Serial Port Protocol

Hardware Product Development > IoTOS Pro-Code Development >

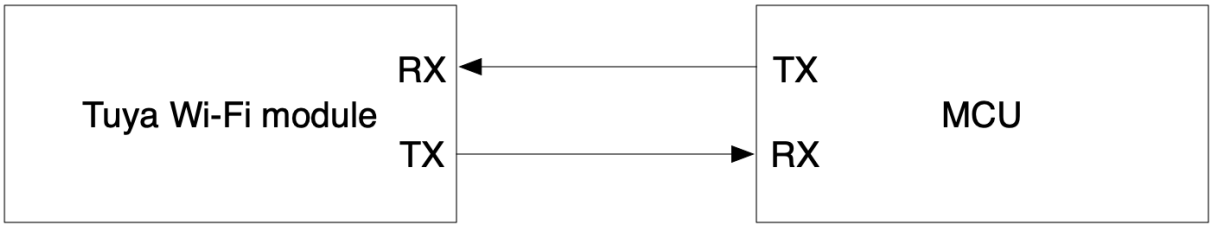
MCU Connection Solution > Wi-Fi Common Solution

Version: 20210412

Online Version

Contents

Tuya Wi-Fi general serial port protocol is a customized protocol for Wi-Fi modules of Tuya. It is mainly used for serial port communication between Tuya Wi-Fi modules and other MCU serial ports. The architecture diagram is shown as follows.



{width=100%}

1 Serial communication convention

Baud: 9600/115200

Data bit: 8

Parity check: none

Stop bit: 1

Data flow control: none

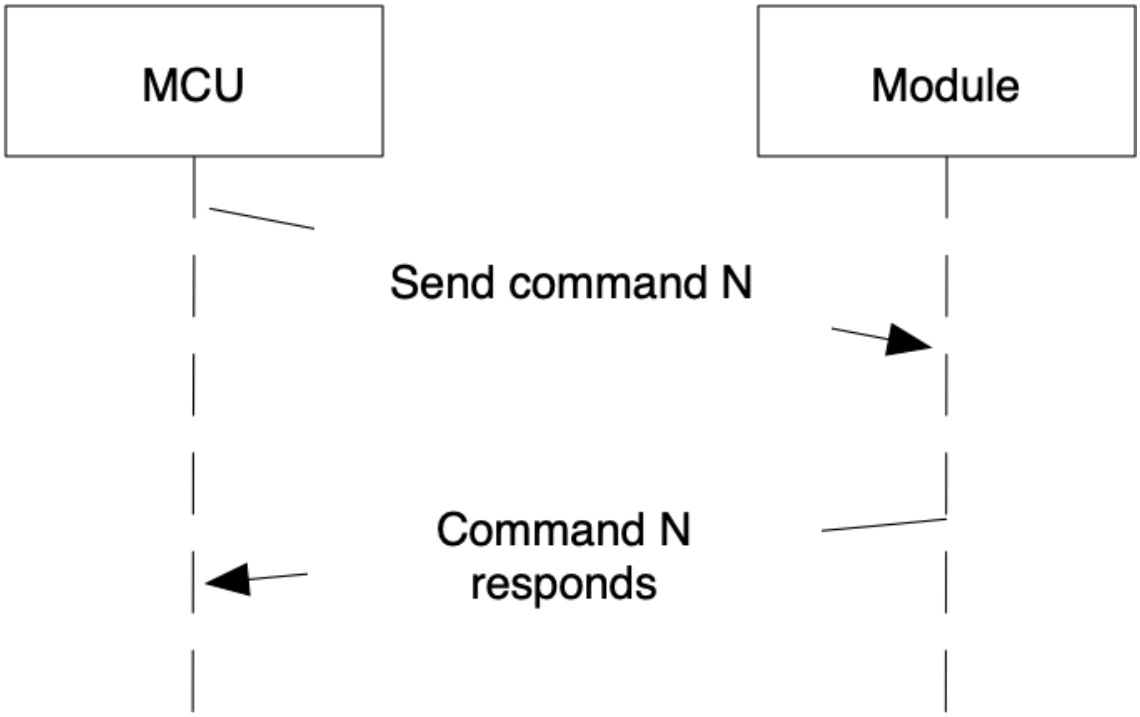
MCU: the control chip of the control board. It connects to the Tuya module through the serial port.

2 Frame format description

Field	Length (byte)	Description
Header	2	It is fixed as `0x55aa`
Version	1	It is used for upgrade and extension
Command	1	Specific frame type
Data length	2	Big-endian
Data	N	Entity data
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Description:

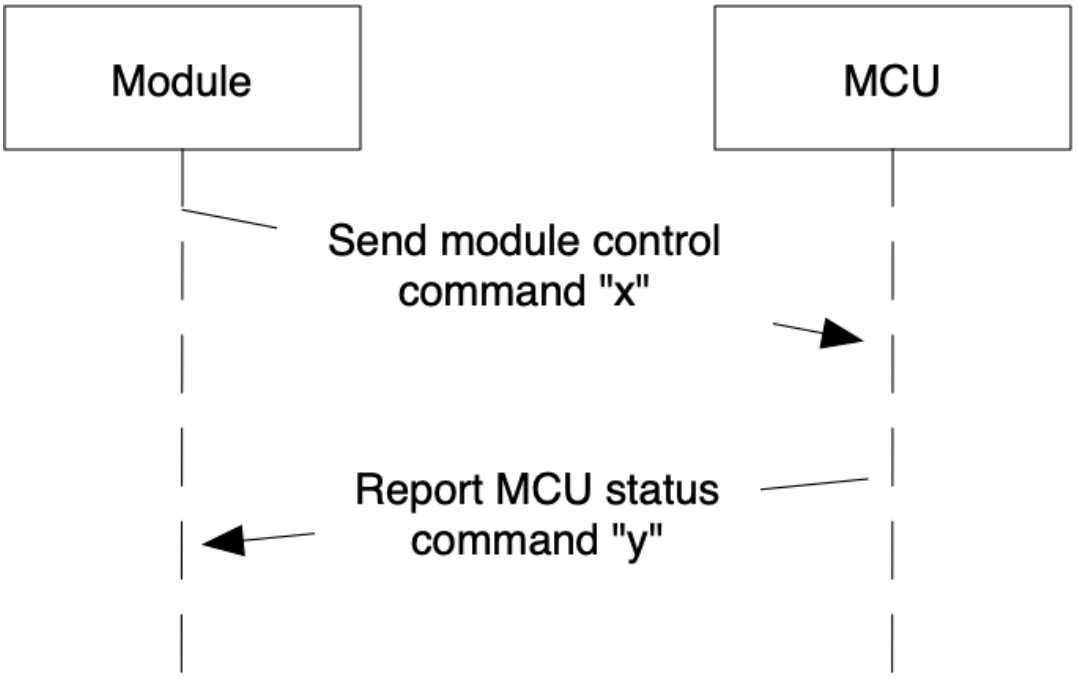
- All data greater than one byte shall be transmitted with big-endian format.
- Generally, one command word is sent by one party and received by the other party in a synchronous way. That is, one party sends the command, and the other party responds. If the sender does not receive the correct response packet within the stipulated time, the transmission times out, as shown in the following figure:



{width=100%}

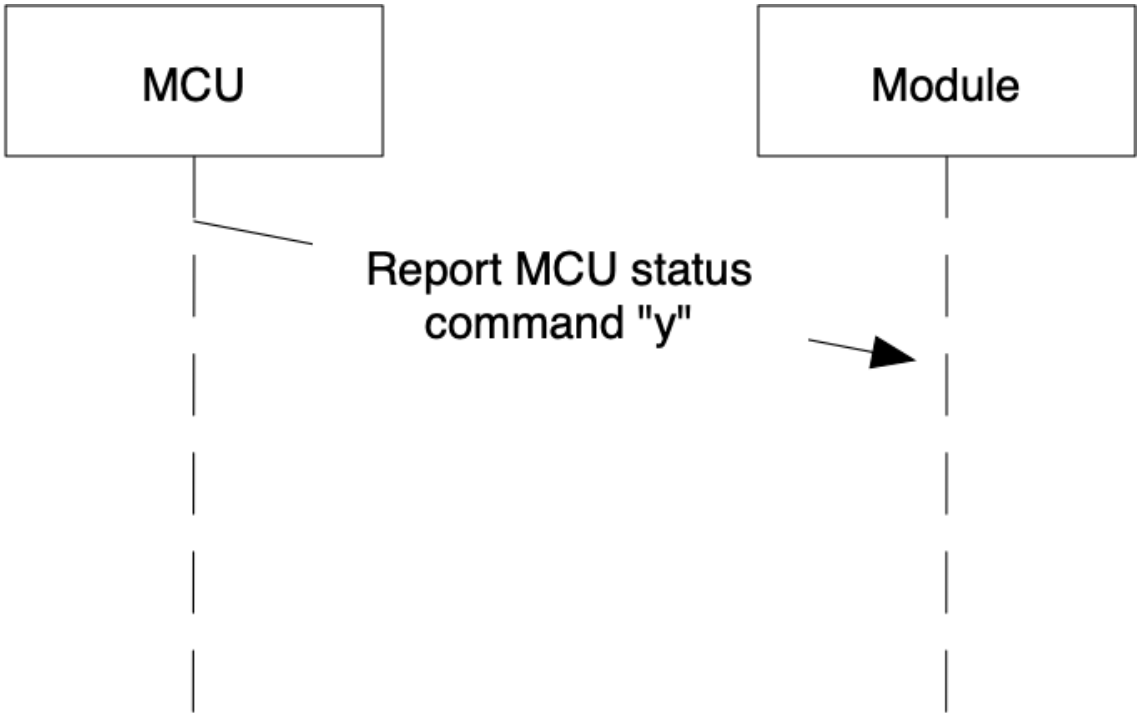
> **Note**: for specific communication mode, see the section of Protocol details.

