**GPS Locator**

**Protocol**

**Important revision records**

**content**

1.communication protocol 3

2.Definition of Terms 3

1.basic rules 4

4.packet format 6

4.1. start bit 6

4.2. package length 6

4.3. protocol number 6

4.4. information 6

4.5. information serial number 6

4.6. error checking 6

4.7. stop bit 7

5.Detailed explanation of terminal sending data packet to server 8

5.1. login information package 8

5.2. Positioning data package (GPS, LBS combined information package) 11

5.3. Alarm Packet ( GPS, LBS, Status Merge Packet) 15

5.4. Heartbeat packets ( status packets) 22

5.5. Base Station Packet ( LBS ONLY ) 26

5.6 Information transfer general package 28

6.The server sends packets to the terminal 29

6.1. server sends 29

6.2. terminal returns 30

6.3. cut off oil and electricity 30

6.4. Restoring gas and electricity 31

6.5. Add SOS number 32

6.6. Delete SOS number 32

6.7. set center number 32

6.8. delete center number 33

6.9. Turn on vibration alarm 33

6.10. Turn off vibration alarm 33

6.11. Turn on overspeed alarm 33

6.12. Turn off overspeed alarm 34

6.13. Turn on power failure alarm 34

6.14. Turn off the power failure alarm 34

6.15. Enable displacement alarm 35

6.16. Turn off displacement alarm 35

6.17. Turn on low battery alarm 35

6.18. Turn off low battery alarm 36

6.19. Turn on the fence alarm 36

6.20. Turn off the fence alarm 36

6.21. Control device restart 37

6.22. The server sends the query address information 37

6.23. GPS, phone number query address information package (0X2A) 37

7.Attached A CRC-ITU table lookup algorithm C language code snippet 42

8.Appendix B Communication Protocol Packet Fragment Example 43

**1.communication protocol**

**introduction**

This document defines the description of the application layer interface protocol of the vehicle GPS locator location service platform. The relevant interface protocol is only applicable to the interaction between the platform and the positioning terminal.

**2.Definition of Terms**

|  |  |  |
| --- | --- | --- |
| **terms, abbreviations** | **English meaning** | **Chinese meaning** |
| CMPP | China Mobile Peer to Peer | China Mobile Peer-to-Peer Agreement |
| GPS | Global Positioning System | Global Positioning System |
| GSM | Global System for Mobile Communication | Global System for Mobile Communications |
| GPRS | General Packet Radio Service | General Radio Packet Service |
| TCP | Transport Control Protocol | Transmission Control Protocol |
| LBS | Location Based Services | Assisted Location Services |
| IMEI | International Mobile Equipment Identity | International Mobile Equipment Identity |
| MCC | Mobile Country Code | The country code of the mobile user |
| MNC | Mobile Network Code | mobile network number |
| LAC | Location Area Code | location area code |
| Cell ID | Cell Tower ID | mobile base station |
| UDP | User Datagram Protocol | User Datagram Protocol |
| SOS | Save Our Ship/Save Our Souls | distress signal |
| CRC | Cyclic Redundancy Check | Cyclic Redundancy Check |
| NITZ | Network Identity and Time Zone, | Time zone |
| GIS | Geographic Information System | geographic information system |

**1.basic rules**

* 1. The GPRS connection is successfully established and the first login information packet is sent to the server. If the server response data packet is received within 5 seconds, the connection is considered to be normal, and the positioning information (GPS, LBS information packet) will be sent. After 3 minutes, the status information packet will be sent. Regularly confirm that the communication is normal.
	2. When the GPRS connection is not established successfully, the terminal cannot send the login information packet. When the GPRS connection fails 3 times, the terminal starts the timing restart function, and the time is 20 minutes. If the terminal successfully establishes a connection with the server within 20 minutes, and receives a data packet that the server responds to the login information packet sent by the terminal , the scheduled restart function will be disabled and the terminal will not restart, otherwise the terminal will automatically restart after 20 minutes.
	3. After receiving the login information packet sent by the terminal, the server returns a response data packet to the terminal. If the terminal does not receive a response packet from the server for more than 5 seconds after sending the login information packet or status information packet, it is considered that the current connection establishment is abnormal and GPS positioning is started. Data retransmission function , disconnect the current GPRS connection, re-establish a new GPRS connection and send the login information package.
	4. The connection is judged to be abnormal, and the login information packet or status information packet sent after the connection is established repeatedly for 3 times cannot receive the data packet responded by the server, and the terminal starts the timing restart function. The timing time is 10 minutes. If the server successfully establishes a connection and receives a data packet from the server, the scheduled restart function will be disabled, and the terminal will not restart. Otherwise, the terminal will automatically restart after 10 minutes.
	5. After the connection is established normally, after the GPS information is changed, the terminal regularly sends GPS and LBS combined information packets to the server, and the server can set the default sending protocol through instructions.
	6. In order to ensure the validity of the connection, status information is sent to the server at regular intervals, and the server returns a response packet for confirmation.
	7. For the terminal without registered IMEI number, the server should reply the login request response and heartbeat packet response, and do not directly disconnect the connection. (If the connection is directly disconnected or does not reply, the terminal will be continuously reconnected, and the GPRS traffic will be seriously consumed).

**data flow diagram**



**4.packet format**

Communication transmission is asynchronous and takes bytes as a unit.

 Total packet length: (10+N) Byte

|  |  |
| --- | --- |
| Format | length (Byte) |
| start bit | 2 |
| package length | 1(2) |
| protocol number | 1 |
| information | N |
| information serial number | 2 |
| error checking | 2 |
| stop bit | 2 |

* + 1. **start bit**

Fixed value, unified as hexadecimal 0x78 0x78 (packet length 1 bit) or 0x79 0x79 (packet length 2 bits)

* + 1. **package length**

length = protocol number + message content + message serial number + error check,

A total of (5+N) Bytes, because the information content is a variable-length field.

* + 1. **protocol number**

|  |  |
| --- | --- |
| Types of | value |
| login information | 0x01 |
| Positioning data (UTC) | 0x22 |
| status information | 0x13 |
| string information | 0x21 |
| LBS information (see the annex for the agreement) | 0x24 |
| Alarm data (UTC) | 0x26 |
| time zone time | 0x27 |
| GPS, phone number query address information (UTC) | 0x2A |
| WIFI packet | 0x2C |
| The server sends instruction information to the terminal | 0x80 |

* + 1. **information**

According to different applications, corresponding to the corresponding "protocol number" to determine the specific content.

* + 1. **information serial number**

The serial number of the first GPRS data (including status packets and GPS, LBS and other data packets) sent after power-on is '1', and the serial number of each data sent after that (including status packets and GPS, LBS data packets) is automatically incremented by 1 .

* + 1. **error checking**

The terminal or server can use the check code to judge whether the received information is wrong. In order to prevent errors in the data transmission process, an error check is added to prevent data misoperation, which increases the security and efficiency of the system. The error check code adopts the CRC-ITU check method.

The CRC-ITU value of the data from "Packet Length" to "Information Sequence Number" (including "Packet Length" and "Information Sequence Number") in the protocol body.

If the received information has a CRC error in the calculation, the receiver ignores and discards the data packet .

* + 1. **stop bit**

Fixed value, unified as hexadecimal 0x0D 0x0A .

**5.Detailed explanation of terminal sending data packet to server**

Separately explain common packet sending and server return

* + 1. **login information package**
			1. **Terminal sends data packet to server**

The login information packet is used to confirm to the server that the connection is established normally, and to submit the terminal ID to the server.

|  |  |  |
| --- | --- | --- |
| Format | length | Example |
| Login information packet (18 Byte) | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x11 |
| protocol number | 1 | 0x01 |
| Terminal ID | 8 | 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45 |
| Type ID | 2 | 0x00 0x00 |
| time zone language | 2 | 0x32 0x00 |
| information serial number | 2 | 0x00 0x01 |
| error checking | 2 | 0x90 0x01 |
| stop bit | 2 | 0x0d 0X0a |

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

See 4.3 Packet Format for details

* + - * 1. **Terminal ID**

Such as: 123456789012345,

Then the terminal ID is: 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45

* + - * 1. **Type ID**

The type identifier occupies two bytes. Determine the terminal type according to this identification code.

