



Serial Communication Protocol

Version: 20241031

Online Version

Contents

1. Terms	2
2. Serial communication	3
3. Feature description	4
3.1. Model comparison	4
3.2. Wake-up pin convention	5
4. Module GPIO and ADC	9
5. Frame format	12
6. Response format for unsupported commands	14
7. Basic services	15
7.1. Heartbeat check (0x00)	15
7.2. Query product information (0x01)	16
7.3. Query working mode (0x02)	20
7.4. Report network status (0x03)	22
7.5. Reset module (0x04)	24
7.6. Get version information (0x71 41)	25
8. DP services	27
8.1. Data units	27
8.2. Send commands (0x06)	28
8.3. Report status (0x07)	29
8.4. Query status (0x08)	30
8.5. Report non-record type status (0x22)	31
8.6. Report record type status (0x26)	32
9. MCU firmware update	35
9.1. Initiate update (0x0a)	37
9.2. Transmit update file (0x0b)	39
9.3. Notify MCU of result (0x34 09)	41
10. Cellular services	43
10.1. Get system time in GMT (0x0c)	43
10.2. Test module functionality (0x0e)	44
10.3. Get local time (0x1c)	46
10.4. Get module's memory (0x0f)	49
10.5. Get Unix timestamp (with time zone) (0x1b)	50
10.6. (Optional) Get signal strength (0x24)	52
10.7. (Optional) Notify module to disable heartbeat (0x25)	53
10.8. Get network status (0x2b)	54

10.9. Get module's MAC address (0x2d)	56
10.10. Set cellular mode (0x05)	57
10.11. Get cellular mode (0x71 01)	58
10.12. Get IMSI (0x71 02)	59
10.13. Get ICCID number (0x71 03)	60
10.14. Get IMEI (0x71 04)	61
10.15. Get cellular network type (0x71 43)	62
10.16. Query the current PLMN (0x71 48)	63
10.17. (Optional) Set or get radio resource control (RRC) (0x71 4A) ..	65
11. Password and weather services	67
11.1. Offline dynamic password (0x16)	67
11.2. Request temporary password (with schedule list) (0x14)	68
11.3. Password positional notation (0x17)	71
11.4. (Optional) Enable weather service (0x20)	76
11.5. (Optional) Send weather data (0x21)	77
12. Bluetooth LE and LTE Cat.1 combo dedicated protocol	80
12.1. Product authorization	80
12.2. Query module's unique identifier (0xD0 00)	81
12.3. Sync authorization information (0xD0 01)	82
12.4. Sync activation information (0xD0 02)	84
12.5. Get key (0xD0 03)	86
13. Positioning services	88
13.1. (Optional) Enable or disable GNSS positioning (0x72 81)	88
13.2. (Optional) Report GNSS information automatically (0x72 90)	91
13.3. (Optional) Enable or disable Wi-Fi positioning (0x72 82)	92
13.4. (Optional) Report Wi-Fi information automatically (0x72 91)	93
13.5. (Optional) Report LBS information automatically (0x72 92)	95
13.6. (Optional) Reset GNSS module (0x72 83)	96
13.7. Get GNSS positioning (longitude, latitude) (0x71 10)	98
13.8. Get GNSS positioning (latitude,longitude) (0x71 29)	99
13.9. (Optional) Get GNSS's SNR (0x71 11)	100
13.10. (Optional) Get GNSS positioning speed (0x71 12)	101
13.11. (Optional) Get Wi-Fi positioning information (0x71 20)	103
13.12. Get LBS positioning information (0x71 21)	104
13.13. Control LZ211 GNSS's main power supply (0x72 85)	106
13.14. Get LZ211 GNSS's power status (0x71 22)	107
13.15. AGNSS on/off (0x72 86)	108
13.16. (Optional) Get the number of visible GNSS satellites (0x71 15) ...	109
13.17. (Optional) Get the enablement of positioning features (0x71 30) .	110

13.18. (Optional) Get GNSS heading (0x71 13)	112
13.19. (Optional) Get GNSS horizontal dilution of precision (HDOP) (0x71 16)	113
13.20. (Optional) Get GNSS orthometric height (0x71 17)	114
13.21. (Optional) Enable GNSS filter algorithm (0x72 18)	115
13.22. (Optional) Set GNSS filter algorithm (0x72 19)	116
13.23. (Optional) Get GNSS positioning information (0x71 1A)	117
14. Phone call services	121
14.1. (Optional) Incoming call notification (0x71 23 00)	121
14.2. (Optional) Outgoing calls (0x71 23 01)	122
14.3. (Optional) Answer calls (0x71 23 02)	123
14.4. (Optional) Hang up calls (0x71 23 03)	124
14.5. (Optional) Query call status (0x71 23 04)	125
14.6. (Optional) Report call status (0x71 23 04)	127
14.7. Enable/disable VoLTE (0x72 94)	127
14.8. Send DTMF commands (0x72 95)	129
14.9. Enable listening for DTMF (0x72 99)	131
14.10. Report DTMF events (0x71 95)	132
14.11. (Optional) Set or query mic gain for calls (0x71 46)	133
14.12. (Optional) Set or query sidetone gain (0x71 47)	135
15. SMS services	138
15.1. (Optional) Receive SMS messages (0x71 24 00)	138
15.2. (Optional) Send SMS messages (0x71 24 01)	140
15.3. (Optional) Set SMS encoding (0x71 24 02)	142
15.4. Set SMS transmission and reception format (0x71 24 03)	144
16. Battery services	146
16.1. (Optional) Query battery level (0x71 25)	146
16.2. (Optional) Query battery charging status (0x71 26)	146
17. Audio services	149
17.1. (Optional) Set volume (0x72 87)	149
17.2. (Optional) Audio playback from SD card (0x71 27)	150
17.3. (Optional) Query local audio playback status (0x71 28)	152
17.4. (Optional) Report local music playback finished (0x71 2a)	153
17.5. (Optional) Set the mute state (0x72 97)	154
17.6. (Optional) Online TTS service (0x71 & 0x45)	155
17.7. (Optional) Audio playback from memory (0x71 49)	157
18. Bluetooth services	162
18.1. (Optional) Turn on/off control over Bluetooth (0x72 89)	162
19. Function control	164
19.1. (Optional) Enable/disable low voltage shutdown (0x72 88)	164

19.2. Send QR code to MCU proactively (0x72 93)	165
19.3. Set timeout for network exceptions (0x72 96)	165
19.4. (Optional) Restart module (0x72 98)	167
19.5. Report cellular network alerts (0x71 9a)	167
19.6. Set external wake-up pin (0x72 9b)	168
19.7. Enable/disable remote logging (0x72 9c)	169
19.8. Notify MCU of restarting device (0x72 9d)	170
19.9. Enable SIM card hot swapping (0x72 9e)	171
19.10. Control ADC (0x72 A0)	172
19.11. Control GPIO (0x72 A1)	176
20. Extended services	180
20.1. (Optional) Enable reset status notification (0x34 04)	180
20.2. (Optional) Notify MCU of reset status (0x34 05)	181
20.3. Send update in progress status (0x34 0A)	182
21. Version history	184

The serial protocol is used for serial communication between Tuya's LTE Cat.1 module and the third-party MCU.

```
1 sequenceDiagram
2     Tuya LTE Cat.1 Module ->> User MCU: RX->TX
3     User MCU ->> Tuya LTE Cat.1 Module: TX<-RX
```

1. Terms

- Module: refers to Tuya's LTE Cat.1 module in this topic. For more information, see [LTE Cat.1 Module](#) .
- GNSS: stands for Global Navigation Satellite System.

2. Serial communication

- Baud rate: 115200, 9600, 460800, and 921600. The baud rate will be automatically detected for two minutes. If not specified, the default baud rate of 115200 will be used.
- Data bit: 8
- Parity check: none
- Stop bit: 1
- Data flow control: none
- MCU: microcontroller unit. Your MCU communicates with Tuya's modules through the serial port.

3. Feature description

3.1. Model comparison

The following table provides a comparison of LTE Cat.1 modules.

- Y represents support.
- N represents not support.
- 0 represents external-based support.

	FE2201C	NL7521CN	FE2201E	FE2211E	FE3800	FE3800	FE3800	FE3800	FE3800	FE3800	FE3800	FE3800
	FE2201C	NL7521CN	FE2201E	FE2211E	FE3800	FE3800	FE3800	FE3800	FE3800	FE3800	FE3800	FE3800
Basic services	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Data point (DP) service	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCU update	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Cellular service	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Password service	Y	Y	Y	Y	Y	N	N	N	N	N	N	N
Weather service	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Bluetooth												
Low Energy (LE) and LTE Cat.1 combo	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Positioning service	0	Y	0	Y	0	0	Y	0	0	0	0	0

Feature	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026
AGNSS service	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Wi-Fi Scan	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	N
Phone call service	Y	Y	Y	Y	Y	Y	N	N	N	N	N
SMS service	Y	Y	Y	Y	Y	Y	N	Y	N	N	N
Battery service	Y	Y	Y	Y	Y	Y	N	N	N	N	N
Audio service	Y	Y	Y	Y	Y	Y	N	N	N	N	N
Bluetooth service	Y	Y	N	N	N	N	N	N	N	N	N
Function control	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Extended services	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
GPIO services	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y
ADC services	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y

3.2. Wake-up pin convention

When the module has data to send to the MCU, the wake-up pin on the module will be pulled down to low for 60 ms by default. The following table lists the wake-up pin for different module models.

Module model	Wake-up pin	UART pin	Description
--------------	-------------	----------	-------------

Module model	Wake-up pin	UART pin	Description
- LZ201CN - LZ211CN - LZ211EAU - LZ201EAU - LZ501CN - TCS600U	- Pin 61: LZ501CN - Pin 62: LZ201CN - LZ211CN - LZ211EAU - LZ201EAU	UART1	- Pull down the specified pin for 60 ms. - UART2 supports interfacing with Tuya's GNSS module.
L610EU	Pin 37	UART1	- Pin 62 is the wake-up pin for the basic firmware. You can refer to the AT+GTWAKE command, but this method is not recommended. - UART2 supports interfacing with Tuya's GNSS module.

Module model	Wake-up pin	UART pin	Description
MA510GL	Pin 61	UART3	- When UART3_TXD is high, the device cannot start up. - UART2 supports interfacing with Tuya's GNSS module. - UART1 supports AT command. - In low power mode, pull down the Pin 41 more than 30ms to wake up the cellular module.
NL668AM	Pin 62	129(TX) 130(RX)	-
TCS800E	Pin 20	UART1(RX17,TX18)	UART2(RX62,TX67) supports interfacing with Tuya's GNSS module.
TCS600E	Pin 20	UART1(RX31,TX32)	UART2(RX62,TX67) supports interfacing with Tuya's GNSS module.
MC665EU	Pin 61	UART1(RX18,TX19)	UART2(RX39,TX40) supports interfacing with Tuya's GNSS module.

Module model	Wake-up pin	UART pin	Description
MC665CNMG661EUMG661LA	Pin 20	MAIN_TXD(18),MAIN_RXD(17)	AUX_UART(RX28,TX29) supports interfacing with Tuya's GNSS module.
			AUX_UART (RX28, TX29) supports interfacing with Tuya's GNSS module.
L511AL511CL511E	Pin 20	MAIN_TXD (18), MAIN_RXD (17)	In low power mode, pull down the Pin 19 more than 30ms to wake up the cellular module.

4. Module GPIO and ADC

Module model	ADC	Reference voltage (mV)	Tuya GPIO	Pinout (physical pin indicated in parentheses)
LZ501	0-1	1250/2444/3233/5000	0 to 3, 5 (wake up), 9 to 10, 17 to 21, 26, 29 to 31	See LZ501-CN Datasheet .
LZ211EAU	0-2	1250/2444/3233/5000	0 to 3, 5, 7 to 11, 12 (wake up), 13 to 15, 17 to 19	See LZ211-EAU Datasheet .
TCS600U	0-1	1250/2444/3233/5000	5 (wake up), 8 (net_led), 9 to 12, 14 to 15, 17 to 22	See TCS600U Datasheet .
TCS600ETCS800E	0-1	3,400 mV	24 (wake up), others	See TCS600E Datasheet and TCS800E Datasheet .

