Overview of migrating Tuya's MCU SDK

Device Development > Access Mode MCU > Wi-Fi General Solution >

Software Reference Wi-Fi

Version: 20200226

Online Version

Contents

1 Introduction **1**

2 Precautions **2**

3 ile Structure **3**

4 Roadmap **4**

4.1 Compile the MCU basic program and migrate the SDK file. 4

4.2 Verify the macro definition in **protocol.h**. 5

4.3 Migrating the protocol.c File and Invoking Functions 8

4.4 Processing DP Data Report and Delivery Functions 9

4.5 Optimize the network configuration and indicator functions. 10

4.6 Optimize the product testing function. 13

1 Introduction

The **mcu_sdk** package contains the MCU code that is automatically generated based on product functions defined on the Tuya Smart platform. The communication and protocol resolution architecture is prepared and can be directly added to the original MCU project to quickly develop MCU programs.

2 Precautions

The SDK package has the following requirements on MCU hardware resources:

- Flash memory: 4 KB
- RAM: tens of bytes (depending on the DP data length), or 260 KB or higher if the OTA upgrade function is required
- The number of nested functions is 9.

Users without sufficient resources can implement protocol interworking without using the MCU SDK.

3 ile Structure

Execution File	Header File	Description
mcu_api.c	mcu_api	Contain Wi-Fi–related functions. Customers can invoke the functions on demand.
protocol.c	protocol.h	Protocol files that contain data processing functions. Users need to modify the two files based on project requirements.
system.c	system.h	Contain detailed implementation of the serial port communication protocol.
	wifi.h	Contains Wi-Fi–related macro definitions.

4 Roadmap

Step 1: Compile the MCU basic program and migrate the SDK file.

Step 2: Verify the macro definition in **protocol.h**.

Step 3: Migrate the **protocol.c** file and invoke functions.

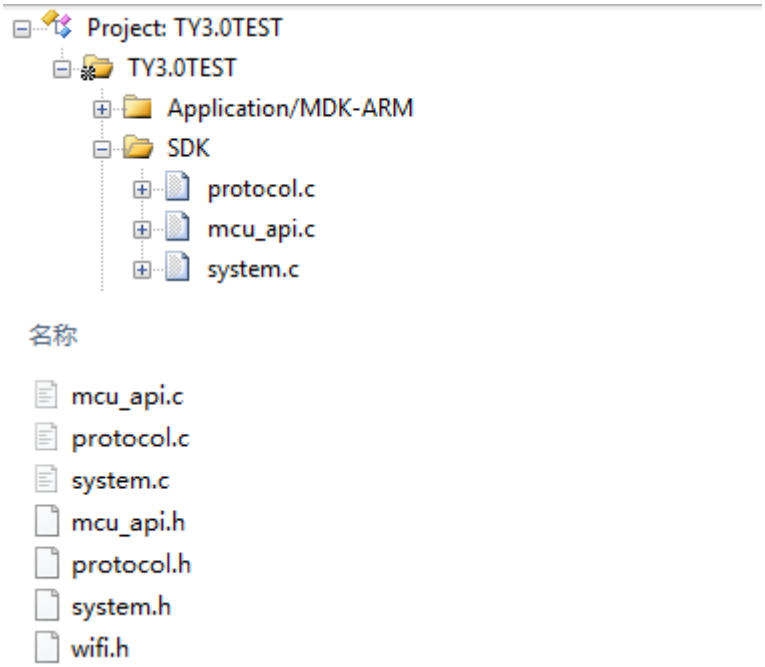
Step 4: Optimize the DP data report and delivery functions.

Step 5: Optimize the network configuration and indicator functions.

Step 6: Optimize the product testing function.