G17H 0x00 0x00

* + - * 1. **time zone language**

|  |  |  |
| --- | --- | --- |
| A nibble bit15—bit4 | 15 | The time zone is expanded by the value of 100 |
| 14 |
| 13 |
| 12 |
| 11 |
| 10 |
| 9 |
| 8 |
| 7 |
| 6 |
| 5 |
| 4 |
| Lower nibble bit4-bit0 | 3 | time zone stuff |
| 2 | No definition yet |
| 1 | language selection bit | 1 |
| 0 | language selection bit | 0 |

B it3 0-------Eastern time zone

1-------West time zone

If: Extended bit: 0X32 0X00 means East Eighth District, GMT+8:00.

Calculation method: 8\*100=800, converted to hexadecimal, 0X0320

Extended bit: 0X4D 0XD8 means the West Twelve District and 3/4 time zone, GMT-12:45.

Calculation method: 12.45\*100=1246, converted to hexadecimal, 0X04, 0XDD.

The algorithm here is to cyclically shift the calculated time zone value to the left by four bits and then combine the east, west and language selection bits of the time zone to save four bytes.

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

* + - 1. **server response packet**

|  |  |  |
| --- | --- | --- |
| illustrate | length | Example |
| Login information packet (18 Byte) | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x05 |
| protocol number | 1 | 0x01 |
| information serial number | 2 | 0x00 0x01 |
| error checking | 2 | 0xd9 0xdc |
| stop bit | 2 | 0x0d 0x0a |

The server sends a response packet to the terminal: (the protocol number in the response packet is the same as the protocol number of the data packet sent by the terminal)

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

See 4.3 Packet Format for details

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

* + 1. **Positioning data package (GPS, LBS combined information package)**
			1. **The terminal sends a general positioning data packet to the server**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) | Example |
| information | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x22 |
| protocol number | 1 | 0x22 |
| GPS information | date time | 6 | 0x0a 0x03 0x17 0xf 0x32 0x17 |
| GPS information satellites | 1 | 0xcc |
| latitude | 4 | 0x02 0x6C 0x6C 0x82 |
| longitude | 4 | 0x0C 0x37 0x16 0x82 |
| speed | 1 | 0x00 |
| heading, status | 2 | 0x15 0x3e |
| LBS information | MCC | 2 | 0x01 0xcc |
| MNC | 1 | 0x00 |
| LAC | 2 | 0x26 0x33 |
| Cell ID | 3 | 0x00 0x0e 0x7f |
| ACC | 1 | 0x01 |
| data reporting mode | 1 | 0x00 |
| GPS real-time supplementary transmission | 1 | 0x00 |
| serial number | 2 | 0x00 0x08 |
| error checking | 2 | 0x60 0xa5 |
| end bit | 2 | 0x0d 0x0a |

* + - 1. **Extended odometer positioning data package**

|  |  |  |  |
| --- | --- | --- | --- |
| information | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x26 |
| protocol number | 1 | 0x22 |
| GPS information | datetime | 6 | 0x0a 0x03 0x17 0xf 0x32 0x17 |
| GPS information satellites | 1 | 0xcc |
| latitude | 4 | 0x02 0x6C 0x6C 0x82 |
| longitude | 4 | 0x0C 0x37 0x16 0x82 |
| speed | 1 | 0x00 |
| heading, status | 2 | 0x15 0x3e |
| LBS information | MCC | 2 | 0x01 0xcc |
| MNC | 1 | 0x00 |
| LAC | 2 | 0x26 0x33 |
| Cell ID | 3 | 0x00 0x0e 0x7f |
| ACC | 1 | 0x01 |
| data reporting mode | 1 | 0x00 |
| GPS real-time supplementary transmission | 1 | 0x00 |
| Mileage Statistics | 4 | 0x00 0x00 0x00 0x00 |
| serial number | 2 | 0x00 0x08 |
| error checking | 2 | 0x1a 0xcf |
| end bit | 2 | 0x0d 0x0a |

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

See 4.3 Packet Format for details

* + - * 1. **datetime**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) | Example |
| year | 1 | 0x0A |
| moon | 1 | 0x03 |
| day | 1 | 0x17 |
| Time | 1 | 0x0F |
| Minute | 1 | 0x32 |
| Second | 1 | 0x17 |

 Such as: March 23, 2010 15:50:23

 Calculation method: 10 (decimal)=0A (hexadecimal)

 3 (decimal)=03 (hexadecimal)

 23 (decimal)=17 (hexadecimal)

 15 (decimal)=0F (hexadecimal)

 50 (decimal)=32 (hexadecimal)

23 (decimal)=17 (hexadecimal)

 The value is: 0x0A 0x03 0x17 0x0F 0x32 0x17

* + - * 1. **GPS information length, number of satellites involved in positioning**

1Byte is displayed with two hexadecimal characters, the first character is the length of GPS information, and the second character is the number of satellites involved in positioning

Example: When the value is 0xCB, it means that the length of GPS information is 12, and the number of satellites involved in positioning is 11.

(C = 12Bit length, B = 11 satellites)

* + - * 1. **latitude**

Occupies 4 bytes, indicating the latitude value of the positioning data. The value range is 0 to 162000000, indicating the range from 0 degrees to 90 degrees. The conversion method is as follows:

Convert the latitude and longitude values output by the GPS module into decimals in units of minutes; then multiply the converted decimals by 30000, and convert the multiplied result into a hexadecimal number.

Such as 22º 32.7658 ' =(22X60+32.7658)X30000=40582974, then convert to hexadecimal number

40582974(decimal)= 26B3F3E (hexadecimal)

The final value is 0x02 0x6B 0x3F 0x3E.

* + - * 1. **longitude**

Occupies 4 bytes, indicating the longitude value of the positioning data. The value range is 0 to 324000000, representing the range from 0 degrees to 180 degrees.

 The conversion method is the same as the conversion method of latitude

* + - * 1. **speed**

Occupies 1 byte, indicating the running speed of GPS, the value range is 0x00～0xFF, which means the range is 0～255km/h.

For example: 0x00 means 0 km/h.

 0x10 means 16 km/h

 0xFF means 255 km/h

* + - * 1. **state heading**

Occupies 2 bytes, indicating the running direction of the GPS, indicating a range of 0 to 360, unit: degree, with true north as 0 degree, clockwise.

|  |  |  |
| --- | --- | --- |
| BYTE\_1 | Bit7 | 0 |
| Bit6 | 0 |
| Bit5 | GPS real-time/differential positioning |
| Bit4 | GPS positioning |
| Bit3 | East and West |
| Bit2 | South Latitude, North Latitude |
| Bit1 | course |
| Bit0 |
| BYTE\_2 | Bit7 |
| Bit6 |
| Bit5 |
| Bit4 |
| Bit3 |
| Bit2 |
| Bit1 |
| Bit0 |

Note: The state information in the data packet is the state at the moment recorded by the time bit in the data packet.

For example: the value is 0x15 0x4C, which becomes 00010 101 in binary 01001100 ,

BYTE\_1 Bit7 0

BYTE\_1 Bit6 0

BYTE\_1 Bit5 0 (real-time GPS)

BYTE\_1 Bit4 1 (GPS has been positioned)

BYTE\_1 Bit3 0 (East longitude)

BYTE\_1 Bit2 1 (North Latitude)

BYTE\_1 Bit1 0

BYTE\_1 Bit0 1

BYTE\_2 Bit7 0

BYTE\_2 Bit6 1

BYTE\_2 Bit5 0 heading 332° ( 0101001100 binary to decimal is 332)

BYTE\_2 Bit4 0

BYTE\_2 Bit3 1

BYTE\_2 Bit2 1

BYTE\_2 Bit1 0

BYTE\_2 Bit0 0

It means that the GPS has been positioned, and the real-time GPS, north latitude, east longitude, and heading are 332°.