Module model	ADC	Reference voltage (mV)	Tuya GPIO	Pinout (physical pin indicated in parentheses)
MG665EU	0-1	VBAT, range (0 to VBAT)	2, 4, 6 (wake up), 8 to 11, 14 to 23, 29	2 (38), 4 (41), 8 (39), 9 (40), 10 (75), 11 (76), 14 (21), 15 (20), 16 (78), 17 (77), 18 (82), 19 (85), 20 (84), 21 (83), 22 (100), 23 (99), 29 (56)
				0 (25), 2 (16), 6 (28), 7 (29), 8 (84), 10 (85), 16 (68), 17 (69), 18 (101), 20 (31), 21 (32), 22 (33), 29 (5), 30 (6), 31 (75), 32 (83), 33 (86), 34 (76), 35 (74), 42 (67), 43 (66)
MC661CNMG661LAAMG661EU0-1		VBAT, range (0 to VBAT)	0, 1 (wake up), 2, 6 to 8, 10, 16 to 18, 20 to 22, 29 to 35, 42 to 43	

Module model	ADC	Reference voltage (mV)	Tuya GPIO	Pinout (physical pin indicated in parentheses)
L511A	0-1	0 to 1.2V	16, 19 (wake up the module in low power mode), 20 (wake up the MCU), 21-23, 25, 26, 30-33, 49-58, 78	/
L511CL511E	0-1	0-1.2V	16, 19 (wake up the module in low power mode and cannot be used as GPIO), 20 (wake up the MCU), 21-23, 25	/

5. Frame format

Field	Byte(s)	Description
Header	2	It is fixed to 0x55aa .
Version	1	It is used for updates and extensions.
Command	1	The frame type.
Data length	2	Big-endian format.
Data	N	-
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

- All data greater than one byte is transmitted in big-endian format.
- Typically, one command is sent by one party and received by the other party synchronously. That is, one party sends a command and waits for a response from the other party. If the sender does not receive a correct response packet within a specified time period, the transmission times out.

```
1 sequenceDiagram
2 MCU ->> Module: Send the command N
3 Module ->> MCU: Respond to the command N
```



For more information, see the [Protocol list](#) .

- If the module does not respond to the MCU command within two minutes, the MCU must restart it. After the MA510 module completes an OTA update, it will restart without notifying the MCU. If the MA510 module starts with `uart3_tx` high, it will fail to start and require the MCU to trigger a restart.
- Both command sending and status reporting use asynchronous transmission. Assume that the module sends a control **command X** and the MCU reports **status Y**. The data transmission is performed as shown below.
- The module sends a control command:

```
1 sequenceDiagram
2 Module ->> MCU: Send the control command X
3 MCU ->> Module: Report the status Y
```

- The MCU reports status:

```
1 sequenceDiagram
2 MCU ->> Module: Report the status Y
```

6. Response format for unsupported commands

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0xFF
Data length	2	2+N (version information)
Data	2+N	• Data[0]: Command • Data[1]: Subcommand • Data[2]: Version information in string format.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 AA 00 FF 00 07 70 24 31 2E 30 2E 31 87

7. Basic services

7.1. Heartbeat check (0x00)

- The module sends heartbeats to the MCU every 15 seconds. If the MCU does not respond within the 90-second timeout period, the module considers a communication exception occurred and will restart the software.
- The MCU can determine whether the module works properly by the regular heartbeat check. If the MCU does not receive a heartbeat packet as expected, it can use the reset pin to reset the module.

The module sends the following data.

Field	Byte(s)	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, 0x55aa 00 00 0000 ff

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x00
Data length	2	0x0001

Field	Byte(s)	Description
Data	1	• 0x00 : The MCU returns this value only one time after a restart. The module uses this value to determine whether the MCU restarts during operation. • 0x01 : The MCU returns this value except for the first response after a restart.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

- The MCU returns 0x55aa 03 00 0001 00 03 after a restart.
- The MCU returns 0x55aa 03 00 0001 01 04 except for the first response after a restart.

7.2. Query product information (0x01)

Product information consists of the product ID (PID) and the MCU software version number.

- PID: a unique identifier assigned to each product created on the [Developer Platform](#) for storing product information in the cloud.
- MCU software version number: It is expressed in dot-decimal notation x.x.x where x is a decimal digit between 0 and 99.

The module sends the following data.

Field	Byte(s)	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.

Example: 0x55aa 00 01 0000 00

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x01
Data length	2	N

```
{“p”:“Alp08kLlftb8x2x0”, “v”:“1.0.0”, “m”:1, “apn”:“xxx”, “mht”:60, “qr”:1
```

```
[{ “ch”:1, “v”:“0.1.1” },
```

```
...
```

```
{ “ch”:n, “v”:“0.1.n” }], “U”:0, “dp_cache”:1, “cops”:“xxx”dp_ack”
```

apn is optional. If the apn is not configured, the system will automatically set it based on the SIM card.

Typically, apn configuration is required for SIM cards from tier 2 carriers. mht , qr , and slave are optional.

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

Example: {"p":"AIp08kLIftb8x2x0", "v":"1.0.0", "m":1}

- **p** : indicates the PID is AIp08kLIftb8x2x0 .
- **v** : indicates the MCU version is 1.0.0.
- **m** : indicates the power consumption of the module.
- **0** : standard power.
- **1** : low power. The period of the provided timer is one second.
- **mht** : indicates the MQTT heartbeat interval for the module, in seconds. The default value is 300 seconds.
- **qr** : specifies whether to get the short URL of the QR code for pairing.
- **1** : Get the short URL.
- **0** : Not get the short URL. If a device has been paired, the short URL will not be pulled.
- **slave** : specifies whether the LTE Cat.1 module works in tandem with a Bluetooth LE module.
- **ext_ota** : OTA update for subordinate MCU firmware. **v** : version information. The maximum length is 10 bytes, with up to six update channels supported. The update channels dedicated to the application firmware and basic firmware must not be used.
- **u** : specifies the protocol for MCU OTA updates.
- **0** : protocol version 0
- **1** : protocol version 1

i

- ▶ If `slave` is set and `u` is not set, protocol version 0 is used.
- ▶ If both `slave` and `u` are set, `0` and `1` represent protocol version 0 and protocol version 1 respectively.
- ▶ If both `slave` and `u` are not set, protocol version 0 is used.
- ▶ If `slave` is not set and `u` is set, `0` and `1` represent protocol version 0 and protocol version 1 respectively.

- `dp_cache` : specifies whether to enable DP cache. DP cache requires an additional cloud-based feature. For more information, contact your account manager.
- `1` : Enable.
- Other values: Disable.
- (Optional) `cops` : specifies the public land mobile network (PLMN) of the mobile operator. The value must be numeric encoding. For example, `cops:"20416"` is equivalent to AT command `AT+COPS=1,2, "xxxxx"` .
- (Optional) `dp_ack` : specifies whether to enable acknowledgment (ACK) response. Set it to `1` to enable ACK. If the module does not receive an ACK message after it sends DP data to the MCU, it will transmit the packet again. A packet can be retransmitted up to three times, with an ACK wait time of 500 ms.
- `online_tts` : setting it to `1` indicates that online text-to-speech (TTS) is supported. This feature applies when the module supports the audio service.
- `gps` : GPS model, defaulting to Unicore UFirebird. Allystar's GPS module also applies. When using the assisted GNSS (AGNSS) service, you need to specify the GPS model. The MCU integration protocol must be v1.1.3 or later.
- `env_pre` : `1` for the staging environment, and other values for the production environment. The MCU integration protocol must be v1.1.3 or later.
- `net_type` (for MA510 only): The MCU integration protocol must be v1.1.4 or later.
- `0` or empty: Auto
- `1` : GSM
- `2` : LTE

- 4 : CATM
- 5 : NB-IoT



When specifying this field, check the supported network types of the module.

7.3. Query working mode (0x02)

The working mode indicates how the network status is indicated and the way to trigger module reset. Theoretically, the module can work in the following two modes:

- The MCU works with the module to process network events.
- The module sends its current work status to the MCU through serial communication. The MCU controls the LED to indicate status accordingly.
- When the MCU detects a request to reset the module, it directs the module to reset through serial communication.
- The module processes network events itself.
- The `NET_MODE` pin on the module drives the LED indicator to indicate the network status.
- The GPIO input signal determines module reset. When the module detects a low level on the reset pin for more than three seconds, it will trigger a reset action. The following command specifies the GPIO pins of the LED indicator and the reset button. This feature is not available in low power mode.



- In the module self-processing mode, the LED indicator can only be defined by `NET_MODE`. In this case, **the module does not report network status**.
- For MA510 modules, it is recommended to reserve the USB port for testing purposes on the local network at the fulfillment location.
- Ensure correct APN configuration, as a wrong one can lead to internet connection failure.

The module sends the following data.

Field	Byte(s)	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, 0x55aa 00 02 0000 01

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x02
Data length	2	0x0000/0x0002 0x0000 : The module works with the MCU to process network events. You need to implement the functionality mentioned in the above description. 0x0002 : The module processes network events itself.

Field	Byte(s)	Description
Data	0/2	The data length is 2 bytes. The first byte indicates the GPIO pin of the LED indicator. The fixed value is 0. The second byte indicates the GPIO pin of the reset button.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

- The module works with the MCU to process network events: 0x55aa 03 02 0000 04 is returned.
- The module processes network events itself: 0x55aa 03 02 0002 0c0d 1f is returned. 0x0c indicates the LED indicator is connected to GPIO12. 0x0d indicates the reset button is connected to GPIO13.

7.4. Report network status (0x03)

Network status	Description	Status value
Status 1	The SIM card is not connected.	0x00
Status 2	The module searches for cellular networks.	0x01
Status 3	The module is registered with the cellular network but not connected to the network.	0x02

Network status	Description	Status value
Status 4	The module is connected to the network and gets an IP address.	0x03
Status 5	The module has been connected to the cloud.	0x04
Status 6	SIM card registration is denied, which might be because it is not subscribed to a cellular service provider.	0x05
Status 7	The module is ready for pairing.	0x06
Status 8	Unknown status	0xFF

- When the module detects that the MCU is restarted or reconnected, it will proactively send the current network status to the MCU.
- When the network status of the module changes, the module will proactively send its current status to the MCU.
- If you choose the **module self-processing mode**, implementing this protocol for your MCU is not necessary.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x03
Data length	2	0x0001
Data	1	See network status .
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

For example, 0x55aa 00 03 0001 00 03

The MCU returns the following data.

Field	Byte(s)	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, 0x55aa 03 03 0000 05

7.5. Reset module (0x04)

When the LTE Cat.1 module receives the reset command, it unbinds the device from the cloud.

The MCU sends the following data.

Field	Byte(s)	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, 0x55aa 03 04 0000 06

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa

Field	Byte(s)	Description
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.

Example: 0x55aa 00 04 0000 03

7.6. Get version information (0x71 41)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x41
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002+n

Field	Byte(s)	Description
Data	N	• Subcommand: 0x41 • Result: • 0 : Failure. • 1 : Success. • Version information: The content is in string format. Example: <code>fw_name:xxx, fw_ver:xxx; app_name:xxx, app_ver:xxx; pro_name:xxx, pr</code>
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

8. DP services

8.1. Data units

The following table defines data units .

Field	Byte(s)	Description
dpid	1	The ID of a DP .
type	1	The data type of a DP defined on the Developer Platform . For more information, see the description of the type field in the following table.
len	2	The length of a value , in bytes. For more information, see the description of the type field in the following table.
value	1/2/4/N	Represented in hexadecimal format. Data greater than one byte is transmitted in big-endian format.

Description of the type field

Type	Data type	Byte(s)	Description
0x00	Raw	N	Represents a DP of raw data type. The module transmits the raw data to the receiver.

Type	Data type	Byte(s)	Description
0x01	Boolean	1	Represents a DP of Boolean data type. Valid values: • 0x00 • 0x01
0x02	Value	4	Represents a DP of integer data type. The data is represented in big-endian format.
0x03	String	N	Represents a DP of string data type.
0x04	Enum	1	Represents a DP of enum data type, ranging from 0 to 255.
0x05	Bitmap	1/2/4	Data greater than one byte is transmitted in big-endian format.

i

- Except for the `Raw` data type, all others belong to the object type.
- **One command** can contain data units of multiple `DPs`.
- Asynchronous communication applies when the module **sends control commands** and the MCU **reports DP status**.

8.2. Send commands (0x06)

The module sends the following data.