4.1 Compile the MCU basic program and migrate the SDK file.

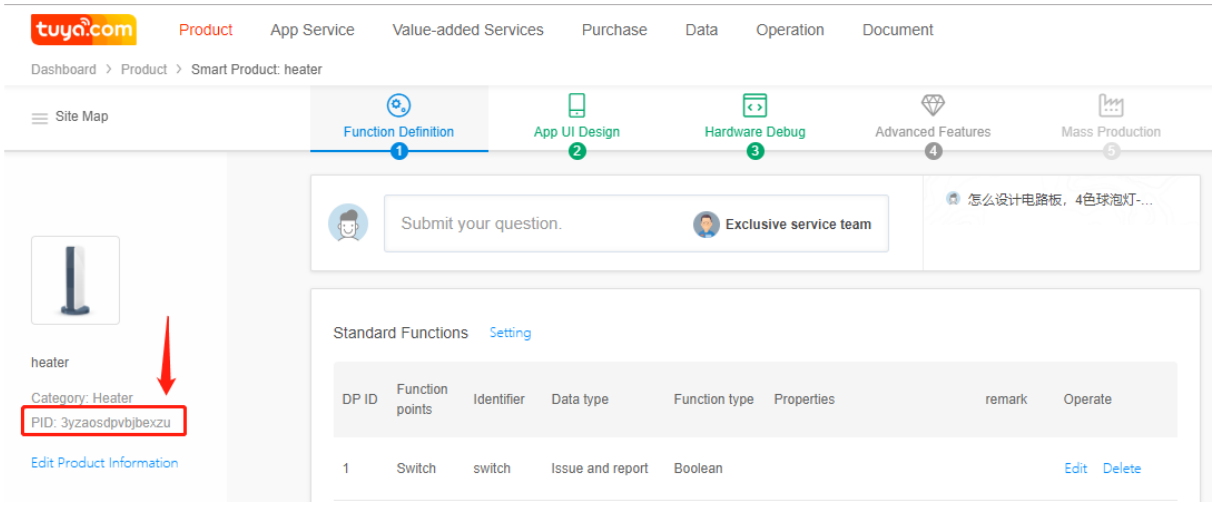
Add the .c and .h files in the mcu_sdk folder and corresponding header file reference path to the original project. Initialize MCU-related peripherals, including the serial port, external interrupt (button), and timer (indicator blinking).



4.2 Verify the macro definition in protocol.h.

4.2.1 Verify the product information

PRODUCT_KEY indicates the macro definition of the product ID (PID), which is the unique identifier of a product. Ensure that the PID is the same as that displayed on the Tuya Smart platform. If the PIDs are different, download the latest SDK package. MCU_VER indicates the software version, which is 1.0.0 by default. If the MCU requires OTA upgrade, you need to update the version number after the OTA upgrade. CONFIG_MODE indicates the network configuration mode, and the typical value is DEFAULT, indicating the default network configuration mode.



4.2.2 Check whether the MCU firmware needs to be upgraded

If OTA upgrade of MCU firmware is required, enable the firmware update macro, which is disabled by default.

```
/*****************************************************************************
                        2:MCU是否需要支固件升级
如需要支持MCU固件升级,请开启该宏
MCU可调用mcu_api.c文件内的mcu_firm_update_query()函数获取当前MCU固件更新情况
                        ********WARNING!!!**********
当前接收缓冲区为关闭固件更新功能的大小,固件升级包为256字节
如需要开启该功能,串口接收缓冲区会变大
*****************************************************************************/
//#define          SUPPORT_MCU_FIRM_UPDATE                  //开启MCU固件升级功能(默认关闭)
```

4.2.3 Define the transmitting and receiving buffers

Modify the buffer size based on the DP definition. The size of the serial port transmitting and receiving buffers must be larger than the maximum DP data length. The default size is 24 bytes. If MCU OTA upgrade is required, a 260-byte buffer is recommended. The receiving buffer size can be reduced if the RAM has insufficient space.

```
                        3:定义收发缓存:
                    如当前使用MCU的RAM不够,可修改为24
*****************************************************************************/
#ifndef SUPPORT_MCU_FIRM_UPDATE
#define WIFI_UART_QUEUE_LMT             16              //数据接收队列大小,如MCU的RAM不够,可缩小
#define WIFI_UART_RECV_BUF_LMT          128              //根据用户DP数据大小量定,必须大于24
#else
#define WIFI_UART_QUEUE_LMT             128             //数据接收队列大小,如MCU的RAM不够,可缩小
#define WIFI_UART_RECV_BUF_LMT          300             //固件升级缓冲区,需大缓存,必须大于260
#endif

#define WIFIR_UART_SEND_BUF_LMT         128              //根据用户DP数据大小量定,必须大于24
```