- Sending module control command and reporting MCU status works in an asynchronous way. Assuming that the command of module control is “x”, and the command of MCU status report is “y”, the data transmission is as follows:
  - Send module control command:



{width=100%}

- Report MCU status:



{width=100%}

3 Status data unit

Data point command and status data unit are shown as follows:

Data segment	Length (byte)	Description
dpid	1	Data point serial number.
type	1	Specific data type of a data point in the IoT Console. For more information, see the description of the type field.
len	2	Number of bytes of the value. For more information, see the description of the type field.
value	1/2/4/N	Expressed in the hexadecimal system, and adopt big-endian transmission when there is more than one byte.

Description of the type field:

type	Type	Length (byte)	Description
0x00	raw	N	Represents raw data point (module pass-through).
0x01	bool	1	Value range: 0x00/0x01.

type	Type	Length (byte)	Description
0x02	value	4	Represents integer type, expressed with big-endian.
0x03	string	N	Represents specific string.
0x04	enum	1	Represents enumeration type, ranging from 0 to 255.
0x05	bitmap	1/2/4	Expressed with big-endian when there is more than one byte.

- For data point command and status data unit, except raw type, all other types belong to the object data point.
- Status data can contain command data units of multiple data points.

4 Protocol details

4.1 Detecting heartbeat

Description:

- After the Wi-Fi module is powered on, it sends heartbeats periodically at an interval of 15 seconds. If the module does not receive a response from the MCU within three seconds, the MCU is considered offline.
- The MCU can also periodically check whether the module is working normally based on the heartbeat. If the module does not send a heartbeat, the MCU can reset the Wi-Fi module through the module hardware reset pin.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x00
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 00 00 00 ff`

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa

Field	Length (byte)	Description
Version	1	0x03
Command	1	0x00
Data length	2	0x0001
Data	1	0x00: the return value of the first heartbeat after the MCU reboots. It is only sent once, used for the module to determine whether the MCU reboots during the working process.
		0x01: this value is returned except for the first return value of 0 after the MCU reboots.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 00 00 01 00 03` (the first return value of the MCU)

`55 aa 03 00 00 01 01 04` (the normal return value except for the first return)

4.2 Querying product information

Description:

- Product ID (PID) is generated in the Tuya IoT console to record product information in the cloud.
- Product information consists of the product ID and MCU software version number.

- MCU software version number is defined as dot-decimal notation in the format of x.x.x (0\<=x\<=99), and x is a decimal digit.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x01
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 01 00 00 00`

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x01
Data length	2	N
Data	N	{“p”:”AIp08kLIftb8x***”,“v”:”1.0.0”,”m”

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `{ “p” :” AIp08kLIftb8x***” , “v” :” 1.0.0” ,” m” :1,” mt” :10,” n” :0,” ir ” :” 5.12” ,” low” :0}`

Product information field description:

- `p` indicates that the product ID is `AIp08kLIftb8x***`, which is the product ID created by the user in Tuya IoT Console.
- `v` indicates that the MCU version is 1.0.0, and the format of the MCU version number is defined in the format of x.x.x.
- `m` indicates the working mode of the module. `0` represents the normal network configuration mode. `1` represents a time-out network configuration mode. `2` represents unexpected-trigger-proof mode.
  - Normal network configuration mode: The module is ready for network configuration after it is powered on for the first time.
  - Time-out network configuration mode: The module is not ready for network configuration after it is powered on for the first time. When the MCU sends a reset command, the module enters network configuration mode accordingly. If the device is not configured within three minutes, the module will enter the non-network configuration mode again until it receives another reset command.
  - Unexpected-trigger-proof mode: After the module is configured and locally reset by the MCU reset command, the device is ready for network configuration. If the device is not configured within three minutes, it will restore the network connection before reset. If the device is powered off unexpectedly after the local reset, it will restore the network connection before reset. In this mode, after the device is removed from the app, it will not record the last network connection and reconnect to that network. This mode suits the scene of avoiding unexpected local network reset.

- `mt` field (optional): set the status switching time between the safe mode and the unexpected-trigger-proof mode. If this field is not uploaded, the switching time is three minutes. The switching time can be set from 3 to 10 minutes.
- `n` field (optional): indicates the network configuration mode. If this field is not uploaded, you can switch between two network configuration modes.
  - `0`: indicates SmartConfig coexists with access point (AP) configuration. The module supports both AP configuration and EZ configuration. You do not need to switch them manually. For the corresponding network configuration status packet, see reporting device network status.
  - `1`: indicates there is only AP configuration. In this mode, the network can only be configured through an AP connection.
- `ir` field (optional): enable the infrared (IR) function of the module to inform the module of I/O interfaces used by IR I/O pins. Without this field, the IR function is disabled by default. 5.12: indicates that the IR output pin is I/O 5, and the IR input pin is I/O 12.

  > Note: if the self-processing mode of the module is used, the button and Wi-Fi indicator I/O interfaces cannot be used for IR I/O pins. For the cross-module I/O setting, you need to add 32 to the I/O pin number of the set module. For example, the number of PB20 is 20+32, which is 52. The IR output pin uses PWM resources. IR input pin requires the I/O interrupt. For the supported I/O interface, see the corresponding module documentation.

- `low` field (optional): indicates whether the module enables low power mode that maintains long-running connections. Without this field, the low power mode is disabled by default. If the product keeps a connection to the router without network access control, you can use this field to keep the average consumption of the module lower than 15 mA. When the dual-mode module enables this function, it only has Bluetooth network configuration function. The Bluetooth control function is disabled. For products that do not have requirements for power consumption, this field is not necessary.
  - `0`: disable low power mode.
  - `1`: enable low power mode.

4.3 Querying working mode

The MCU settings determine the method of triggering and indicating network configuration. The working mode of the module mainly includes how to indicate Wi-Fi working status and how to reset the Wi-Fi network. There are two cases:

- The coordinated processing mode of the MCU and module: The MCU detects the trigger signal of network configuration. When receiving the serial port command, the Wi-Fi module resets network configuration. The module informs MCU of the current Wi-Fi working status. The MCU supports the status display. It is recommended to select this mode for home appliances.
- The self-processing mode of the module: The working status of the Wi-Fi module is displayed through the Wi-Fi GPIO pin to drive the LED indicator. The Wi-Fi module resets the network configuration by detecting the GPIO input pin. Reset Wi-Fi in self-processing mode: Wi-Fi detects the low electrical level of GPIO input pin for more than 5 seconds to trigger Wi-Fi reset. The GPIO pins used by the indicators and buttons are configured by the following commands.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x02
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 02 00 00 01`

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x02
Data length	2	0x0000: indicates that the module works in coordination with the MCU, and the MCU needs to implement the functions mentioned in the above explanation.
		0x0002: indicates that the module works in self-processing mode.
Data	0/2	The data length is 2: the first byte is the GPIO number of the Wi-Fi status indicator. The second byte is the GPIO number of the Wi-Fi reset button.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example:

The coordinated processing of the MCU and module.

```
55 aa 03 02 00 00 04
```

In module self-processing mode, indicator 0x0c represents GPIO12, and reset button 0x0d represents GPIO13.

```
55 aa 03 02 00 02 0c 0d 1f
```

4.4 Reporting device network status

Device network status	Description	Status value
Status 1	SmartConfig configuration status.	0x00
Status 2	AP configuration status.	0x01
Status 3	Wi-Fi has been configured but not connected to the router.	0x02
Status 4	Wi-Fi has been configured and connected to the router.	0x03
Status 5	Wi-Fi has been connected to the router and the cloud.	0x04
Status 6	Wi-Fi device is in the low power mode.	0x05
Status 7	Wi-Fi device is in SmartConfig and AP configuration mode.	0x06

Description:

- Device network status:
  - SmartConfig configuration status.
  - AP configuration status.
  - Wi-Fi has been successfully configured but not connected to the router.
  - Wi-Fi has been successfully configured and connected to the router.
  - The device has been connected to the router and the cloud.
- The status of an LED indicator in module self-processing mode:

  - Status 1: flicker at 250 milliseconds intervals.
  - Status 2: flicker at 1,500 milliseconds intervals.
  - Status 3 or 6: always off.
  - Status 4 or 5: always on.