Field	Byte(s)	Description
-------	---------	-------------

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x06
Data length	2	It depends on the types and the number of data units .
Data	N	Data units
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Suppose that DP 3 of Boolean type is used for on/off control, and 1 means to turn on the device, the module sends 0x55aa 00 06 0005 03 01 0001 01 10 to the MCU.

8.3. Report status (0x07)

- For more information, see [Data units](#) .
- This is an asynchronous command. The MCU uses it to report device status to the module, which can be triggered by three mechanisms.
- After the MCU executes the command received from the module, it reports the changed DP status to the module.
- When the MCU proactively detects status changes of DPs, it reports the changed DP status to the module.
- When the MCU receives the [DP status query](#) , it sends the status of all DPs to the module.
- A piece of status data can contain data units of multiple DPs.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x07

Field	Byte(s)	Description
Data length	2	It depends on the types and the number of data units.
Data	N	Data units
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Suppose that DP 5 of value type is used for reporting the current humidity, and the current humidity is 30%, the MCU reports `0x55aa 03 07 0008 05 02 0004 0000001e 3a` to the module.

8.4. Query status (0x08)

- The module asynchronously queries the status of all object DPs. When the MCU receives a query, it sends the status of DPs to the module through the command `0x07`.
- The module sends a status query to the MCU when the following two events occur.
- When powered on for the first time, the module sends a status query after it connects to the MCU and the Developer Platform.
- When the module detects the MCU is restarted or disconnected and then goes online, the module sends a status query.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x08
Data length	2	0x0000
Data	0	None

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

Example: 0x55aa 00 08 0000 07

8.5. Report non-record type status (0x22)

- This is a synchronous command. The MCU reports DP status and then waits for the result from the module.
- The MCU should not send a new request until receiving a response to the previous status reporting.
- If the data fails to be reported to the cloud due to poor network quality, the module will return a failure code five seconds later. In this case, the MCU should wait for more than five seconds.
- For more information, see [Data units](#).
- A piece of status data can contain data units of multiple DPs.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x22
Data length	2	It depends on the types and the number of data units.
Data	N	Data units
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x23
Data length	2	0x01
Data	1	0x00 : Failure. 0x01 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

8.6. Report record type status (0x26)

- This is a synchronous command. The MCU reports DP status and then waits for the result from the module.
- The MCU should not send a new request until receiving a response to the previous status reporting.
- If the data fails to be reported to the cloud due to poor network quality, the module will return a failure code five seconds later. In this case, the MCU should wait for more than five seconds.
- For more information, see [Data units](#) .
- A piece of status data can contain data units of multiple DPs.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x26
Data length	2	The length of the DP data unit plus seven

Field	Byte(s)	Description
Timestamp	7	- Data[0]: specifies whether the local time is included. 0 : The local time provided by the MCU is not included. The module considers the received time data invalid. 1 : The local time is included. 2 : The time in GMT is included. - Data[1]: indicates the year. 0x00 indicates the year 2000. - Data[2]: indicates the month, ranging from 1 to 12. - Data[3]: indicates the day, ranging from 1 to 31. - Data[4]: indicates the hour, ranging from 0 to 23. - Data[5]: indicates the minute, ranging from 0 to 59. - Data[6]: indicates the second, ranging from 0 to 59.

Field	Byte(s)	Description
Data	N	Data units
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x26
Data length	2	0x0001
Data	1	0x00 : Failure. 0x01 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

9. MCU firmware update

- You can specify the method for the OTA firmware update. The module only serves as the channel for update transmission, without any data parsing operation.
- Developer Platform provides four update methods.
- Update notification: Users receive a firmware update notification on the app and choose whether to install updates.
- Automatic update: Users will not receive any update pop-up window. The module checks for firmware updates within one minute after power-on. If any new version is available, it will automatically pull the updates. The module checks for updates every 24 hours after the first-time power-on.
- Forced update: Users receive a firmware update notification on the app and have no option but to update the firmware.
- Check for updates: Users will not receive a firmware update notification on the app but need to manually check for new updates.
- After the update packet transmission is finished, the module will send the command `0x01` to the MCU to [query the product information](#). The MCU must reply with the new MCU firmware version number within one minute. The new version number should be consistent with that configured on the Developer Platform. The following flowchart shows how the OTA firmware update works.

graph TD

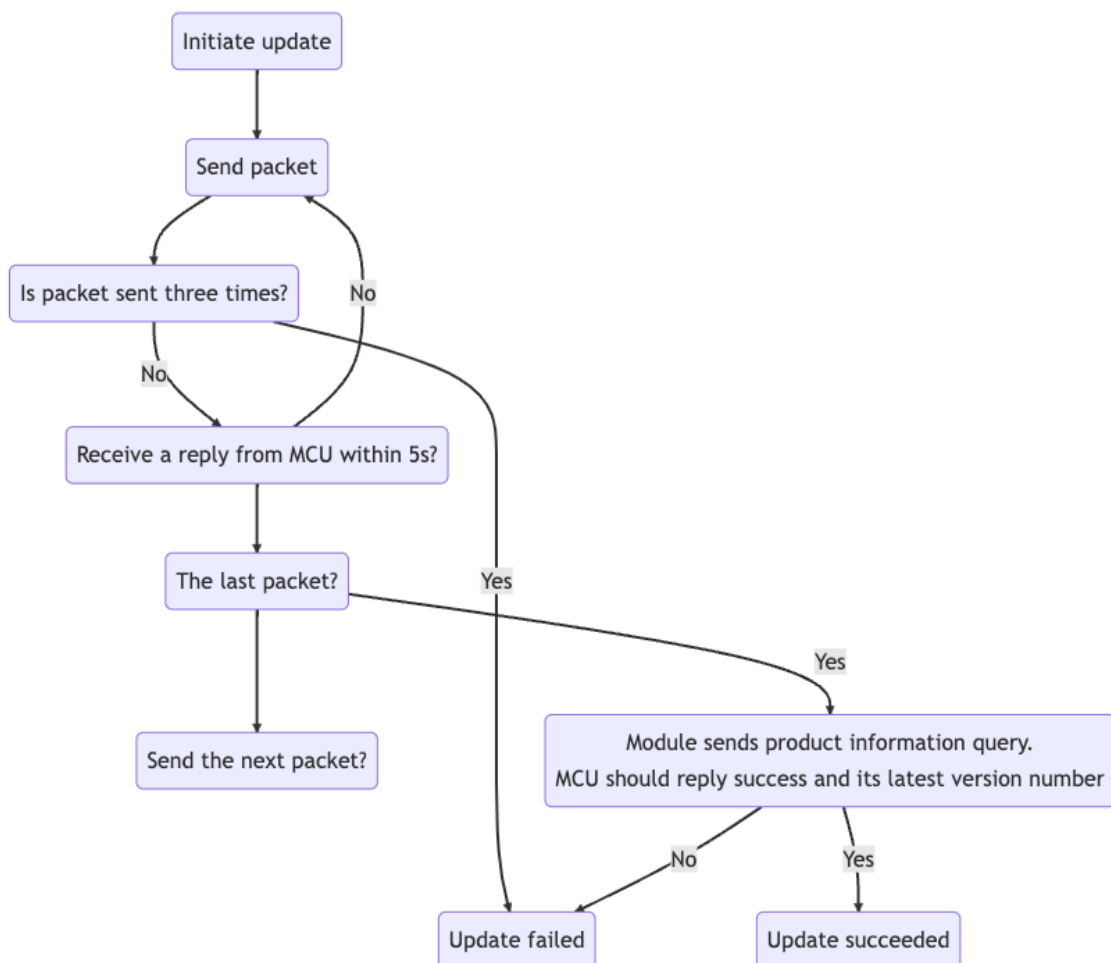
```

a("Initiate update")
b("Send packet")
c("Is packet sent three times?")
d("Receive a reply from MCU within 5s?")
e("The last packet?")
f("Send the next packet?")
g("Update failed")
h("Module sends product information query. MCU should reply success and its latest version number")
i("Update succeeded")
a --> b
b --> c
c --No--> d
c --Yes--> g
d --> e
e --> f
f --> b
e --Yes--> h
e --No--> g
  
```

```

h --Yes--\> i
d --No--\> b
graph TD
a("Initiate update")
b("Send packet")
c("Is packet sent three times?")
d("Receive a reply from MCU within 5s?")
e("The last packet?")
f("Send the next packet?")
g("Update failed")
h("Module sends product information query. MCU should reply success and its latest version number")
i("Update succeeded")
a --\> b --\> c --No--\> d --\> e --\> f
c --Yes--\> g
e --Yes--\> h --No--\> g
h --Yes--\> i
d --No--\> b

```



9.1. Initiate update (0x0a)

The OTA update can be initiated automatically or manually.

- For automatic updates, when the module detects MCU firmware updates from the cloud, the transmission of the update file automatically starts.
- For manual updates, the module initiates updates only when updates are confirmed on the app.

The module sends the following data.

- **u : protocol version 0** (0x00)

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0a
Data length	2	0x0004
Data	4	The size of the update file in bytes. The data type is an unsigned integer, and the data is transmitted 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, 0x55aa 00 0a 0004 00006800 75 indicates the size of the update file is 26624 bytes (26 KB).

- **u : protocol version 1** (0x01)

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x01
Command	1	0x0a
Data length	2	0x004F

Field	Byte(s)	Description
Data	40	4-byte update file. The data type is an unsigned integer, and the data is transmitted in big-endian format. 32-byte MD5 hash. 1-byte update channel. 3-byte version number. For example, version 01.2.04 is 010204 .
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0a
Data length	2	0x0001
Data	1	The options for the maximum size of each packet: 0x00 : 256 bytes. It is the default value and is compatible with legacy firmware. 0x01 : 512 bytes 0x02 : 1,024 bytes

Field	Byte(s)	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, 0x55aa 03 0a 0001 00 0d

9.2. Transmit update file (0x0b)

- Data format: the channel ID (for protocol version 01) + packet offset (unsigned short) + payload.
- When the MCU receives a frame with a data length equal to 4 bytes and the packet offset is equal to or greater than the size of the update, the transfer is completed.
- For the last packet, the packet offset is the firmware size, and the packet length is 0 bytes.

The module sends the following data.

- **u : protocol version 0 (0x00)**

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0b
Data length	2	The data length is the sum of 0x0004 and the packet length.
Data	4+N	The first four bytes are fixed as packet offset, and the latter bytes are the actual data.

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

• **U : protocol version 1 (0x01)**

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x01
Command	1	0x0b
Data length	2	The data length is the sum of 0x0005 and the packet length.
Data	5+N	• 1-byte update channel. • The first four bytes are fixed as packet offset. • The latter bytes are the actual data.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Assume that the size of the update is 530 bytes, and the MCU does not need to respond to the last packet.

- For the first packet, the packet offset is 0x00000000 , and the packet length is 256 bytes. The module sends 0x55aa 00 0b 0104 00000000 xx ... xx XX .
- For the second packet, the packet offset is 0x00000100 , and the packet length is 256 bytes. The module sends 0x55aa 00 0b 0104 00000100 xx ... xx XX .
- For the third packet, the packet offset is 0x00000200 , and the packet length is 18 bytes. The module sends 0x55aa 00 0b 0016 00000200 xx ... xx XX .

- For the last packet, the packet offset is 0x00000212 , and the packet length is 0 bytes. The module sends 0x55aa 00 0b 0004 00000212 xx ... xx XX .

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0b
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, 0x55aa 03 0b 0000 0d

9.3. Notify MCU of result (0x34 09)

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	2	0x09 (subcommand): one byte. 0x01 (status): one byte, indicating verification of the update failed.

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

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0001
Data	1	Subcommand: 0x09
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

10. Cellular services

10.1. Get system time in GMT (0x0c)

- As the international standard of civil time, GMT is independent of the time zone and DST.
- After connecting to the network, the module returns success and valid time data after the local timestamp is synced.

The MCU sends the following data.

Field	Byte(s)	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, 0x55aa 03 0c 0000 0e

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0c
Data length	2	0x0007

Field	Byte(s)	Description
Data	7	- Data[0]: indicates whether the time is obtained successfully. 0 : Failure. 1 : Success. - Data[1]: indicates the year. 0x00 indicates the year 2000. - Data[2]: indicates the month, ranging from 1 to 12. - Data[3]: indicates the day, ranging from 1 to 31. - Data[4]: indicates the hour, ranging from 0 to 23. - Data[5]: indicates the minute, ranging from 0 to 59. - Data[6]: indicates the second, ranging from 0 to 59.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

For example, 0x55aa 00 0c 0007 01 10 04 13 05 06 07 4c indicates the time 05:06:07 on April 19, 2016 in GMT.

