**GPS Tracker**

**Communication Protocol**

（S5E/S5L）

Copyright

This document is copyrighted by SEEWORLD Technology Co.,Ltd. All rights reserved.

Any unauthorized copy or transmission of the document partially or wholly shall be subject to prosecution.

**CONTENT**

**[I. COMMUNICATION PROTOCOL 错误！未定义书签。](#_Toc19804667)**

**[II. TERMS DEFINITIONS 4](#_Toc19804668)**

**[III. ASIC RULES 错误！未定义书签。](#_Toc19804669)**

**[IV. DATA PACKET FORMAT 错误！未定义书签。](#_Toc19804670)**

[4.1. S TART BIT 8](#_Toc19804671)

[4.2. PACKET LENGTH 8](#_Toc19804672)

[4.3. PROTOCOL NUMBER 8](#_Toc19804673)

[4.4. INFORMATION CONTENTS 8](#_Toc19804674)

[4.5. INFORMATION SERIAL NUMBER 9](#_Toc19804675)

[4.6. ERROR CHECK 9](#_Toc19804676)

[4.7. STOP BIT 9](#_Toc19804677)

**[V. Details about Data Packet sent by Server to Terminal 错误！未定义书签。](#_Toc19804678)**

**[5.1. LOGIN M ESSAGE PACKET 10](#_Toc19804679)**

[5.1.1. Terminal Sending Data Packet to Server 10](#_Toc19804680)

[5.1.2. Server Response Packet 11](#_Toc19804681)

**[5.2. LOCATION DATA PACKET ( COMBINED INFORMATION PACKAGE OF GPS AND LBS） 13](#_Toc19804682)**

[5.2.1. Terminal Sending Location Data Packet to Server 13](#_Toc19804683)

**[5.3. HEARTBEAT PACKET 13](#_Toc19804684)**

[5.3.1. Terminal Sending Heartbeat Packet to Server 20](#_Toc19804685)

[5.3.2. Server Response Packet 22](#_Toc19804686)

**[5.4. ALARM PACKET (GPS, LBS, COMBINED STATUS INFORMATION PACKET ) 23](#_Toc19804687)**

[5.4.1. Terminal Sending Alarm Packet to Server 23](#_Toc19804688)

[5.4.2. Server Sending the Alarm Packet Reply to Terminal 24](#_Toc19804689)

**[VI. Server sending Data Packet to Terminal 错误！未定义书签。](#_Toc19804690)**

[6.1. Packet Sent by Server 25](#_Toc19804691)

[6.1.1. Start Bit 25](#_Toc19804692)

[6.1.2. Packet Length 25](#_Toc19804693)

[6.1.3. Protocol Number 25](#_Toc19804694)

[6.1.4. Length of Command 26](#_Toc19804695)

[6.1.5. Server Flag Bit 26](#_Toc19804696)

[6.1.6. Command Content 26](#_Toc19804697)

[6.1.7. Information Serial Number 26](#_Toc19804698)

[6.1.8. Error Check 26](#_Toc19804699)

[6.1.9. Stop Bit 26](#_Toc19804700)

[6.2. Packet Replied by Terminal 26](#_Toc19804701)

[6.2.1. Start Bit 27](#_Toc19804702)

[6.2.2. Packet Length 27](#_Toc19804703)

[6.2.3. Protocol Number 27](#_Toc19804704)

[6.2.4. Length of Command 27](#_Toc19804705)

[6.2.5. Server Flag Bit 27](#_Toc19804706)

[6.2.6. Command Content 27](#_Toc19804707)

[6.2.7. Information Serial Number 27](#_Toc19804708)

[6.2.8. Error Check 27](#_Toc19804709)

[6.2.9. Stop Bit 28](#_Toc19804710)

**[VII. Attached A A U CRC-ITU look-up table algorithm C C language code fragment 错误！未定义书签。](#_Toc19804711)**

1. **Communication Protocol**

**Introduction**

This document defines instructions about interface protocol on application layer of vehicles GPS tracker and location-based service platform. Related interface protocol only applies in the interaction between the platform and the position terminal.