- When the module detects that the MCU reboots or is disconnected and then go back online, it will send the Wi-Fi status to the MCU.
- When the Wi-Fi status of the module changes, it will send the Wi-Fi status to the MCU.
- If the module working mode is set to module self-processing, the MCU does not need to implement this protocol.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x03
Data length	2	0x0001
Data	1	Indicates Wi-Fi working status:
		0x00: status 1
		0x01: status 2
		0x02: status 3
		0x03: status 4
		0x04: status 5
		0x05: status 6
		0x06: status 7

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 03 00 01 00 03`

MCU returns:

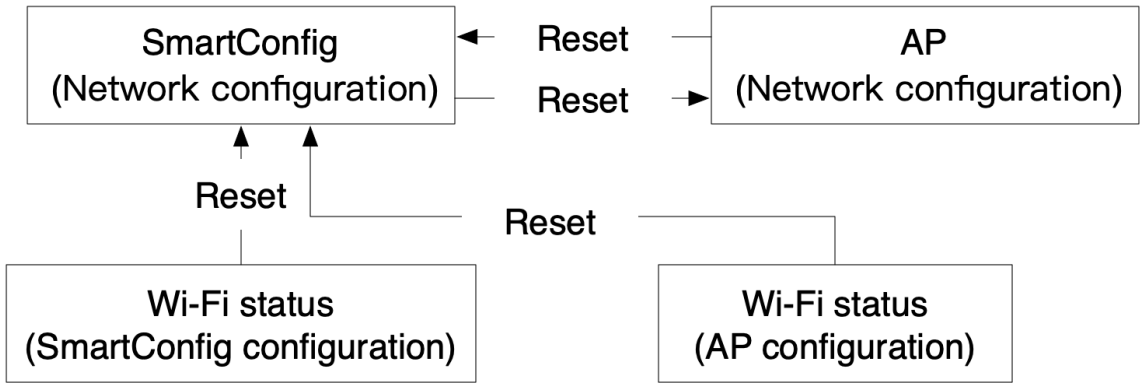
Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x03
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 03 00 00 05`

4.5 Resetting Wi-Fi

Description:

- Status transfer of resetting Wi-Fi is shown as follows:



{width=100%}

> **Note**: When sending the reset command, please send it after the module initialization is completed. Otherwise, it may be invalid. For more information, see Wi-Fi Module Initialization Flow.

- If the working mode of the module is set to module self-processing, the MCU does not need to implement this protocol.

Reset Wi-Fi in self-processing mode: Wi-Fi detects the low electrical level of GPIO input pin for more than 5 seconds to trigger Wi-Fi reset.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x04
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 04 00 00 06`

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x04
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 04 00 00 03`

4.6 Resetting Wi-Fi and select configuration mode

Description:

- Compared to resetting Wi-Fi, with this frame, MCU can select the required configuration mode after the Wi-Fi is reset.
- You can implement this protocol selectively.
- If the module working mode is set to module self-processing, the MCU does not need to implement this protocol.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x05

Field	Length (byte)	Description
Data length	2	0x0001
Data	1	0x00: enter SmartConfig configuration mode
		0x01: enter AP configuration mode
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, enter SmartConfig configuration mode

```
55 aa 03 05 00 01 00 08
```

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x05
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 05 00 00 04`

4.7 Sending command

Description:

- Command sending can contain status data units of multiple data points.
- Command sending is an asynchronous processing protocol, corresponding to the data point status report of the MCU.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x06
Data length	2	It depends on the type and number of the status data unit
Data	N	See the section of status data unit
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, the system switch corresponds to DP 3, uses the Boolean data type, and the value of 1 indicates power-on.

```
55 aa 00 06 00 05 03 01 00 01 01 10
```

4.8 Reporting Status

Description:

- For the description of the status data unit of a data point, see the section of the status data unit.

- The status report is an asynchronous processing protocol. Its trigger mechanism has three types:
  - After receiving the processing frame of command sending, the MCU executes the command of the corresponding data point, and then sends the changed status to the module through the status report frame.
  - MCU actively detects that the data point has changed and sends the changed status to the module.
  - After the MCU receives a status query frame, it sends the status of all data points to the module.
- Status report can contain status data units of multiple data points.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x07
Data length	2	It depends on the type and number of the status data unit
Data	N	See the section of status data unit
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, humidity corresponds to DP 5, uses a valve variable, and the humidity is 30°C.

```
55 aa 03 07 00 08 05 02 00 04 00 00 00 1e 3a
```

Examples of reporting multiple status data units:

DP 109 is a Boolean variable, and the value is 1.

The DP 102 is a string variable, and the value is an ASCII code of 201804121507.

```
55 aa 03 07 00 15 6d 01 00 01 01 66 03 00 0c 32 30 31 38 30 34 31 32 31 35 30 37 62
```

4.9 Querying Status

Description:

- Status query is an asynchronous processing command that is mainly used by the module to query the status of all object data points of MCU. When MCU receives this frame, it reports the status of data points through the status report frame.
- Status query is sent at two time points:
  - The module is powered on for the first time, after connecting to the MCU through the heartbeat, it sends the query.
  - When the module detects that the MCU reboots or is disconnected and then go back online, it sends the query.

The module sends:

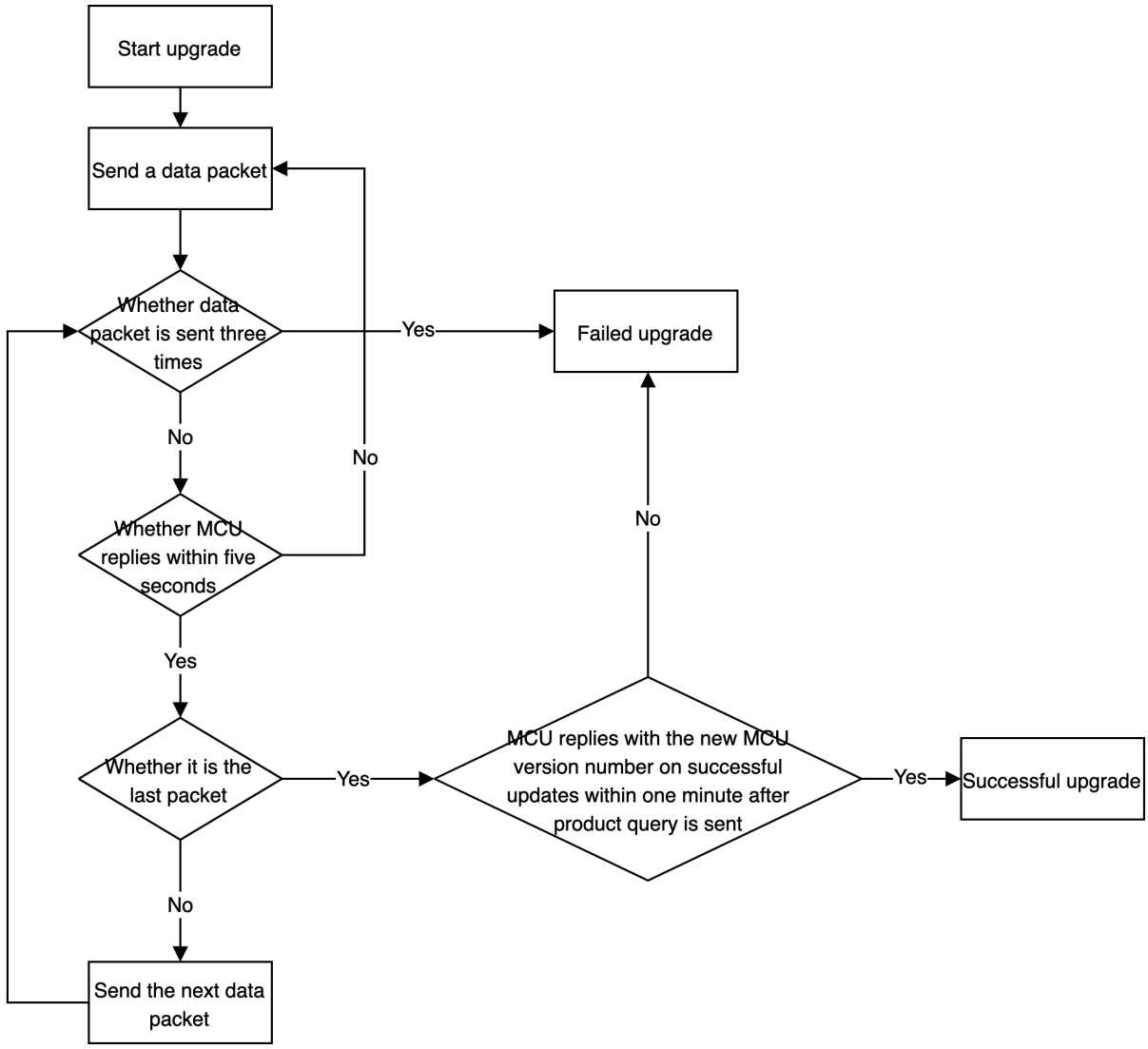
Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x08
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 08 00 00 07`

