

**Huace New Protocol**

**Reference Manual**

**(Draft)**

Ver 1.0.1

2014.12.03



**About This Manual**

This manual aims to describe the new device communication protocol developed for the new generation of GNSS receiver platform settings by Shanghai HuaCe Navigation Technology Limited (CHC), which providing with more efficient communication transmission, better expansibility and stronger compatibility.

Currently, the new device communication protocol is suit for the X10 Series, i80 Series and N72 Series GNSS receivers of CHC.

**Change Log**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Modify** | **Note** |
| **2014-12-03** | **1.0.1** | **First Draft** | **2014-12-03** |

**Contents**

[1. Overview 1](#_Toc405380623)

[2. Transport Layer Protocol 1](#_Toc405380624)

[3. Business Layer Protocol 1](#_Toc405380625)

[4. Predefined Data Type 2](#_Toc405380626)

[5. Business Layer Command Explanation 3](#_Toc405380627)

[5.1. Command (cmd block) 3](#_Toc405380628)

[5.2. Command Set (cmd block list) 3](#_Toc405380629)

[5.3. Command ID and Command Type 3](#_Toc405380630)

[5.4. Main Command ID 4](#_Toc405380631)

[5.5. Reply Command ID 5](#_Toc405380632)

[5.6. COM Related Command 5](#_Toc405380633)

[5.6.1. HC\_CMD\_ID\_COM\_BAUD[=0x0103] 6](#_Toc405380634)

[5.6.2. HC\_COM\_PORT[=0x0104] 7](#_Toc405380635)

[5.6.3. HC\_CMD\_ID\_COM\_DIRECT[=0x0108] 8](#_Toc405380636)

[5.7. System Related Command 9](#_Toc405380637)

[5.7.1. HC\_CMD\_ID\_SYSTEM\_POWER\_OFF[=0x0303] 9](#_Toc405380638)

[5.7.2. HC\_CMD\_ID\_SYSTEM\_POWER\_STATUS[=0x0304] 10](#_Toc405380639)

[5.7.3. HC\_CMD\_ID\_SYSTEM\_REBOOT[=0x0305] 10](#_Toc405380640)

[5.7.4. HC\_CMD\_ID\_SYSTEM\_HARDWARE\_INFO[=0x0306] 11](#_Toc405380641)

[5.7.5. HC\_CMD\_ID\_SYSTEM\_FIRMWARE\_VERSION[=0x0307] 11](#_Toc405380642)

[5.7.6. HC\_CMD\_ID\_SYSTEM\_ADMIN\_CMD[=0x0308] 12](#_Toc405380643)

[5.7.7. HC\_CMD\_ID\_SYSTEM\_GSENSOR\_VALID[=0x0309] 12](#_Toc405380644)

[5.7.8. HC\_CMD\_ID\_SYSTEM\_REGISTER\_CODE[=0x030A] 12](#_Toc405380645)

[5.7.9. HC\_CMD\_ID\_SYSTEM\_REGISTER\_TIME[=0x030B] 12](#_Toc405380646)

[5.8. GNSS related command 13](#_Toc405380647)

[5.8.1. HC\_CMD\_ID\_GNSS\_FREQ[=0x0403] 15](#_Toc405380648)

[5.8.2. HC\_CMD\_ID\_GNSS\_IO\_ID[=0x0404] 17](#_Toc405380649)

[5.8.3. HC\_CMD\_ID\_GNSS\_DIFF\_TYPE[=0x0405] 19](#_Toc405380650)

[5.8.4. HC\_CMD\_ID\_GNSS\_DAT\_RAW[=0x0406] 20](#_Toc405380651)

[5.8.5. HC\_CMD\_ID\_GNSS\_DAT\_EPH[=0x0407] 20](#_Toc405380652)

[5.8.6. HC\_CMD\_ID\_GNSS\_DAT\_GGA[=0x0408] 21](#_Toc405380653)

[5.8.7. HC\_CMD\_ID\_GNSS\_DAT\_GSV[=0x0409] 22](#_Toc405380654)

[5.8.8. HC\_CMD\_ID\_GNSS\_DAT\_BLH[=0x040A] 23](#_Toc405380655)

[5.8.9. HC\_CMD\_ID\_GNSS\_DAT\_TIME[=0x040B] 24](#_Toc405380656)

[5.8.10. HC\_CMD\_ID\_GNSS\_DAT\_DOPS[=0x040C] 24](#_Toc405380657)

[5.8.11. HC\_CMD\_ID\_GNSS\_DAT\_SVTRACK[=0x040D] 25](#_Toc405380658)

[5.8.12. HC\_CMD\_ID\_GNSS\_DAT\_PRE[=0x040E] 28](#_Toc405380659)

[5.8.13. HC\_CMD\_ID\_GNSS\_DAT\_BASE\_POSITION[=0x040F] 28](#_Toc405380660)

[5.8.14. HC\_CMD\_ID\_GNSS\_DAT\_POSINFO[=0x0410] 29](#_Toc405380661)

