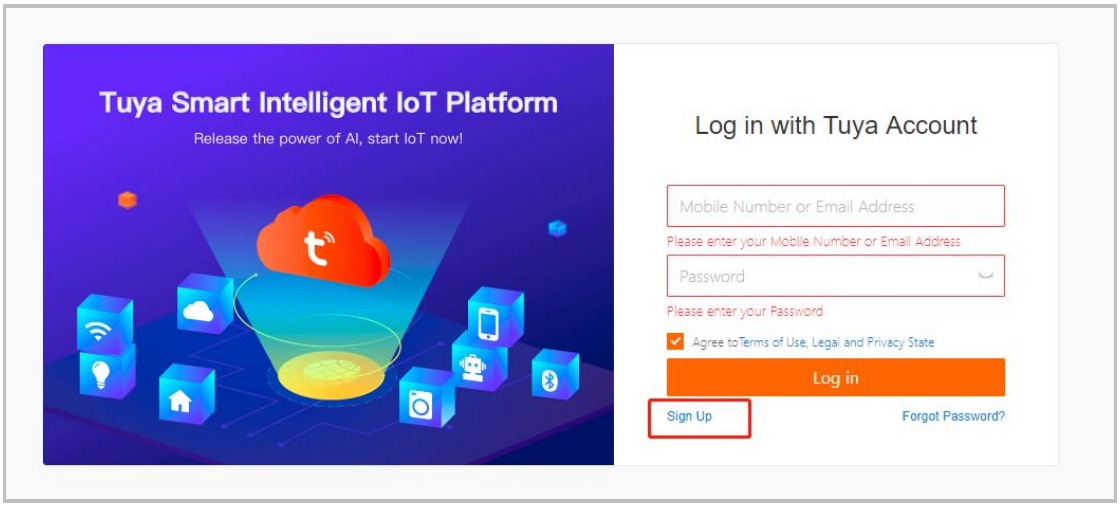
Guide to Interworking with the Tuya MCU

Contents

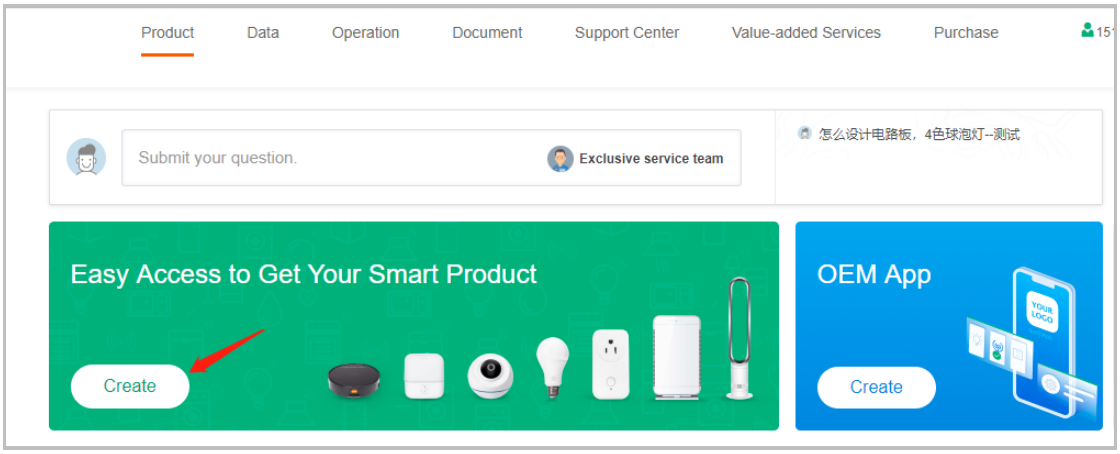
1 Creating a Product on the Tuya Smart Platform and Downloading the MCU Development Package

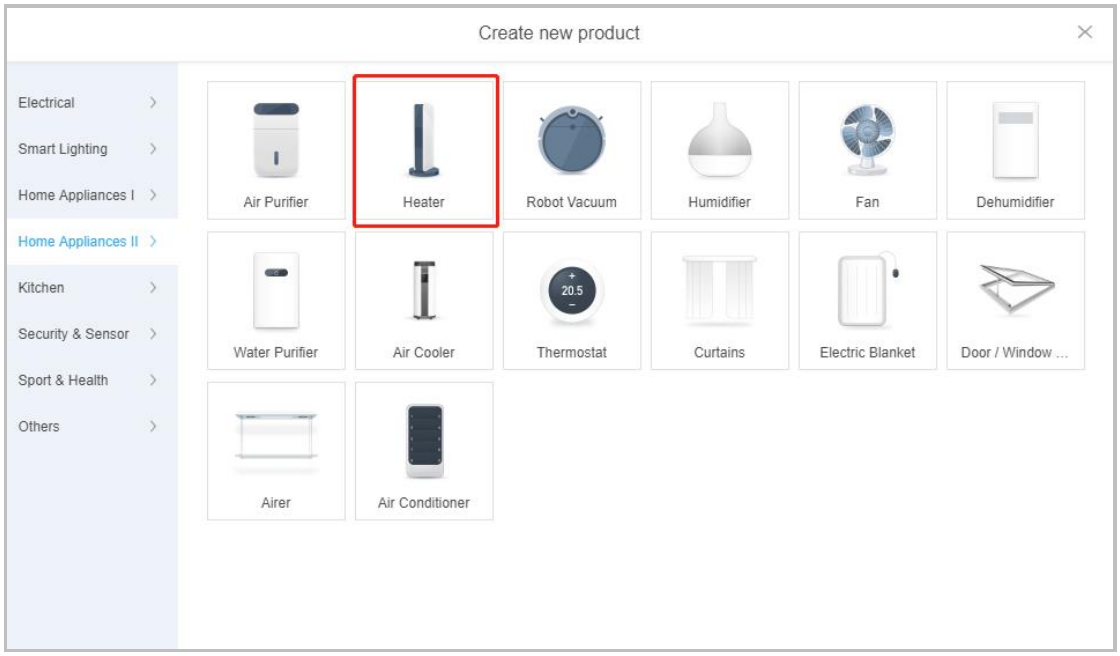
This section describes how to create a heater product on the Tuya Smart platform as an individual developer.

1) Visit https://iot.tuya.com and register a developer account. Then, log in to the platform using the account.