* + - * 1. **MCC**

Mobile Country Code (MCC)

For example: China's mobile country number is: China's mobile country number is 460 (decimal) 0x01 0xCC (decimal 460 is converted to hexadecimal, and the left side is filled with 0 if the hexadecimal is less than four digits)

The value range here is: 0x0000 ~ 0x03E7

* + - * 1. **MNC**

Mobile Network Code(MNC)

Such as: China Mobile is 0x00.

* + - * 1. **LAC**

Location Area Code (LAC) is included in the LAI and consists of two bytes, coded in hexadecimal. The available range is 0x0001-0xFFFE, and the code groups 0x0000 and 0xFFFF cannot be used (see GSM specifications 03.03, 04.08 and 11.11).

* + - * 1. **Cell ID**

Mobile base station Cell Tower ID (Cell ID) , the value range is 0x000000 ~ 0xFFFFFF

* + - * 1. **ACC**

ACC status ACC low is 00, ACC high is 01

* + - * 1. **data reporting mode**

**reserved, currently inactive**

* + - * 1. **data reporting mode**

**reserved, currently inactive**

* + - * 1. **GPS real-time supplementary transmission**

0x00 real-time upload

0x01 Supplementary transmission

* + - * 1. **Mileage Statistics**

Unit: m

Such as: 0x00 0x00 0x03 0xe8 is 1000 meters

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

* + 1. **Alarm Packet (** **GPS, LBS, Status Merge Packet)**

* + - 1. **The terminal sends an alarm packet to the server**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) | Example |
| information | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x25 |
| protocol number | 1 | 0x26 |
| datetime | 6 | 0x12 0x08 0x1b 0x06 0x00 0x02 |
| GPS information | GPS information satellites | 1 | 0x00 |
| latitude | 4 | 0x02 0x6c 0x6d 0x72 |
| longitude | 4 | 0x0c 0x37 0x15 0x86 |
| speed | 1 | 0x00 |
| heading, status | 2 | 0x04 0x21 |
| LBS information | LBS length | 1 | 0x08 |
| MCC | 2 | 0x01 0xcc |
| MNC | 1 | 0x00 |
| LAC | 2 | 0x26 0x33 |
| Cell ID | 3 | 0x00 0x10 0x04 |
| status information | Terminal information content | 1 | 0x12 |
| Voltage level | 1 | 0x06 |
| GSM signal strength | 1 | 0x04 |
| Alarm/language/extension port status | 2 | 0x02 0x01 |
| serial number | 2 | 0x01 0x33 |
| error checking | 2 | 0x47 0x9d |
| end bit | 2 | 0x0d 0x0a |

**The alarm package is composed of adding status information (alarm information) on the basis of the positioning package, and the encoding protocol format is also composed of adding status information on the basis of the positioning package .**

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

See 4.3 Packet Format for details

* + - * 1. **datetime**

For details, please refer to 5.2.1.4 of positioning data packet format

* + - * 1. **GPS information length, number of satellites involved in positioning**

For details, please refer to 5.2.1.5 of positioning data packet format

* + - * 1. **latitude**

For details, please refer to 5.2.1.6 of positioning data packet format

* + - * 1. **longitude**

For details, please refer to 5.2.1.7 Positioning Packet Format

* + - * 1. **speed**

For details, please refer to 5.2.1.8 of positioning data packet format

* + - * 1. **state heading**

For details, please refer to 5.2.1.9 of positioning data packet format

* + - * 1. **MCC**

For details, please refer to 5.2.1.10 Positioning Packet Format

* + - * 1. **MNC**

For details, please refer to 5.2.1.11 Positioning Packet Format

* + - * 1. **LAC**

For details, please refer to 5.2.1.12 of positioning data packet format

* + - * 1. **Cell ID**

For details, please refer to 5.2.1.13 Positioning Packet Format

* + - * 1. **Terminal information**

Occupies 1 byte and is used to represent various status information of the mobile phone.

|  |  |
| --- | --- |
| bit | Code meaning |
| BYTE | Bit7 | 1: Oil and electricity disconnected |
| 0: Oil and electricity connected |
| Bit6 | 1: GPS has been positioned |
| 0: GPS not positioned |
| Bit3～Bit5 | 100: SOS for help (not supported) |
| 011: Low battery alarm |
| 010: Power off alarm |
| 001: Vibration alarm |
| 000: normal |
| Bit2 | 1: Connected to the power supply for charging |
| 0: not connected to the power supply for charging |
| Bit1 | 1: ACC high |
| 0: ACC low |
| Bit0 | 1: Fortification |
| 0: disarm |

Such as: 0x44, the corresponding binary is 01000100

Indicates that the terminal status is: oil and electricity connected, GPS has been positioned, normal without alarm, power is connected to charging, ACC is low, disarmed

* + - * 1. **Voltage level**

The range is from 0 to 6, indicating that the voltage is from low to high.

0: No power (shutdown)

1: The battery is extremely low (not enough to make calls, send text messages, etc.)

2: Lighting is very low (low battery alarm)

3: Low battery (can be used normally)

4: In battery

5: High battery

6: Very high battery

Such as: 0x02 low battery send location alarm

* + - * 1. **GSM signal strength level**

0x00: no signal;

0x01: Very weak signal

0x02: weak signal

0x03: good signal

0x04: strong signal

Such as: 0x03 GSM signal is good

* + - * 1. **Alarm/language**

0x00 (front bit) 0x01 (rear bit)

Front position: terminal alarm status (applicable to alarm package and needs electronic fence function project)

After bit: the current language bit of the terminal

|  |  |
| --- | --- |
| byte 1 | 0x00: normal |
| 0x01: SOS for help |
| 0x02: Power off alarm |
| 0x03: Vibration alarm |
| 0x04: Alarm when entering the fence |
| 0x05: Alarm when leaving the fence |
| 0x06 Overspeed alarm |
| 0x09 Displacement alarm |
| 0x0A Enter GPS blind zone alarm |
| 0x0B GPS blind area alarm |
| 0x0C boot alarm |
| 0x0E External low power alarm |
| 0x0F External power low point protection alarm |
| 0X11 shutdown alarm |
| 0X13 (removal alarm) |
| 0X14 door alarm |
| 0X15 low power shutdown |
| 0X2C Collision Alarm |
| 0x 2D flip alarm |
| 0x 2E sharp turn warning |
|  | 0x 28 Rapid deceleration alarm |
|  | 0X29 Rapid acceleration alarm |
| byte 2 | 0x01 Chinese0x02 English |

For example :

No alarm Chinese: 0x00 0x01 ; No alarm English: 0x00 0x02

**In order to increase the reliability of the alarm information, the alarm information is repeatedly marked. In most cases, the alarm information is consistent with the upper terminal information. The inconsistencies are as follows:**

A. The terminal information appears low power alarm B. Alarm/language information enters and exits the fence alarm

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

Note: The state information in the data packet is the state at the moment recorded by the time bit in the data packet.

* + - 1. **The server sends an alarm packet reply to the terminal**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) |  |
| information | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x05 |
| protocol number | 1 | 0x26 |
| serial number | 2 | 0x01 0x33 |
| error checking | 2 | 0x5d 0xab |
| end bit | 2 | 0x0d 0x0a |

The alarm packet is composed of adding status information (alarm information) on the basis of the positioning packet, and the encoding protocol format is also composed of adding status information on the basis of the positioning packet .