4.2.4 (Mandatory) Define the working mode of the Wi-Fi module

1) If the MCU controls network configuration triggering and indication, that is, the Wi-Fi reset button and Wi-Fi indicator are on the MCU side, enable cooperative processing by the Wi-Fi module and MCU (common mode) and ensure that #define is commented (the line of code starts with "//").

```
/*****************************************************************************
                    4:定义模块工作方式
模块自处理:
        wifi指示灯和wifi复位按钮接在wifi模块上(开启WIFI_CONTROL_SELF_MODE宏)
        并正确定义WF_STATE_KEY和WF_RESET_KEY
MCU自处理:
        wifi指示灯和wifi复位按钮接在MCU上(关闭WIFI_CONTROL_SELF_MODE宏)
        MCU在需要处理复位wifi的地方调用mcu_api.c文件内的mcu_reset_wifi()函数,并可调用mcu_get_reset_wifi_flag()函数返回复位wifi结果
        或调用设置wifi模式mcu_api.c文件内的mcu_set_wifi_mode(WIFI_CONFIG_E mode)函数,并可调用mcu_get_wifi_work_state()函数返回设置wifi结果
*****************************************************************************/
//#define         WIFI_CONTROL_SELF_MODE                  //wifi自处理按键及LED指示灯;如为MCU外界按键/LED指示灯请关闭该宏
```

2) If the Wi-Fi indicator and Wi-Fi reset button are on the Wi-Fi module, execute the following statement to enable processing by the Wi-Fi module: `#ifdef WIFI_CONTROL_SELF_MODE` Then, add information about the GPIO pins connected to the Wi-Fi indicator and Wi-Fi reset button, as shown in the following figure.

```
#define         WIFI_CONTROL_SELF_MODE                  //wifi自处理按键及LED指示灯;如为MCU外界按键/LED指示灯请关闭该宏
#ifdef          WIFI_CONTROL_SELF_MODE                  //模块自处理
  #define     WF_STATE_KEY            14              //wifi模块状态指示按键, 请根据实际GPIO管脚设置
  #define     WF_RESERT_KEY           0               //wifi模块重置按键, 请根据实际GPIO管脚设置
#endif
```

4.2.5 Check whether the MCU needs time verification

If the time verification function is required, enable the RTC check macro.

```
/*****************************************************************************
                    5:MCU是否需要支持校时功能
如需要请开启该宏,并在Protocol.c文件内mcu_write_rtctime实现代码
mcu_write_rtctime内部有#err提示,完成函数后请删除该#err
mcu在wifi模块正确联网后可调用mcu_get_system_time()函数发起校时功能
*****************************************************************************/
//#define         SUPPORT_MCU_RTC_CHECK               //开启校时功能
```

Write mcu_write_rtctime in the Protocol.c file to implement the code. After the Wi-Fi module successfully connects to the network, the MCU can invoke the `mcu_get_system_time()` function to initiate time verification.

4.2.6 Check whether the Wi-Fi product testing function is enabled

To ensure mass production efficiency and quality, we recommend that you enable the product testing macro. For details about implementation of the product testing function, see section 3.3.6 "Optimizing the Product Testing Function."

```
                    6:MCU是否需要支持wifi功能测试
如需要请开启该宏,并且mcu在需要wifi功能测试处调用mcu_api.c文件内mcu_start_wifitest
并在protocol.c文件wifi_test_result函数内查看测试结果,
wifi_test_result内部有#err提示,完成函数后请删除该#err
*****************************************************************************/
#define         WIFI_TEST_ENABLE                    //开启WIFI产测功能
```

4.3 Migrating the protocol.c File and Invoking Functions

1. Use #include "wifi.h" in the files (for example, the main.c file) that require Wi-Fi–related files.
2. After MCU peripherals are initialized, invoke the wifi_protocol_init() function in the mcu_api.c file.
3. Add the single-byte sending function of the MCU serial port to the uart_transmit_output function in the protocol.c file and delete #error. The following figure shows an example.

protocol.c

```
124
125 /*****************************************************************************
126 函数名称 : uart_transmit_output
127 功能描述 : 发数据处理
128 输入参数 : value:串口收到字节数据
129 返回参数 : 无
130 使用说明 : 请将MCU串口发送函数填入该函数内,并将接收到的数据作为参数传入串口发送函数
131 *****************************************************************************/
132 void uart_transmit_output(unsigned char value)
133 {
134 //  #error "请将MCU串口发送函数填入该函数,并删除该行"
135   UART3_SendByte(value);
136 /*
137   //示例:
138   extern void Uart_PutChar(unsigned char value);
139   Uart_PutChar(value);                                    //串口发送函数
140 */
141 }
```