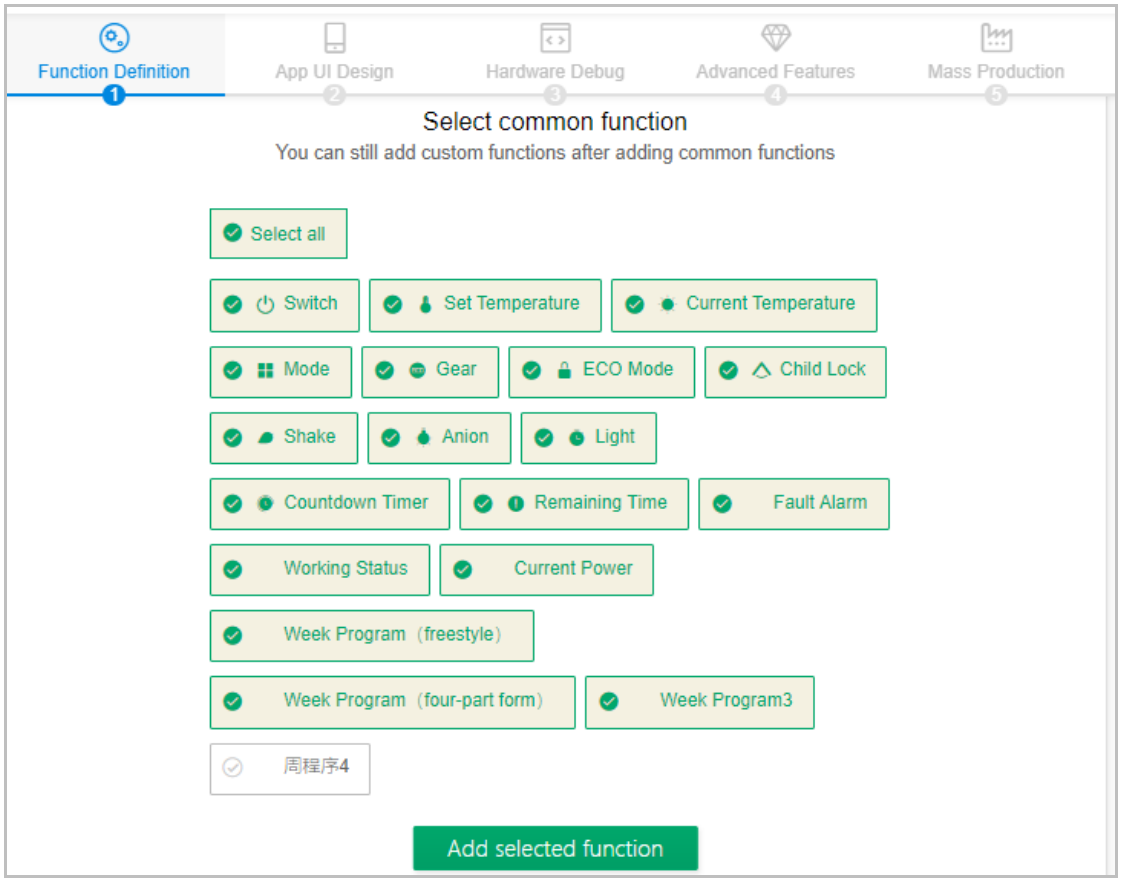


2) Click **Create** on the **Product** page. Choose **Home Appliances II** and select **Heater**.

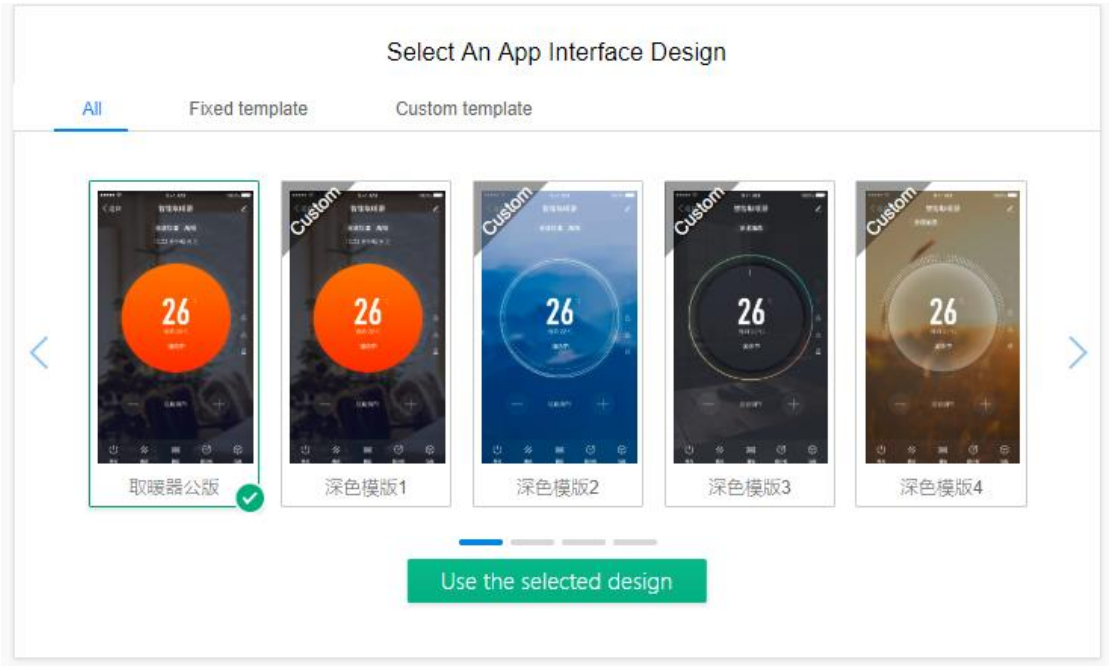




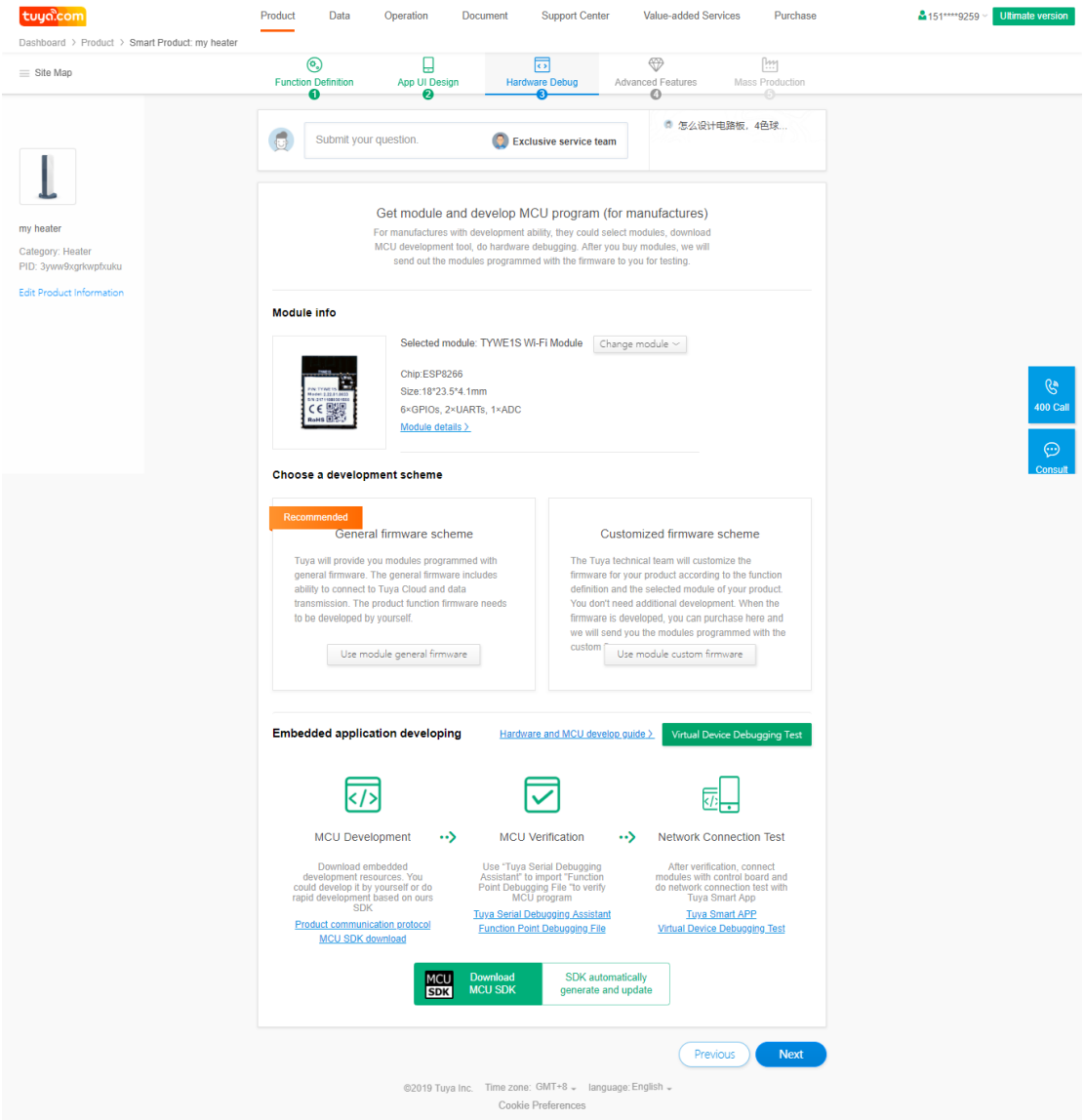
3) Select data points (DPs) based on product requirements. If there are custom functions, add them as required. For details about custom product functions, see https://docs.tuya.com/en/product/function.html.