# **Terms, Definitions**

|  |  |  |
| --- | --- | --- |
| **Terms, Abbreviation** | **Definition in English** | **Definition in Chinese** |
| CMPP | China Mobile Peer to Peer | 中国移动点对点协议 |
| GPS | Global Positioning System | 全球卫星定位系统 |
| GSM | Global System for Mobile Communication | 全球移动通信系统 |
| GPRS | General Packet Radio Service | 通用无线分组业务 |
| TCP | Transport Control Protocol | 传输控制协议 |
| LBS | Location Based Services | 辅助定位服务 |
| IMEI | International Mobile Equipment Identity | 国际移动设备识别码 |
| MCC | Mobile Country Code | 移动用户所属国家代号 |
| MNC | Mobile Network Code | 移动网号码 |
| LAC | Location Area Code | 位置区码 |
| Cell ID | Cell Tower ID | 移动基站 |
| UDP | User Datagram Protocol | 用户数据报协议 |
| SOS | Save Our Ship/Save Our Souls | 遇难求救信号 |
| CRC | Cyclic Redundancy Check | 循环冗余校验 |
| NITZ | Network Identity and Time Zone, | 时区 |
| GIS | Geographic Information System | 地理信息系统 |

1. **Basic Rules**

**1.** If a GPRS connection is established successfully, the terminal will send a first login message packet to the server, and the terminal can receive a data packet responded by the server within five seconds, which means the connection is considered to be a normal connection. Then the terminal will begin to send location information (i.e., GPS, LBS information package). A status information package will be sent by the terminal after three minutes to regularly confirm the connection.

**2.** If the GPRS connection is established unsuccessfully, the terminal will not be able to send the login message packet. The terminal will start schedule reboot in twenty minutes if the GPRS connection is failed three times. 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 turn off and the terminal will not restart, otherwise the terminal restarts automatically after 20 minutes.

**3.** After receiving the login message packet, the server will return a response data packet. If the terminal doesn’t receive packet from the server within five seconds after sending the login message packet or the status information package, the current connection is regarded as an abnormal connection. The terminal will start a retransmission function for GPS tracking data, which will cause the terminal to disconnect the current GPRS connection, rebuild a new GPRS connection and send a login message packet again.

**4.** If the connection is regarded to be abnormal, and the data packet as a response from the server is failed to be received three times after a connection has established and a login message packet or status information package has sent, the terminal will start schedule reboot and the scheduled time is ten minutes. Within ten minutes, if the terminal successfully connects to the server and receives the data packet responded by the server, the schedule reboot will be turned off and the terminal will not be rebooted. Otherwise, the terminal will be rebooted automatically in ten minutes.

1. After the normal connection, the terminal will send a combined information package of GPS and LBS to the server after the GPS information is changed, and the server may set a default protocol for transmission by using commands.
2. To ensure the effectiveness of the connection, the terminal will send status information to the server at regular intervals, and the server will return response data packets to confirm the connection.
3. For the terminal which doesn’t register an IMEI number, the server will reply the terminal with a login request response and heartbeat packet response, rather than directly disconnect the connection. (If the connection is directly disconnected or the server doesn’t reply to the terminal, it will lead to a continuous reconnected by the terminal and the GPRS traffic will be consumed heavily.)

**Data Flow Diagram**



1. **Data Packet Format**

The communication is transferred asynchronously in bytes.

The total length of packets is (10+N) Bytes.

|  |  |
| --- | --- |
| Format | Length(Byte) |
| Start Bit | 2 |
| Packet Length | 1 |
| Protocol Number | 1 |
| Information Content | N |
| Information Serial Number | 2 |
| Error Check | 2 |
| Stop Bit | 2 |

## 4.1. Start Bit

Fixed value in HEX 0x78 0x78.

## 4.2. Packet Length

Length = Protocol Number + Information Content + Information Serial Number + Error Check, totally (5+N)Bytes, because the Information Content is a variable length field.

## 4.3. Protocol Number

|  |  |
| --- | --- |
| Type | Value |
| Login Message | 0x01 |
| Location Data | 0x12 |
| Status information | 0x13 |
| String informatio | 0x15 |
| Alarm data | 0x16 |
| Command information sent by the server to the terminal | 0x80 |

## 4.4. Information Contents

The specific contents are determined by the protocol numbers corresponding to different applications.

## 4.5. Information Serial Number

The serial number of the first GPRS data (including status packets and GPS, LBS and other data packets) sent after booting is "00 01". After that the serial number of each data sent (including status packets and GPS, LBS data packets) will be automatically added ‘1’. When the serial number is "FF FF", reset the serial number to "00 01".