4.10 MCU upgrade service

Description:

- The upgrade time is triggered by related upgrade options that you configure in Tuya IoT Console. The module only serves as a data transmission channel that supports MCU upgrades and does not perform any data analysis.
- Currently, the Tuya IoT platform provides the following four MCU upgrade configurations.
  - App notification upgrade: you choose to upgrade or not when the app receives an upgrade prompt.
  - App hardware silent upgrade: there is no upgrade prompt on the app. The module detects upgrade automatically within one minute after the module is powered on and automatically pulls the upgrade package if the latest upgrade package is found. After the first power-on, the module detects whether there is an upgrade package configuration in the cloud every 24 hours.
  - Forced upgrade: there is an upgrade prompt on the app. You are required to upgrade the device before you can continue to use it.
  - Manual detection upgrade: there is no upgrade prompt on the app. You must click the relevant firmware version to detect on the app. If there is the latest firmware configuration, an upgrade prompt will appear.
- MCU upgrade flowchart: after the Wi-Fi module has sent all upgrade packages, it resends the command 01 (see the section of query product information). The MCU needs to reply with the upgraded MCU software version number in the product information within one minute. The version number must be consistent with that configured in the IoT Console.



{width=100%}

Initiating upgrade (notification of upgrade package size)

Upgrade methods include automatic and manual upgrades. For an automatic upgrade, when the module detects the latest MCU firmware in the cloud, it will automatically initiate the interaction with the MCU upgrade package. For a manual upgrade, only when you confirm the upgrade on the app, can the module initiate the interaction with the MCU upgrade package.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0a
Data length	2	0x0004
Data	4	The size of the firmware file is in byte measurement, the data type is unsigned integer, and the data storage is in big-endian format
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 00 0a 00 04 00 00 68 00 75`

It indicates that the length of the firmware package is 26,624, which is 26 KB.

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0a
Data length	2	0x0001
Data	1	Packet size of the upgrade package:

Field	Length (byte)	Description
		0x00: 256 byte by default (compatible with old firmware)
		0x01: 512 byte
		0x02: 1,024 byte
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 0a 00 01 00 0d`

Transmitting upgrade package

Description:

- The data format of upgrade package transmission: packet offset + packet data.
- If the MCU receives the frame with data length equal to 4 bytes and the upgrade packet offset is equal to or greater than the size of firmware, the packet transmission ends.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0b
Data length	2	The data length is the sum of 0X0004 and the data packet length

Field	Length (byte)	Description
Data	N	The first four bytes are fixed as packet offset, and the latter bytes are the packet content
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example:

If the file to be upgraded has 530 bytes in size, under this circumstance, the MCU can skip the response to the last data packet.

- For the first packet data, the packet offset is 0x00000000, and the data packet length is 256. `55 aa 00 0b 01 04 00000000 xx···xx XX`
- For the second packet data, packet offset is 0x00000100, and data packet length is 256. `55 aa 00 0b 01 04 00000100 xx···xx XX`
- For the second to last packet data, packet offset is 0x00000200, and the data packet length is 18. `55 aa 00 0b 00 16 00000200 xx···xx XX`
- For the last packet data, packet offset is 0x00000212, and data packet length is 0. `55 aa 00 0b 00 04 00000212 xx...xx XX`

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0b
Data length	2	0x0000
Data	0	None

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 0b 00 00 0d`

4.11 Obtaining system time (GMT)

Description:

- As a standard time, Greenwich Mean Time (GMT) is independent of time zone and daylight saving time.
- When the module is connected to the network, after the previous local timestamp is calibrated, the module returns success and valid time data.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0c
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 0c 00 00 0e`

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0c
Data length	2	0x0007
Data	7	Data length is 7 bytes:
		Data[0] indicates whether the time is obtained successfully. 0 represents failure, and 1 represents success.
		Data[1] is the year, and 0x00 represents the year of 2000.
		Data[2] is the month, ranging from 1 to 12.
		Data[3] is the day, ranging from 1 to 31.
		Data[4] is the hour, ranging from 0 to 23.
		Data[5] is the minute, ranging from 0 to 59.
		Data[6] is the second, ranging from 0 to 59.

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, at 05:06:07 on April 19, 2016 (GMT).

```
55 aa 00 0c 00 07 01 10 04 13 05 06 07 4c
```

4.12 Obtaining local time

Description:

- GMT plus local (device activation location) time zone and daylight saving time is the local time.
- When the module is connected to the network, after the previous local timestamp is calibrated, the module returns success and valid time data.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x1c
Data length	2	0x0000
Data	xxxx	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x1c
Data length	2	0x0008
Data	Data	Data length is 8 bytes:
		Data[0] indicates whether the time is obtained successfully. 0 represents failure, and 1 represents success.
		Data[1] is the year, and 0x00 represents the year of 2000.
		Data[2] is the month, ranging from 1 to 12.
		Data[3] is the day, ranging from 1 to 31.
		Data[4] is the hour, ranging from 0 to 23.
		Data[5] is the minute, ranging from 0 to 59.
		Data[6] is the second, ranging from 0 to 59.
		Data[7] is the week, ranging from 1 to 7, and 1 represents Monday.

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

- For example, if the device is activated in mainland China, the local time is Beijing time (GMT+08:00). For example, at 05:06:07 on April 19, 2016 (GMT).
  `55 aa 00 1c 00 08 01 10 04 13 05 06 07 02 5f`
- If the device is activated in other countries or regions, the local time is the time zone in which the device is located.

4.13 Wi-Fi functional test (scan the designated router)

Description:

- The module scans the SSID of `tuya_mdev_test` and returns results and signal strength percentage.
- In order to prevent defective products to the greatest extent, it is recommended that the distance between the router and the device should be about 5 meters. The test result is qualified if the signal strength is greater than or equal to 60%. It can be adjusted based on the actual production line and factory environment.

> **Note**: To send the test command, please send it after the module initialization is completed. Otherwise it may be invalid. For more information, see Wi-Fi Module Initialization Flow.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03

Field	Length (byte)	Description
Command	1	0x0e
Data length	2	0x0000
Data	Data	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0e
Data length	2	0x0002
Data	2	Data length has 2 bytes.
		Data[0]: 0x00 indicates failure, and 0x01 indicates success.
		When Data[0] is 0x01, which indicates success, Data[1] is signal strength (from 0 to100, 0 represents the weakest signal, and 100 represents the strongest signal).

Field	Length (byte)	Description
		When Data[0] is 0x00, which indicates failure, if Data[1] is 0x00, the specified SSID is not found; if Data[1] is 0x01, an authorization key is not burned to the module.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.14 Obtaining module memory

Description:

Obtain the remaining memory of the Wi-Fi module.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0f
Data length	2	0x0000
Data	Data	None

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0f
Data length	2	0x0004
Data	4	The data length is 4 bytes in big-endian format.
		For example, 0x00 0x00 0x28 0x00 represents the remaining memory is 10,240 bytes.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.15 Enabling the function of obtaining weather data (optional)

Description:

Enable the function of obtaining weather data.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x20
Data length	2	N((L+K)+(L+K)…)
Data	Data	L: occupies 1 byte, indicating the length of K.
		K: represents the request parameter name.
		For example:
		L: 0x06 K: w.temp
		L: 0x06 K: w.pm25
		L: 0x0a K: w.humidity
		L: 0x0b K: w.condition
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00

Field	Length (byte)	Description
Command	1	0x20
Data length	2	0x0002
Data	2	Data[0]:
		0x00 indicates failure.
		0x01 indicates success.
		Data[1]:
		0x00 indicates no error.
		0x01 is error code, indicating invalid data format.
		0x02 is error code, indicating exception.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.16 Sending weather data (optional)

Description:

The module will send the weather data regularly after the weather data function is enabled. It will send the data immediately once the function is enabled, and later it sends at 30-minute intervals.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa

Field	Length (byte)	Description
Version	1	0x00
Command	1	0x21
Data length	2	N((LKTLV)+(LKTLV)+...)
Data	Data	0x00: indicates failure
		0x01: is an error code, indicating the parameter service is not authorized (check if you have purchased this parameter service or not)
		0x01: indicates success
		L: indicates parameter name length
		K: indicates parameter name
		T: 0x00 indicates integer type, and 0x01 indicates string type
		L: indicates field name length
		V: indicates field value
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x21
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Remark:

- For example, when the request parameter is w.temp,w.pm25, w.temp is returned. If the return parameter is less than the request parameter, check whether the request parameter name is correct.
- For the field value of w.condition, see Weather UTF-8 Code Reference Table or the following Weather UTF-8 Code Table for w.condition.