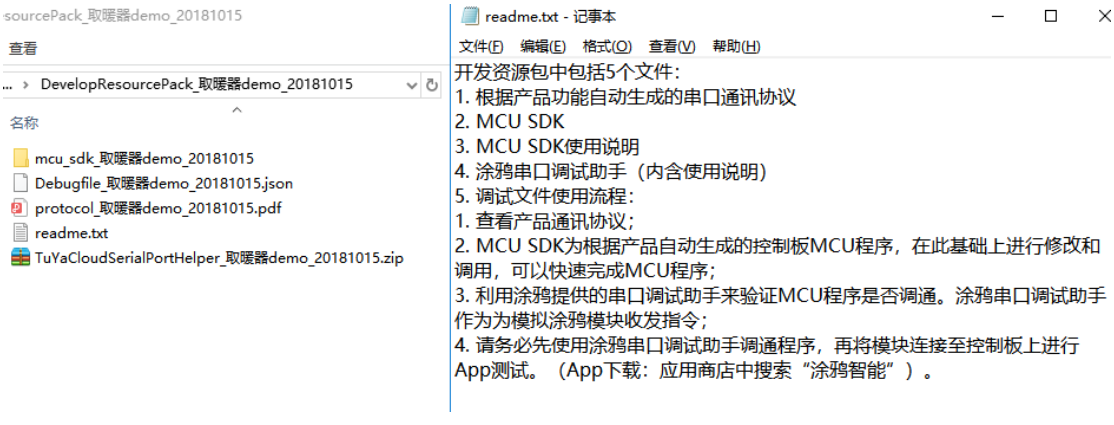
4) Select a favorite app control panel template and scan the QR code to verify the effect. More panel templates are provided for enterprise accounts. To upgrade your account, contact Tuya business personnel.


Select An App Interface Design
All
Fixed template
Custom template
26
Custom
取暖器公版
深色模版1
深色模版2
深色模版3
深色模版4
Use the selected design

5) Download an MCU development package with one click.



6) Check the downloaded package. The following figure shows the materials contained in an MCU development package.



2 Protocol Resolution

Protocols are classified into basic and functional protocols. Basic protocols are independent of products. They are common protocols of modules and include module initialization commands and some extended functional commands. Functional protocols are DP data transmitting and receiving commands that the platform automatically generates based on the definition of each product DP.

2.1 Protocol Framework

The MCU interworks with the Wi-Fi module through a serial port and common firmware. Settings of the communication parameters are as follows:

- Bits per second: 9600
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow control: None

The following table describes the data frame format.

Field	Length (Byte)	Description
Frame header	2	Fixed value of 0x55aa
Version	1	Used during upgrade and extension
Command word	1	Detailed frame type
Data length	2	Big endian
Data	xxxx	
Checksum	1	Reminder of the byte sum starting from the frame header to 256

The following table describes command words.

Command Word	Description
0x00	Heartbeat detection.
0x01	Query product information.
0x02	Query the working mode of the Wi-Fi module.
0x03	Report the network connection status of the device.
0x04	Reset the Wi-Fi module and switch the network configuration mode.
0x05	Reset the Wi-Fi module and select a network configuration mode.
0x06	Deliver DP commands.
0x07	Report DP status.
0x08	Query the device initialization status.
0x0a	(Optional) Start OTA upgrade.
0x0b	(Optional) Transmit the OTA upgrade package.

0x1c	(Optional) Obtain the local time.
0x0e	Test the Wi-Fi function (product testing command).

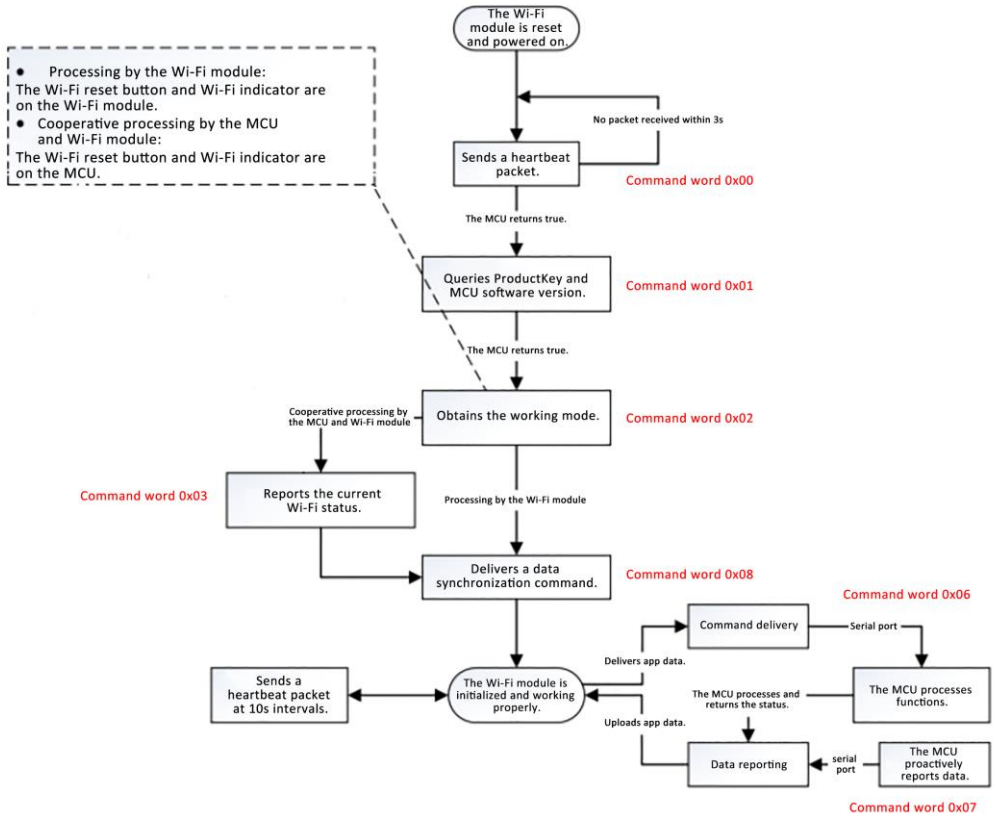
2.1.1 Basic Protocol

Basic protocols are the same for each product and mandatory for the Wi-Fi module. Basic protocols include heartbeat detection and query for product information, working mode of the Wi-Fi module, and Wi-Fi status.