## 4.6. Error Check

A check code may be used by the terminal or the server to distinguish whether the received information is error or not. To prevent errors occur during data transmission, error check is added to against data misoperation, so as to increase the security and efficiency of the system. The check code is generated by the CRC-ITU checking method.

The check codes of data in the structure of the protocol, from the Packet Length to the Information Serial Number (including “Packet Length” and “Information Serial Number”) , are values of CRC-ITU. CRC error occur when the received information is calculated, the receiver will ignore and discard the data packet.

## 4.7. Stop Bit

Fixed value in HEX 0x0D 0x0A.

# Details about Data Packet sent by Server to Terminal

The commonly used information packages sent by the terminal and those sent by the server will be interpreted separately.

## 5.1. Login Message Packet

## 5.1.1. Terminal Sending Data Packet to Server

The login message packet is used to be sent to the server with the terminal ID so as to confirm the established connection is normal or not.

|  |  |  |  |
| --- | --- | --- | --- |
| Descriptions | | Bits | Examples |
| Login Message Packet(18 Byte) | Start Bit | 2 | 0x78 0x78 |
| Packet Length | 1 | 0x0D |
| Protocol Number | 1 | 0x01 |
| Terminal ID | 8 | 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45 |
| Information Serial Number | 2 | 0x00 0x01 |
| Error Check | 2 | 0x8C 0xDD |
| Stop Bit | 2 | 0x0D 0x0A |

#### 5.1.1.1. Start Bit

For details see Data Packet Format section 4.1.

#### 5.1.1.2. Packet Length

#### For details see Data Packet Format section 4.2.

#### 5.1.1.3. Protocol Number

For details see Data Packet Format section 4.3.

#### 5.1.1.4. Terminal ID

The terminal ID applies IMEI number of 15 bits.

Example: if the IMEI is 123456789012345,

the terminal ID is 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45.

#### 5.1.1.5. Information Serial Number

For details see Data Packet Format section 4.5.

#### 5.1.1.6. Error Check

For details see Data Packet Format section 4.6.

#### 5.1.1.7. Stop Bit

For details see Data Packet Format section 4.7.

### 5.1.2. **Server Responds the Data Packet**

|  |  |  |  |
| --- | --- | --- | --- |
| Description | | Bits | Examples |
| Login Message Packet (18 | Start Bit | 2 | 0x78 0x78 |
| Packet Length | 1 | 0x05 |
| Protocol Number | 1 | 0x01 |
| Information Serial Number | 2 | 0x00 0x01 |
| Error Check | 2 | 0XD9 0xDC |
| Stop Bit | 2 | 0x0D 0x0A |

The response packet from the server to the terminal: the protocol number in the response packet is identical to the protocol number in the data packet sent by the terminal.

#### 5.1.2.1. Start Bit

For details see Data Packet Format section 4.1.

#### 5.1.2.2. Packet Length

For details see Data Packet Format section 4.2.

#### 5.1.2.3. Protocol Number

For details see Data Packet Format section 4.3.

#### 5.1.2.4. Information Serial Number

For details see Data Packet Format section 4.5

#### 5.1.2.5. Error Check

For details see Data Packet Format section 4.6

#### 5.1.2.6. Stop Bit

For details see Data Packet Format section 4.7

## 5.2. Location Data Packet (combined information package of GPS and LBS)

### 5.2.1. Terminal Sending Location Data Packet to Server

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Format | | | Bite（Byte） | Examples |
| Information Content | Start Bit | | 2 | 0x78 0x78 |
| Packet Length | | 1 | 0x24 |
| Protocol Number | | 1 | 0x12 |
| GPS Information | Date Time | 6 | 0x0B 0x08 0x1D 0x11 0x2E 0x10 |
| Quantity of GPS information satellites | 1 | 0xCF |
| Latitude | 4 | 0x02 0x7A 0xCF 0xEB |
| Longitude | 4 | 0xCC 0x46 0x58 0x49 |
| Speed | 1 | 0x00 |
| Course, Status | 2 | 0x14 0x8F |
| LBS Information | MCC | 2 | 0x01 0xCC |
| MNC | 1 | 0x00 |
| LAC | 2 | 0x28 0x7D |
| Cell ID | 3 | 0x00 0x1F 0xB8 |
| Status Information | Teminal Information Details | 1 | 0x00 |
| Power Voltage | 2 | 0x04 0xA1 |
| GSM Signal Intensity | 1 | 0x04 |
| Alarm expansion package | 1 | 0x01 |
| Expand Information | Oil Data | 2 | 0x01 0xF4 |
| Temperature Data | 1 | 0x16 |
| Mileage Data | 4 |  |
| Serial Number | | 2 | 0x00 0x03 |
| Error Check | | 2 |  |
| Stop Bit | | 2 | 0x0D 0x0A |