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

See 4.3 Packet Format for details

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

* + - 1. **The server sends an alarm data address packet reply to the terminal**
				1. Chinese reply

The Chinese reply packet is as follows:

|  |  |  |
| --- | --- | --- |
| The command packet sent by the server to the terminal (15+M+N Byte) | start bit | 2 |
| data bit length | 1 |
| protocol number | 1 |
| information | instruction length | 1 |
| server flag | 4 |
| Instruction content | ALARMSMS | 8 |
| && | 2 |
| address content | M |
| && | 2 |
| telephone number | twenty one |
| ## | 2 |
| information serial number | 2 |
| Check Digit | 2 |
| stop bit | 2 |

Request Chinese address reply protocol number: 0X17.

Command content: ALARMSMS &&address content&&phone number (all 0)##( ALARMSMS , && , ## are fixed strings)

Chinese address content is delivered in **UNICODE** encoding.

**Example of replying to Chinese address information:**

7878 //start bit

85 //Data length

17 //Reply protocol number

7E //The command length is the length of the sent content information

00000001 // server flag

414C41524D534D 53 // ALARMSMS

2626 //&& separator

624059044F4D7F6E0028 //Chinese location is sent in UNICODE

004C004200530029003A

5E7F4E1C77015E7F5DDE

5E0282B190FD533AFF17

FF15FF144E6190530028

004E00320033002E0033

00390035002C00450031

00310032002E00390038

0038002996448FD1

2626 //&& separator

0000000000000000000000000000000000000000 //phone number

2323 //## Content information terminator

0106 //serial number

3825 //Check Digit

0D0A // stop bit

* + - * 1. **English reply**

Considering that English or other foreign addresses are longer, one data bit is not enough, and it is increased to 2 bytes. Notice:

Among them, the data bit length corresponding to the protocol number only for the return address information is changed to 2.

|  |  |  |
| --- | --- | --- |
| The command packet sent by the server to the terminal (15+M+N Byte) | start bit | 2 |
| data bit length | 2 |
| protocol number | 1 |
| information | instruction length | 2 |
| server flag | 4 |
| Instruction content | ALARMSMS | 8 |
| && | 2 |
| address content | M |
| && | 2 |
| telephone number | twenty one |
| ## | 2 |
| information serial number | 2 |
| Check Digit | 2 |
| stop bit | 2 |

**Request English address reply protocol number: 0X97**

Command content: ALARMSMS &&address content&&phone number (all 0)##( ALARMSMS , && , ## are fixed strings)

**Example reply English address information example:**

7979 //The start bit is changed from 7878 to 7979

00D 2 //Data length

97 //Reply protocol number

00CA //The command length is the length of the sent content information

00000001 //Server flag bit

414C41524D534D 53 / / ALARMSMS

2626 //&& separator

0053004F00530028004C //The English position is sent in UNICODE

0029003A005300680069

006D0069006E00200046

0061006900720079006C

0061006E006400200057

00650073007400200052

0064002C004800750069

006300680065006E0067

002C004800750069007A

0068006F0075002C0047

00750061006E00670064

006F006E00670028004E

00320033002E00310031

0031002C004500310031

0034002E003400310031

0029004E006500610072

00620079

2626 //&& separator

0000000000000000000000000000000000000000 //phone number

2323 //## Content information terminator

0007 //serial number

72b5 //check digit

0D0A //stop bit

Note: Since some alarm functions do not require the platform to reply to the address information, the platform does not need to reply to the resolution address after receiving the terminal alarm packet, and does not require the address to reply to the alarm type as follows:

1. Low battery alarm 2. Over-speed alarm 3. GPS blind spot

* + 1. **Heartbeat packets (** **status packets)**

Heartbeat packets are data packets that maintain the connection between the terminal and the server

* + - 1. **The terminal sends a heartbeat packet to the server**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) | Example |
| information | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x0a |
| protocol number | 1 | 0x13 |
| status information | Terminal information content | 1 | 0x04 |
| Voltage level | 1 | 0xf0 |
| GSM signal strength | 1 | 0x04 |
| language/extension port status | 2 | 0x8d 0x01 |
| serial number | 2 | 0x00 0x7f |
| error checking | 2 | 0x71 0x7f |
| end bit | 2 | 0x0d 0x0a |

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

0x13

* + - * 1. **Terminal information**

Occupies 1 byte and is used to represent various status information of the mobile phone.

|  |  |
| --- | --- |
| bit | Code meaning |
| BYTE | Bit7 | 1: Oil and electricity disconnected |
| 0: Oil and electricity connected |
| Bit6 | 1: GPS has been positioned |
| 0: GPS not positioned |
| Bit3～Bit5 |

|  |
| --- |
| 100: SOS for help (not supported) |
| 011: Low battery alarm |
| 010: Power off alarm |
| 001: Vibration alarm |
| 000: normal |

 |
| Bit2 | 1: Connected to the power supply for charging |
| 0: not connected to the power supply for charging |
| Bit1 | 1: ACC high |
| 0: ACC low |
| Bit0 | 1: Fortification |
| 0: disarm |

Such as: 0x44, the corresponding binary is 01000100

Indicates that the terminal status is: oil and electricity connected, GPS has been positioned, power is connected to charging, ACC is low, disarmed

* + - * 1. **Voltage level**

On battery power

The range is from 0 to 6, indicating that the voltage is from low to high.

0: No power (shutdown)

1: The battery is extremely low (not enough to make calls, send text messages, etc.)

2: Lighting is very low (low battery alarm)

3: Low battery (can be used normally)

4: In battery

5: High battery

6: Very high battery

Such as: 0x02 low battery send location alarm

When connected to external power, this bit is external power voltage\*10 byte 1

byte1 0xFX F is a fixed value, representing the state of external power connection

* + - * 1. **GSM signal strength level**

0x00: no signal;

0x01: Very weak signal

0x02: weak signal

0x03: good signal

0x04: strong signal

Such as: 0x03 GSM signal is good

* + - * 1. **language/extension port status**

0x00 (front bit) 0x01 (rear bit)

Front position: terminal expansion port status (external voltage\*10 byte0)

External voltage analysis example

byte1 byte0

0xF3 0x20 Take 0x320 to decimal 800 External voltage is 80.0V

After bit: the current language bit of the terminal

|  |  |
| --- | --- |
| front |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| rear | 0x01 Chinese0x02 English |

For example :

No alarm Chinese: 0x00 0x01

No alarm English: 0x00 0x02

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

* + - 1. **server response packet**

|  |  |
| --- | --- |
| Format | length |
| Heartbeat information packet (18 Byte) | start bit | 2 |
| package length | 1 |
| protocol number | 1 |
| information serial number | 2 |
| error checking | 2 |
| stop bit | 2 |

The server sends a response packet to the terminal: (the protocol number in the response packet is the same as the protocol number of the data packet sent by the terminal)

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

0x13

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

* + - 1. **Data instance**

|  |
| --- |
| Terminal sending example78 78 08 13 4B 04 03 00 01 00 11 06 1F 0D 0Aexplain |
| 0x78 0x78 | 0x08 | 0x13 | 0x4B 0x04 0x03 | 0x00 0x01 | 0x00 0x11 | 0x06 0x1F | 0x0D 0x0A |
| start bit | length | protocol number | information | reserved space (language) | serial number | error checking | stop bit |
| Server Reply Example78 78 05 13 00 11 F9 70 0D 0Aexplain |
| 0x78 0x78 | 0x05 | 0x13 | 0x00 0x11 | 0xF9 0x70 | 0x0D 0x0A |
| start bit | length | protocol number | serial number | error checking | stop bit |