Command words 0x00 to 0x08 are basic commands of the Wi-Fi module. Command words 0x0a to 0x0e are basic functions of the Wi-Fi module, including the MCU OTA upgrade, local time acquisition, and product testing.

To enable the Wi-Fi module to work properly, you need to initialize the module and configure the network connection.

The following figure shows command words involved in the module initialization protocol and the initialization process.



After being powered on, the Wi-Fi module sends heartbeat packets continuously. After the MCU responds, the preceding initialization process starts.

1. **Heartbeat detection:** After the MCU is powered on, it returns 0x00 for the first heartbeat packet and 0x01 for the second and later heartbeat packets. After receiving 0x00, the Wi-Fi module is automatically initialized for data synchronization. Later heartbeat packets are used to determine whether the device is online and automatically connects to the network upon disconnection.

		Frame Header	Version	Command Word	Data	Checksum
Heartbeat detection	Sent by the Wi-Fi module	0x55aa	0x00	0x0000		0xff
	Reported by the MCU	0x55aa	0x03	0x0001	**0x00 (first packet) or 0x01 (later packets)**	Checksum

For example, the Wi-Fi module sends 55 aa 00 00 00 00 ff, and the MCU returns 55 aa 03 00 00 01 00 03 for the first packet and 55 aa 03 00 00 01 01 04 for other packets.

2. **Query product information.** After receiving a heartbeat response, the Wi-Fi module sends a command to query the product information. The MCU reports the product information, including the PID, version, and mode. Note that characters, such as curly brackets ({}), colons (:), and double quotation marks ("") also need to be included. For details about the format, see the following table.

Querying product information	Sent by the Wi-Fi module	0x55aa 0x00	0x01	0x0000		0x00
	Reported by the MCU	0x55aa 0x03	0x01	0x0015 (0x0010-0x0018)	Mode: 0: default network configuration 1: low power consumption 2: special network configuration Format: {“p”:“svizlf0dzs4rz85c”, “v”:“1.0.0”, “m”:0}	Checksum

For example, the Wi-Fi module sends 55 aa 00 01 00 00 00, and the MCU returns PID:RN2FVAgXG6WfAktU, as shown in the following figure. You need to convert your product ID to an ASCII code and use the ASCII code to replace RN2FVAgXG6WfAktU.

Example: {"p":"RN2FVAgXG5WfAktU", "v":"1.0.0", "m":0}

"p" indicates the product ID, and the value is RN2FVAgXG5WfAktU. "v" indicates the MCU version, and the value is 1.0.0. "m" indicates the network configuration mode, and the value is 0 (the value 0 indicates default network configuration, the value 1 indicates low power consumption, and the value 2 indicates special network configuration).

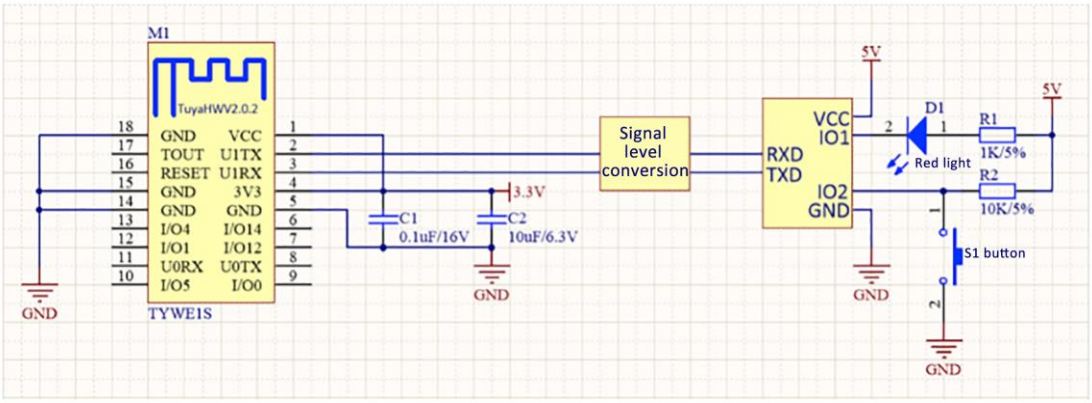
55	aa	03	01	00	2a	7b	22	70	22	3a	22	52	4e	32	46
Frame Header						{	"	P	"	:	"	R	N	2	F
56	41	67	58	47	36	57	66	41	6b	74	55	22	2c	22	76
V	A	g	X	G	6	W	f	A	k	t	U	"	,	"	v
22	3a	22	31	2e	30	2e	30	22	2c	22	6d	22	3a	30	7d
"	:	"	1	.	0	.	0	"	,	"	m	"	:	0	}
0c															
Parity Bit															

3. **Query the working mode of the set Wi-Fi module.** After receiving the product information, the Wi-Fi module sends command word 0x02 to query the working mode of the set Wi-Fi module.

The working mode of the Wi-Fi module instructs how to show the Wi-Fi status and how to reset

the Wi-Fi module.

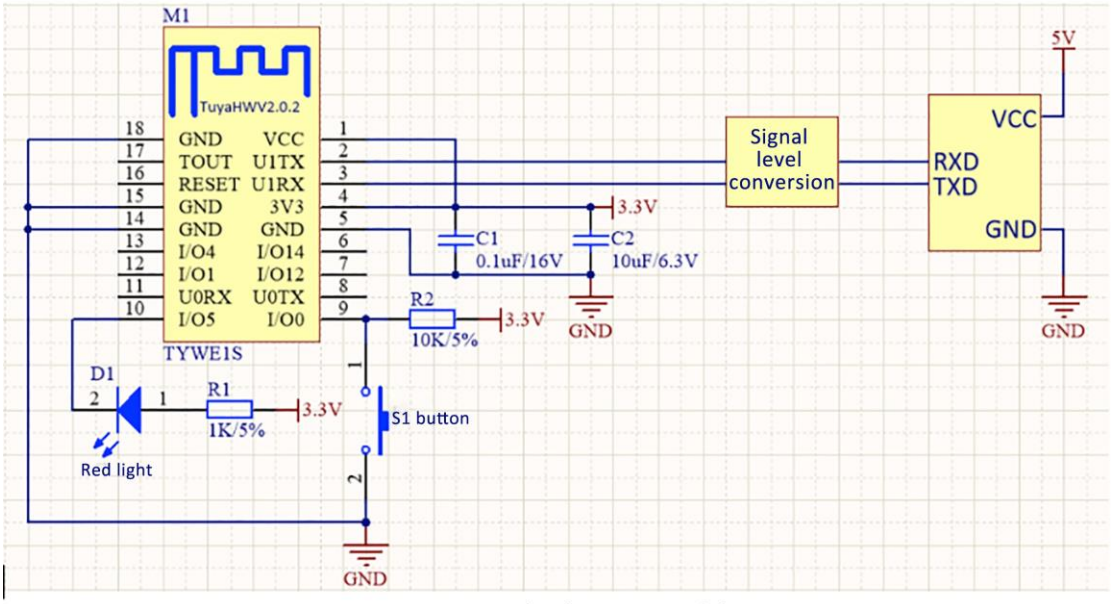
a. Cooperative processing by the MCU and Wi-Fi module The Wi-Fi module notifies the MCU of the current Wi-Fi status over a serial port. The MCU controls status of the Wi-Fi indicator.