4. Invoke the uart_receive_input function in the mcu_api.c file in the serial port receiving interrupt service function, and use the received characters as parameter input. The following figure shows an example.

```
213 void USART3_IRQHandler(void)
214 {
215   /* USER CODE BEGIN USART3_IRQn 0 */
216   unsigned char Res=0;
217
218     if((USART3->SR&UART_FLAG_RXNE) != 0)
219     {
220       Res=USART3->DR;
221       uart_receive_input(Res);
222     }
223
224 }
225
```

5. Invoke the wifi_uart_service() function in the mcu_api.c file after the MCU enters the while cycle. The following shows an example of code structure in main.c.

include "wifi.h" … void main(void) { wifi_protocol_init(); … while(1) { wifi_uart_service(); … } } Note: The MCU must directly invoke the wifi_uart_service() function in the mcu_api.c file in while. After the program is successfully initialized, it is recommended that the serial port interrupt not be disabled. If the serial port interrupt

must be disabled, ensure that the interrupt is disabled for only a short time to prevent serial port data loss. Do not invoke the report function in the interrupt.

4.4 Processing DP Data Report and Delivery Functions

4.4.1 Reporting data of all DPs

After the Wi-Fi module restarts or the network is reconfigured, the Wi-Fi module proactively delivers a status query command. The MCU needs to report the status of the device's DPs to the Wi-Fi module for synchronization. (1) Open protocol.c and locate the all_data_update(void) function. (2) Enter initial values of all DPs to be reported into corresponding report functions. The values will be displayed on the App control panel. Note: Do not invoke the all_data_update() function manually. This function is automatically invoked at a specific time.

protocol.c*

```
157
158  //自动化生成数据上报函数
159
160 /*******************************************************************************
161  函数名称 : all_data_update
162  功能描述 : 系统所有dp点信息上传,实现APP和muc数据同步
163  输入参数 : 无
164  返回参数 : 无
165  使用说明 : 此函数SDK内部需调用;
166             MCU必须实现该函数内数据上报功能;包括只上报和可上报可下发型数据
167 *******************************************************************************/
168  void all_data_update(void)
169 {
170  //  #error "请在此处理可下发可上报数据及只上报数据示例,处理完成后删除该行"
171
172    //此代码为平台自动生成，请按照实际数据修改每个可下发可上报函数和只上报函数
173    mcu_dp_bool_update(DPID_POWER,1); //BOOL型数据上报;
174    mcu_dp_value_update(DPID_TEMPSET,10); //VALUE型数据上报;
175    mcu_dp_value_update(DPID_TEMPCURRENT,10); //VALUE型数据上报;
176    mcu_dp_enum_update(DPID_MODE,0); //枚举型数据上报;
177    mcu_dp_bool_update(DPID_ECO,0); //BOOL型数据上报;
178    mcu_dp_bool_update(DPID_CHILDLOCK,1); //BOOL型数据上报;
179    mcu_dp_raw_update(DPID_PROGRAM,RAWBuffer,54); //RAW型数据上报;
180    mcu_dp_value_update(DPID_FLOORTEMP,20); //VALUE型数据上报;
181    mcu_dp_enum_update(DPID_TEMPSWITCH,1); //枚举型数据上报;
182    mcu_dp_bool_update(DPID_FLOORTEMPFUNCTION,0); //BOOL型数据上报;
183 }
```

4.4.2 Reporting data of a single DP

When the status of a DP is changed, the MCU proactively reports the new DP status to the Wi-Fi module, and the DP status displayed on the App will be updated accordingly. The report data format is mcu_dp_xxxx_updata(DPID_X,n). DPID_X indicates the DP whose status has changed. Functions in all_data_update() can be independently invoked. Example: mcu_dp_bool_update(DPID_SWITCH,1); //Boolean data reporting mcu_dp_value_update(DPID_TEMPER_SET,25); //Value data reporting mcu_dp_string_update(DPID_DAY,"1234",4); //String data reporting