* + 1. **Base Station Packet ( LBS ONLY )**
			1. **The terminal sends the positioning data packet to the server**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) | Example |
| information | start bit | 2 | 0x78 0x78 |
| package length | 1 | 0x31 |
| protocol number | 1 | 0x24 |
| datetime | 6 | 0x11 0x06 0x1b 0x06 0x2f 0x0d |
|  | TA | 1 | 0x00 |
| LBS information | MCC | 2 | 0x01 0xcc |
| MNC | 1 | 0x00 |
| CellNum | 1 | 0x04 |
| base station 1 | LAC | 2 | 0x26 0x33 |
| Cell ID | 3 | 0x00 0x0e 0x7f |
| RSSI | 1 | 0x00 |
| base station 2 | LAC | 2 | 0x26 0x33 |
| Cell ID | 3 | 0x00 0x0e 0x8a |
| RSSI | 1 | 0x24 |
| base station 3 | LAC | 2 | 0x26 0x33 |
| Cell ID | 3 | 0x00 0x0e 0x80 |
| RSSI | 1 | 0x23 |
| base station 4 | LAC | 2 | 0x26 0x33 |
| Cell ID | 3 | 0x00 0x0f 0xf2 |
| RSSI | 1 | 0x21 |
| base station 5 | LAC | 2 | 0x00 0x00 |
| Cell ID | 3 | 0x00 0x00 0x00 |
| RSSI | 1 | 0x00 |
| reserved | 1 | 0x00 |
| reserved | 1 | 0x00 |
| reserved | 1 | 0x00 |
| serial number | 2 | 0x00 0x06 |
| error checking | 2 | 0x9f 0xc8 |
| end bit | 2 | 0x0d 0x0a |

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

* + - * 1. **protocol number**

See packet format 4.3 0x24 for details

* + - * 1. **datetime**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) | Example |
| year | 1 | 0x0A |
| moon | 1 | 0x03 |
| day | 1 | 0x17 |
| Time | 1 | 0x0F |
| Minute | 1 | 0x32 |
| Second | 1 | 0x17 |

 Such as: March 23, 2010 15:50:23

 Calculation method: 10 (decimal)=0A (hexadecimal)

 3 (decimal)=03 (hexadecimal)

 23 (decimal)=17 (hexadecimal)

 15 (decimal)=0F (hexadecimal)

 50 (decimal)=32 (hexadecimal)

23 (decimal)=17 (hexadecimal)

 The value is: 0x0A 0x03 0x17 0x0F 0x32 0x17

* + - * 1. **MCC**

Mobile Country Code (MCC)

For example: China's mobile country number is: China's mobile country number is 460 (decimal) 0x01 0xCC (decimal 460 is converted into hexadecimal, and if the hexadecimal is less than four digits, the left side is filled with 0)

The value range here is: 0x0000 ~ 0x03E7

* + - * 1. **MNC**

Mobile Network Code(MNC)

Such as: China Mobile is 0x00.

* + - * 1. **LAC**

Location Area Code (LAC) is included in the LAI and consists of two bytes, coded in hexadecimal. The available range is 0x0001-0xFFFE, and the code groups 0x0000 and 0xFFFF cannot be used (see GSM specifications 03.03, 04.08 and 11.11).

* + - * 1. **Cell ID**

Mobile base station Cell Tower ID (Cell ID) , the value range is 0x000000 ~ 0xFFFFFF

5.5.1.9. RSSI

The signal strength of the primary cell , the value range is 0x00 ~ 0xFF , the actual signal strength is a negative value, upload its absolute value.

See Glossary – RSSI .

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

**5.6 Information transfer general package**

**Used for terminal transmission of various types of non-location data**

**Example data: 79 79 00 20 94 0A 08 69 81 41 01 11 51 23 04 60 04 24 29 80 22 56 89 86 04 04 10 18 C0 00 22 55 00 03 14 07 0D 0A**

|  |  |  |
| --- | --- | --- |
|  | **length** | **Detailed explanation** |
| **start bit** | **2** | **0x79 0x79** |
| **package length** | **2** | **0x00 0x20** |
| **protocol number** | **1** | **0x94** |
| **information** | **Type of information ( sub-protocol number )** | **1** | **00 External voltage****01~03 (customized)****04 Terminal state synchronization****05 Door Status****08 Self-test parameters****09 Positioning satellite information****0A ICCID and other information****.....to be added** |
| **data content** | **N** | **The content of the transmission varies according to the content of the information** |
| **information serial number** | **2** | **After booting, the serial number is automatically incremented by 1 every time the data is sent** |
| **error checking** | **2** | **CRC-ITU value from "Packet Length" to "Information Sequence Number". If the received information has a CRC error in the calculation, the receiver will ignore it (discard the packet)** |
| **stop bit** | **2** | **Fixed value, unified as 0x0D 0x0A** |

**When the type is 0A, this bit transmits the following information, which is a hexadecimal number**

**08 69 81 41 01 11 51 23 04 60 04 24 29 80 22 56 89 86 04 04 10 18 C0 00 22 55**

|  |  |  |
| --- | --- | --- |
| **IMEI** | **8** | **Example: IMEI number is 0869 814101115123, then the terminal ID is 0x08 0x69 0x81 0x41 0x01 0x11 0x51 0x23** |
| **IMSI** | **8** | **Example: IMSI number is 0460042429802256, then the terminal ID is 0x04 0x60 0x04 0x24 0x29 0x80 0x22 0x56** |
| **ICCID** | **10** | **Example: ICCID number is 898604041018 C0002255, then the terminal ID is 0x89 0x86 0x04 0x04 0x10 0x18 0xC0 0x00 0x22 0x55** |

**6.The server sends packets to the terminal**

The server

****

1. The server issues instructions first

2、Information returned by the terminal after execution

terminal

* + 1. **server sends**

|  |  |
| --- | --- |
| Format | length (Byte) |
| start bit | 2 |
| package length | 1 |
| protocol number | 1 |
| information | instruction length | 1 |
| server flag | 4 |
| Instruction content | M |
| language | 2 |
| information serial number | 2 |
| error checking | 2 |
| stop bit | 2 |

* + - 1. **start bit**

See packet format 4.1 for details

* + - 1. **package length**

See 4.2 Packet Format for details

* + - 1. **protocol number**

The terminal sends the protocol number to use: **0x80**

* + - 1. **instruction length**

Server flag bit + command content length

For example: the unit of byte length, 0x0A, means that the flag bit + instruction content occupies 10 bytes

* + - 1. **server flag**

It is reserved for the server to identify, and the terminal will return the received data in the return packet as it is.

* + - 1. **Instruction content**

It is represented by ASCII of character string, and the command content is compatible with SMS command

* + - 1. **language**

language bit of the terminal .

Chinese: 0x00 0x01

English: 0x00 0x02

* + - 1. **information serial number**

See 4.5 Packet Format for details

* + - 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - 1. **stop bit**

See 4.7 Packet Format for details

* + 1. **terminal returns**

|  |  |  |
| --- | --- | --- |
| Format | length (Byte) | Example |
| start bit | 2 | 0x79 0x79 |
| package length | 2 | 0x00 0x09 |
| protocol number | 1 | 0x21 |
| information | server flag | 4 | 0x 00 0x 00 0x 00 0x 01 |
| content encoding | 1 |  |
| content | M |  |
| information serial number | 2 | 0x00 0x01 |
| error checking | 2 | 0xD9 0xDC |
| stop bit | 2 | 0x0D 0x0A |

**6.2.1 Start bit**

Fixed value 0x79 0x79

**6.2.2 Packet length**

occupies 2 bytes

**6.2.3 Protocol number**

use **0x21**

* + 1. **server flag**

It is reserved for the server to identify, and the terminal will return the received data in the return packet as it is.

* + 1. **content encoding**

0x01 ASC II code

0x02 UTF16-BE encoding

* + 1. **content**

data to be sent.