Cooperative processing by the MCU and Wi-Fi module

b. Processing by the Wi-Fi module The GPIO pins of the Wi-Fi module change status of the Wi-Fi indicator (LED indicator). The Wi-Fi module is reset based on the GPIO inputs.

In processing by the Wi-Fi module mode, the Wi-Fi module triggers a reset when it detects that the GPIO input is at a low level for more than 5s. GPIO pins used by the Wi-Fi indicator and Wi-Fi reset button are configured by using command word 0x02.



Processing by the Wi-Fi module

If the MCU selects cooperative processing by the MCU and Wi-Fi module, it reports 0. If the MCU selects processing by the Wi-Fi module, it reports the I/O interfaces of the Wi-Fi indicator and Wi-Fi reset button. If the MCU selects processing by the Wi-Fi module, the following steps 4 to 6 can be ignored.

Querying the working mode of the set Wi-Fi module	Sent by the Wi-Fi module	0x55aa 0x00	0x02	0x0000		0x01
	Reported by the MCU (cooperative processing by the MCU and Wi-Fi module)	0x55aa 0x03	0x02	0x0000		Checksum
	Reported by the MCU (processing by the Wi-Fi module)	0x55aa 0x03	0x02	0x0002	The first and second bytes indicate the GPIO pin SNs of the Wi-Fi status indicator and Wi-Fi reset button, respectively.	Checksum

For example, the Wi-Fi module sends 55 aa 00 02 00 00 01.

The MCU returns 55 aa 03 02 00 00 04 (cooperative processing by the MCU and Wi-Fi module) or 55 aa 03 02 00 02 05 00 0b (processing by the Wi-Fi module). "05" and "00" indicate the I/O interfaces 5 and 0 that are connected to the Wi-Fi indicator and Wi-Fi reset button, respectively.

4. **Report the Wi-Fi status**. When the Wi-Fi module detects that the MCU restarts or the Wi-Fi status is changed, the Wi-Fi module proactively reports the Wi-Fi status to the MCU. Based on the Wi-Fi status indicated by the command word 0x03, the MCU controls blinking of the Wi-Fi indicator. The following table describes six states in the protocol V03.

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.

Reporting the Wi-Fi status	Sent by the Wi-Fi module	0x55aa 0x00	0x03	0x0001	Indicates the Wi-Fi status. 0x00: smart network configuration, in which mode, the indicator blinks quickly 0x01: AP network configuration, in which mode, the indicator blinks slowly 0x02: Wi-Fi configuration is successful, but the device fails to connect to the router. The indicator is off. 0x04: The device connects to the router and cloud. The indicator is steady on.	Checksum
	Reported by the MCU	0x55aa 0x03	0x03	0x0000		Checksum

For example, the Wi-Fi module sends the checksum of 55 aa 00 03 00 01 01 ("01" indicates AP network configuration), and the MCU returns 55 aa 03 03 00 00 05.

5. **Reset the Wi-Fi module**.

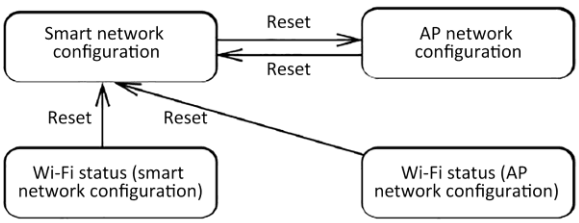
   Network configuration command: You can reset the Wi-Fi module to enable the device to enter network configuration state. The network configuration modes include:

a. Smart network configuration, in which mode, the Wi-Fi indicator blinks quickly. This mode is simple and convenient.

b. AP network configuration, in which mode, the Wi-Fi indicator blinks slowly. This mode is stable and reliable.

We recommend that you use both modes. You can customize the triggering mechanism to control quick or slow blinking of the Wi-Fi indicator.

When receiving command word 0x04 sent from the MCU, the Wi-Fi module changes the network configuration mode. The default mode is smart network configuration, and the Wi-Fi module switches between the smart and AP network configuration modes.



Resetting the Wi-Fi module	Sent by the MCU	0x55aa 0x03	0x04	0x0000		Checksum
	Reported by the Wi-Fi module	0x55aa 0x00	0x04	0x0000		0x03

For example, the MCU sends 55 aa 03 04 00 00 06, and the Wi-Fi module returns 55 aa 00 04 00 00 03.

6. **Reset the Wi-Fi module and select a network configuration mode**. Based on the parameters

sent by the MCU, the Wi-Fi module selects the smart or AP network configuration mode. Similar to command word 0x04, the command can be used for network configuration. It also enables the Wi-Fi module to select a network configuration mode.

Resetting the Wi-Fi module and selecting a network configuration mode	Reported by the MCU (smart network configuration mode)	0x55aa 0x03	0x05	0x0001	0x00	Checksum
	Sent by the MCU (AP network configuration mode)	0x55aa 0x03	0x05	0x0001	0x01	Checksum
	Sent by the Wi-Fi module	0x55aa 0x00	0x05	0x0000		0x04

For example, the MCU sends 55 aa 03 05 00 01 00 08 (indicating the smart network configuration mode) or 55 aa 03 05 00 01 01 09 (indicating the AP network configuration mode), and the Wi-Fi module returns 55 aa 00 05 00 00 04.

7. **Query the MCU working status.** The Wi-Fi module uses command word 0x08 to query the status of all MCU DPs as the initial values that are displayed on the app. After receiving the command word, the MCU reports data of all DPs one by one. The Wi-Fi module queries the DP status in the following scenarios:

a. The Wi-Fi module is powered on for the first time and connects to the MCU through heartbeat packets.

b. The Wi-Fi module detects that the MCU has restarted or gone offline and then online.

Querying the MCU working status	Sent by the Wi-Fi module	0x55aa 0x00	0x08	0x0000		Checksum
	Reported by the MCU	0x55aa 0x03	0x07	N	Report data of all DPs as the initial values to be displayed on the app.	Checksum

For example, the Wi-Fi module sends 55 aa 00 08 00 00 07, and the MCU returns the checksum of 55 aa 03 07 N **** (DP 1), the checksum of 55 aa 03 07 N **** (DP 2), or the checksum of ... (DP *N*).

8. **Test product functions.** The product testing command is used to test the RF performance of the Wi-Fi module during mass production of the product. We recommend that you invoke the product testing command 5s after the Wi-Fi module is powered on and initialized. After receiving the product testing command, the Wi-Fi module automatically searches for the "tuya_mdev_test" WLAN network and returns the search result with the signal strength (0 to 100 with a step of 20).