For example: `w.humidity`: 69, `w.temp`: 32, `w.pm25`: 10, `w.condition`: cloudy. The UTF-8 code is E5A49AE4BA91. `55 AA 00 21 00 40 01 0A 77 2E 68 75 6D 69 64 69 74 79 00 04 00 00 00 45 06 77 2E 74 65 6D 70 00 04 00 00 00 20 06 77 2E 70 6D 32 35 00 04 00 00 00 10 0B 77 2E 63 6F 6E 64 69 74 69 6F 6E 01 06 E5 A4 9A E4 BA 91 1E`

- The weather UTF-8 code table of `w.condition` is as follows:

Text	Hexadecimal	Text	Hexadecimal
Sunny	E699B4	Heavy rain	E5A4A7 E99BA8
Thunderstorm	E99BB7 E69AB4	Sandstorm	E6B299 E5B098 E69AB4
Light snow	E5B08F E99BAA	Snow	E99BAA

Text	Hexadecimal	Text	Hexadecimal
Freezing fog	E586BB E99BBE	Rainstorm	E69AB4 E99BA8
Isolated showers	E5B180 E983A8 E998B5 E99BA8	Dust	E6B5AE E5B098
Thunder and lightning	E99BB7 E794B5	Light showers	E5B08F E998B5 E99BA8
Rain	E99BA8	Sleet	E99BA8 E5A4B9 E99BAA
Dust devil	E5B098 E58DB7 E9A38E	Ice pellet	E586B0 E7B292
Strong sandstorm	E5BCBA E6B299 E5B098 E69AB4	Sand blowing	E689AC E6B299
Light to moderate rain	E5B08F E588B0 E4B8AD E99BA8	Mostly clear	E5A4A7 E983A8 E699B4 E69C97
Fog	E99BBE	Showers	E998B5 E99BA8
Heavy showers	E5BCBA E998B5 E99BA8	Heavy snow	E5A4A7 E99BAA
Extraordinary rainstorm	E789B9 E5A4A7 E69AB4 E99BA8	Blizzard	E69AB4 E99BAA
Hail	E586B0 E99BB9	Light to moderate snow	E5B08F E588B0 E4B8AD E99BAA
Partly cloudy	E5B091 E4BA91	Light snow shower	E5B08F E998B5 E99BAA
Moderate snow	E4B8AD E99BAA	Overcast	E998B4
Needle ice	E586B0 E99288	Downpour	E5A4A7 E69AB4 E99BA8
Thundershower and hail	E99BB7 E998B5 E99BA8 E4BCB4 E69C89 E586B0 E99BB9	Freezing rain	E586BB E99BA8

Text	Hexadecimal	Text	Hexadecimal
Snow showers	E998B5 E99BAA	Light rain	E5B08F E99BA8
Haze	E99CBE	Moderate rain	E4B8AD E99BA8
Cloudy	E5A49A E4BA91	Thundershower	E99BB7 E998B5 E99BA8
Moderate to heavy rain	E4B8AD E588B0 E5A4A7 E99BA8	Heavy to rainstorm	E5A4A7 E588B0 E69AB4 E99BA8

4.17 Reporting Status (synchronous)

Description:

- It is a synchronous command. After the MCU data status is reported, you need to wait for the module to return the result.
- The module responds to each report. A repeated report is not allowed before the Wi-Fi module responds.
- When the poor network quality causes a failed data report, the module will return failure after five minutes. The MCU needs to wait for more than five seconds.
- For the description of the status data unit of the data point, see the section of the status data unit.
- Status report can contain command data units of multiple data points.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x22
Data length	2	It depends on the type and number of the status data unit

Field	Length (byte)	Description
Data	N	See the section of status data unit
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x23
Data length	2	0x0001
Data	Data	0x00: indicates failure
		0x01: indicates success
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.18 Obtaining current Wi-Fi signal strength (optional)

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa

Field	Length (byte)	Description
Version	1	0x03
Command	1	0x24
Data length	2	0
Data	N	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x24
Data length	2	0x0001
Data	Data	0x00: indicates failure. Less than 0: indicates signal strength, such as -60db.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.19 Notifying Wi-Fi module to disable the heartbeat (optional)

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x25
Data length	2	0
Data	N	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x25
Data length	2	0
Data	N	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

> Note: for MCU modules that require hibernation to reduce power consumption, before entering hibernation status, this command is sent to disable the heartbeat of the Wi-Fi module. This command must not be sent when the device is just powered on. The Wi-Fi module needs to establish a heartbeat connection with the MCU after powered on.

4.20 Interface for serial port network configuration (optional)

Description:

- You can use the Tuya app or the app developed based on Tuya app SDK to obtain the network configuration parameters. The module can receive the parameters through the serial port and then complete network configuration through serial port communication.
- When the module is ready for network configuration, the network configuration can be implemented through the serial port.
- The module returns the response of successfully receiving network configuration information of the serial port. With this information, the module will be connected to the router and then activated in the cloud.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2A
Data length	2	xx
Data	Data	`{ “s” :” xxx” ,” p” :” yyy” ,” t” :” zzz” }`
		`s`: ssid
		`p`: password
		`t`: a token that is generated by the app

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2A
Data length	2	0x0001
Data	x	0x00: indicates the data is successfully received
		0x01: indicates the module is not ready for network configuration
		0x02: indicates JSON data is invalid
		0x03: indicates other errors
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.21 Obtaining current Wi-Fi connection status

Device network status	Description	Status value
Status 1	SmartConfig configuration status.	0x00
Status 2	AP configuration status.	0x01
Status 3	Wi-Fi has been configured but not connected to the router.	0x02
Status 4	Wi-Fi has been configured and connected to the router.	0x03
Status 5	Wi-Fi has been connected to the router and the cloud.	0x04
Status 6	Wi-Fi device is in the low power mode.	0x05
Status 7	Wi-Fi device is in SmartConfig and AP configuration mode.	0x06

> Note: follows the description in the section of reporting device network status.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2B
Data length	2	0x0000
Data	Data	None