* + 1. **information serial number**

See 4.5 Packet Format for details

* + 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + 1. **stop bit**

See 4.7 Packet Format for details

* + 1. **cut off oil and electricity**

**Function description:** Cut off the vehicle oil and electricity control circuit

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| RELAY,1 # |
| **terminal returns** |
| return successfully |
| DYD=Success! |
| return on failure |
|  |
|  |
|  |

* + 1. **Restoring gas and electricity**

**Function description:** Turn on the vehicle oil and electricity control circuit

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| RELAY,0 # |
| **terminal returns** |
| return successfully |
|  |
| return on failure |
|  |

**illustrate:**

**various responses when the oil and electricity are cut off , and the success can only be judged by searching for the Success character.**

**RELAY,ERROR :104**

**command error**

**RELAY,ERROR :103**

**command parameter error**

**RELAY,FAIL ! ACC ON, GPS has Not FIXED or speed > 20KM/H, relay delay!!**

**Power failure failed! , When the ACC is turned on, the GPS is not positioned or the speed is greater than 20KM/H, the fuel cut operation will be delayed!**

**RELAY, The oil has been cut already!!**

**Power failure failed! Has been in a state of power outage!**

**RELAY, The oil has been Resume already!!**

**Failed to restore gas and electricity! Already in a state of oil recovery !**

**Cut off the fuel supply: Success!Speed :0 km/h.**

**Oil and power cut off successfully**

**Restore fuel supply:Success !**

**Successful recovery of oil and power**

**RELAY,Restore fuel supply:Success!**

**Successful recovery of oil and power**

**RELAY, Cut off the fuel supply: Success! Speed: 0 km/h.**

**The end game is successful!**

* + 1. **Add SOS number**

**Function description:** Add the SOS number for receiving alarm SMS and phone calls

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| SOS,A,NUM1,NUM2,NUM3 #  |
| **terminal returns** |
| return successfully |
| OK!SOS1:NUM1SOS2:NUM2SOS3NUM3 |
| return on failure |
| ERROR:XXX |

* + 1. **Delete SOS number**

**Function description:** delete the SOS number for receiving alarm SMS and phone calls

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| SOS,D,NUM1,NUM2,NUM3# |
| **terminal returns** |
| return successfully |
| OK!SOS1:NUM1SOS2:NUM2SOS3NUM3 |
| return on failure |
| ERROR:XXX |

* + 1. **set center number**

**Function description:** Set the center number that can control the oil and power cutoff

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| CENTER,A,NUM #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

* + 1. **delete center number**

**Function description:** delete the center number that can control oil and power cutoff

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| CENTER,D #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

* + 1. **Turn on vibration alarm**

**Function description:** Turn on the vibration alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| SENALM, ON, alarm mode #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

**Note: The alarm mode is 0: platform; 1: platform + SMS; 2: platform + SMS + phone call; 3 platform + phone call;**

* + 1. **Turn off vibration alarm**

**Function description:** Turn off the vibration alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| SENALM, OFF #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

* + 1. **Turn on overspeed alarm**

**Function description:** Enable overspeed alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| SPEED, ON, TIME, SPEED, alarm mode # |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

**Note: The alarm mode is 0: platform; 1: platform + SMS;**

* + 1. **Turn off overspeed alarm**

**Function description:** Turn off the overspeed alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| SPEED, OFF #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

* + 1. **Turn on power failure alarm**

**Function description:** Turn on the power failure alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| POWERALM, ON, alarm mode #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

**Note: The alarm mode is 0: platform; 1: platform + SMS; 2: platform + SMS + phone call; 3 platform + phone call;**

* + 1. **Turn off the power failure alarm**

**Function description:** Turn off the power failure alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| POWERALM, OFF #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |
|  |

* + 1. **Enable displacement alarm**

**Function description:** Enable displacement alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| MOVING, NO, displacement radius, alarm mode #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

**Note: The alarm mode is 0: platform; 1: platform + SMS; 2: platform + SMS + phone call; 3 platform + phone call;**

**Displacement radius:** 100 ~ 1000

* + 1. **Turn off displacement alarm**

**Function description:** Turn off the displacement alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| MOVING ,OFF# |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

* + 1. **Turn on low battery alarm**

**Function description:** Turn on low battery alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| BATALM, ON, alarm mode #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

**Note: The alarm mode is 0: platform; 1: platform + SMS;**

* + 1. **Turn off low battery alarm**

**Function description:** Turn off the low battery alarm

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| BATALM, OFF #  |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

* + 1. **Turn on the fence alarm**

**Function description:** Open the electronic fence

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| FENCE, ON, 0, center latitude , center longitude , fence radius , X, alarm mode #For example: FENCE,ON,0,N1.2971,E103.822349,61,IN,0# |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

**Note: The alarm mode is 0: platform; 1: platform + SMS;**

**X=IN/OUT; IN: alarm when entering the fence, OUT: alarm when exiting the fence, if it is empty, it means that the alarm will be issued when entering/exiting the fence; the default is that the alarm will be issued when entering/exiting the fence;**

* + 1. **Turn off the fence alarm**

**Function description:** Open the electronic fence

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| FENCE, OFF # |
| **terminal returns** |
| return successfully |
| OK |
| return on failure |
| ERROR:XXX |

* + 1. **Control device restart**

**Function description:** Make the device restart after 1 minute

In the example, the sent and returned strings are converted to ASCII to generate the command content

|  |
| --- |
| **server sends** |
| RESET #  |
| **terminal returns** |
| return successfully |
| The terminal will restart after 1 minute! |
| return on failure |
| ERROR:XXX |

* + 1. **The server sends the query address information**

In the example, the sent and returned strings are converted to ASCII to generate the command content

server sends

ADDRESS,Address content,Phone number

Note: Chinese address content is issued in UNICODE code.

* + 1. **GPS, phone number query address information package (0X2A)**



* + - 1. **Terminal sends server information**

Terminal receives

The format is basically the same as that mentioned in the content of GPS information above, with the addition of a telephone number for querying the address.

|  |  |
| --- | --- |
| Format | length (Byte) |
| start bit | 2 |
| package length | 1 |
| protocol number | 1 |
| information | datetime | 6 |
| GPS information | GPS information length, number of satellites involved in positioning | 1 |
| latitude | 4 |
| longitude | 4 |
| speed | 1 |
| heading, status | 2 |
| telephone number | twenty one |
| language | 2 |
| information serial number | 2 |
| error checking | 2 |
| stop bit | 2 |

* + - * 1. **start bit**

See packet format 4.1 for details

* + - * 1. **package length**

See 4.2 Packet Format for details

For example: the unit of byte length, 0x2E, means that the instruction content occupies 46 bytes

* + - * 1. **protocol number**

Use **0x2A** .

* + - * 1. **datetime**

For details, please refer to 5.2.1.4 of positioning data packet format

* + - * 1. **GPS information length, number of satellites involved in positioning**

For details, please refer to 5.2.1.5 of positioning data packet format

* + - * 1. **latitude**

For details, please refer to 5.2.1.6 of positioning data packet format

* + - * 1. **longitude**

For details, please refer to 5.2.1.7 Positioning Packet Format

* + - * 1. **speed**

For details, please refer to 5.2.1.8 of positioning data packet format

* + - * 1. **course**

For details, please refer to 5.2.1.9 of positioning data packet format

* + - * 1. **telephone number**

The SOS phone number of the request address query, converted through ASCII, and the right side is filled with 0 if it is less than 21 digits.

* + - * 1. **language**

language bit of the terminal .

Chinese: 0x00 0x01

English: 0x00 0x02

* + - * 1. **information serial number**

See 4.5 Packet Format for details

* + - * 1. **error checking**

For details, please refer to 4.6 of Packet Format

* + - * 1. **stop bit**

See 4.7 Packet Format for details

* + - 1. **server response**

According to the extended command request to reply to the Chinese address or English address, the reply data packets are inconsistent.