Testing the Wi-Fi function (Note: Scan the specified SSID of "tuya_mdev_test".)	Reported by the MCU	0x55aa 0x03	0x0e	0x0000		Checksum
	Sent by the Wi-Fi module	0x55aa 0x00	0x0e	0x0002	The data contains two bytes. If Data[0] is 0x00, the test fails. If Data[0] is 0x01, the test is successful. When Data[0] is 0x01, Data[1] indicates the signal strength, and its value range is from 0 to 100. A larger value indicates a higher signal strength, and the value 100 indicates the strongest signal strength. When Data[0] is 0x00 and Data[1] is 0x00, the specified SSID is not scanned. When Data[0] is 0x00 and Data[1] is 0x01, the authorized key is not burnt into the Wi-Fi module.	Checksum

For example, the MCU sends the checksum of 55 aa 03 0e 00 00, and the Wi-Fi module returns 55 aa 00 0e 00 02 01 28 38, indicating that the product testing is successful and that the signal strength is 40.

2.1.2 Functional Protocol

Functional protocols are used for delivering and reporting DP data. The command word for the Wi-Fi module to deliver DP data is 0x06, and that for the MCU to report DP data is 0x07.
After receiving a data delivery command, the MCU performs corresponding logical control. When the DP status is changed, the MCU reports the DP data and changes the DP status displayed on the app. The Wi-Fi module filters out duplicated DP data that the MCU reports.
Example:

ID	Function		Frame Header Version	Command Word	Data Length	dpID	Date Type	Function Length	Function Command	Verification Method
1	Switch	Sent by the Wi-Fi module	0x55aa 0x00	0x06	0x00 0 x05	0x01	0x01	0x00 0 x01	off:0x00 on:0x01	Checksum
		Reported by the MCU	0x55aa 0x03	0x07	0x00 0 x05	0x01	0x01	0x00 0 x01		Checksum
2	Target temperature	Sent by the Wi-Fi module	0x55aa 0x00	0x06	0x00 0 x08	0x02	0x02	0x00 0 x04	0x1e-0x50	Checksum
		Reported by the MCU	0x55aa 0x03	0x07	0x00 0 x08	0x02	0x02	0x00 0 x04		Checksum
11	Remaining countdown time	Sent by the Wi-Fi module	0x55aa 0x00	0x06	0x00 0 x05	0x0b	0x04	0x00 0 x01	1hour:0x00 2hour:0x01 3hour:0x02	Checksum
		Reported by the MCU	0x55aa 0x03	0x07	0x00 0 x05	0x0b	0x04	0x00 0 x01		Checksum
13	Fault Alarm	Reported by the MCU	0x55aa 0x03	0x07	0x00 0 x06	0x0d	0x05	0x00 0 x02	bit0:1 bit1:2 bit2:4 bit3:8 bit4:16 bit5:32 bit6:64 bit7:128 bit8:512	Checksum
17	Week program	Sent by the Wi-Fi module	0x55aa 0x00	0x06	N	0x11	0x00	N	0x00-0xff	Checksum
		Reported by the MCU	0x55aa 0x03	0x07	N	0x11	0x00	N		Checksum
102	Data string	Sent by the Wi-Fi module	0x55aa 0x00	0x06	N	0x66	0x03	N	0x00-0xff	Checksum
		Reported by the MCU	0x55aa 0x03	0x07	N	0x66	0x03	N		Checksum

Note:

a. Value data has four bytes. If a value contains less than four bytes, 0 is supplemented before the value.

For example, if the MCU sends the checksum of 55 aa 03 07 00 08 02 02 00 04 00 00 00 1e, the target temperature is 30°C.

b. Alarm data can contain multiple alarms reported simultaneously. Each bit represents an alarm. The value 1 indicates that the fault occurs, and the value 0 indicates that the fault does not occur.

For example, if the MCU sends the checksum of 55 aa 03 07 00 06 0d 05 00 02 00 09, the faults represented by bit 0 and bit 3 occur.

c. The meaning and display of string data must be the same as that on the panel. Customized string data needs to be negotiated with the panel developer.

d. Raw data is transparent and typically used for implementing complex functions. We do not recommend that you use raw data yourself.

3 Migrating Tuya's MCU SDK

3.1 Precautions

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.

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.

Execution File	Header File	Description
mcu_api.c	mcu_api.h	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.

3.2 Roadmap

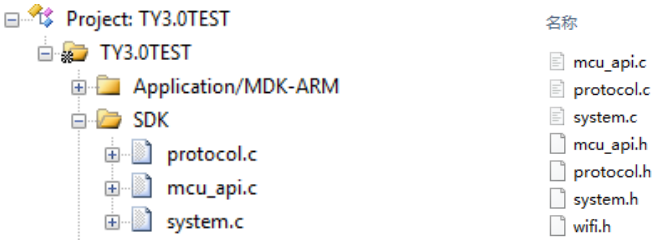
Step 1: Compile the MCU basic program and migrate the SDK file.
Step 2: Verify the macro definition in **protocol.h**.
Step 3: Migrate the **protocol.c** file and invoke functions.
Step 4: Optimize the DP data report and delivery functions.
Step 5: Optimize the network configuration and indicator functions.
Step 6: Optimize the product testing function.

3.3 Procedure

3.3.1 Compiling the MCU Basic Program and Migrating 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).



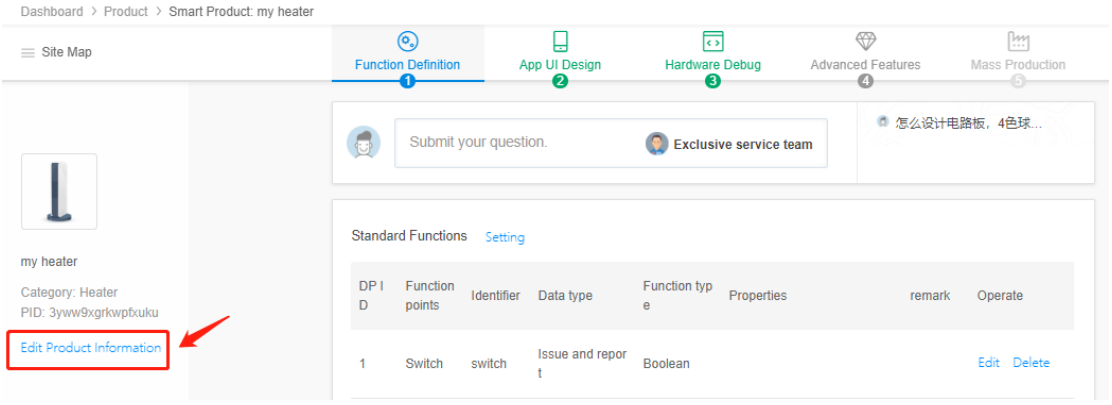
3.3.2 Verifying the Macro Definition in protocol.h

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.



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)
```

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. (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
```

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.

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                1                  //开启WIFI产测功能
```

3.3.3 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

```
/*******************************************************************************
函数名称 : uart_transmit_output
功能描述 : 发数据处理
输入参数 : value:串口收到字节数据
返回参数 : 无
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
*******************************************************************************/
void uart_transmit_output(unsigned char value)
{
//  #error "请将MCU串口发送函数填入该函数,并删除该行"
  UART3_SendByte(value);
/*
  //示例:
  extern void Uart_PutChar(unsigned char value);
  Uart_PutChar(value);                                    //串口发送函数
*/
}
```

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.

