Guide to Interworking with the Tuya MCU

Contents

1 Creating a Product on the Tuya Smart Platform and Downloading the MCU Development Package.............................................................................................................2
2 Protocol Resolution.........................................................................................................................6
  2.1 Protocol Framework..................................................................................................................6
  2.1.1 Basic Protocol....................................................................................................................7
  2.1.2 Functional Protocol.............................................................................................................13
3 Migrating Tuya’s MCU SDK.........................................................................................................15
  3.1 Precautions..............................................................................................................................15
  3.2 Roadmap ................................................................................................................................15
  3.3 Procedure ................................................................................................................................15
    3.3.1 Compiling the MCU Basic Program and Migrating the SDK File ..............................15
    3.3.2 Verifying the Macro Definition in protocol.h ...............................................................16
    3.3.3 Migrating the protocol.c File and Invoking Functions ...............................................18
    3.3.4 Processing DP Data Report and Delivery Functions ...................................................19
    3.3.5 Optimizing Network Configuration and Indicator Functions ...................................20
    3.3.6 Optimizing the Product Testing Function ..................................................................22
    Optional Function: MCU Online Upgrade............................................................................23
4 Serial Port Simulation Tools.........................................................................................................24
  4.1 Tuya Cloud Serial Port Debugging Assistant ......................................................................24
  4.2 Tuya MCU Simulation Debugging Assistant .......................................................................25
5 SDK Function Architecture Breakdown.....................................................................................26
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

Fixed template

Custom template

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.

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

The following table describes command words.

<table>
<thead>
<tr>
<th>Command Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Heartbeat detection.</td>
</tr>
<tr>
<td>0x01</td>
<td>Query product information.</td>
</tr>
<tr>
<td>0x02</td>
<td>Query the working mode of the Wi-Fi module.</td>
</tr>
<tr>
<td>0x03</td>
<td>Report the network connection status of the device.</td>
</tr>
<tr>
<td>0x04</td>
<td>Reset the Wi-Fi module and switch the network configuration mode.</td>
</tr>
<tr>
<td>0x05</td>
<td>Reset the Wi-Fi module and select a network configuration mode.</td>
</tr>
<tr>
<td>0x06</td>
<td>Deliver DP commands.</td>
</tr>
<tr>
<td>0x07</td>
<td>Report DP status.</td>
</tr>
<tr>
<td>0x08</td>
<td>Query the device initialization status.</td>
</tr>
<tr>
<td>0x0a</td>
<td>(Optional) Start OTA upgrade.</td>
</tr>
<tr>
<td>0x0b</td>
<td>(Optional) Transmit the OTA upgrade package.</td>
</tr>
<tr>
<td>0x1c</td>
<td>(Optional) Obtain the local time.</td>
</tr>
<tr>
<td>------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>0x0e</td>
<td>Test the Wi-Fi function (product testing command).</td>
</tr>
</tbody>
</table>

### 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.
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.

<table>
<thead>
<tr>
<th>Querying product information</th>
<th>Sent by the Wi-Fi module</th>
<th>Reported by the MCU</th>
<th>Command Word</th>
<th>Data</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x55aa 0x00</td>
<td>0x55aa 0x03</td>
<td>0x0000</td>
<td>0x00 (first packet) or 0x01 (later packets)</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

For example, the Wi-Fi module sends 55 aa 00 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":"RN2FVAgXG6WfAktU", "v":"1.0.0", "m"=0"

"p" indicates the product ID, and the value is RN2FVAgXG6WfAktU. "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).

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.

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.

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.
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.