#### 5.2.1.1. Start Bit

For details see Data Packet Format section 4.1

#### 5.2.1.2. Packet Length

For details see Data Packet Format section 4.2

#### 5.2.1.3. Protocol Number

For details see Data Packet Format section 4.3

#### 5.2.1.4. Date Time

|  |  |  |
| --- | --- | --- |
| Format | Length(Byte) | Example |
| Year | 1 | 0x13 |
| Month | 1 | 0x08 |
| Day | 1 | 0x11 |
| Hour | 1 | 0x0F |
| Minute | 1 | 0x32 |
| Second | 1 | 0x17 |

For Example: 2010-03-23 15:50:23

Calculated as follows: 10(Decimal)=0A(Hexadecimal)

3 (Decimal)=03(Hexadecimal)

23(Decimal)=17(Hexadecimal)

15(Decimal)=0F(Hexadecimal)

50(Decimal)=32(Hexadecimal)

23(Decimal)=17(Hexadecimal)

Then the value is: 0x0A 0x03 0x17 0x0F 0x32 0x17

Upload as UTC time

#### 5.2.1.5. Length of GPS information, quantity of positioning satellites

The field is 1 Byte displayed by two hex digits, wherein the first one is for the length of GPS information and the second one for the number of the satellites join in positioning.

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

(C = 12Bit Lenght ，B = 11 satellites)

#### 5.2.1.6. Latitude

Four bytes are consumed, defining the latitude value of location data. The range of the value is 0-162000000, indicating a range of 0°-90°. The conversion method thereof is as follow:

Converting the value of latitude and longitude output by GPS module into a decimal based on minute; multiplying the converted decimal by 30000; and converting the multiplied result into hexadecimal.

Example: 22º32.7658’=(22X60+32.7658)X3000=40582974, then converted into a hexadecimal number

40582974(Decimal)= 26B3F3E(Hexadecimal)

At last the value is 0x02 0x6B 0x3F 0x3E.

#### 5.2.1.7. Longitude

Four bytes are consumed, defining the longitude value of location data. The range of the value is 0-324000000, indicating a range of 0°-180°.

The conversion method herein is same to the method mentioned in Latitude (see section 5.2.1.6).

#### 5.2.1.8. Speed

One byte is consumed, defining the running Speed of GPS. The value ranges from 0x00 to 0xFF indicating a range from 0 to 225km/h.

e.g. 0x00 represents 0 km/h.

0x10 represents 16km/h.

0xFF represents 255 km/h.

#### 5.2.1.9. Course Status

Occupies 2 bytes and converts to binary. The first six digits of the first byte indicate the positioning status of the device, its last two digits and the eight digits of the second byte indicate the direction of the GPS, indicating the range of 0 to 360, unit: Degrees, measured clockwise from north of 0°.

|  |  |  |
| --- | --- | --- |
| BYTE\_1 | Bit7 | Reserved |
| Bit6 | Reserved |
| Bit5 | GPS real-time/differential positioning |
| Bit4 | GPS having been positioning or not |
| Bit3 | East Longitude, West Longitude |
| Bit2 | South Latitude, North Latitude |
| Bit1 | Course |
| Bit0 |
| BYTE\_2 | Bit7 |
| Bit6 |
| Bit5 |
| Bit4 |
| Bit3 |
| Bit2 |
| Bit1 |
| Bit0 |

Note: The status information in the data packet is the status corresponding to the time bit recorded in the data packet.