10.2. Test module functionality (0x0e)

- Insert the SIM card into the module to test whether the card can be detected.
- Test whether the module is authorized.
- Test whether the RF is calibrated.
- Get the 4G signal strength. A SIM card must be inserted. Signal strength ranges from 0 to 31. You can set a quality standard as needed.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0e
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.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x0e
Data length	2	0x0004

Field	Byte(s)	Description
Data	4	- Data[0]: 0 indicates the SIM card is not detected. 1 indicates the SIM card is detected. - Data[1]: 0 indicates the module is not authorized. 1 indicates the module has been authorized. - Data[2]: 0 indicates RF is not calibrated. 1 indicates RF has been calibrated. - Data[3]: Signal strength, ranging from 0 to 31.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

10.3. Get local time (0x1c)

- The local time is calculated by adding the time zone offset and DST to the GMT. The time zone is where the device is activated.
- After connecting to the network, the module returns success and valid time data after the local timestamp is synced.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa

Field	Byte(s)	Description
Version	1	0x03
Command	1	0x1c
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.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x1c
Data length	2	0x0008

Field	Byte(s)	Description
Data	8	- Data[0]: indicates whether the time is obtained successfully. 0 : Failure. 1 : Success. - Data[1]: indicates the year. 0x00 indicates the year 2000. - Data[2]: indicates the month, ranging from 1 to 12. - Data[3]: indicates the day, ranging from 1 to 31. - Data[4]: indicates the hour, ranging from 0 to 23. - Data[5]: indicates the minute, ranging from 0 to 59. - Data[6]: indicates the second, ranging from 0 to 59. - Data[7]: indicates the week, ranging from 1 to 7. 1 indicates Monday.

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

Example:

- If the device is activated in mainland China, the local time is Beijing time (GMT+08:00). For example, the module returns 0x55aa 00 1c 0008 01 10 04 13 05 06 07 02 5f to indicate Beijing time at 05:06:07 on April 19, 2016.
- If the device is activated in other countries or regions, the local time is the time zone in which the device is located.

10.4. Get module's memory (0x0f)

This command can be used to get the module's remaining memory. If the module returns -1, getting the module's remaining memory is not supported.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x0f
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.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00

Command	1	0x0f
Field	Byte(s)	Description
Data length	2	0x0004
Data	4	The value is represented in big-endian format. For example, 0x00 0x00 0x28 0x00 represents 10240 bytes remaining memory.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

10.5. Get Unix timestamp (with time zone) (0x1b)

- The MCU can get a Unix timestamp, time zone, and daylight saving time (DST).
- This command only applies to door locks with the keep-alive feature.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x1B
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.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00

Command	1	0x1B
Field	Byte(s)	Description
Data length	2	N
		- Data[0]: indicates whether the time is obtained successfully. 0 : Failure. 1 : Success. - Data[1] to Data[4]: Unix timestamp. - Data[5]: indicates whether the time zone is obtained successfully. 0 : Failure. 1 : Success. - Data[6]: 0 indicates local time ahead of GMT. 1 indicates local time behind GMT. - Data[7]: time zone. - Data[8]: indicates whether DST is in use in a time zone. 1 : in use. 0 : not in use. - Data[9] to Data[12]: indicates timestamp when DST starts, in big-endian format. - Data[13] to Data[16]: indicates timestamp when DST ends, in big-endian format.
Data	N	

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

i

Currently, getting the DST is not supported.

10.6. (Optional) Get signal strength (0x24)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x24
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.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x24
Data length	2	0x0001

Field	Byte(s)	Description
Data	1	The RSSI value ranges from 0 to 31. The larger the value, the better the signal quality. 0 indicates -113 dBm or less. 31 indicates -51 dBm or above. If the return value is not within the range, the value is invalid and the operation fails.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

10.7. (Optional) Notify module to disable heartbeat (0x25)

The MCU sends the following data.

- Version 03:

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x25
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.

- Version 04:

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x04
Command	1	0x25
Data length	2	0x0001
Data	1	• 0 : Disable. • 1 : Enable.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x25
Data length	2	0x0000 or 0x0001
Data	0 or 1	• 0x00 : Failure. • 0x01 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Before the module goes to sleep for reducing power consumption, the MCU can send this command to notify the module to disable the heartbeat. Because heartbeats are required for building communication between the module and the MCU, this command must not be sent when the device is just powered on.

10.8. Get network status (0x2b)

Network status	Description	Status value
----------------	-------------	--------------

Network status	Description	Status value
Status 1	The SIM card is not connected.	0x00
Status 2	The module searches for cellular networks.	0x01
Status 3	The module is registered with the cellular network but not connected to the network.	0x02
Status 4	The module is connected to the network and gets an IP address.	0x03
Status 5	The module is connected to the cloud.	0x04
Status 6	SIM card registration is denied.	0x05



The status definition must be consistent with that in [Report network status](#).

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2b
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.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2b
Data length	2	0x0001
Data	1	See network status .
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

10.9. Get module's MAC address (0x2d)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x2d
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.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x2d
Data length	2	0x0007

Field	Byte(s)	Description
Data	7	- Data[0]: indicates whether the MAC address is obtained successfully. 0x00 : Success. The next 6 bytes denote a valid MAC address. 0x01 : Failure. The next 6 bytes denote an invalid MAC address. - Data[1] to Data[6]: indicates the valid MAC address of the module on a success.

10.10. Set cellular mode (0x05)

The module can work in fully operational mode or airplane mode.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x05
Data length	2	0x0001
Data	1	1 : Fully-operational mode. 4 : Airplane 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, 0x55aa 03 04 0000 06

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x05
Data length	2	0x0001
Data	1	• 0 : Success. • 1 : Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 0x55aa 00 04 0000 03

10.11. Get cellular mode (0x71 01)

The module can work in fully operational mode or airplane mode.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x01
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 71 00 01 01 75

The module returns the following data.

Field	Byte(s)	Description
-------	---------	-------------

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002
Data	2	• Subcommand: 0x01 • Data: • 1 : Fully-operational mode. • 4 : Airplane mode.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

10.12. Get IMSI (0x71 02)

The MCU can send this command to the module to get the international mobile subscriber identity (IMSI) of the module.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x02
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
-------	---------	-------------

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0010
Data	16	• Subcommand: 0x02 • For example, a 15-digit IMSI code is 460113012467340 .
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

55 AA 00 71 00 10 02 34 36 30 31 31 33 30 31 32 34 36 37 33 34 30 7C

10.13. Get ICCID number (0x71 03)

The MCU can send this command to the module to get the integrated circuit card identifier (ICCID). The module returns a 20-digit ICCID number to the MCU.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x03
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0015
Data	21	• Subcommand: 0x03 • For example, a 20-digit ICCID code is 89861118249000363490 .
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

55 AA 00 71 00 15 03 38 39 38 36 31 31 31 38 32 34 39 30 30 30 33 36 33 34 39 30 9A

10.14. Get IMEI (0x71 04)

The MCU can send this command to request the 15-digit international mobile equipment identity (IMEI) of the module.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x04
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0010
Data	16	- Subcommand: 0x04 - For example, a 15-digit IMEI number is 864237040014733 .
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

55 AA 00 71 00 10 04 38 36 34 32 33 37 30 34 30 30 31 34 37 33 33 88

10.15. Get cellular network type (0x71 43)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x43
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa

Version	1	0x00
Command	1	0x71
Data length	2	0x0002
		• Subcommand: 0x43 • Type: • 1 : GPRS • 2 : LTE Cat.1 • 4 : CATM • 5 : NB-IoT network
Data	2	error code
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

10.16. Query the current PLMN (0x71 48)

- This command applies to the MA510 module only and is available when `cops` is specified by the MCU in the product information.
- The module can return the PLMN only after it successfully registers to the network.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	3	Data[0] Subcommand: 0x48

Field	Bytes	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002 + N × PLMN
Data	3	- Subcommand: 0x48 - Data[1]: 1 : Success. 0 : Failure. - PLMN: The PLMN of the country where the device is currently registered. A PLMN value is 4 bytes. An array of PLMN in big endian is returned. Current PLMN + PLMN1 ... + PLMNn
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

- 55 AA 03 71 00 01 48 BC
- 55 AA 00 71 00 12 48 01 00 04 BA FC 00 04 BA FC 00 04 BC 8A 00 04 BB F4 3C

10.17. (Optional) Set or get radio resource control (RRC) (0x71 4A)

The MCU integration protocol must be v1.1.1 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002/0x0003
		• Subcommand: 0x4A • Command (1 byte): • 0 : Get • 1 : Set • For the set command, the 1-byte RRC value ranges from 0 to 20, corresponding to 0 to 10 seconds.
Data	0x0002/0x0003	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003/0x0004

Field	Bytes	Description
Data	0x0003/0x0004	- Subcommand: 0x4A - Command (1 byte): 0 : Get 1 : Set - For the set command, the 1-byte RRC value ranges from 0 to 20, corresponding to 0 to 10 seconds. - For the get command: - The 1-byte RRC value ranges from 0 to 20, corresponding to 0 to 10 seconds.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

11. Password and weather services

11.1. Offline dynamic password (0x16)

Offline dynamic passwords can be used to unlock the door if a device is disconnected for long periods of time.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x16
Data length	2	N
Data	7+N	Year(1) + mon(1) + day(1) + hour(1) + min(1) + sec(1) + code_len(1) + code(n), indicating GMT + password length + password
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

Suppose that the offline password is 1849455172 and GMT is 08:18:42 on January 11, 2021, the MCU sends 55 AA 00 16 00 11 15 01 0B 08 12 2A 0A 01 08 04 09 04 05 05 01 07 02 C3 .

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x16

Field	Byte(s)	Description
Data length	2	Result(1) + type(1) + decode_len(1) + decode(n) Result: 0 : correct password. Other values: error, no data returned.
Data	3+N	- type : 0 : timed password. 1 : one-time password. 2 : clear password. - decode_len : data length after decoding. - decode : decoded data.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

11.2. Request temporary password (with schedule list) (0x14)

- To maximize power saving, when detecting multiple temporary passwords, the MCU then calls this interface to get password data for updating locally stored data.



The module sends the MCU time data in GMT, so the MCU should sync its clock with the module through Getting GMT.



- The server sends full temporary passwords to the device. The MCU updates locally stored data accordingly.
- Users can set a weekly schedule based on the current validity period of a temporary password. Up to three schedules can be added.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x14
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.

The door lock gets the currently valid temporary password:

55 aa 00 14 00 00 13

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x14
Data length	2	0x0001+N (temporary password)

Field	Byte(s)	Description
		- Data[0]: indicates whether the password is obtained successfully. 0 indicates failure with a data length of one byte, and the data of password and valid time will not be transmitted in serial communication. 1 indicates success. - Data[1]: indicates the number of passwords, ranging from 0 to 10. ** 0 indicates no password is created on the app. The data of password and valid time will not be transmitted in serial communication. The number of passwords indicates the password data contained in the current packet.** - Data[2]: indicates password length. The length of each password is consistent. - Data[3]: Bit 7 indicates whether there is a
		subsequent packet. Bit 6 to bit 0: indicates the sequence number of

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

11.3. Password positional notation (0x17)

The positional notation uses a set of consecutive digits to represent any real number. The positional notation adopted by the password is determined by two factors:

- **The base:** indicates the number of unique digits that a positional notation system uses to represent numbers. Base-4 through base-10 are supported.
- **The starting value:** indicates the number starts with zero or one.

Number of digits	Password starts with 0 or 1	Dynamic password	Offline password	The min length of online password	Security level	Multilingual online password
4	Configurable	Not supported	Not supported	8	1/29W	Password of 8 to 12 digits
5	Configurable	Support	Not supported	8	1/39W	Password of 8 to 12 digits
6	Configurable	Support	Not supported	8	1/100W	Password of 8 to 12 digits
7	Configurable	Support	Support	8	1/100W	Password of 8 to 11 digits

Number of digits	Password starts with 0 or 1	Dynamic password	Offline password	The min length of online password	Security level	Multilingual online password
8	Configurable	Support	Support	7	1/100W	Password of 7 to 11 digits
9	Configurable	Support	Support	7	1/100W	Password of 7 to 10 digits
10	Not configurable	Support	Support	7	1/100W	Password of 7 digits

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x17
Data length	2	0x0002
Data	2	Data[0]: indicates the base number. Data[1]: indicates the starting value.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

- To adopt a set of **consecutive digits 1 through 5**, the base number is set to 5 and the starting value is set to 1 .
- To adopt a set of **consecutive digits 0 through 5**, the base number is set to 6 and the starting value is set to 0 .