* + - * 1. **Chinese reply**

The Chinese reply packet is as follows:

|  |  |
| --- | --- |
| Format | length (Byte) |
| The command packet sent by the server to the terminal (15+M+N Byte) | start bit | 2 |
| data bit length | 1 |
| protocol number | 1 |
| information | instruction length | 1 |
| server flag | 4 |
| Instruction content | ADDRESS | 7 |
| && | 2 |
| address content | M |
| && | 2 |
| telephone number | twenty one |
| ## | 2 |
| information serial number | 2 |
| Check Digit | 2 |
| stop bit | 2 |

Request Chinese address reply protocol number: 0X17.

Command content: ADDRESS&&address content&&phone number##(ADDRESS, &&, ## are fixed strings)

Chinese address content is delivered in **UNICODE** encoding.

**Example of replying to Chinese address information:**

7878 //start bit

84 //Data length

17 //Reply protocol number

7E //The command length is the length of the sent content information

00000001 // server flag

41444452455353 //ADDRESS

2626 //&& separator

624059044F4D7F6E0028 //Chinese location is sent in UNICODE

004C004200530029003A

5E7F4E1C77015E7F5DDE

5E0282B190FD533AFF17

FF15FF144E6190530028

004E00320033002E0033

00390035002C00450031

00310032002E00390038

0038002996448FD1

2626 //&& separator

313337313038313931333500000000000000000000 //phone number

2323 //## Content information terminator

0106 //serial number

3825 //Check Digit

0D0A // stop bit

* + - * 1. **English reply**

Considering that English or other foreign addresses are long, one data bit is not enough, and it is increased to 2 bytes. Notice:

Among them, the data bit length corresponding to the protocol number only for the return address information is changed to 2.

|  |  |
| --- | --- |
| Format | length (Byte) |
| The command packet sent by the server to the terminal (15+M+N Byte) | start bit | 2 |
| data bit length | 2 |
| protocol number | 1 |
| information | instruction length | 2 |
| server flag | 4 |
| Instruction content | ADDRESS | 7 |
| && | 2 |
| address content | M |
| && | 2 |
| telephone number | twenty one |
| ## | 2 |
| information serial number | 2 |
| Check Digit | 2 |
| stop bit | 2 |

**Request English address reply protocol number: 0X97**

Command content: ADDRESS&&address content&&phone number##(ADDRESS, &&, ## are fixed strings)

**Example reply English address information example:**

7878 //Start bit

00D1 //Data length

97 //Reply protocol number

00CA //The command length is the length of the sent content information

00000001 //Server flag bit

41444452455353 / /ADDRESS

2626 //&& separator

0053004F00530028004C //The English position is sent in UNICODE

0029003A005300680069

006D0069006E00200046

0061006900720079006C

0061006E006400200057

00650073007400200052

0064002C004800750069

006300680065006E0067

002C004800750069007A

0068006F0075002C0047

00750061006E00670064

006F006E00670028004E

00320033002E00310031

0031002C004500310031

0034002E003400310031

0029004E006500610072

00620079

2626 //&& separator

313235323031333739303737343035310000000000 //Phone number

2323 //## Content information terminator

0007 //serial number

72b5 //check digit

0D0A //stop bit

**7.Attached A CRC-ITU table lookup algorithm C language code snippet**

CRC-ITU table lookup algorithm C language code snippet

static const U16 crctab16[] =

{

0X0000, 0X1189, 0X2312, 0X329B, 0X4624, 0X57AD, 0X6536, 0X74BF,

0X8C48, 0X9DC1, 0XAF5A, 0XBED3, 0XCA6C, 0XDBE5, 0XE97E, 0XF8F7,

0X1081, 0X0108, 0X3393, 0X221A, 0X56A5, 0X472C, 0X75B7, 0X643E,

0X9CC9, 0X8D40, 0XBFDB, 0XAE52, 0XDAED, 0XCB64, 0XF9FF, 0XE876,

0X2102, 0X308B, 0X0210, 0X1399, 0X6726, 0X76AF, 0X4434, 0X55BD,

0XAD4A, 0XBCC3, 0X8E58, 0X9FD1, 0XEB6E, 0XFAE7, 0XC87C, 0XD9F5,

0X3183, 0X200A, 0X1291, 0X0318, 0X77A7, 0X662E, 0X54B5, 0X453C,

0XBDCB, 0XAC42, 0X9ED9, 0X8F50, 0XFBEF, 0XEA66, 0XD8FD, 0XC974,

0X4204, 0X538D, 0X6116, 0X709F, 0X0420, 0X15A9, 0X2732, 0X36BB,

0XCE4C, 0XDFC5, 0XED5E, 0XFCD7, 0X8868, 0X99E1, 0XAB7A, 0XBAF3,

0X5285, 0X430C, 0X7197, 0X601E, 0X14A1, 0X0528, 0X37B3, 0X263A,

0XDECD, 0XCF44, 0XFDDF, 0XEC56, 0X98E9, 0X8960, 0XBBFB, 0XAA72,

0X6306, 0X728F, 0X4014, 0X519D, 0X2522, 0X34AB, 0X0630, 0X17B9,

0XEF4E, 0XFEC7, 0XCC5C, 0XDDD5, 0XA96A, 0XB8E3, 0X8A78, 0X9BF1,

0X7387, 0X620E, 0X5095, 0X411C, 0X35A3, 0X242A, 0X16B1, 0X0738,

0XFFCF, 0XEE46, 0XDCDD, 0XCD54, 0XB9EB, 0XA862, 0X9AF9, 0X8B70,

0X8408, 0X9581, 0XA71A, 0XB693, 0XC22C, 0XD3A5, 0XE13E, 0XF0B7,

0X0840, 0X19C9, 0X2B52, 0X3ADB, 0X4E64, 0X5FED, 0X6D76, 0X7CFF,

0X9489, 0X8500, 0XB79B, 0XA612, 0XD2AD, 0XC324, 0XF1BF, 0XE036,

0X18C1, 0X0948, 0X3BD3, 0X2A5A, 0X5EE5, 0X4F6C, 0X7DF7, 0X6C7E,

0XA50A, 0XB483, 0X8618, 0X9791, 0XE32E, 0XF2A7, 0XC03C, 0XD1B5,

0X2942, 0X38CB, 0X0A50, 0X1BD9, 0X6F66, 0X7EEF, 0X4C74, 0X5DFD,

0XB58B, 0XA402, 0X9699, 0X8710, 0XF3AF, 0XE226, 0XD0BD, 0XC134,

0X39C3, 0X284A, 0X1AD1, 0X0B58, 0X7FE7, 0X6E6E, 0X5CF5, 0X4D7C,

0XC60C, 0XD785, 0XE51E, 0XF497, 0X8028, 0X91A1, 0XA33A, 0XB2B3,

0X4A44, 0X5BCD, 0X6956, 0X78DF, 0X0C60, 0X1DE9, 0X2F72, 0X3EFB,

0XD68D, 0XC704, 0XF59F, 0XE416, 0X90A9, 0X8120, 0XB3BB, 0XA232,

0X5AC5, 0X4B4C, 0X79D7, 0X685E, 0X1CE1, 0X0D68, 0X3FF3, 0X2E7A,

0XE70E, 0XF687, 0XC41C, 0XD595, 0XA12A, 0XB0A3, 0X8238, 0X93B1,

0X6B46, 0X7ACF, 0X4854, 0X59DD, 0X2D62, 0X3CEB, 0X0E70, 0X1FF9,

0XF78F, 0XE606, 0XD49D, 0XC514, 0XB1AB, 0XA022, 0X92B9, 0X8330,

0X7BC7, 0X6A4E, 0X58D5, 0X495C, 0X3DE3, 0X2C6A, 0X1EF1, 0X0F78,

};

// Calculate the 16-bit CRC for the given length of data.