[5.8.15. Hc\_cmd\_id\_gnss\_dat\_gsensor[=0x0411] 30](#_Toc405380662)

[5.8.16. Hc\_cmd\_id\_gnss\_dat\_gsa[=0x0412] 30](#_Toc405380663)

[5.8.17. HC\_CMD\_ID\_GNSS\_DAT\_GST[=0x0413] 31](#_Toc405380664)

[5.8.18. HC\_CMD\_ID\_GNSS\_DAT\_RMC[=0x0414] 32](#_Toc405380665)

[5.8.19. HC\_CMD\_ID\_GNSS\_DAT\_GLL[=0x0415] 33](#_Toc405380666)

[5.8.20. HC\_CMD\_ID\_GNSS\_DAT\_VTG[=0x0416] 34](#_Toc405380667)

[5.8.21. HC\_CMD\_ID\_GNSS\_DAT\_ZDA[=0x0417] 34](#_Toc405380668)

[5.8.22. HC\_CMD\_ID\_GNSS\_DAT\_GRS[=0x0418] 35](#_Toc405380669)

[5.8.23. HC\_CMD\_ID\_GNSS\_DAT\_PJK[=0x0419] 36](#_Toc405380670)

[5.8.24. HC\_CMD\_ID\_GNSS\_DAT\_PJT[=0x041a] 37](#_Toc405380671)

[5.8.25. HC\_CMD\_ID\_GNSS\_DAT\_MAGNETOMETER[=0x041b] 38](#_Toc405380672)

[5.8.26. HC\_CMD\_ID\_GNSS\_PDOP\_MASK[=0x0450] 38](#_Toc405380673)

[5.8.27. HC\_CMD\_ID\_GNSS\_ANGLE\_MASK[=0x0451] 38](#_Toc405380674)

[5.8.28. HC\_CMD\_ID\_GNSS\_BASE\_POSITION[=0x0452] 39](#_Toc405380675)

[5.8.29. HC\_CMD\_ID\_GNSS\_SEARCH\_RESET[=0x0453] 40](#_Toc405380676)

[5.8.30. HC\_CMD\_ID\_GNSS\_UNLOGALL[=0x0454] 40](#_Toc405380677)

[5.8.31. HC\_CMD\_ID\_GNSS\_SATELLITE\_ENABLE[=0x0455] 40](#_Toc405380678)

[5.8.32. HC\_CMD\_ID\_GNSS\_DATA\_CONFIG\_LIST[=0x0456] 40](#_Toc405380679)

[5.9. Radio related command 42](#_Toc405380680)

[5.9.1. HC\_CMD\_ID\_RADIO\_FREQ[=0x0703] 43](#_Toc405380681)

[5.9.2. HC\_CMD\_ID\_RADIO\_POWER[=0x0704] 43](#_Toc405380682)

[5.9.3. HC\_CMD\_ID\_RADIO\_STEPPER[=0x0709] 44](#_Toc405380683)

[5.9.4. HC\_CMD\_ID\_RADIO\_AIR\_BAUD[=0x070b] 44](#_Toc405380684)

[5.9.5. HC\_CMD\_ID\_RADIO\_PROTOCOL[=0x070c] 44](#_Toc405380685)

[5.9.6. HC\_CMD\_ID\_RADIO\_INFOLIST[=0x070D] 45](#_Toc405380686)

[5.9.7. HC\_CMD\_ID\_RADIO\_POWER\_ON[=0x070E] 45](#_Toc405380687)

[5.9.8. HC\_CMD\_ID\_RADIO\_POWER\_OFF[=0x070f] 45](#_Toc405380688)

[5.9.9. HC\_CMD\_ID\_RADIO\_AUTO\_POWER\_ON[=0x0710] 46](#_Toc405380689)

[5.9.10. HC\_CMD\_ID\_RADIO\_MODULE\_TYPE[=0x0711] 46](#_Toc405380690)

[5.9.11. HC\_CMD\_ID\_RADIO\_POWER\_STATUS[=0x0712] 47](#_Toc405380691)

[5.10. WIFI related commands 48](#_Toc405380692)

[5.10.1. HC\_CMD\_ID\_WIFI\_PARA\_AP[=0x0A03] 48](#_Toc405380693)

[5.10.2. HC\_CMD\_ID\_WIFI\_MODE[=0x0A04] 49](#_Toc405380694)

[5.10.3. HC\_CMD\_ID\_WIFI\_OFF[=0x0A05] 50](#_Toc405380695)

[5.10.4. HC\_CMD\_ID\_WIFI\_ON[=0x0A06] 50](#_Toc405380696)

[5.10.5. HC\_CMD\_ID\_WIFI\_AUTO\_OPEN[=0x0A07] 50](#_Toc405380697)

[5.10.6. HC\_CMD\_ID\_WIFI\_POWER\_STATUS[=0x0A08] 50](#_Toc405380698)

[5.11. File record related command 51](#_Toc405380699)

[5.11.1. HC\_CMD\_ID\_FILE\_RECORD\_ON[=0x0B03] 52](#_Toc405380700)