For example: the value is 0xD5 0x4C，the corresponding binary is 11010101 01001100，

BYTE\_1 Bit7 1（Reserved bit, default is 1）

BYTE\_1 Bit6 1（Reserved bit, default is 1）

BYTE\_1 Bit5 0 （(real time GPS）——0：(real time GPS，1：differential positioning

BYTE\_1 Bit4 1 （GPS Already positioned——0：GPS Not position，1：GPS Already positioned

BYTE\_1 Bit3 0 （East longitude）——0：East longitude，1：West longitude

BYTE\_1 Bit2 1 （north latitude）——0：South latitude，1：Noth latitude

BYTE\_1 Bit1 0

BYTE\_1 Bit0 1

BYTE\_2 Bit7 0

BYTE\_2 Bit7 1

BYTE\_2 Bit7 0 Course 332° (0101001100 in Binary, or 332 in decimal)

BYTE\_2 Bit7 0

BYTE\_2 Bit7 1

BYTE\_2 Bit7 1

BYTE\_2 Bit7 0

BYTE\_2 Bit7 0

That means ACC OFF, GPS is positioned, real-time GPS, north latitude, east longitude, and the course is 332°.

#### 5.2.1.10. MCC

The country code to which a mobile user belongs, i.e., Mobile Country Code(MCC).

Example: Chinese MCC is 460 in decimal, or 0x01 0xCC in Hex (that is, a decimal value of 460 converting into a hexadecimal value, and 0 is added at the left side because the converted hexadecimal value is less than four digits).

Herein the range is 0x0000 ~ 0x03E7.

#### 5.2.1.11. MNC

Mobile Network Code(MNC)

Example: Chinese MNC is 0x00.

#### 5.2.1.12. LAC

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

#### 5.2.1.13. Cell ID

Cell Tower ID (Cell ID), which value ranges from 0x000000 to 0xFFFFFF.

#### 5.2.1.14. Terminal Information

Occupy 1 byte, converted to binary, used to represent the status information of the terminal

|  |  |  |
| --- | --- | --- |
| Bit | | Code Meaning |
| BYTE | Bit7 | 1：Oil cut off |
| 0：Oil connect |
| Bit6 | 1：GPS positioned |
| 0：GPS not position |
| Bit5-Bit3 | 111：Overspeed alarm |
| 110：Enter GEO fence alarm |
| 101：Displacement alarm |
| 100：SOS alarm |
| 011：Low power alarm |
| 010：Power off alarm |
| 001：Vibration alarm |
| 000：Normal |
| Bit2 | 1：connected outside power |
| 0：Not connect outside power |
| Bit1 | 1：ACC ON |
| 0：ACC OFF |
| Bit0 | 1：Set up defence |
| 0：Withdraw defence |

Bit5-Bit3 The alarm bit is only valid in the alarm package, not in the positioning package and heartbeat package.

#### 5.2.1.15. Voltage

The voltage value multiplied by 100 and convert to hexadecimal.

For example, 11.85V, multiplied by 100 equals is 1185, converted to hexadecimal is 0x04 0xA1

Note: If the power bit (bit2) of the terminal information of 5.2.1.14 is 0, it means that when the power is off, the platform will assign the voltage value to 0.

#### 5.2.1.16. GSM [Signal](D:/%E4%B8%8B%E8%BD%BD%E8%BD%AF%E4%BB%B6/Youdao/Dict/8.9.3.0/resultui/html/index.html" \l "/javascript:;) [Intensity](D:/%E4%B8%8B%E8%BD%BD%E8%BD%AF%E4%BB%B6/Youdao/Dict/8.9.3.0/resultui/html/index.html" \l "/javascript:;)

0x00：no signal；

0x01：signal is very weak;

0x02：signal is weak;

0x03：signal is good;

0x04：signal is very strong.

For example：0x03 GSM signal is good

#### 5.2.1.17. Alarm Extension Bit

|  |  |
| --- | --- |
| Byte | Code Meaning |
| 00 | Normal |
| 01 | Reserved |
| 02 | ACC OFF alarm |
| 03 | Oil alarm |
| 04 | ACC ON alarm |
| 05 | Out of fence alarm |
| 06 | Light alarm |
| 07 | High temperature alarm |
| 08 | Loe temperature alarm |
| … | Reserved |

The alarm extension is only valid in the alarm packet, not in the positioning packet and the heartbeat packet.