```
void USART3_IRQHandler(void)
{
  /* USER CODE BEGIN USART3_IRQn 0 */
  unsigned char Res=0;

    if((USART3->SR&UART_FLAG_RXNE) != 0)
    {
      Res=USART3->DR;
      uart_receive_input(Res);
    }

}
```

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.

3.3.4 Processing DP Data Report and Delivery Functions

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 }
```

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
```

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.

3.3.5 Optimizing Network Configuration and Indicator Functions

When protocol migration is successful, the network configuration command and indicator function need to be optimized for network configuration. Skip this section if processing by the Wi-Fi module is used.

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.

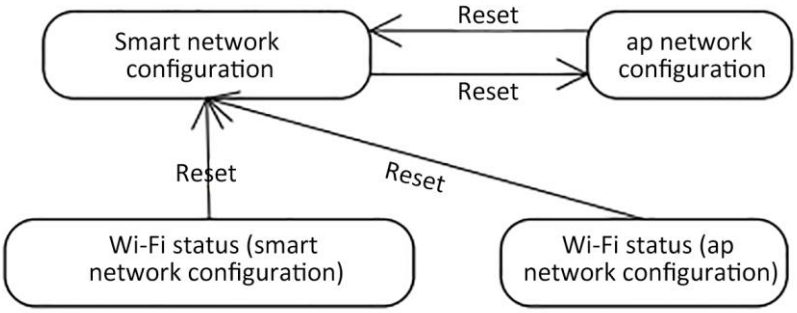
Smart network configuration mode: The operation is simple and convenient, and the Wi-Fi indicator blinks quickly.

AP network configuration mode: Network configuration is reliable, and the Wi-Fi indicator blinks slowly.

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.

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	0x04	The indicator is

	and cloud.		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:

```
void main(void)

{           ...

    while(1)
    {
        ...
        switch(mcu_get_wifi_work_state())
        {
            case SMART_CONFIG_STATE:
                //Smart network configuration mode, and the Wi-Fi indicator blinks quickly. You need to enter the Wi-Fi indicator status control code.
            break;
            case AP_STATE:
                //AP network configuration mode, and the Wi-Fi indicator blinks slowly. You need to enter the Wi-Fi indicator status control code.
            break;
            case WIFI_NOT_CONNECTED:
                //Wi-Fi configuration is completed, and the device is connecting to the router. The Wi-Fi indicator is steady off.
            break;
            case WIFI_CONNECTED:
                //The device successfully connects to the router, and the Wi-Fi indicator is steady on.
            break;
            case WIFI_CONN_CLOUD:
                //The device successfully connects to the cloud, and the Wi-Fi indicator is steady on.
            default:break;
        }
        ...

    }
}
```

3.3.6 Optimizing the Product Testing Function

1. The MCU needs to support the Wi-Fi testing function. Open **protocol.h** and define the

following macro:

#define WIFI_TEST_ENABLE //Enable the Wi-Fi testing function.

2. The MCU invokes the mcu_start_wifitest() function from the **mcu_api.c** file when Wi-Fi testing is required.
3. Invoke the wifi_test_result function from the **protocol.c** file to view the test result.

We recommend that you invoke the product testing command 5s after the Wi-Fi module is powered on and initialized. Triggering conditions are user-defined. After the product testing function is enabled, the module automatically searches for the "tuya_mdev_test" WLAN network and returns the signal strength. The wireless hotspot name needs to be changed to "tuya_mdev_test". During the test, you can change the hotspot name on your mobile phone.

Optional Function: MCU Online Upgrade

To support MCU online upgrade, open the **protocol.h** file, define the following macros:

#define SUPPORT_MCU_FIRM_UPDATE //Enable the MCU firmware upgrade function, which is disabled by default.

#define MCU_VER "1.0.0"

//Set your software version, which is used during MCU firmware upgrade. The version number needs to be modified after an MCU upgrade.

If the data packet is large, adjust the buffer size based on actual requirements as follows:

WIFI_UART_RECV_BUF_LMT 300 //Firmware upgrade buffer size, which must be greater than 260 bytes

The corresponding upgrade function is in **protocol.c**.

protocol.c

```
/*****************************************************************************
函数名称 : mcu_firm_update_handle
功能描述 : MCU进入固件升级模式
输入参数 : value:固件缓冲区
           position:当前数据包在于固件位置
           length:当前固件包长度(固件包长度为0时,表示固件包发送完成)
返回参数 : 无
使用说明 : MCU需要自行实现该功能
*****************************************************************************/
unsigned char mcu_firm_update_handle(const unsigned char value[],unsigned long position,unsigned short length)
{
  #error "请自行完成MCU固件升级代码,完成后请删除该行"
  if(length == 0)
  {
    //固件数据发送完成

