段必填，版本号必须是合法版本字符串。
*   fw_key 产品固件key，用户需传入正确的应用固件KEY。
*   ext_attch 扩展固件配置，如果x没有，传入NULL。通道0和通道10已经被内部使用，请勿设置
*   ext_attch_count 扩展固件数量
*   mq_keepalive MQTT 心跳间隔时间，单位秒
*   ble_netcfg 是否启动蓝牙配网(MA510GL设置FALSE)
*   ble_ctrl 是否启动蓝牙DP控制。如果需要TUYA APP通过蓝牙连接蜂窝设备，需要产品的通讯类型里有BLE通讯类型。(MA510GL设置FALSE)
* @return
*   OPRT_OK 成功
*   其它 失败
*/

```

```

OPERATE_RET
tuya_cniot_start_iot_device_with_ble(TY_IOT_CBS_S* cbs,
                                     CONST CHAR_T* pid,
                                     CONST CHAR_T* appv,
                                     CONST CHAR_T*
fw_key,
                                     GW_ATTACH_ATTR_T
*ext_attch,
                                     UCHAR_T
ext_attch_count,
                                     UINT_T
mq_keepalive_tm,
                                     BOOL_T ble_netcfg,
                                     BOOL_T ble_ctrl);

```

1.1.25 OPENCPU 应用示例

```

THREAD_HANDLE device_TaskHandle = NULL;

OPERATE_RET tuya_cell_pdp_active(CHAR_T *apn_val, CHAR_T
*username, CHAR_T*password)
{
    OPERATE_RET ret;

    INT8_T simid = tal_cellular_base_get_default_simid();
    ret = tal_cellular_mds_pdp_active(simid, apn_val,
username, password);
}

```

```

        if (ret == OPRT_OK) {
            tuyaa_cellular_health_check(60*10);
        }
        return ret;
    }
    STATIC OPERATE_RET ex_ota_init(VOID)
    {
        OPERATE_RET op_ret = OPRT_OK;

        // 注册OTA回调函数
        op_ret =
        tuyaa_cmiot_register_ext_fota_inform_hook(ex_fota_inform);
        if (op_ret != OPRT_OK) {
            TAL_PR_ERR("register mcu fota inform failed!");
            return op_ret;
        }

        op_ret =
        tuyaa_cmiot_register_ext_fota_process_hook(ex_fota_process)
        ;
        if (op_ret != OPRT_OK) {
            TAL_PR_ERR("register mcu fota inform failed!");
            return op_ret;
        }

        op_ret =
        tuyaa_cmiot_register_ext_fota_notify_hook(ex_fota_notify);
        if (op_ret != OPRT_OK) {
            TAL_PR_ERR("register mcu fota inform failed!");
            return op_ret;
        }

        return op_ret;
    }
    OPERATE_RET tuyaa_cloud_server(void)
    {
        // 启动拨号,用户根据实际情况选择拨号时机
        tuyaa_cell_pdp_active(NULL, NULL, NULL);

        GW_ATTACH_ATTR_T att = {0};
        //可选, 作为给外设进行OTA的通道, 通道号为
        DEV_ATTACH_MOD_2~DEV_ATTACH_MOD_10

        att.tp = DEV_ATTACH_MOD_2;
        strncpy(att.ver, dev_sw_ver, SW_VER_LEN);

        ex_ota_init();
    }

```

```

        op_ret =
tuya_cmiot_start_iot_device_mqtt_keepalive(&iot_cbs, product_key,
tuya_get_app_ver(), APP_FW_KEY, &att, 1, mqtt_alive_interval);
        if(OPRT_OK != op_ret) {
            TAL_PR_ERR("device_thread err:%d", op_ret);
            return -1;
        }