You must set the positional notation after the product information query (0x01) and before any password-related operation.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x17
Data length	2	0x0001
Data	1	0 : Success. Other values: Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

11.3.1. Request temporary password (with schedule list) (0x14)

- To maximize power saving, when detecting multiple temporary passwords, the MCU then calls this interface to get password data for updating locally stored data.
- The module sends the MCU time data in GMT, so the MCU should sync its clock with the module through Getting GMT.
- The server sends full temporary passwords to the device. The MCU updates locally stored data accordingly.
- Users can set a weekly schedule based on the current validity period of a temporary password. Up to three schedules can be added.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03



Field	Byte(s)	Description
Command	1	0x14
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.

The door lock gets the currently valid temporary password:

55 aa 00 14 00 00 13

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x14
Data length	2	0x0001+N (temporary password)

Field	Byte(s)	Description
		• Data[0]: indicates whether the password is obtained successfully. 0 indicates failure with a data length of one byte, and the data of password and valid time will not be transmitted in serial communication. 1 indicates success. • Data[1]: indicates the number of passwords, ranging from 0 to 10. 0 indicates no password is created on the app. The data of password and valid time will not be transmitted in serial communication. The number of passwords indicates the password data contained in the current packet. • Data[2]: Bit 7 indicates whether there is a subsequent packet. Bit 6 to bit 0: indicates the sequence number of the current packet, starting from 0. The data of the first password, including

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

11.4. (Optional) Enable weather service (0x20)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x20
Data length	2	$N((L+K)+(L+K)...)N$
Data	N	- L : a 1-byte value, representing the length of K . - K : parameter name.For example: - L : 0x06 K : w.temp - L : 0x06 K : w.pm25 - L : 0x0a K : w.humidity
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00

Field	Byte(s)	Description
Command	1	0x20
Data length	2	0x0002
Data	2	- Byte 1: 0x00 : Failure. 0x01 : Invalid data format. 0x02 : Exception. - Byte 2: 0x01 : Success. 0x00 : Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

11.5. (Optional) Send weather data (0x21)

After the weather service is enabled, the module regularly sends weather data received from the cloud to the MCU.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x21
Data length	2	N((LKTLV)(LKTLV)...)

Field	Byte(s)	Description
Data	N	• 0x00 : Failure. • 0x01 : Access denied. Check whether you have subscribed to the weather service. • 0x01 : Success. • L : length of the parameter name. • K : parameter name. • T : 0x00 indicates an integer and 0x01 indicates a string. • L : length of the field name. • v : value of the field.
		Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
Checksum	1	

The MCU returns the following data.

Field	Byte(s)	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.

i

If the request parameter is `w.temp,w.pm25` , only `w.temp` returns a value, you need to check whether the request parameter name is correct.

12. Bluetooth LE and LTE Cat.1 combo dedicated protocol

12.1. Product authorization

The Bluetooth LE chip stores authorization information. When connected to an LTE Cat.1 module, the Bluetooth LE module transmits activation and authorization data to the Cat.1 module via UART communication. The LTE Cat.1 module is not allowed to proactively bind with and unbind from the cloud. When the Bluetooth LE module has been unbound, it should notify the LTE Cat.1 module of its unbinding.

This block diagram illustrates the interaction between the Bluetooth LE module and the LTE Cat.1 module.

```

1 sequenceDiagram
2 participant LTE Cat.1
3 participant Bluetooth LE
4 LTE Cat.1->>Bluetooth LE: 1. Send heartbeat packet
5 Bluetooth LE-->>LTE Cat.1: 2. Respond to heartbeat packet
6 LTE Cat.1->>Bluetooth LE: 3. Query product information
7 Bluetooth LE-->>LTE Cat.1: 4. Respond with product information
8 LTE Cat.1->>LTE Cat.1: 5. Save Bluetooth LE<br>version number<br>Start PDP
9 activation<br>and cloud service
10 LTE Cat.1->>Bluetooth LE: 6. Query working mode
11 Bluetooth LE-->>LTE Cat.1: 7. Respond with working mode
12 Bluetooth LE->>LTE Cat.1: 8. Query LTE Cat.1 unique ID
13 LTE Cat.1->>Bluetooth LE: 9. Return LTE Cat.1 IMEI, device type<br>and binding
14 status
15 Bluetooth LE-->>LTE Cat.1: 10. Get the key
16 LTE Cat.1->>LTE Cat.1: 11. Generate public key and shared key.<br>Save shared
17 key.
18 LTE Cat.1->>Bluetooth LE: 12. Respond with LTE Cat.1 public key
19 Bluetooth LE-->>LTE Cat.1: 13. Sync authorization information
20 LTE Cat.1->>LTE Cat.1: 14. Get uuid, auth_key, and (optional) psk_key<br>Call
21 tuya_iot_set_ext_auth_info to set uuid<br>auth_key, and psk_key
22 LTE Cat.1->>Bluetooth LE: 15. Return authorization information<br>and the 32-
23 byte key that is encrypted with <br>16-byte random number and shared key using
24 HMAC-SHA256.
25 Bluetooth LE-->>LTE Cat.1: 16. Sync activation information
26 LTE Cat.1->>LTE Cat.1: 17. Start a 3s timer on<br>receiving activation
27 information
28 LTE Cat.1->>Bluetooth LE: 18. Return activation status 0
29 LTE Cat.1->>LTE Cat.1: 19. The timer calls <br>tuya_iot_set_ext_active_info
30 LTE Cat.1->>Bluetooth LE: 20. Return activation status 4 and save the activation
31 flag

```

12.2. Query module's unique identifier (0xD0 00)

The Bluetooth LE module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0xD0
Data length	2	0x0001
Data	1	Subcommand: 0x00
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0xD0
Data length	2	N
Data	1	Subcommand: 0x00
	1	- Result: 0 : Success. - Other values: Failure. - The payload is invalid if the operation fails.
	1	The binding status of the module. 0 : Unbound. 1 : Bound.

Field	Byte(s)	Description
	1	The module type: 1 : NB-IoT 2 : Wi-Fi 3 : LTE Cat.1 4 : Zigbee 5 : Bluetooth LE Only 2 and 3 are currently used.
	N	XXX (The unique identifier of the module)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

12.3. Sync authorization information (0xD0 01)

The Bluetooth LE module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0xD0
Data length	2	N

Field	Byte(s)	Description
Data	N	- Subcommand: 0x01 - The encryption method. 0x00 : AES-CBC 0x01 : AES-CCM 0x02 : Other - Random number: 16 bytes <pre>{“uuid”:“xxxx”,“auth_key”:“xxxx”,“psk_key”:“xxxx”}</pre> uuid , auth_key , and psk_key
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0xD0
Data length	2	0x0002

Field	Byte(s)	Description
Data	35	- Subcommand: 0x01 - Result: 0 : Success. Other values: Failure. - The encryption method. 0x00 : AES-CBC 0x01 : AES-CCM 0x02 : Other - Use HMAC-SHA256 to encrypt the key with both the random number and shared key. key_len is 32 bytes.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

12.4. Sync activation information (0xD0 02)

The Bluetooth LE module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0xD0
Data length	2	N

Field	Byte(s)	Description
		Subcommand: 0x02
		<pre>{“devId”:“xxxx”,“localKey”:“xxxx”,“secKey”:“xxxx”,“env”:“xxxx”,“r</pre>
Data	1+N	The data is encrypted with AES. The algorithm depends on the field of the encryption method. - Key: the first 16 bytes of the shared key. - iv : all zeros. - Padding: The data must be padded to ensure that its total length is an integral multiple of 16.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0xD0
Data length	2	1
Data	1	Subcommand: 0x02

Field	Byte(s)	Description
	Result	0 : indicates the activation information is received and activation is in progress. After successful activation, the device reports status . Other values indicate failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

12.5. Get key (0xD0 03)

The Bluetooth LE module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0xD0
Data length	2	0x42 (66)
		• Subcommand: 0x03 • Curves for elliptic curve cryptography (ECC).
Data	66	• 1 : secp192r1 • 2 : secp224r1 • 3 : secp256r1 • 4 : secp256k1 • The public key of the MCU.

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

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0xD0
Data length	2	0x42 (66)
Data	66	• Subcommand: 0x03 • Curves for elliptic curve cryptography (ECC). • 1 : secp192r1 • 2 : secp224r1 • 3 : secp256r1 • 4 : secp256k1 • The public key of the MCU.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13. Positioning services

13.1. (Optional) Enable or disable GNSS positioning (0x72 81)

The MCU sends this command to the module to enable or disable the GNSS positioning function. Because the GNSS module performs many operations, the maximum timeout period for a response is three seconds.



The values of positioning modes are valid only when the GNSS positioning is enabled.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	3	0x0003

Field	Byte(s)	Description
Data	3	- Subcommand: 0x81 - Control command: 0x01 : Enable. 0x00 : Disable. - Positioning mode (applies when positioning is enabled): 0x00 : Global Positioning System (GPS) plus BeiDou Navigation Satellite System (BDS) 0x01 : GPS plus GLONASS (depending on the used GNSS modules) 0x02 : GPS 0x03 : BDS (depending on the used GNSS modules) 0x04 : GLONASS (depending on the used GNSS modules) 0x05 : Galileo satellite navigation (depending on the used GNSS modules)

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

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002/0x0003
		Subcommand: 0x81.
		The result of enabling or disabling positioning:
		0 : Failure. 1 : Success. - Status of the
Data	3	positioning mode, which is invalid when positioning is disabled. Some GPS modules do not support this feature, and the return value can be ignored.
		0 : Failure. 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.



The **LZ211 module** supports GPS plus BDS only. Modification is not allowed.

13.2. (Optional) Report GNSS information automatically (0x72 90)

The MCU sends this command to the module to enable or disable the GNSS positioning function. Because the GNSS module performs many operations, the maximum timeout period for a response is three seconds. You must enable GNSS positioning before using this feature.



The values of positioning modes are valid only when the GNSS positioning is enabled.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0004
Data	4	• Subcommand: 0x90 • The reporting period: 2-byte data in seconds, transmitted in big-endian format. • DP ID: 1-byte data.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	3	• Subcommand: 0x90 • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

i

- If the reporting period is 0 , it indicates automatic reporting is canceled.
- The reporting period must be greater than five seconds.
- The data that is automatically reported is transmitted over the cellular network.
- Only DPs of string type are supported.

13.3. (Optional) Enable or disable Wi-Fi positioning (0x72 82)

The MCU sends the DP ID and the reporting period to the module. The module automatically reports the positioning information to the cloud accordingly. When Wi-Fi positioning is enabled, Bluetooth will be disabled.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72

Field	Byte(s)	Description
Data length	2	0x0002
Data	2	- Subcommand: 0x82 - Control commands: 0x01 : Enable. 0x00 : Disable.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	- Subcommand: 0x82 - Status: 0 : Failure. 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.4. (Optional) Report Wi-Fi information automatically (0x72 91)

The MCU sends the DP ID and the reporting period to the module. The module automatically reports the positioning information to the cloud accordingly. You must enable Wi-Fi positioning before using this feature.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0004
Data	4	Subcommand: 0x91 The reporting period: 2-byte data in seconds, transmitted in big-endian format. The DP ID: 1-byte data.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x91 • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

i

- If the reporting period is 0, it indicates automatic reporting is canceled.
- The reporting period must be greater than five seconds.
- The data that is automatically reported is transmitted over the cellular network.
- Only DPs of string type are supported.

13.5. (Optional) Report LBS information automatically (0x72 92)

The MCU sends the DP ID and the reporting period to the module. The module automatically reports the positioning information to the cloud accordingly.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0004
Data	4	• Subcommand: 0x92 • The reporting period: 2-byte data in seconds, transmitted in big-endian format. • DP ID: 1-byte data.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa

Field	Byte(s)	Description
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x92 • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

- If the reporting period is 0, it indicates automatic reporting is canceled. The reporting period must be greater than five seconds. The data that is automatically reported is transmitted over the cellular network. Only DPs of string type are supported.
- We recommend you choose only one of the positioning modes from Wi-Fi, GNSS, or LBS for automatic positioning reporting. Otherwise, there will be a deviation of one to two seconds between the actual reporting period and the preset period. If the module fails to receive positioning information as scheduled, it will pause the timer and retry after one or two seconds. The timer will reset once the information is obtained.
- It takes about one second to get the LBS positioning.