#### 5.2.1.18. Oil Data

Oil sensor resistance value, hexadecimal, unit is Ohm, occupy 2 bytes

For example, 2000Ω is 0x07 0xD0

#### 5.2.1.19. Temperature Data

Temperature value, hexadecimal, the first bit is the sign bit, 1 is negative, the unit is Celsius, occupy 1 byte.

For example, 23 degrees is 0x17; -23 degrees is 0x97

#### 5.2.1.20. Mileage Data

Mileage value, hexadecimal, converted to decimal and divided by 100, the unit is Km, occupying 4 bytes.

Such as 231231.34Km: 231231.34\*100 🡪 0x01 0x60 0xD4 0xBE

#### 5.2.1.18. Information Serial Number

For details see Data Packet Format section 4.5

#### 5.2.1.19. Error Check

For details see Data Packet Format section 4.6

#### 5.2.1.20. Stop Bit

For details see Data Packet Format section 4.7

## 5.3. Heartbeat Packet

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

### 5.3.1. Terminal Sending Heartbeat Packet to Server

|  |  |  |  |
| --- | --- | --- | --- |
| Format | | | Length（Byte） |
| Information Content | Start Bit | | 2 |
| Packet Length | | 1 |
| Protocol Number | | 1 |
| Status Information | Terminal Information Content | 1 |
| Voltage Level | 2 |
| GSM Signal Strength | 1 |
| Alarm Expansion Pack | 1 |
| Serial Number | | 2 |
| Error Check | | 2 |
| Stop Bit | | 2 |

#### 5.3.1.1. Start Bit

For details see Data Packet Format section 4.1

#### 5.3.1.2. Packet Length

For details see Data Packet Format section 4.2

#### 5.3.1.3. Protocol Number

For details see Data Packet Format section 4.3

#### 5.3.1.4. Terminal Information

Occupy 1 byte, converted to binary, used to represent the status information of the terminal

|  |  |  |
| --- | --- | --- |
| Bit | | Code Meaning |
| BYTE | Bit7 | 1：Oil cut off |
| 0：Oil connect |
| Bit6 | 1：GPS positioned |
| 0：GPS not position |
| Bit3-Bit5 | 111：Overspeed alarm |
| 110：Out of GEO fence alarm |
| 101：Overspeed alarm |
| 100：SOS alarm |
| 011：Low power alarm |
| 010：Power off alarm |
| 001：Vibration alarm |
| 000：Normal |
| Bit2 | 1：Connected outside power |
| 0：Not connect outside power |
| Bit1 | 1：ACC ON |
| 0：ACC OFF |
| Bit0 | 1：Set up defence |
| 0：Withdraw defence |

Bit5-Bit3 alarm bit is only valid in the alarm package, not in the positioning package and heartbeat package

#### 5.3.1.5. Voltage Level

Multiply the voltage value by 100 and convert to hexadecimal.

For example, 11.85V, multiplied by 100 equals 1185, converted to hexadecimal is 0x04 0xA1

#### 5.3.1.6. GSM Signal Strength

0x00：no signal；

0x01：signal is very weak;

0x02：signal is weak;

0x03：signal is good;

0x04：signal is very strong.

For example：0x03 GSM signal is good

#### 5.3.1.7. Alarm Expansion Pack

|  |  |
| --- | --- |
| Byte | Code Meaning |
| 00 | Normal |
| 01 | Reserved |
| 02 | ACC OFF alarm |
| 03 | Oil alarm |
| 04 | ACC ON alarm |
| 05 | Out of GEO fence alarm |
| 06 | Light alarm |
| 07 | High temperature alarm |
| 08 | Low temperature alarm |
| … | Reserved |

The alarm extension bit is only valid in the alarm package, not in the positioning package and heartbeat package