4.4.3 DP data delivery

Each deliverable DP has an independent data delivery processing function in the protocol.c file. The function format is dp_download_xxx_handle(), and xxx indicates a deliverable DP. After the function parses a DP, the MCU performs logical control in the corresponding position. The following shows an example of receiving switch data.

```
/*****************************************************************************
函数名称 : dp_download_switch_handle
功能描述 : 针对DPID_SWITCH的处理函数
输入参数 : value:数据源数据
         : length:数据长度
返回参数 : 成功返回:SUCCESS/失败返回:ERROR
使用说明 : 可下发可上报类型,需要在处理完数据后上报处理结果至app
*****************************************************************************/
static unsigned char dp_download_switch_handle(const unsigned char value[], unsigned short length)
{
  //示例:当前DP类型为BOOL
  unsigned char ret;
  //0:关/1:开
  unsigned char  switch1;

  switch1 = mcu_get_dp_download_bool(value,length);
  if(switch1 == 0)
  {
    MCU_ON_switch1();//开关关
  }
  else
  {
    MCU_OFF_switch1();//开关开
  }

  //处理完DP数据后应有反馈
  ret = mcu_dp_bool_update(DPID_SWITCH,switch1);
  if(ret == SUCCESS)
    return SUCCESS;
  else
    return ERROR;
}
```

The MCU uses MCU_ON_switch1() and MCU_OFF_switch1() to turn on and off a switch, respectively. When the device status is changed under non-App control, the MCU invokes mcu_dp_bool_update(DPID_SWITCH_1,switch_1) to upload the real status of the switch. Typically, the receiving processing function automatically invokes the function.

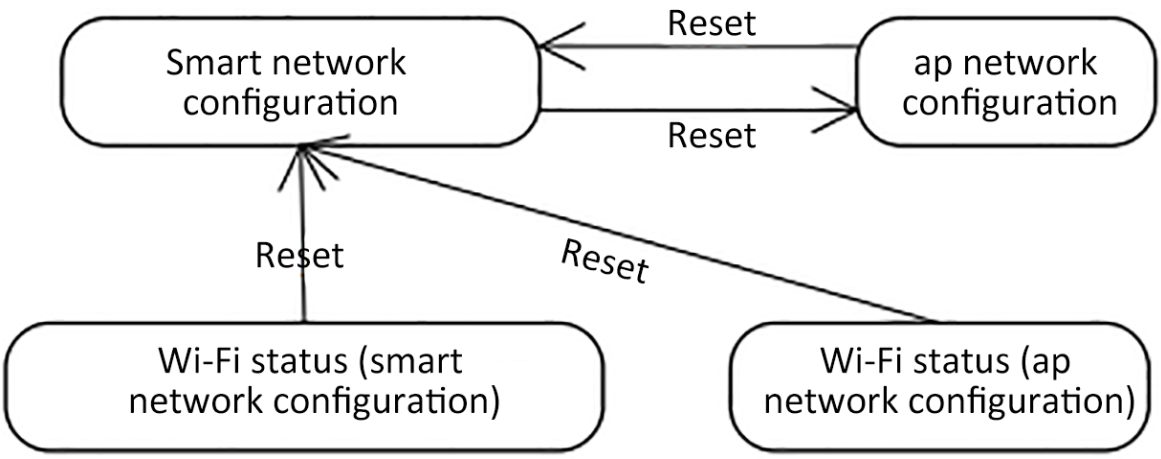
4.5 Optimize the network configuration and indicator functions.

Skip this section if processing by the Wi-Fi module is used. When protocol migration is successful, the network configuration command and indicator function need to be optimized for network configuration. In mode of cooperative processing by the Wi-Fi module and MCU, the MCU can select the network configuration triggering and indication modes based on actual requirements. Typically, network configuration is triggered by the Wi-Fi reset button and indicated by quick or slow blinking of the

Wi-Fi indicator. We recommend that you enable both network configuration modes for your product. Smart network configuration mode: The operation is simple and convenient, and the Wi-Fi indicator blinks quickly. AP network configuration mode: Network configuration is reliable, and the Wi-Fi indicator blinks slowly.