13.6. (Optional) Reset GNSS module (0x72 83)

- The MCU sends this command to the module to reset the GNSS module.
- The LZ201 module supports this feature.
- The LZ211 module does not support this feature.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa

Field	Byte(s)	Description
Version	1	0x03
Command	1	0x72
Data length	2	0x0003
Data	3	• Subcommand: 0x83 • GPIO reset pin: a value from 0 to 31, according to the actual hardware connection. • The voltage level on the reset pin: • 0 : low level. • 1 : high level.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x83 • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.



The LZ211 module includes a GNSS device that is reset every time it is powered on.

13.7. Get GNSS positioning (longitude, latitude) (0x71 10)

The MCU sends this command to the module to get GNSS positioning information.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x10
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002+N

Field	Byte(s)	Description
		- Subcommand: 0x10 - Status: 0 : Failure. 1 : Success. - The returned N-byte positioning information string takes the format longitude,latitude .
Data	2+N	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns information in string type, so the MCU gets the information and reports it back to the cloud through [status reporting](#) . If the cloud requires DP data in latitude and longitude format, the MCU must convert the format of the received positioning information.

13.8. Get GNSS positioning (latitude,longitude) (0x71 29)

The MCU sends this command to the module to get GNSS positioning information.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x29

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

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002+N
		- Subcommand: 0x29 - Status: 0 : Failure. 1 : Success. - The returned N-byte positioning information string takes the format longitude,latitude .
Data	2+N	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns information in string type, so the MCU gets the information and reports it back to the cloud through [status reporting](#) .

13.9. (Optional) Get GNSS's SNR (0x71 11)

The MCU sends this command to the module to get the signal-to-noise ratio (SNR) of GNSS signals.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x11
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003
Data	3	• Subcommand: 0x11 • Status: • 0 : Failure. • 1 : Success. • The 1-byte SNR value ranges from 0 to 100.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.10. (Optional) Get GNSS positioning speed (0x71 12)

The MCU sends this command to the module to get GNSS instantaneous velocity.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x12
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0004
Data	4	• Data[0]: Subcommand 0x12 • Data[1]: status • 0 : Failure. • 1 : Success. • Data[2]: indicates the speed of the high data bit. • DATA[3]: indicates the speed of the low data bit. For example, Data[2] is SPEED>>8 , and Data[3] is SPEED&0xFF .The speed value has 2 bytes, in the unit of 100 m/h.

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

13.11. (Optional) Get Wi-Fi positioning information (0x71 20)

The MCU sends this command to the module to get Wi-Fi positioning information.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x20
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002+N

Field	Byte(s)	Description
Data	2+N	- Subcommand: 0x20 - The value that indicates the number of pieces of positioning information has one byte. - The N-byte positioning information string. For example, 18:56:80:2F:7C:36, -61;18:56:80:2F:7C:36, -61;1
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns information in string type, so the MCU gets the information and reports it back to the cloud through **status reporting** .

Up to 10 APs sorted by signal strength can be reported.

13.12. Get LBS positioning information (0x71 21)

The MCU sends this command to the module to get location-based service (LBS) positioning information.

- LBS positioning information takes the format: carrier code (mobile country code (MCC) + mobile network code (MNC)) + location area code + base station number + signal strength.
- For example, the LBS information for China Telecom is 46000,1816,875a521,-40 .



Typically, the base station number has two bytes (04ba). The base station number is designed to support four bytes, so when reporting LBS DP information, remove the padded 0000 from a value such as 04ba0000 .

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x21
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002+N
Data	2+N	• Subcommand: 0x21 • Status (one byte): • 0 : Failure. • 1 : Success. • N-byte LBS positioning information string, such as 460,00,6166,27556130,-75 .
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example:

```
55 AA 00 71 00 1A 21 01 34 36 30 2C 30 30 2C 36 31 36 36 2C 32 37 35 35 36 31 33 30
2C 2D 37 35 5F
```

13.13. Control LZ211 GNSS's main power supply (0x72 85)

The MCU sends this command to the LZ211 module to turn on or off the GNSS's power supply. The LZ211 module has a built-in GNSS chip. As long as the device is powered on, the backup power for GNSS works. The on/off control of GNSS's main power supply is supported. The chip performs a warm boot when the main power is on.

i

The GNSS chip takes about four seconds to load its firmware after being powered on. The GNSS positioning will restart when the main power is off and on again. For more information, see [\(Optional\) Enable or disable GNSS positioning](#).

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x85 • Data[1]: • 1 : Turn on the power supply. • 0 : Turn off the power supply.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa

Field	Byte(s)	Description
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x85 • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.14. Get LZ211 GNSS's power status (0x71 22)

The MCU sends this command to the LZ211 module to query GNSS's power status.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x22
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00

Field	Byte(s)	Description
Command	1	0x71
Data length	2	0x0002
Data	2	• Subcommand: 0x22 • Status: • 0 : Power is off. • 1 : Firmware is being loaded. • 2 : Power is on, and firmware loading is finished.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.15. AGNSS on/off (0x72 86)

- Turn on or off the AGNSS service. This service uses mobile data. AGNSS is turned off by default.
- AGNSS data is updated once every two hours.
- The MCU integration protocol must be v1.1.4 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0001
Data	2	• Subcommand: 0x86 • Data: Data[1] • 1 : On • 0 : Off

Field	Bytes	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x86 • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.16. (Optional) Get the number of visible GNSS satellites (0x71 15)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x15

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

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003
Data	3	• Subcommand: 0x15 • Status: • 0 : Failure. • 1 : Success. • The number of visible satellites (one byte)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.17. (Optional) Get the enablement of positioning features (0x71 30)

Get whether positioning via GNSS, Wi-Fi, or LBS is enabled or not.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71

Field	Byte(s)	Description
Data length	2	0x0001
Data	1	Subcommand: 0x30
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0004
Data	4	• Subcommand: 0x30 • Status: • GNSS positioning: • 0x01 : Enabled. • 0x00 : Disabled. • Wi-Fi positioning: • 0x01 : Enabled. • 0x00 : Disabled. • LBS positioning: • 0x01 : Enabled. • 0x00 : Disabled.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa

Field	Bytes	Description
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x13
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002/0x0004
Data	2/4	- Data[0]: Subcommand 0x13 - Data[1]: Status 0 : Failure. 1 : Success. - Heading (2 bytes): Its value is 10 times the actual value. No data is returned on failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.18. (Optional) Get GNSS heading (0x71 13)

13.19. (Optional) Get GNSS horizontal dilution of precision (HDOP) (0x71 16)

The MCU integration protocol must be v1.1.3 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x16
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002/0x0004
Data	2/4	• Data[0]: Subcommand 0x16 • Data[1]: Status • 0 : Failure. • 1 : Success. • HDOP (2 bytes): Its value is 10 times the actual value. No data is returned on failure.

Field	Bytes	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.20. (Optional) Get GNSS orthometric height (0x71 17)

The MCU integration protocol must be v1.1.3 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x17
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002/0x0004

Field	Bytes	Description
Data	2/4	- Data[0]: Subcommand 0x17 - Data[1]: Status 0 : Failure. 1 : Success. - Orthometric height (2 bytes): Its value is 10 times the actual value. No data is returned on failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.21. (Optional) Enable GNSS filter algorithm (0x72 18)

The MCU integration protocol must be v1.1.3 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0001
Data	1	Subcommand: 0x18
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
		Data[0]: Subcommand 0x18
Data	2	- Data[1]: Status 0 : Failure. 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.22. (Optional) Set GNSS filter algorithm (0x72 19)

The MCU integration protocol must be v1.1.3 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0003
		Subcommand: 0x19
Data	3	HDOP threshold (2 bytes): Its value is 10 times the actual value. The HDOP threshold ranges from 0 to 99.9, defaulting to 2.0.

Field	Bytes	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Data[0]: Subcommand 0x19 • Data[1]: Status • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

13.23. (Optional) Get GNSS positioning information (0x71 1A)

The MCU integration protocol must be v1.1.3 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x1A

Field	Bytes	Description
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002/0x0014

			• Data[0]: Subcommand 0x1A • Data[1]: Status • 0 : Failure. There is no following data. If latitude and longitude are returned, but other values fail, the result is 0 . • 1 : Success. • Longitude: 4-byte value in floating-point format. • Latitude: 4-byte value in floating-point format. • Signal strength: 1-byte value. • Number of satellites: 1-byte value. • Speed: 2-byte value, in the unit of 100 m/h. • Heading: 2-byte value, 10 times the actual value. • HDOP: 2-byte value, 10 times the actual value. • Orthometric height: 2-byte value, 10 times the actual value.
Data	2/20		

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

14. Phone call services

14.1. (Optional) Incoming call notification (0x71 23 00)

The MCU and the module communicate in a request-response manner. The MCU should respond to an incoming call within three seconds. Otherwise, the call will be hung up.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002+N (phone number)
Data	2+N	• Subcommand: 0x23 • Control command: 0x00 • N-byte phone number string
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 00 71 00 0D 23 00 31 35 38 39 30 39 38 36 37 36 37 F2

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002
Data	2	• Subcommand: 0x23 • Control command: 0x00

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

Example: 55 aa 03 71 00 02 23 00 98

14.2. (Optional) Outgoing calls (0x71 23 01)

The return value can only indicate whether the command of an outgoing call is sent successfully. Whether the call is received depends on the return value of a call status query.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002+N (phone number)
Data	2+N	- Subcommand: 0x23 - Control command: 0x01 - Phone number string
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 03 71 00 07 23 01 31 30 30 38 36 9D

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00

Command	1	0x71
Data length	2	0x0003
Data	3	- Subcommand: 0x23 - Control command: 0x01 - Result: 1 : Success. 0 : Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 00 71 00 03 23 01 01 98

14.3. (Optional) Answer calls (0x71 23 02)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002
Data	2	- Subcommand: 0x23 - Control command: 0x02
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 03 71 00 02 23 02 9A

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003
Data	3	• Subcommand: 0x23 • Control command: 0x02 • Result: • 1 : Success. • 0 : Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 00 71 00 03 23 02 00 9A

14.4. (Optional) Hang up calls (0x71 23 03)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002
Data	2	• Subcommand: 0x23 • Control command: 0x03
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 03 71 00 02 23 03 9B

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003
		• Subcommand: 0x23 • Control command: 0x03
Data	3	• Result: • 1 : Success. • 0 : Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 00 71 00 03 23 03 00 99

14.5. (Optional) Query call status (0x71 23 04)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002
Data	2	• Subcommand: 0x23 • Control command: 0x04

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

Example: 55 aa 03 71 00 02 23 04 9C

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003
Data	3	- Subcommand: 0x23 - Control command: 0x04 - Result: 0 : Dialing. 1 : Idle. 2 : Call failed. 3 : On a call.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 00 71 00 03 23 04 01 9B

14.6. (Optional) Report call status (0x71 23 04)

When the voice call status changes, the module proactively reports the current status to avoid additional memory usage caused by repeated queries.

The module sends the following data.

Field	Byte(s)	Description
-------	---------	-------------

Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003
- Subcommand: 0x23 - Control command: 0x04		
Data	3	- Result: 0 : Dialing. 1 : Idle. 2 : Call failed. 3 : On a call.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: 55 aa 00 71 00 03 23 04 01 9B

14.7. Enable/disable VoLTE (0x72 94)

Enable VoLTE to support voice calls over 4G. When VoLTE is disabled, voice calls fall back to the 2G network. If the SIM card supports voice calls, but the mobile operator does not support 2G, VoLTE must be enabled to support voice calls. If the mobile operator supports both 2G and 4G, you can choose whether to enable VoLTE. VoLTE setting is stored in non-volatile memory.



LZxxx-CN series modules support the 4G network only, so VoLTE is enabled by default. LZ211-EAU module supports both 2G and 4G. VoLTE is enabled by default. You can call this function to disable VoLTE as needed.

The MCU sends the following data.