#### 5.3.1.8. Serial Number

For details see Data Packet Format section 4.5

#### 5.3.1.9. Error Check

For details see Data Packet Format section 4.6

#### 5.3.1.10. Stop Bit

For details see Data Packet Format section 4.7

### 5.3.2. Server Response Packet

For details see Data Packet Format section 5.1.2

## 5.4. Alarm Packet (GPS, LBS, Combined Status Information Packet)

### 5.4.1. Terminal Sending Alarm Packet to Server

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Format | | | Length（Byte） | Examples |
| Information  Content | Start Bit | | 2 | 0x78 0x78 |
| Packet Length | | 1 | 0x24 |
| Protocol Number | | 1 | 0x16 |
| GPS Information | Date Time | 6 | 0x0B 0x08 0x1D 0x11 0x2E 0x10 |
| Quantity of GPS information satellites | 1 | 0xCF |
| Latitude | 4 | 0x02 0x7A 0xCF 0xEB |
| Longitude | 4 | 0xCC 0x46 0x58 0x49 |
| Speed | 1 | 0x00 |
| Course, Status | 2 | 0x14 0x8F |
| LBS Information | MCC | 2 | 0x01 0xCC |
| MNC | 1 | 0x00 |
| LAC | 2 | 0x28 0x7D |
| Cell ID | 3 | 0x00 0x1F 0xB8 |
| Status Information | erminal Information Content | 1 | 0x00 |
| Voltage Level | 2 | 0x04 0xA1 |
| GSM Signal Strength | 1 | 0x04 |
| Alarm expansion packet | 1 | 0x01 |
| Expansion Information | Oil data | 2 | 0x01 0xF4 |
| Temperature data | 1 | 0x16 |
| Mileage data | 4 |  |
| Serial number | | 2 | 0x00 0x03 |
| Error check | | 2 |  |
| Stop Bit | | 2 | 0x0D 0x0A |

The protocol number is 16, please refer to the data package 5.2 for other information formats.

### 5.4.2. Server Sending the Alarm Packet Reply to Terminal

Details package can be referenced to 5.1.2

# Server sending Data Packet to Terminal

## 

## 6.1. Packet Sent by Server

|  |  |  |
| --- | --- | --- |
| Format | | Length（Byte） |
| Start Bit | | 2 |
| Packet Length | | 1 |
| Protocol Number | | 1 |
| Information Content | Command Length | 1 |
| Server Flag Bit | 4 |
| Command Details | M |
| Information Serial number | | 2 |
| Error Check | | 2 |
| Stop Bit | | 2 |

### 6.1.1. Start Bit

Packet format can be referenced to 4.1

### 6.1.2. Packet Length

Packet format can be referenced to 4.2

### 6.1.3. Protocol Number

Terminal sends protocol number using 0x80

### 6.1.4. Command Length

Server flag bit and command length

For example: in the unit of byte length, 0x0A, it means that the instruction content occupies 10 bytes

### 6.1.5. Server Flag Bit

It is reserved for server identification, and the terminal will return the received data binary in the return packet.

### 6.1.6. Command Details

Expressed as a string ASCⅡ, the command content is compatible with SMS commands.

### 6.1.7. Information Serial Number

Packet format can be referenced to 4.5

### 6.1.8. Error Check

Packet format can be referenced to 4.6

### 6.1.9. Stop Bit

Packet format can be referenced to 4.7

## 6.2. Packet Replied by Termina

|  |  |  |
| --- | --- | --- |
| Format | | Length（Byte） |
| Start Bit | | 2 |
| Packet Length | | 1 |
| Protocol Number | | 1 |
| Information  Content | Command Length | 1 |
| Server Flag Bit | 4 |
| Command Content | M |
| Information Serial number | | 2 |
| Error Check | | 2 |
| Stop Bit | | 2 |

### 6.2.1. Start Bit

Packet format can be referenced to 4.1

### 6.2.2. Packet Length

Packet format can be referenced to 4.2

### 6.2.3. Protocol Number

The terminal responds to the command sent by the server. The format of the data packet is consistent with the format of the “command sent by the server to the terminal”.

The protocol number is different, using 0x15.

### 6.2.4. Command Length

Server flag bit and command length

For example: in the unit of byte length, 0x0A, it means that the command content occupies 10 bytes.

### 6.2.5. Server Flag Bit

It is reserved for server identification, and the terminal will return the received data binary in the return packet.

### 6.2.6. Command Content

Expressed as a string ASCⅡ, the command content is compatible with SMS commands.

### 6.2.7. Information Serial number

Packet format can be referenced to 4.5

### 6.2.8. Error check

Packet format can be referenced to 4.6

### 6.2.9. Stop Bit

Packet format can be referenced to 4.7

# Attached A A U CRC-ITU look-up table algorithm C C language code fragment

CRC-ITU table lookup algorithm C language code fragment

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,

};

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

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; // negated

}