U16 GetCrc16(const U8\* pData, int nLength)

{

U16 fcs = 0xffff; // initialization

while(nLength> 0){

fcs = (fcs >> 8) ^ crctab16[(fcs ^ \*pData) & 0xff];

nLength--;

pData++;

}

return ~fcs; // negate

}

**8.Appendix B Communication Protocol Packet Fragment Example**

The following data is intercepted from the communication between the terminal and the server. The data displayed in hexadecimal is sent out by the terminal, and received by the server:

Login package:

Issued: 78 78 0D 01 03 53 41 35 32 15 03 62 00 02 2D 06 0D 0A

Reception: 78 78 05 01 00 02 EB 47 0D 0A

GPS data packets (06 uses GPSLBS to merge information packets):

Issued: 78 78 1F 12 0B 08 1D 11 2E 10 CF 02 7A C7 EB 0C 46 58 49 00 14 8F 01 CC 00 28 7D 00 1F B8 00 03 80 81 0D 0A

**Status package:**

Issued: 78 78 0A 13 44 01 04 00 01 00 05 08 45 0D 0A

Reception: 78 78 05 13 00 05 AF D5 0D 0A

**The terminal obtains the address information from the server:**

**Chinese:**

Issue: 78 78 2E 1A 0B 0B 0F 0E 21 17 CF 02 7A C8 87 0C 46 57 E3 00 14 02 36 36 33 36 36 00 03 00 04 00 00 00 00 00 00 00 00 00 0 00 00 E9 0D 0A

Receive: 78 78 94 17 8E 00 00 00 01 41 44 44 52 45 53 53 26 26 4F 4D 7F 6E 00 3A 5E 7F 4E 1C 77 01 60 E0 5D DE 5E 02 4E 91 5C 80 89 5E 7F 53 4E 4E 00 8D EF 00 2E 79 BB 60 E0 5D DE 5B 89 4F 17 4F 1A 8B A1 5E 08 4E 8B 52 A1 62 40 7E A6 00 33 00 32 7C 73 00 2E 79 02 59 E0 5D DE 55 46 62 95 8D 44 67 0D 52 A1 4E 2D 5F C3 7E A6 00 33 00 32 7C 73 00 2E 26 26 36 36 33 36 36 00 03 00 04 00 00 00 00 00 2 2 0 0 00 0 00 01 E4 2A 0D 0A

Content sent by the server: Location: Yunshan West Road, Huizhou City, Guangdong Province. Wenhua 1st Road. It is about 32 meters away from Huizhou Anzhong Accounting Firm, and about 32 meters away from Huizhou Foreign Investment Service Center.

The phone number is: 66366

**English:**

Issue: 78 78 2E 1A 0B 0B 0F 0E 1E 08 CF 02 7A C8 A2 0C 46 57 D7 00 14 02 36 36 33 36 36 00 03 00 04 00 00 00 00 00 00 00 00 00 2 00 000 3A 0D 0A

Receive: 78 78 00 E9 97 00 E2 00 00 00 01 41 44 44 52 45 53 53 26 26 00 50 00 72 00 65 00 63 00 69 00 73 00 60 00 6C 00 79 00 60 00 00 4 00 74 00 69 00 6E 00 67 00 3A 00 31 00 75 00 20 00 59 00 75 00 73 00 68 00 61 00 7E 00 20 00 57 00 65 00 74 00 20 00 52 00 64 00 2c 00 48 00 75 00 69 00 63 00 68 00 65 00 67 00 2c 00 48 00 75 00 75 00 2c 00 47 00 75 00 61 00 6E 00 67 00 64 00 6E 00 67 00 64 00 6E 00 67 00 64 00 6E 00 6E 00 67 00 2c 00 35 00 31 00 30 00 33 00 28 00 4e 00 32 00 33 00 31 00 31 00 31 00 37 00 37 00 2C 00 45 00 31 00 31 00 34 00 2E 00 34 00 30 00 39 00 32 00 32 00 29 26 26 36 36 33 36 36 00 03 00 04 00 00 00 00 00 00 00 00 00 00 00 00 23 23 00 01 AF 4D 0D 0A

Content delivered by the server: Precisely Locating: No. 10 Yunshan West Rd, Huicheng, Huizhou, Guangdong, 516003 (N23.11177, E114.40922)

The phone number is: 66366

**Alarm package process:**

**Chinese SMS:**

Issued: 78 78 25 16 0B 0B 0F 0E 24 1D CF 02 7A C8 87 0C 46 57 E6 00 14 02 09 01 CC 00 28 7D 00 1F 72 65 06 04 01 01 00 36 56 A4 0D 0A

Reception: 78 78 05 16 00 36 95 70 0D 0A

Receive: 78 78 BE 17 B8 00 00 00 01 41 4C 41 52 4D 53 4D 53 26 26 7D 27 60 25 54 7C 53 EB 00 3A 5E 7F 4E 1C 77 01 60 E0 5D DE 5E 718D 4E 891 EF 00 2E 65 87 53 4E 4E 00 8D EF 00 2E 79 BB 4E 2D 88 4C 00 41 00 54 00 4D 7E A6 00 33 00 31 7C 73 00 2E 79 BB E 4E 2D 88 4C 6C 7C 5C 58F 5 A6 00 33 00 31 73 00 2E 00 31 00 31 00 2d 00 31 00 31 00 31 00 35 00 31 00 34 00 »33 00 36 00 32 00 39 26 26 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 D 23 0 A 00

SMS content: Emergency call: Yunshan West Road, Huizhou City, Guangdong Province. Wenhua 1st Road. About 31 meters away from Bank of China ATM. About 31 meters away from Bank of China Jiangbei Sub-branch., 11-11-15 14:36:29

The specific meaning of the above instructions can be found in the protocol description document by looking up the table.

|  |
| --- |
| **Description: English symbols must be used, the command setting is correct Reply: OK, the command is abnormal: no reply or the reply command format is incorrect** |
| NO. | functional item | Instruction format | Remarks/Examples |
| 1 | Address query | 123 or DW |  |
| 2 | Status query | STATUS# |  |
| 3 | version query | VERSION# |  |
| 4 | query parameter settings | PARAM# |  |
| 5 | restart command | RESET# |  |
| 6 | Add SOS number | SOS,A , Number1 , Number2 , Number3 # |  |
| 7 | Delete SOS number | SOS, D, number 1, number 2, number 3 # |  |
| 8 | Query SOS number | SOS# |  |
| 9 | Change IP settings | SERVER,0,IP,Port,0# | SERVER,0,120.24.248.12,8005,0# |
| 10 | Change domain settings | SERVER,1,domain name,port,0# | SERVER,1,GW.CARHERE.NET,8005,0# |
| 11 | Enable power failure alarm settings | POWERALM, A ,M,T1,T2,#A=ONM=0 ～ 2 ; 0 only GPRS , 1 SMS+GPRS , 2 GPRS+SMS+CALLT1=2 ~ 60 seconds; power failure detection time T2=1-3600 seconds; minimum charging time |  |
| 12 | Turn off the power failure alarm | POWERALM,OFF# |  |
| 13 | Query power failure alarm status | POWERALM# |  |
| 14 | Enable low battery alarm settings | BATALM,A ,M#A=ONM=0 ～ 10 means: GPRS only , 1 represents: SMS+GPRS , |  |
| 15 | Turn off low battery alarm settings | BATALM,OFF# |  |
| 16 | Query the low battery alarm status | BATALM# |  |
| 17 | Enable displacement alarm settings | MOVING,A ,R,M#A=ONR=100 ~ 1000 ; displacement radiusM=0 ～ 2 ; 0 only GPRS , 1 SMS+GPRS ; 2 GPRS+SMS+CALL |  |
| 18 | Turn off displacement alarm | MOVING,OFF# |  |
| 29 | Query the displacement setting status | MOVING# |  |