Field	Byte(s)	Description
-------	---------	-------------

Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x94 • Control commands: • 0x01 : Enable. • 0x00 : Disable.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x94 • Status: • 0x00 : Failure. • 0x01 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

14.8. Send DTMF commands (0x72 95)

Dual-tone multi-frequency (DTMF) is used for sending numbers and issuing commands when dialing. When the user enters a command on the keypad during

a call, this function sends the signal tone to the voice channel. Note that calling this function when not on a call will fail.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
Data	N	• Data[0]: Subcommand 0x95 • Data[1]: The type of signal tones. • 0 DTMF 0 • 1 DTMF 1 • 2 DTMF 2 • 3 DTMF 3 • 4 DTMF 4 • 5 DTMF 5 • 6 DTMF 6 • 7 DTMF 7 • 8 DTMF 8 • 9 DTMF 9 • 10 DTMF A • 11 DTMF B • 12 DTMF C • 13 DTMF D • 14 DTMF # • 15 DTMF * • Data[2] to Data[5]: The duration of a signal tone in milliseconds, high byte first.

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	N	• Subcommand: 0x95 • Status: • 0x00 : Failure. • 0x01 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

14.9. Enable listening for DTMF (0x72 99)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x99 • Control commands: • 00 : Disable. • 01 : Enable.

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x99 • Result: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

14.10. Report DTMF events (0x71 95)

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002

Data	2	• Data[0]: Subcommand 0x95 • Data[1]: The type of signal tones. • 0 DTMF 0 • 1 DTMF 1 • 2 DTMF 2 • 3 DTMF 3 • 4 DTMF 4 • 5 DTMF 5 • 6 DTMF 6 • 7 DTMF 7 • 8 DTMF 8 • 9 DTMF 9 • 10 DTMF A • 11 DTMF B • 12 DTMF C • 13 DTMF D • 14 DTMF # • 15 DTMF *
		Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
Checksum	1	

14.11. (Optional) Set or query mic gain for calls (0x71 46)

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71

Data length	2	0x0002/0x0003
		- Data[0]: Subcommand 0x46 . - The query command: Data[1] 0: Query the mic gain. - The set command: Data[1] 1: Set the mic gain. Data[2]: The set value, ranging from 0 to 30. 0 means mute.
Data	2	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003/0x0004

		- Subcommand: 0x46 - The query command: Data[1]: 0: Query. Data[2]: 0: Failure. 1: Success. Data[3]: The set value, ranging from 0 to 30. 0 means mute. - The set command: Data[1]: 1: Set. Data[2]: 0: Failure. 1: Success.
Data	2	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

i

The setting is stored in the flash memory. During a call, the mic gain setting will be automatically applied to the mic driver. If the mic gain is set during a call, the value will be applied instantly.

If no mic gain is set, the module will respond to the query with failure.

Example:

- The query command: 55 aa 03 71 00 02 46 00 BB
- The set command: 55 aa 03 71 00 03 46 01 10 CD

14.12. (Optional) Set or query sidetone gain (0x71 47)

This command takes effect on a call.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa

Version	1	0x03
Command	1	0x71
Data length	2	0x0002/0x0003
Data	3	- Data[0]: Subcommand 0x47 . - The query command: Data[1] 0: Query the sidetone gain. - The set command: Data[1] 1: Set the sidetone gain. Data[2]: The set value, ranging from 0 to 100.
		Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
Checksum	1	

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0004/0x0003

		• Subcommand: 0x47 • Data[1]: Type selection: Data[1]: 0: Query. • Data[2]: 0: Failure. 1: Success. • Data[3]: The set value, ranging from 0 to 100. • The set command: Data[1] 1: Set. Data[2]: 0: Failure.1: Success.
Data	3	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

i

The setting is stored in the flash memory. During a call, the sidetone gain setting will be automatically applied to the driver. If the sidetone gain is set during a call, the value will be applied instantly.

If no sidetone gain is set, the module will respond to the query with failure.

Example:

- The query command: 55 aa 03 71 00 02 47 00 BC
- The set command: 55 aa 03 71 00 03 47 01 05 C3

15. SMS services

15.1. (Optional) Receive SMS messages (0x71 24 00)

The module sends the received SMS message to the MCU and deletes this message when it confirms that the MCU receives it. This way, there will be enough space for the module to receive a new message.

i

If not specified, the default format of SMS transmission and reception is JSON.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002+N

		- Subcommand: 0x24 - Control command: 0x00 - N-byte message content - Protocol version 0x03 takes the format <pre>{ "n": "10086", "t": "2021-3-29 16:58" "c": "xxxd fasfd" } .</pre> - Protocol version 0x04 takes the format:
Data	2+N	- The length of the phone number: 1 byte - The phone number: N byte(s) - The length of the date and time: 1 byte - The date and time: N byte(s) - The length of the message content: 2 bytes - The message content: N byte(s)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The MCU returns the following data.

Field	Byte(s)	Description
-------	---------	-------------

Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0003
Data	3	• Subcommand: 0x24 • Control command: 0x0 • Result: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The SMS message reported to the cloud is in JSON format, in which `n` represents the phone number, `t` represents the date and time, and `c` represents the message content. The encoding format of GB2312 is adopted.

15.2. (Optional) Send SMS messages (0x71 24 01)

If not specified, the default format of SMS transmission is JSON in protocol version 0x03 and hexadecimal in protocol version 0x04 .

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0002+N

Data	2+N	- Subcommand: 0x24 - Control command: 0x01 - Protocol version 0x03 takes the format: <pre>{ "n": "10086", "c": "xxxfasfd" }</pre> - Protocol version 0x04 takes the format: - The length of the phone number: 1 byte - The phone number: N byte(s) - The length of the message content: 2 bytes - The message content: N byte(s)
		Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
Checksum	1	

Example: Send a message with the content of **Hello** in Chinese.

55aa037100202401 7b226e223a223135383930393836373533222c2263223a22c4e3bac3227d A2

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003

		• Subcommand: 0x24 • Control command: 0x01 • Result: • 0 : Failure. • 1 : Success.
Data	3	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

Example: Send a message with the content of **Hello** in Chinese.

```
55aa00710003240101 99
```

15.3. (Optional) Set SMS encoding (0x71 24 02)

Set the encoding of SMS transmission and reception. LZ211-EAU uses UTF-8 as its default encoding, while the LZxxx-CN series modules use CP936. The setting is stored in the volatile memory.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0003

		• Subcommand: 0x24 • Control command: 0x02 • Encoding options: • 0x00 : ISO8859-1 (Chinese characters are not supported) • 0x01 : UTF-8 • 0x02 : UTF-16BE • 0x03 : UTF-16LE • 0x04 : GSM • 0x05 : CP936
Data	3	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003
Data	3	• Subcommand: 0x24 • Control command: 0x02 • Result: • 0 : Failure. • 1 : Success.

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

15.4. Set SMS transmission and reception format (0x71 24 03)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0003
Data	3	• Subcommand: 0x24 • Control command: 0x03 • Encoding options: • 0x00 : JSON • 0x01 : Hexadecimal
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003

		• Subcommand: 0x24 • Control command: 0x03 • Result: • 0 : Failure. • 1 : Success.
Data	3	

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

16. Battery services

16.1. (Optional) Query battery level (0x71 25)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x25
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002
Data	2	• Subcommand: 0x25 • Battery level: 0 to 100
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

16.2. (Optional) Query battery charging status (0x71 26)

The MCU sends the following data.

Field	Byte(s)	Description
-------	---------	-------------

Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x26
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002

		• Subcommand: 0x26 • Charging state: • 0 : Initial state. When the device starts up, no charging state is detected. • 1 : Start charging. • 2 : Charging is finished. • 3 : Battery is low. • 4 : Battery is extremely low. • 5 : Battery is unplugged. • 6 : Charger is unplugged. • 7 : Charging fault occurs.
Data	2	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

17. Audio services

17.1. (Optional) Set volume (0x72 87)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0003
Data	3	• Subcommand: 0x87 • Type (1 byte): • 1 : System volume. • 2 : Voice call volume. • Volume: 0 to 100. 0 means mute.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0003
Data	3	• Subcommand: 0x87 • Type (1 byte): • 1 : System volume. • 2 : Voice call volume. • Result: • 0 : Success. • Other values: Failure.

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

17.2. (Optional) Audio playback from SD card (0x71 27)

An SD card support is required to implement this feature.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0004+N

		• Subcommand: 0x27 • Interface (1 byte): • 0 : Play on the local speaker. • 1 : Play on the voice call uplink. • Command (1 byte): • 0 : Stop. • 1 : Play. • 2 : Pause. • 3 : Resume. • Audio format: • 1 : PCM • 2 : WAVPCM • 3 : MP3 • 4 : AMR-NB • 5 : AMR-WB • Audio file path (N byte): For example, /sdcard0/ice_8k_16bit.pcm
Data	4+N	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0005

		• Subcommand: 0x27 • Interface (1 byte) • Control command (1 byte)
Data	5	• Audio format • Result: • 0 : Success. • Other values: Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

17.3. (Optional) Query local audio playback status (0x71 28)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	0x0001
Data	1	Subcommand: 0x28
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0003

		• Subcommand: 0x28 • Status • 0 : Idle. • 1 : Playing. • 2 : Paused. • 3 : Finished. • Result: • 0 : Success. • Other values: Failure.
Data	3	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

17.4. (Optional) Report local music playback finished (0x71 2a)

The module reports the completion status when local music playback is finished.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002
Data	2	• Subcommand: 0x2A • Status: 1 , indicating playback is finished.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

17.5. (Optional) Set the mute state (0x72 97)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0003
Data	3	• Data[0] Subcommand 0x97 • Data[1]: The mute options. • 0 : Mute SMS messages. • 1 : Mute incoming calls. • 2 : Mute active calls. • Data[2]: The set value. • 0 : Unmute. • 1 : Mute.
		Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
Checksum	1	

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0003

		• Subcommand: 0x97 • Mute mode: • 0 : Mute SMS messages. • 1 : Mute incoming calls. • Status: • 0x00 : Failure. • 0x01 : Success.
Data	3	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

17.6. (Optional) Online TTS service (0x71 & 0x45)

- To enable online TTS, you need to subscribe to Tuya's online TTS service for your product.
- The product must get connected to the internet before it can access the online TTS service.
- The MCU must send UTF-8 text. Other formats are not supported.
- This online service uses internet data, and latency may occur if the network is poor.
- A longer text results in a larger synthesized audio received from the cloud. It is recommended that a single text should not exceed 64 characters.
- The service operates asynchronously. If the device fails to receive audio data from the cloud within 10 seconds, the request will be terminated.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71

Data length	2	0x0002
Data	2	- Subcommand: 0x45 {“opt”:“play”,“timeout”:xxx,“context”:“xxx”}
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

opt : indicates the mode. Valid values: play and query .

- play : The cellular module plays the audio directly after receiving it.
- timeout : The timeout for receiving the audio from the cloud. The audio playback time is excluded. The timeout defaults to 10s and must be at least 5 seconds.
- query : Query the current status of TTS playback. When query is selected, context and timeout fields can be left empty.

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	N
Data	3	- Subcommand: 0x45 - Data[1]: result
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

result

result	Meaning
-7	The TTS server is not connected.
-6	Incorrect parameter.
-5	Text encoding error.

-4	Network error.
-3	Device is busy.
-2	Failure.
-1	Timeout.
0	Success.
1	TTS service is idle.
2	TTS information is obtained.
3	TTS audio is being requested.
4	TTS audio is downloading.
5	TTS audio is playing.
6	Failed to request TTS audio.
7	Failed to download TTS audio.
8	TTS request timed out.
9	TTS audio finishes playing.

17.7. (Optional) Audio playback from memory (0x71 49)

- Transfer and control audio data stored in the memory. The module supports up to four audio IDs, with a maximum of 16 KB per audio ID.
- The audio data is stored in the memory of the module, which gets lost when the device is restarted. Therefore, the MCU should transfer the memory data to the module again.
- To add audio data, the length of an audio data packet should not exceed 1024 bytes.

The MCU sends the following data.

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x71
Data length	2	00009+N : Add audio data. 0004 : Play. 0003 : Delete or stop. 0002 : Query.

		• Subcommand: 0x49 • Control command (1 byte) • 1 : Add audio data. • 2 : Delete an audio ID. • 3 : Play an audio ID. • 4 : Pause an audio ID. • 5 : Query the audio IDs stored in the module's memory. • Audio ID (1 byte): applies to the control commands 1, 2, 3, and 4. • Port (1 byte): applies to the control command 3. • 0 : Play audio on a speaker. • 1 : Play audio on the voice call's uplink. • Audio file format (1 byte): applies to the control command 1. • 1 : PCM • 2 : WAVPCM • 3 : MP3 • 4 : AMR-NB • 5 : AMR-WB • total_size (2 bytes): applies to the control command 1. • offset (2 bytes): applies to the control command 1. • Audio data (N bytes):
Data	Data	

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

Command

Command	Combined content
1	Audio ID, audio format, total_size , offset , and audio data
2	Audio ID
3	Audio ID and port
4	Audio ID
5	None

The module returns the following data.