4.5.1 Network configuration command

The network configuration command can be implemented by the mcu_reset_wifi() and mcu_set_wifi_mode() functions. Typically, these two functions are invoked in the button processing function after the button is pressed for network configuration. After mcu_reset_wifi() is invoked, the Wi-Fi module is reset and the previous network configuration information is cleared. The function invoking also triggers a switchover between the AP and smart network configuration modes.



After mcu_set_wifi_mode() with parameter SMART_CONFIG or AP_CONFIG is invoked, the network configuration information is cleared, and smart or AP network configuration mode is used. This function has the same function as the mcu_reset_wifi() function. You can select one as needed.

4.5.2 Network configuration indication

Typically, the mcu_get_wifi_work_state() function is invoked at while(1) to return the Wi-Fi status. Then, you write the indicator blinking mode in based on the Wi-Fi status.

Device Network Connection Status	Description	Status Value	LED Indicator Status
State 1	Smart network configuration	0x00	The indicator blinks at 250 ms intervals.
State 2	AP network configuration	0x01	The indicator blinks at 1500 ms intervals.
State 3	The Wi-Fi is configured. However, the device fails to connect to the router.	0x02	The indicator is off.
State 4	The Wi-Fi is configured, and the device successfully connects to the router.	0x03	The indicator is steady on.
State 5	The device connects to the router and cloud.	0x04	The indicator is steady on.
State 6	The Wi-Fi device is in low power consumption mode.	0x05	The indicator is off.

Invoke the mcu_get_wifi_work_state() function to obtain the Wi-Fi status. The function architecture is as follows:

```c
void main(void)
{
    ...

    while(1)
    {
        switch(mcu_get_wifi_work_state())
        {
            case SMART_CONFIG_STATE:
            //smart config configuration state: LED flash quickly; the user needs to complete the configuration
            break;
            case AP_STATE:
            //AP configuration state: LED flash slowly
            break;
            case WIFI_NOT_CONNECTED:
            //Wi-Fi configuration is finished; the router is being connected to; LED keeps long dark
            break;
            case WIFI_CONNECTED:
            //The router is successfully connected to; LED keeps long bright
            break;
            default:break;
        }
        ...
    }
}
```

4.6 Optimize the product testing function.

Notes: product test is used only in production, and it is mainly used to test the Wi-Fi function of modules and the communication capability of module and control panel.

The test requires no network connection, and the product test process is triggered by pressing button. The test process takes about 5s.

Note:

1. There shall be as less routers as possible to speed up test of products. 2、Please wait for 2 seconds after the system is connected to the power supply so that the module is started. 3、Product test is not required in daily use.