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

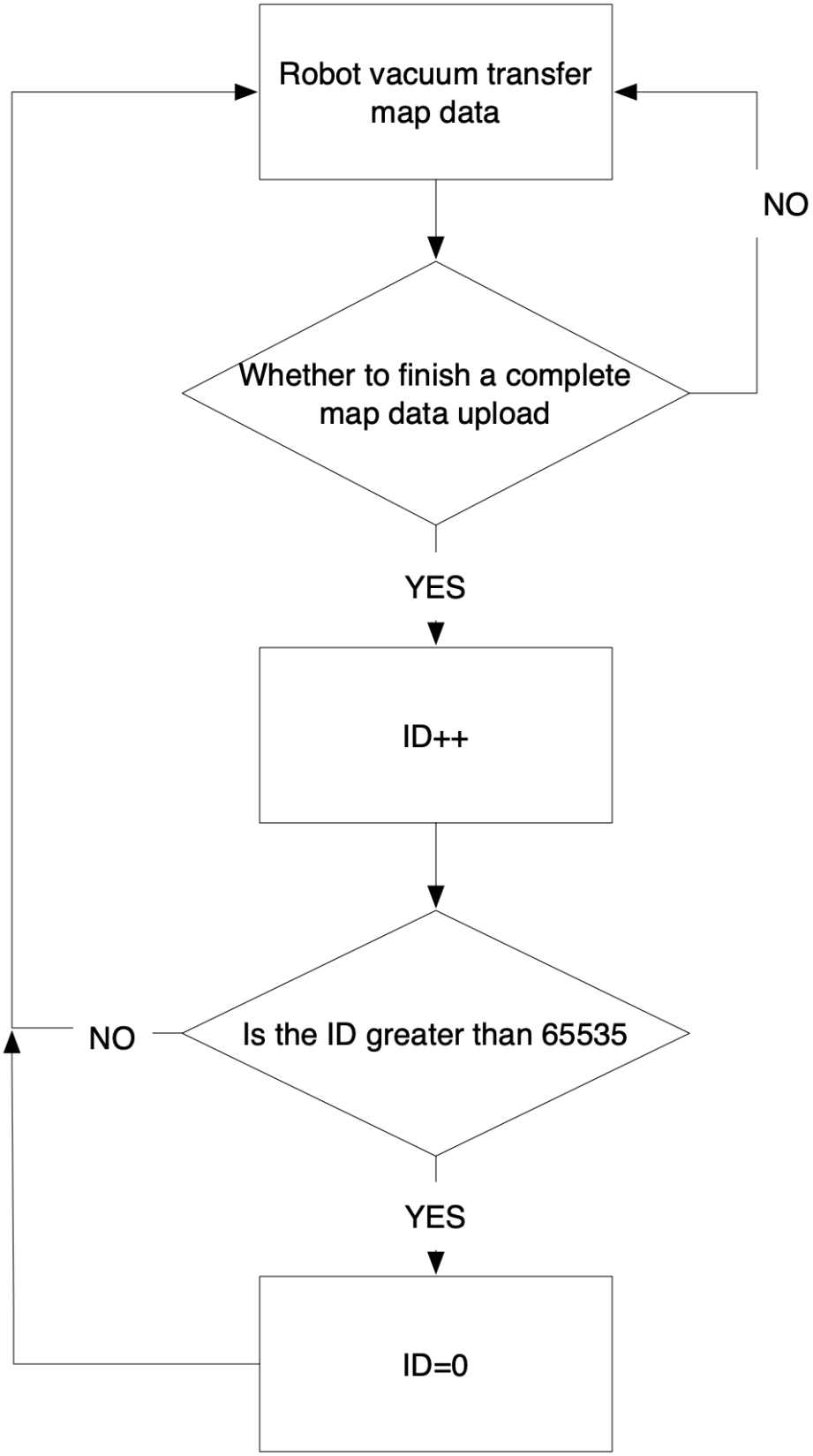
Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2B
Data length	2	0x0001
Data	1	0x00: indicates SmartConfig configuration status
		0x01: indicates AP configuration status
		0x02: indicates Wi-Fi is configured but not connected to the router
		0x03: indicates Wi-Fi is configured and connected to the router
		0x04: indicates the device is connected to the router and the cloud
		0x05: indicates Wi-Fi device is in low power mode

Field	Length (byte)	Description
		0x06: indicates Wi-Fi device is in SmartConfig and AP configuration status
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.22 Map data service of robot vacuum (optional)

Description:

- Robot vacuum streaming service commands are not supported by all modules. To use this service, the device must adopt the specific module and activate related service.
- This service currently works to transmit robot vacuum map data, acting as an online channel of communication between the robot vacuum and map data on the Tuya app.
- The data of a complete map generated by the robot vacuum is classified by the map ID. The data under one map ID is considered to belong to the same map by the app and will be accumulated.
- The robot vacuum moves around the entire house during cleaning. Poor Wi-Fi signal in some areas may cause data upload failure. If the module memory is sufficient, it can currently cache 24 pieces of data.


Robot vacuum transfer map data
NO
Whether to finish a complete map data upload
YES
ID++
NO
Is the ID greater than 65535
YES
ID=0

{width=100%}

Map streaming data transmission

Description:

- The offset represents the total length of data that the MCU has sent for one map.
- Currently, the maximum data that can be buffered by the serial port of the module can reach 1,024 bytes. The data in one map packet cannot exceed 1,024 bytes. The data content of one map packet is recommended to be 512 bytes.
- The map ID is an identifier of the data of a complete map. It will be changed after the cleaning is completed, that is, after the data of a map is complete and before new cleaning starts. Generally, the map ID is created in an incremental manner. When the map ID changes, the current map data is displayed on the app. The previous data will be cleared from the app interface.
- When data transmission starts, the module will stop sending heartbeat packets to ensure priority transmission of the map data. If the module is not powered off later, the heartbeat will not be initiated.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x28
Data length	2	0x0006+N
Data	2	Map ID: used as the identifier of the data of one map
	4	Offset (the first packet is 0)
	N	Entity data (in big-endian format)

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x28
Data length	2	0x0001
Data	Data	0x00: indicates success
		0x01: indicates the streaming service function is disabled
		0x02: indicates that it fails to connect to streaming server
		0x03: indicates data push times out
		0x04: indicates the length of the transmitted data is error
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.23 Wi-Fi functional test (connected to the designated router)

Description:

- MCU sends router information to the module, and the module returns data successfully. Then, the module uses relevant information to connect the router.
- MCU checks whether the router is connected according to the received packet. Status 4 indicates Wi-Fi has been configured and connected to the router. If MCU fails to receive a reply or status packet of a successful connection to the router for more than 15 seconds, the production test is regarded to be a failure.
- When the production test is successful, if you start the production test again, the test command can be resent. If the packet of successful connection to the router is not received, it indicates the module is being tested. You need to reset the module or power the module on again to send the test command.
- The module can complete a connection test when the network is not configured.
- Before the production test of connecting to the router, the heartbeat packet and product query packet must be replied, and the module has completed initialization.
- Router name string supports maximum of 32 bytes, and the router password string supports maximum of 64 bytes.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2C
Data length	2	xxxx
Data	Data	{ “ssid” :” xxx” ,” password” :” xxxxxxxx” } ssid: router name password: router password

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2C
Data length	2	0x0001
Data	1	Data length is 1 byte.
		Data[0]:
		0x00: indicates router information is fails to be received. Check whether the sent router JSON data packet is intact.
		0x01: indicates the router information is received successfully (for the result, see network status packet description in the section of reporting device network status.

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.24 Obtaining module MAC

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2d
Data length	2	0x0000
Data	Data	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2d

Field	Length (byte)	Description
Data length	2	0x0007
Data	Data	Data[0]: indicates whether the MAC address is obtained successfully.
		0x00 indicates success, the last 6 bytes of the MAC address is valid.
		0x01 indicates failure, the last 6 bytes of the MAC address is invalid.
		Data[1]–Data[6]: indicates the valid MAC address when Data[0] is success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.25 IR status notification (optional)

Device IR status	Description	Status value
Status 1	Send IR code	0x00
Status 2	The sending of IR code ends	0x01
Status 3	IR learning begins	0x02

Device IR status	Description	Status value
Status 4	IR learning ends	0x03

Description:

- The IR function can be configured in Tuya IoT Console, or the project manager activates it and sends the sample to you.
- The time period of IR code transmission is very short and requires real-time performance. The serial port data here are sent directly without re-transmission.
- You can set the status display as needed.
- Two I/O pins of the module are used to send and receive IR codes. If your device is in self-processing mode of the module, the I/O interface for IR function cannot be set for other I/O pins.

The module sends:

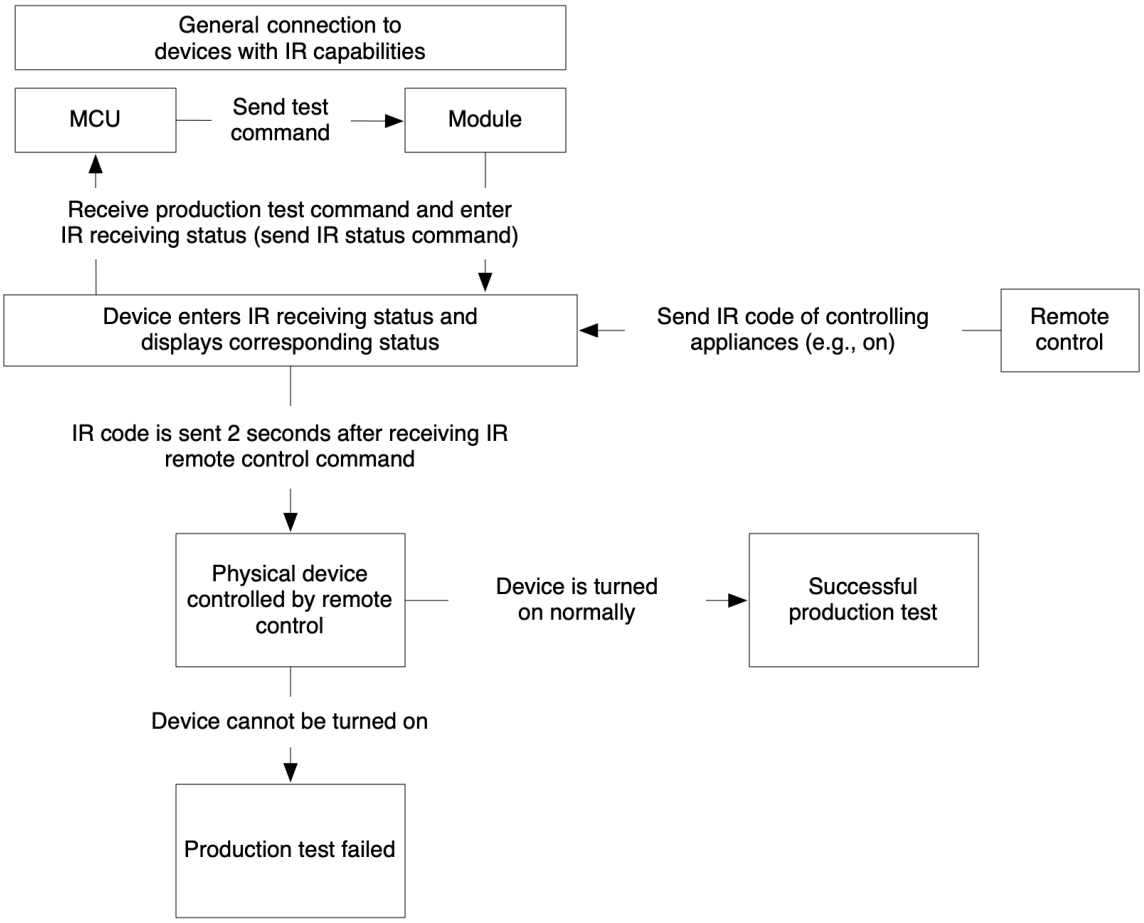
Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2e
Data length	2	0x0001
Data	Data	Indicates IR working status:
		0x00: status 1
		0x01: status 2
		0x02: status 3
		0x03: status 4