Field	Length (byte)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0004/0x0005
Data	5	• Subcommand: 0x49 • Control command (1 byte) • For control commands 1 through 4 : Status (1 byte): 0 indicates success. Other values indicate failure. • For control command 5 : • The number of audio clips (1 byte) • Audio ID (1 byte)

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

Status value

Code	Meaning
0	Success.
1	The upper limit of 16 KB is exceeded.
2	Audio ID is busy.
3	The maximum number of audio IDs is reached.
4	The audio ID does not exist.
5	The data of the audio ID is incomplete.
6	Audio device is busy.
7	Failed to play the audio ID.
8	The size of a single packet exceeds 1024 bytes.

Example

- Play an audio ID.
- 55 AA 03 71 00 04 49 03 01 00 C4

18. Bluetooth services

18.1. (Optional) Turn on/off control over Bluetooth (0x72 89)

Turn on or off the Bluetooth connection after the device is registered. It is turned on by default.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x89 • Control command: 0x01 : Turn on. 0x00 : Turn off.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x89 • Status: 0 : Failure. 1 : Success.

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

19. Function control

19.1. (Optional) Enable/disable low voltage shutdown (0x72 88)

Enable or disable low voltage shutdown for battery-powered devices. This feature is disabled by default.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x88 • Control command: 0x01 : Enable. 0x00 : Disable.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x88 • Status: 0 : Failure. 1 : Success.

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

19.2. Send QR code to MCU proactively (0x72 93)

After receiving the short URL of the QR code from the cloud, the module proactively sends it to the MCU. This feature is usually used on devices that have a screen to show the QR code.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0001+N
Data	1+N	- Subcommand: 0x93 - Short URL string.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.3. Set timeout for network exceptions (0x72 96)

After a successful dial, if the module fails to attach to the network within the specified consecutive time period, the device will restart.

- The minimum timeout is 180 seconds, with a deviation of 60 seconds.
- If the timeout is set to 0, it indicates this feature is disabled. The device will not restart if it is in airplane mode, has no SIM card inserted, or does not start dialing.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0003
		Data[0] Subcommand 0x96
Data	3	Data[1] to Data[2]: The timeout in seconds, high byte first.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
		Subcommand: 0x96
Data	2	- Status: 0 : Failure. 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.4. (Optional) Restart module (0x72 98)

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0001
Data	1	Data[0]: Subcommand 0x98
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module restarts after receiving the command.

19.5. Report cellular network alerts (0x71 9a)

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x71
Data length	2	0x0002

		• Data[0]: Subcommand 0x9A • Data[1]: Types of alerts • 1 : Network registration error. • 2 : Packet data protocol (PDP) activation error. • 3 : TCP connection failure, which occurs after the device has been connected to the cloud.
Data	2	
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.6. Set external wake-up pin (0x72 9b)

Configure the external wake-up pin for the module.

LZ501, LZ201, and LZ211 have the default external wake-up pin configured, so this command is not needed for them.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0003

Data	3	- Subcommand: 0x9BGPIO - The specified pin: one byte - Low-level duration: one byte, in the unit of 10 ms.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	- Subcommand: 0x9B - Status: 0 : Failure. 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.7. Enable/disable remote logging (0x72 9c)

By default, this feature is disabled. The configuration is stored in the key-value storage.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
		Subcommand: 0x9C
Data	2	• 00 : Disable. • 01 : Enable.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
		• Subcommand: 0x9C
Data	2	• Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.8. Notify MCU of restarting device (0x72 9d)

This feature applies to the MA510 module only.

The module sends the following data.

Field	Byte(s)	Description
-------	---------	-------------

Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0001
Data	1	Subcommand: 0x9D
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x9D • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.9. Enable SIM card hot swapping (0x72 9e)

The default state is based on the actual state of the module. The configuration is stored in the key-value storage.

The MCU sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03

Command	1	0x72
Data length	2	0x0001
Data	1	• Subcommand: 0x9E • Control command: • 1 : Enable • 0 : Disable.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0002
Data	2	• Subcommand: 0x9E • Status: • 0 : Failure. • 1 : Success.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.10. Control ADC (0x72 A0)

- Scheduled ADC reading will be stopped if you set the interval to 0 or choose a one-time reading task.
- The MCU integration protocol must be v1.1.3 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0003 + 2/1/3/0
		- Subcommand: 0xA0 - Command: 0x0 : ADC initialization 0x1 : One-time ADC reading 0x2 : Scheduled ADC reading 0x4 : ADC deinitialization - Port (1 byte) - When the command is 0 : - Reference voltage (2 bytes) - When the command is 1 : - Number of readings (1 byte) - When the command is 2 : - Reading interval (2 bytes) in milliseconds - Number of readings (1 byte)
Data	0x0003 + 2/1/3/0	

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	$0x0003 + 2/(2+2N)/1/(2+2N)/1$

		• Subcommand: 0xA0 • Command: • 0x0 : ADC initialization • 0x2 : Scheduled ADC reading • 0x1 and 0x3 : ADC reporting • 0x4 : ADC deinitialization • Port (1 byte) • When the command is 2 or 4 :
Data	$0x0003 + 2/(2+2N)/1/$ $(2+2N)/1$	• Result (1 byte): 0 for success, and 1 for failure. • When the command is 1 or 3 (1 + 2N bytes): • Result (1 byte): 0 for success, and 1 for failure. • Number of readings (1 byte): N • ADC value (2 bytes × N)
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

19.11. Control GPIO (0x72 A1)

The MCU integration protocol must be v1.1.3 or later.

The MCU sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x72
Data length	2	0x0003 + 3/0/1/0/5

		- Subcommand: 0xA1 - Command: 0x0 : GPIO initialization 0x1 : GPIO deinitialization 0x2 : Writing GPIO 0x3 : Reading GPIO 0x4 : Periodic output - Pin number (1 byte) - When the command is 0 : - Set the direction (1 byte): 0 for input and 1 for output. - Set the mode (1 byte): - Input mode: 0 for pull-up. 1 for pull-down. 2 for high impedance. - Output mode: 4 for push-pull. - Initial level (1 byte): 0 : Low level 1 : High levelThis parameter is invalid in input mode. - When the command is 2 (1 byte): 0 : Low level 1 : High level - When the command is 4 :
Data	0x0003 + 3/0/1/0/5	- Initial level (1 byte): 0 : Low level 1 : High level

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

The module returns the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x72
Data length	2	0x0003 + 1/1/1/2/1
Data	0x0003 + 1/1/1/2/1	- Subcommand: 0xA1 - Command: 0x0 : GPIO initialization 0x1 : GPIO deinitialization 0x2 : Writing GPIO 0x3 : Reading GPIO 0x4 : Periodic output - Pin number (1 byte) - When the command is 0 , 1 , 2 , or 4 : - Result (1 byte): 0 for success, and 1 for failure. - When the command is 3 (1 byte): - Result (1 byte): 0 for success, and 1 for failure. - Voltage level (1 byte)

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

20. Extended services

20.1. (Optional) Enable reset status notification (0x34 04)

The module supports local reset, reset via the app, and factory reset. This feature enables the module to proactively send its reset status to the MCU.

The MCU sends the following data.

Field	Length	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0001
Data	1	Subcommand: 0x04
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The module returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	1	Subcommand: 0x04
		Result:
	1	• 0x00 : Success • 0x01 : Failure.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

20.2. (Optional) Notify MCU of reset status (0x34 05)

Reset status	Description	Status value
Status 1	Local reset.	0x00
Status 2	Reset via app.	0x01
Status 3	Factory reset via app.	0x02

After the first attempt fails, the module will try two more times with a one-second interval.

The module sends the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	1	Subcommand: 0x05
		Reset status:
	1	• 0x00 : Local reset. • 0x01 : Reset via app. • 0x02 : Factory reset via app.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

The MCU returns the following data.

Field	Byte(s)	Description
Header	2	0x55aa
Version	1	0x03
Command	1	0x34
Data length	2	0x0001
Data	1	Subcommand: 0x05

Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.
----------	---	---

20.3. Send update in progress status (0x34 0A)

- This command applies to MA510 modules only. The MCU integration protocol must be v1.1.4 or later.
- When the module is updating its firmware, it will respond with this status upon receiving a command from the MCU instead of executing it. The MCU must wait for the update to complete.
- During a firmware update, the module will automatically stop reporting all data except for the update status. If the module restart depends on the MCU controlling `uart3_tx`, the MCU should pull down `uart3_tx` when receiving the update status. After the module restarts, the MCU should restore the configuration of `uart3_tx` when receiving a heartbeat from the module.

The module sends the following data.

Field	Bytes	Description
Header	2	0x55aa
Version	1	0x00
Command	1	0x34
Data length	2	0x0002
Data	2	0x0A (subcommand) Update status: 0x01 indicates the module is updating its firmware.
Checksum	1	Start from the header, add up all the bytes, and then divide the sum by 256 to get the remainder.

21. Version history

Version	Description
1.1.4	• Added the mandatory network type field to product information. • Added AGNSS control commands. • Added the command to send system firmware OTA update status, applying to MA510 modules only.
1.1.3	• Added the command to get GNSS HDOP. • Added the command to get GNSS orthometric height. • Added the command to get GNSS heading. • Added GNSS filter algorithm. You can enable or disable the filter algorithm and set the HDOP threshold (defaulting to 2). • Added the GPS module configuration to product information. • Added ADC and GPIO control. • Added the staging environment option to product information.
1.1.1	• Added RRC commands 0x71 4A . • Allowed you to disable the MCU wake-up pin when the MCU firmware is being updated.

1.1.0	- Added online TTS playback command 0x45 . - Added unknown status to the description of network status. - Added the command 0x49 to transfer and play audio data. - Added the status “module is ready for pairing” to the description of network status.
1.0.20	- Added the option to ACK packets of DP commands. - Added the command to set and read the gain of the mic and sidetone. - Changed the default wake-up pin of the TCS600U module to 70. - Added support for the MC665EU module. - Added the method of requesting PLMN (only supported by the MA510 module). - Removed the command to request the GPS version.
1.0.19	Added notification of firmware update status.
1.0.18	Added notification of the reset status.
1.0.17	- Added NB-IoT and CATM to the network type. - Added the command to instruct the MCU to restart the device. - Added the command to enable SIM card hot swapping. - Added the field of carrier code to the product information query.

1.0.16	• Added the command to enable/disable remote logging. • Added support for selecting the MCU update version. • Added the configuration of DP cache. • Added support for reporting record-type DP data. • Added 460800/921600 auto-baud detection.
1.0.15	• Added the command to request the GNSS version. • Added the command to request the visible GNSS satellite. • Added the key configuration of the external wake-up pin. • Added the command to read Bluetooth status. • Added the command to report and read network status.
1.0.14	• Added an option to turn on the heartbeat feature. • Added the command to request the version. • Added listening for and reporting DTMF events. • Added reporting cellular network alerts.

1.0.13

- Added the mute option for phone calls.
 - Added the definition for hex SMS transmission and reception.
 - Added the configuration of SMS transmission and reception format.
-

1.0.12

- Added the command to set the timeout for network exceptions.
- Added the mute option for SMS and incoming calls.
- Added the response to unsupported commands.
- Added support for Bluetooth Low Energy (LE) and LTE Cat.1 combo.
- Added the D0 command and its subcommands.
- Added the `slave` field in the response to a product information query.
- Added the command version `01` to the MCU update packets. Added the CRC32 field for Bluetooth LE firmware update.
- Added SMS encoding support for UTF-16BE, UTF-16LE, and GSM.
- Added the command to restart the cellular module.
- Added support for Bluetooth LE and LTE Cat.x in basic firmware.
- The modules deployed in mainland China use the Code page 936 encoding.
- The modules deployed outside mainland China use the UTF-8 encoding.

	Added the subcommands for LTE Cat.1 specific commands.
1.0.1	• 0x71: Read the LTE Cat.1 module. • 0x72: Write to the LTE Cat.1 module.
1.0.0	• The first release.