  }
  else
  {
    //固件数据处理
  }

  return SUCCESS;
}
#endif
```

The MCU can invoke the mcu_firm_update_query() function from the **mcu_api.c** file to obtain the MCU firmware upgrade information.

Note: The upgrade is initiated by the mobile phone. Click **Upgrade** to start. When debugging, you can use the Tuya serial port debugging assistant to start the upgrade.

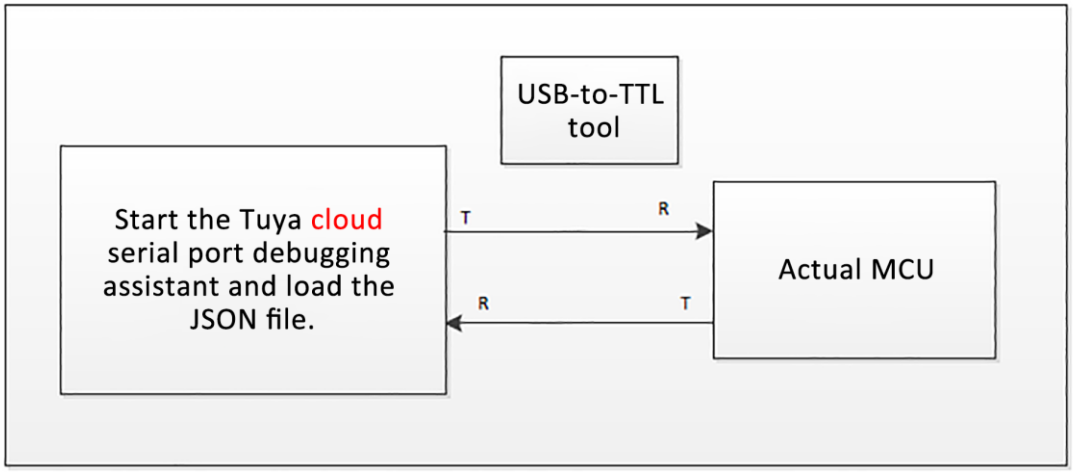
4 Serial Port Simulation Tools

Tuya provides two simulation assistants to help you improve the interworking efficiency, understand the protocol format, and verify data. One assistant simulates the Wi-Fi module, and the other simulates the MCU. Using both assistants can effectively improve the development efficiency. For details, visit https://docs.tuya.com/cn/mcu/debug_assistant.html.

4.1 Tuya Cloud Serial Port Debugging Assistant

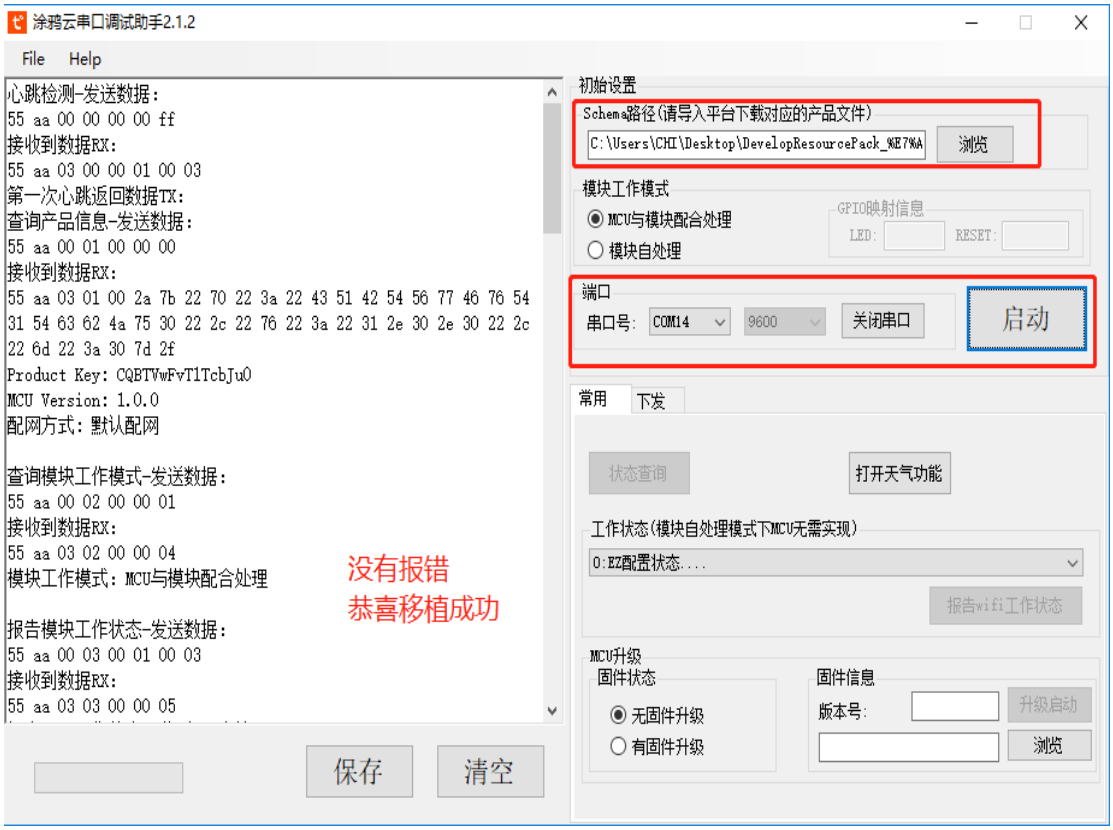
The Tuya cloud serial port debugging assistant simulates data transmitting and receiving of the Wi-Fi module. After connecting the assistant to the MCU, you can check whether MCU data sending meets requirements of the Tuya communication protocol and whether migration is successful.

Note: The Tuya cloud serial port debugging assistant can only verify the sending and receiving protocol formats and does not support network connection.



Use method:

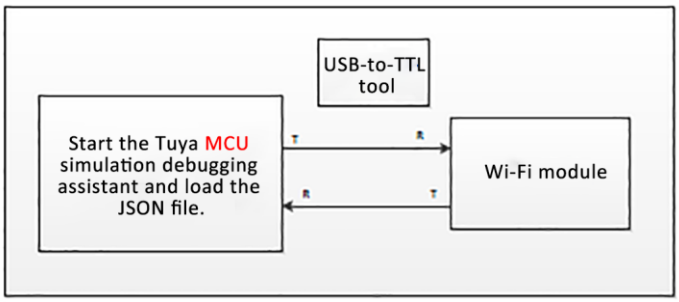
1. Use the USB-to-TTL tool to connect the serial port on the MCU to the serial port on a computer.
2. Double-click the .exe file of the Tuya cloud serial port debugging assistant.
3. Click **Browse**, import the JSON file in the material package, and click **Start**.

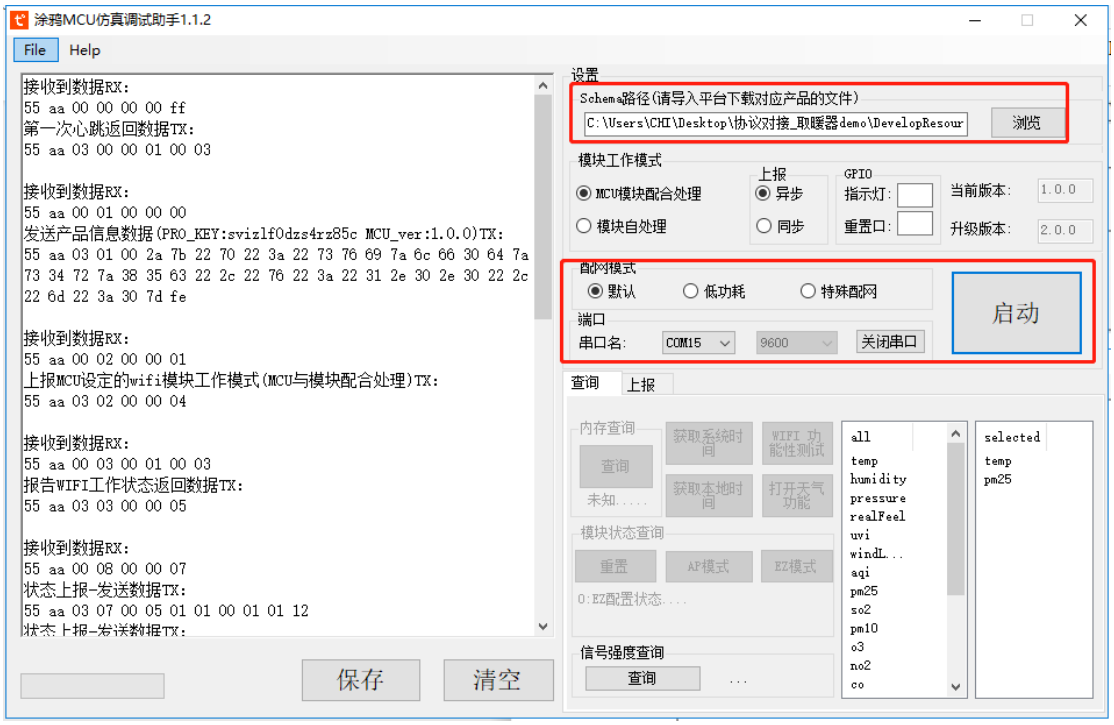


4.2 Tuya MCU Simulation Debugging Assistant

The Tuya MCU simulation debugging assistant simulates data sending and receiving of the MCU. After building a minimum system of the Wi-Fi module, connect the Wi-Fi module to the Tuya MCU simulation assistant to achieve the following functions:

1. Check whether the Wi-Fi module is working properly.
2. Before the MCU is developed, debug app panel display.
3. Refer to the simulation assistant data for how to send or return data to the Wi-Fi module.





5 SDK Function Architecture Breakdown