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2e
Data length	2	0x0000
Data	Data	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.26 Production test of IR receiving and sending (optional)



{width=100%}

Description:

- IR production test function is available when the network is not configured.
- After entering IR production test status, the module enters IR learning status.
- Once the module enters IR production test mode, the module remains in the production test status, learns continuously, and sends the learned data. The module exits the production test after the network is configured, or the module is powered off.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2f
Data length	2	0x0000
Data	Data	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2f
Data length	2	0x0001
Data	Data	0x00 indicates that it entered the production test of IR receiving and sending successfully.
		0x01 indicates that it fails to enter the production test of IR receiving and sending.

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.27 Map streaming data transmission (optional)

Description:

- Robot vacuum streaming service commands are not supported by all modules. To use this service, the device must adopt the specific module and activate related service.
- This service currently works to transmit robot vacuum map data, acting as an online channel of communication between the robot vacuum and map data on the Tuya app.
- A robot vacuum map is composed of multiple maps with different types of data. This type of robot vacuum can use this channel.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x30
Data length	2	0x0009+N
Data	1	Map service protocol version: 0x00.
	2	Map service session ID: a mark of a map display.

Field	Length (byte)	Description
	6	BUF[0] sub-map ID (one map session can be synthesized from multiple map data, such as a path map).
		BUF[1] indicates the processing method of map ID data. 0x00: accumulates data 0x01: clear the data uploaded by the sub-map ID.
		BUF[2]–BUF[5] indicates sub-map data offset (the first packet is 0).
	N	Entity data (in big-endian format)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x30
Data length	2	0x0001
Data	Data	0x00: indicates success

Field	Length (byte)	Description
		0x01: indicates failure
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.28 Downloading service of other files (optional)

Description:

- This channel is used to download application files other than MCU firmware. The files must be uploaded to the Tuya product management background, and they are used based on the app version.
- The firmware only serves as a data transmission channel and does not process the download files. You need to check the integrity of relevant files.
- DP data can be sent and reported normally while the file is being downloaded. It is not recommended to perform high-volume data communication during the file download, so as not to affect the efficiency of data download.

Initiating download (notification of file package size)

Description:

Before starting file download, the module firstly sends the file size to the MCU as a notification to start the download. When the MCU confirms that the package length is valid, it will pull the file package, otherwise, the pulling process will be interrupted and exit.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00

Field	Length (byte)	Description
Command	1	0x31
Data length	2	0x0004
Data	4	The file size is in byte measurement, the data type is unsigned integer, and the data storage is in big-endian format.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, the length of the file is 26,624, which is 26 KB.

```
55 aa 00 31 00 04 00006800 9C
```

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x31
Data length	2	0x0001
Data	1	Packet size of file package:
		0x00: 256 byte
		0x01: 512 byte
		0x02: 1,024 byte

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 31 00 01 00 34`

Transmitting file package

Description:

- The data format of upgrade package transmission: packet offset + packet data.
- If the MCU receives the frame with data length equal to 4 bytes and the upgrade packet offset is equal to or greater than the size of firmware, the packet transmission ends.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x32
Data length	2	The data length is the sum of 0X0004 and the data packet length
Data	N	The first four bytes are fixed as packet offset, and the latter bytes are the packet content

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example:

If the file to be upgraded has 530 bytes in size, under this circumstance, the MCU can skip the response to the last data packet.

- For the first packet data, the packet offset is 0x00000000, and the data packet length is 256. `0x55aa 00 32 0104 00000000 xx···xx XX`
- For the second packet data, packet offset is 0x00000100, and data packet length is 256. `0x55aa 00 32 0104 00000100 xx···xx XX`
- For the second to last packet data, the packet offset is 0x00000200, and the data packet length is 18. `0x55aa 00 32 0016 00000200 xx···xx XX`
- For the last packet data, packet offset is 0x00000212, and data packet length is 0. `0x55aa 00 32 0004 00000212 xx···xx XX`

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x32
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

For example, `55 aa 03 32 00 00 34`

4.29 Voice module protocol (optional)

Description:

- The protocols in this section apply to the general connection of the voice module VWXR2.
- The general firmware of other non-voice modules does not have the relevant protocol functions in this section.

Obtaining voice status code (optional)

Description:

The voice status code of the voice module will be returned automatically, and the MCU can actively query it.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x60
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x60
Data length	2	0x0001
Data	1	Voice status code:
		0: idle
		1: mic mute
		2: wake up
		3: recording
		4: recognizing
		5: recognized successfully
		6: failed to recognize
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Mic mute setting (optional)

Description:

This command can mute the mic and query mute status.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03

Field	Length (byte)	Description
Command	1	0x61
Data length	2	0x0001
Data	1	Mute setting value:
		0: mic is on
		1: mic is mute
		0xA0: query the mute status
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x61
Data length	2	0x0001
Data	1	Mute status value:
		0: mic is on
		1: mic is mute

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Speaker volume setting (optional)

Description:

This command can set the volume and query the volume.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x62
Data length	2	0x0001
Data	1	Volume value: 0-10
		Query volume: 0xA0
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa

Field	Length (byte)	Description
Version	1	0x00
Command	1	0x62
Data length	2	0x0001
Data	1	Volume value: 0-10
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Audio production test (optional)

Description:

The audio production test is to record and play at the same time and compare the input and output audio signals of the module through acoustic instruments.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x63
Data length	2	0x0001
Data	1	Audio production test value:
		0: close audio production test
		1: mic1 audio loop test
		2: mic2 audio loop test

Field	Length (byte)	Description
		0xA0: query current production test status
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x63
Data length	2	0x0001
Data	1	Current production test value
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Wake-up production test (optional)

Description:

After entering the wake-up test, it is required to play the electrical signal of the wake-up word for 10 seconds. It will return failure after 10 seconds.

MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x64
Data length	2	0x0000
Data	0	None
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x64
Data length	2	0x0001
Data	0	Wake-up return result
		0: wake-up failed
		1: wake-up succeeded
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Extension functions

Description:

Extend the system functions of the voice module

- Status notification and settings of play/pause, Bluetooth on/off, local alarm clock, and voice control group.
- Play/pause: play and pause music, poems, and jokes.
- Bluetooth on/off: turn on/off Bluetooth speaker.
- Local alarm clock: synchronization notification of the clock data set by the voice and app.
- Voice control group: notification of voice control command of the previous/next song.

- **MCU function setting**

  MCU sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x65
Data length	2	N
Data	1	Sub-command: 0x00
	`{ “play” :true,” bt_play” :true}`	`play`: play/pause function. `true` is for play, and `false` is for pause.
		`bt_play`: Bluetooth on/off function. `true` is for on, and `false` is for off.
		MCU settings currently only support play/pause and Bluetooth on/off.