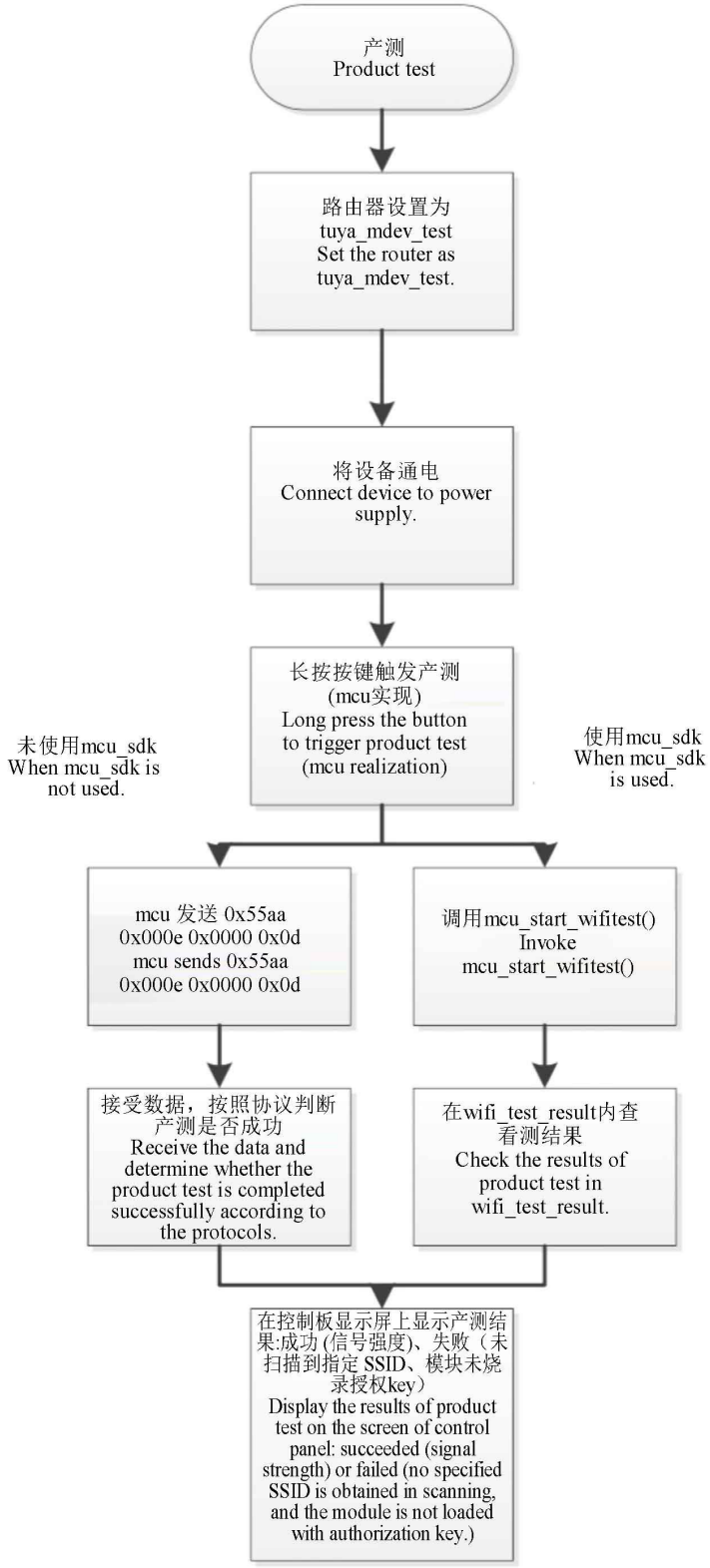


Figure 1: cmd-markdown-logo

4.6.1 Preparation

One 2.4G wireless router and power supply, and network connection is not required. The SSID of router will be set to **tuya_mdev_test**, and it will be placed in the workshop of the production line.

4.6.2 Connect the device to be tested to the power supply.

4.6.3 Trigger the product test.

mcu will press button and hold it to trigger product test (the trigger mode is realized by mcu, it is recommended to use key combination that is uncommon or press some keys for a long period to trigger product test function).

Then the following interfaces will be invoked to trigger product test:

- When mcu_sdk is used, mcu invokes `mcu_start_Wi-Fi test()`
  - When mcu_sdk is not used, mcu sends`0x55 0xaa 0x00 0x0e 0x0000 0x0d`

4.6.4 Check results of test

The following methods can be used to check test results based on whether the mcu_sdk provided by Tuya is used.

mcu_sdk is used. Check results of tests in the `wifi_test_result()`function of the protocol.c file.

```
1 void wifi_test_result(unsigned char result,unsigned char rssi)
2 {
```

#error "Please add your own codes for successful or failed Wi-Fi function test and delete this line when codes are added."

if(result == 0) { //Test failed if(rssi == 0x00) { //The tuya_mdev_test router is not found in the scanning, please check it } else if(rssi == 0x01) { //The module is not authorized } } else { //Test succeeded //rssi represents signal strength (0-100, 0 represents the weakest signal, and 100 represents the strongest signal) } }

mcu_sdk is not used Check results of tests according to received data.

WiFi功能 产测（注：扫描 tuya_mde v_test的 指定SSID）	MCU上报	0x55aa 0x00	0x0e	0x0000		0x0d
	模块发送	0x55aa 0x00	0x0e	0x0002	数据长度为2字节： Data[0]:0x00失败，0x01成功；当Data[0]为0x01，即成功时， Data[1]表示信号强度（0-100，0信号最差，100信号最强）当Data[0]为0x00，即失败时， Data[1]为0x00表示未扫描到指定的ssid， Data[1]为0x01 表示模块未烧录授权key	校验和

Figure 2: cmd-markdown-logo

4.6.5 Display test results

The tests have three kinds of test results, and test results will be displayed on the display screen of the control panel.

1. Signal strength
2. The tuya_mdev_test router is not found in the scanning, please check it
3. he module is not authorized