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

<table>
<thead>
<tr>
<th>Reporting the Wi-Fi status</th>
<th>Sent by the Wi-Fi module</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0x55aa 0x00 0x03 0x0001</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reported by the MCU</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x55aa 0x00 0x03</td>
<td>0x0000</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Resetting the Wi-Fi module and selecting a network configuration mode</th>
<th>Reported by the MCU (smart network configuration mode)</th>
<th>0x55aa 0x03 0x05 0x0001 0x00</th>
<th>Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent by the MCU (AP network configuration mode)</td>
<td>0x55aa 0x03 0x05 0x0001 0x01</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Sent by the Wi-Fi module</td>
<td>0x55aa 0x00 0x05 0x0000 0x04</td>
<td>Checksum</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Querying the MCU working status</th>
<th>Sent by the Wi-Fi module</th>
<th>0x55aa 0x00 0x08 0x0000</th>
<th>Checksum</th>
</tr>
</thead>
</table>
| Reported by the MCU             | 0x55aa 0x03 0x07 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).
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:
<table>
<thead>
<tr>
<th>ID</th>
<th>Function</th>
<th>Frame Header Version</th>
<th>Command Word</th>
<th>Data Length</th>
<th>DoI D</th>
<th>Data Type</th>
<th>Function Length</th>
<th>Function Command</th>
<th>Verification Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Switch</td>
<td>Sent by the Wi-Fi module</td>
<td>0x55aa 0x00</td>
<td>0x06</td>
<td>0x01</td>
<td>0x01</td>
<td>0x09 0x01</td>
<td>off: 0x00</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reported by the MCU</td>
<td>0x55aa 0x03</td>
<td>0x07</td>
<td>0x01</td>
<td>0x01</td>
<td>0x09 0x01</td>
<td>on: 0x01</td>
<td>Checksum</td>
</tr>
<tr>
<td>2</td>
<td>Target temperature</td>
<td>Sent by the Wi-Fi module</td>
<td>0x55aa 0x00</td>
<td>0x06</td>
<td>0x02</td>
<td>0x02</td>
<td>0x09 0x01</td>
<td>0x1e-0x50</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reported by the MCU</td>
<td>0x55aa 0x03</td>
<td>0x07</td>
<td>0x02</td>
<td>0x02</td>
<td>0x09 0x01</td>
<td></td>
<td>Checksum</td>
</tr>
<tr>
<td>11</td>
<td>Remaining countdown time</td>
<td>Sent by the Wi-Fi module</td>
<td>0x55aa 0x00</td>
<td>0x06</td>
<td>0x06</td>
<td>0x04</td>
<td>0x09 0x01</td>
<td>1hour: 0x00</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reported by the MCU</td>
<td>0x55aa 0x03</td>
<td>0x07</td>
<td>0x06</td>
<td>0x04</td>
<td>0x09 0x01</td>
<td>2hour: 0x01</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3hour: 0x02</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Fault Alarm</td>
<td>Reported by the MCU</td>
<td>0x55aa 0x03</td>
<td>0x06</td>
<td>0x0d</td>
<td>0x05</td>
<td>0x00 0x02</td>
<td></td>
<td>Checksum</td>
</tr>
<tr>
<td>17</td>
<td>Week program</td>
<td>Sent by the Wi-Fi module</td>
<td>0x55aa 0x00</td>
<td>0x06</td>
<td>0x11</td>
<td>0x00</td>
<td>N</td>
<td>0x00-0xff</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reported by the MCU</td>
<td>0x55aa 0x03</td>
<td>0x07</td>
<td>0x11</td>
<td>0x00</td>
<td>N</td>
<td></td>
<td>Checksum</td>
</tr>
<tr>
<td>102</td>
<td>Data string</td>
<td>Sent by the Wi-Fi module</td>
<td>0x55aa 0x00</td>
<td>0x06</td>
<td>0x66</td>
<td>0x03</td>
<td>N</td>
<td>0x00-0xff</td>
<td>Checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reported by the MCU</td>
<td>0x55aa 0x03</td>
<td>0x07</td>
<td>0x66</td>
<td>0x03</td>
<td>N</td>
<td></td>
<td>Checksum</td>
</tr>
</tbody>
</table>

**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.

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

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.

   ![Protocol.h](image)

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.
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.

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 "//").

(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:

```
#define 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.

5. Check whether the MCU needs time verification.

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

```
#define WIFI_TEST_ENABLE // 开启Wi-Fi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.

```
#define UART3_SendByte(value) 

void uart_transmit_output(unsigned char value) 
{
    // 示例：
    //uart_putchar(value); 
    // UART SendByte(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.
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.

```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.
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_update(DPID_X,n)`. `DPID_X` indicates the DP whose status has changed. Functions in `all_data_update()` can be independently invoked.

Example:

```c
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.

```c
// 示例：开关类为BOOL
unsigned char ret;
//0:1;1:开
unsigned char switch1;
switch1 = mcu_get_dp_download_bool(value,length);
if(switch1 == 1)
    // MCU_ON_SWITCH1(); // 开开关
else
    // MCU_OFF_SWITCH1(); // 关开关

if(res == 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.

![Network configuration diagram](image)

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.

<table>
<thead>
<tr>
<th>Device Network Connection Status</th>
<th>Description</th>
<th>Status Value</th>
<th>LED Indicator Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 1</td>
<td>Smart network configuration</td>
<td>0x00</td>
<td>The indicator blinks at 250 ms intervals.</td>
</tr>
<tr>
<td>State 2</td>
<td>AP network configuration</td>
<td>0x01</td>
<td>The indicator blinks at 1500 ms intervals.</td>
</tr>
<tr>
<td>State 3</td>
<td>The Wi-Fi is configured. However, the device fails to connect to the router.</td>
<td>0x02</td>
<td>The indicator is off.</td>
</tr>
<tr>
<td>State 4</td>
<td>The Wi-Fi is configured, and the device successfully connects to the router.</td>
<td>0x03</td>
<td>The indicator is steady on.</td>
</tr>
<tr>
<td>State 5</td>
<td>The device connects to the router</td>
<td>0x04</td>
<td>The indicator is...</td>
</tr>
</tbody>
</table>
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 network configuration mode, and the Wi-Fi indicator blinks quickly.
                break;
            case AP_STA:
                // 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.
                break;
            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:

```c
#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:

```c
#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:

```c
#define 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`.

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