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x65
Data length	2	0x0002
Data	1	Sub-command: 0x00
	1	Result:
		0x00: indicates success
		0x01: indicates failure
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

- **Status notification**

  The module sends:

Field	Length (byte)	Description
Header	2	0x55aa

Field	Length (byte)	Description
Version	1	0x00
Command	1	0x65
Data length	2	1+N
Data	1	Sub-command: 0x01
	{ “play” :true,” bt_play” :true,” alarm” :” xxxx” ,” ctrl_group” :” xxxx” }	play: play/pause function.true is for play, and false is for pause.
		bt_play: Bluetooth on/off function. true is for on, and false is for off.
		alarm: local alarm clock. xxx is string.
		ctrl_group: voice control group. xxx is string.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x65
Data length	2	2
Data	1	Sub-command: 0x01

Field	Length (byte)	Description
	1	Result:
		0x00: indicates success
		0x01: indicates failure
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.30 Extended service of the module

Enabling time service notification of the module

Description:

- If MCU requires the notification that whether the module time is calibrated in the cloud every time the module is powered on again, you can enable the module time service notification, and the module will actively send the time data to the MCU.
- When the module is powered on, the service is enabled. If the time is calibrated, the module will actively send the relevant time data to the MCU.
- When the module is not powered on again, the service cannot be enabled repeatedly after it is enabled. If the module does not reboot, you need to implement time obtaining protocol.
- Currently, for the protocol of obtaining system time (GMT) and obtaining local time, the timestamp will not be calibrated after the module is connected to the server. A time period determines that MCU will continuously attempt to obtain the time every time the module is powered on. Enabling this service will allow the module to actively notify MCU of the calibrated time.

MCU sends:

Field	Length	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0002
Data	1	0x01 (sub-command)
	1	0x00: indicates GMT
		0x01: indicates the local time
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	1	0x01 (sub-command)
	1	0x00: indicates the service is successfully enabled. 0x01: indicates that it fails to enable the service

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Time service notification of the module

Description:

- The module sends the corresponding time data according to the time service notification requirements.
- When the module is powered on again, this service is disabled. The MCU needs to resend the command to enable the service.

The module sends:

Field	Length	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0009
Data	1	0x02 (sub-command)
	1	0x00: indicates GMT
		0x01: indicates the local time
	7	Data length is 6 bytes:
		Data[0] is the year, and 0x00 represents the year of 2000.

Field	Length	Description
		Data[1] is the month, ranging from 1 to 12.
		Data[2] is the day, ranging from 1 to 31.
		Data[3] is the hour, ranging from 0 to 23.
		Data[4] is the minute, ranging from 0 to 59.
		Data[5] is the second, ranging from 0 to 59.
		Data[6] is the week, ranging from 1 to 7, and 1 represents Monday.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0001
Data	1	0x02 (sub-command)

Field	Length (byte)	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Active request for weather service data

Description:

- For certain products, it is required to actively obtain weather data based on the service of actively sending weather data every 30 minutes. This command is used to actively obtain weather service interface data.
- The frequency of using this command cannot be less than one minute, and multiple requests within one minute are processed only once.
- This command is used to confirm the data request. Sending data is still implemented through command 0x21.

MCU sends:

Field	Length	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0001
Data	1	0x03 (sub-command)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	1	0x03 (sub-command)
	1	Result:
		0x00: indicates success
		0x01: indicates failure
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Enabling module reset status notification (optional)

Description:

Currently, the module can be removed locally or on the app, and restore the factory settings on the app. But the MCU does not know the module status. This service is used to enable module status notification.

MCU sends:

Field	Length	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0001

Field	Length	Description
Data	1	0x04 (sub-command)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	1	0x04 (sub-command)
	1	Result:
		0x00: indicates success
		0x01: indicates failure
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module reset status notification (optional)

Device reset status	Description	Status value
Status 1	Locally reset the module	0x00

Device reset status	Description	Status value
Status 2	Remotely reset the moduleon the app	0x01
Status 3	Restore factory setting on the app	0x02

Description:

The reset status will be sent twice at most at the interval of one second.

The module sends:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	1	0x05 (sub-command)
	1	Reset status:
		0x00: locally reset the module
		0x01: remotely reset the module on the app
		0x02: restore factory setting on the app
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

MCU returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0001
Data	1	0x05 (sub-command)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

4.31 Bluetooth function (optional)

Bluetooth functional test (scan the specified Bluetooth beacon)

Description:

- The module scans the Bluetooth beacon of `ty_mdev` and returns results and signal strength percentage.
- In order to prevent defective products to the greatest extent, it is recommended that the distance between the router and the device should be about 5 meters. The test result is qualified if the signal strength is greater than or equal to 60%. It can be adjusted based on the actual production line and factory environment.

MCU sends:

Field	Length	Description
Header	2	0x55aa
Version	1	0x03

Field	Length	Description
Command	1	0x35
Data length	2	0x0001
Data	1	0x01 (sub-command)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

Module returns:

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x35
Data length	2	0x0003
Data	1	0x01 (sub-command)
	2	Data length has 2 bytes.
		Data[0]: 0x00 indicates failure, and 0x01 indicates success.
		When Data[0] is 0x01, which indicates success, Data[1] is signal strength (from 0 to100, 0 represents the weakest signal, and 100 represents the strongest signal).

Field	Length (byte)	Description
		When Data[0] is 0x00, which indicates failure, if Data[1] is 0x00, the specified Bluetooth beacon is not found; if Data[1] is 0x01, an authorization key is not burned to the module.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder

5 Version history

Version	Description	Date	Remark
1.2.3	Modified	20200521	1. Add the IR function field and low power selection field to the product information packet.
			2. Add the command to actively obtain weather service.
			3. Add the command to notify reset status.
			4. Modify the name description of the module working mode.
1.2.2	Modified	20200411	Add the extension functions of voice module VWXR2, supporting play/pause, Bluetooth on/off, local alarm clock, and control group.
1.2.1	Modified	20200409	1. Add Wi-Fi production test.

Version	Description	Date	Remark
			2. Add Wi-Fi remote control function.
1.2.0	Modified	20200331	Add functions that adapt to CI baselines, including network status packet and product information field.
1.1.9	Modified	20200326	1. Reorganize the voice module protocols.
			2. Add module extension services.
			3. Add dual-mode Bluetooth production test.
1.1.8	Modified	20200218	1. Add volume setting and mic control protocol to voice module VWXR2.
			2. Add audio production test and wake-up production test protocol.

Version	Description	Date	Remark
1.1.7	Modified	20191119	1. Add data protocol for multiple maps of the robot vacuum.
			2. Add a third-party file download channel.
			3. Improve the document protocol description.
1.1.6	Modified	20190828	Add IR capability protocols.
1.1.5	Modified	20190824	1. Add the interface to obtain the MAC address of the module.
			2. Improve the function description of obtaining weather data.
1.1.4	Modified	20190617	1. Add the performance test of the Wi-Fi connection.

Version	Description	Date	Remark
			2. Optimize the streaming service process, and only retain the streaming data report protocol. The firmware is compatible with the old version of the streaming service mechanism.
1.1.3	Modified	20190415	1. Improve the description of the function command mechanism.
			2. Add the map service command for the robot vacuum.
1.1.2	Modified	20181217	1. Add serial port network protocol.
			2. Add the protocol of MCU obtaining Wi-Fi network status.
1.1.1	Modified	20180810	Modify the return package content of initiating MCU firmware upgrade (compatible with old firmware).

Version	Description	Date	Remark
1.1.0	Modified	20180329	Add the protocol to disable Wi-Fi module heartbeat.
1.0.9	Modified	20180119	1. Add the function of the synchronizing command report. After receiving the synchronization command, the Wi-Fi module will notify MCU when the report fails or succeeds.
			2 Add the function of obtaining Wi-Fi signal strength, in unit of DB.
1.0.8	Modified	20170512	1. Add the interface for enabling weather data obtaining.
			2. Add the interface for sending weather data.
			3. Adjust the heartbeat detection interval to 15 seconds.

Version	Description	Date	Remark
1.0.7	Modified	20170216	1. Add the network configuration mode setting (extend the interface for product information query).
			2. The MCU protocol version number is uniformly upgraded to 0x03.
1.0.6	Modified	20161110	1. Add Wi-Fi working status.
			2. The MCU protocol version number is uniformly upgraded to 0x02.
1.0.5	Modified	20160607	1. Delete the upgrade query command.
			2. Delete the command to notify MCU to enter the production test mode.

Version	Description	Date	Remark
			3. Modify the upgrade startup protocol, supporting file sizes above 64 KB.
1.0.4	Modified	20160512	1. Add the command to obtain the local time.
			2. Add the Wi-Fi functional test.
			3. Add the command to obtain the module memory.
1.0.3	Modified	20151114	1. Add the function of MCU obtaining time.
			2. Add the function of MCU obtaining time zone.
			3. Add the function of entering the production test.

Version	Description	Date	Remark
1.0.2	Modified	20151017	1. Add the function of the MCU rebooting test to the heartbeat detection protocol.
			2. The heartbeat detection interval is adjusted to 10 seconds.
			3. The module Wi-Fi status report is updated to the module actively reporting the status to the MCU.
1.0.1	Modified	20151013	Change product ID query to module information query, and add the function of returning device version information.
1.0.0	The first release.	20151010	Create