        /* tuya云服务授权, 校准校验, 未授权, 打印错误信息 */
        op_ret =
tuya_iot_cellular_dev_is_auth_info_restored();
        if (!op_ret) {
            TAL_PR_ERR("authorization information not download
%d", op_ret);
        }

        op_ret =
tuya_iot_reg_get_cellular_stat_cb(__get_cell_status);
        if(OPRT_OK != op_ret) {
            TAL_PR_ERR("tuya_iot_reg_get_cellular_stat_cb
err:%d", op_ret);
            return op_ret;
        }
    }

static void device_thread(void* arg)
{
    tuya_cloud_server();
    /* 线程执行完毕, 退出 */
    tal_thread_delete(device_TaskHandle);
}

//appimg_enter为固定的函数名称
int appimg_enter(void *param)
{
    //opencpu底层服务初始化, 必须调用
    tuya_cmiot_init();
    //opencpu底层蜂窝网络初始化, 必须调用
    tuya_cmiot_start_cellular();
    //关闭UART0作为AT命令发送口。(默认开启)

    tkl_cellular_base_ioctl(CELL_IOCTL_SET_ATUART_OFF, NULL);
    //设置应用固件的名称和版本号, 使用tuya 云服务及涂鸦产测必选

    tuya_cmiot_set_mftest_fw_info(USR_APP_NAME, USR_APP_VER);
    //启动涂鸦产测
    tuya_cmiot_start_mftest();

```

```

//低电压关机功能是否使能（可选）
tal_cellular_vbat_low_volt_poweroff_enable(FALSE);
// net_mode LED初始化服务（可选）
tuya_cellular_sys_netmode_init();
// 创建 应用 线程
cfg.priority = THREAD_PRIO_2;
cfg.stackDepth = 1024 * 8;
cfg.thrdname = "device_start";
ret = tal_thread_create_and_start(&device_TaskHandle,
NULL, NULL,
                                device_thread, NULL,
                                &cfg);
if (OPRT_OK != ret) {
    TAL_PR_ERR("[APPDEMO]: create device_start task
failed: %d", ret);
    return -1;
} else {
    TAL_PR_NOTICE("[APPDEMO]: create device_start task
success");
}
TAL_PR_INFO("init system success");
}

```

模组外设说明

UART说明

支持2个UART, uart id支持TUYA_UART_NUM_0及TUYA_UART_NUM_1

- UART2串口
 - 开机过程中，有boot的日志输出。启动完成后，就可以作为正常UART使用
- UART3串口
 - UART3_TXD 在模块开机过程中不能被拉高，否则有可能开机异常，请将其在开机过程中保持悬空状态，开机后可以拉高或拉低操作。

UART_ID	TX_PIN	RX_PIN	备注
TUYA_UART_NUM_0	78	77	硬件串口3
TUYA_UART_NUM_1	40	39	硬件串口2

GPIO说明

TUYAOS 完成移植的GPIO接口。

TUYA_GPIO_NUM_E	PIN	PIN NAME
TUYA_GPIO_NUM_0	40	UART2_TXD
TUYA_GPIO_NUM_1	53	GPIO_01
TUYA_GPIO_NUM_2	54	GPIO_02
TUYA_GPIO_NUM_3	58	GPIO_03
TUYA_GPIO_NUM_4	62	GPIO_04
TUYA_GPIO_NUM_5	41	WAKEUP_IN
TUYA_GPIO_NUM_5	61	WAKEUP_OUT

其他GPIO需要使用，请使用原生API实现。

参考文档：FIBOCOM MA510-GL-00-90 Series FibocomOpen Application Guide_V1.0.1.pdf。

GPIO复用表和PIN的对应表请参考 FIBOCOM MA510-GL-00-90 GPIO Multiplex Table V1.0.0.xls。

I2C说明

请直接使用原生接口。

参考文档：

FIBOCOM MA510-GL-00-90 Series FibocomOpen Application Guide_V1.0.1.pdf。

FIBOCOM MA510 Series Basic QAPI Application Guide_V1.0.0.PDF

SPI说明

请直接使用原生接口

参考文档：

FIBOCOM MA510-GL-00-90 Series FibocomOpen Application Guide_V1.0.1.pdf

FIBOCOM MA510 Series Basic QAPI Application Guide_V1.0.0.pdf