[5.11.2. HC\_CMD\_ID\_FILE\_RECORD\_OFF[=0x0B04] 52](#_Toc405380701)

[5.11.3. HC\_CMD\_ID\_FILE\_RECORD\_AUTO\_OPEN[=0x0b05] 52](#_Toc405380702)

[5.11.4. HC\_CMD\_ID\_FILE\_RECORD\_PARA[=0x0B06] 53](#_Toc405380703)

[5.11.5. HC\_CMD\_ID\_FILE\_RECORD\_STATUS[=0x0B07] 54](#_Toc405380704)

[5.11.6. HC\_CMD\_ID\_FILE\_RECORD\_PPK\_CMD[=0x0B08] 55](#_Toc405380705)

[5.12. Netlink related command 55](#_Toc405380706)

[5.12.1. HC\_CMD\_ID\_NETLINK\_INDEX[=0x1102] 56](#_Toc405380707)

[5.12.2. HC\_CMD\_ID\_NETLINK\_IP\_ADDR[=0x1103] 56](#_Toc405380708)

[5.12.3. HC\_CMD\_ID\_NETLINK\_SERVER\_TYPE[=0x1104] 56](#_Toc405380709)

[5.12.4. Hc\_cmd\_id\_netlink\_status[=0x1105] 57](#_Toc405380710)

[5.12.5. HC\_CMD\_ID\_NETLINK\_SOURCE\_LIST[=0x1106] 59](#_Toc405380711)

[5.12.6. HC\_CMD\_ID\_NETLINK\_OPEN[=0x1107] 60](#_Toc405380712)

[5.12.7. HC\_CMD\_ID\_NETLINK\_CLOSE[=0x1108] 60](#_Toc405380713)

[5.12.8. HC\_CMD\_ID\_NETLINK\_AUTO\_OPEN[=0x1109] 61](#_Toc405380714)

[5.12.9. HC\_CMD\_ID\_NETLINK\_DATA\_SOURCE[=0x110a] 61](#_Toc405380715)

[5.12.10. HC\_CMD\_ID\_NETLINK\_ACCOUNT[=0x110b] 61](#_Toc405380716)

[5.13. Mobile network related command 62](#_Toc405380717)

[5.13.1. HC\_CMD\_ID\_3G\_DIAL\_PARA[=0x1403] 62](#_Toc405380718)

[5.13.2. HC\_CMD\_ID\_3G\_DIAL\_STATUS[=0x1404] 63](#_Toc405380719)

[5.13.3. HC\_CMD\_ID\_3G\_DIAL\_ON[=0x1405] 65](#_Toc405380720)

[5.13.4. HC\_CMD\_ID\_3G\_DIAL\_OFF[=0x1406] 65](#_Toc405380721)

[5.13.5. HC\_CMD\_ID\_3G\_POWER\_ON[=0x1407] 65](#_Toc405380722)

[5.13.6. HC\_CMD\_ID\_3G\_POWER\_OFF[=0x1408] 66](#_Toc405380723)

[5.13.7. HC\_CMD\_ID\_3G\_AUTO\_DIAL[=0x1409] 66](#_Toc405380724)

[5.13.8. HC\_CMD\_ID\_3G\_AUTO\_POWER\_ON[=0x140a] 66](#_Toc405380725)

[5.13.9. Hc\_cmd\_id\_3g\_power\_status[=0x140b] 66](#_Toc405380726)

[5.13.10. HC\_CMD\_ID\_3G\_SIGNAL\_INTENSITY[=0x140C] 67](#_Toc405380727)

[5.13.11. HC\_CMD\_ID\_3G\_BAND\_MODE[=0x140D] 68](#_Toc405380728)

[5.14. Datalink related command 69](#_Toc405380729)

[5.14.1. HC\_CMD\_ID\_DATALINK\_MODE[=0x1503] 69](#_Toc405380730)

[5.14.2. HC\_CMD\_ID\_DATALINK\_DATA[=0x1504] 69](#_Toc405380731)

# Overview

This protocol applies to the data transmission and communication of i80 Series, N72 Series and newer model receivers. This protocol is achieved by two layers; the first layer is transport layer and the second one is business layer. The transport layer is responsible for entire transmission and control of underlying data bit stream, which providing security control for the continuity of data transmission. The business layer is responsible for the interpretation and encapsulation of transmission and control command content from a business perspective.

# Transport Layer Protocol

Data structure of first layer (transport layer) protocol:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| head | PID | count | len | data | tail |

Description of each field:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Count** | **Range** | **Description** |
| head | 2 | head[0]==’$’, head[1]==’$’ | The start character of transport packet. |
| PID | 1 | [1,250] | channel flag (temporarily reserved) |
| count | 1 | [0,250] | The values of the multiple consecutive packets must be continuous (The 250 and 0 are consecutive). |
| len | 1 | [1,250] | Used to indicate the data length. For the multiple consecutive packets, the value of each packet must be equal, except the last packet. |
| data | n | random length | The transported business data. |
| tail | 2 | tail[0]==0x0D, tail[1]==0x0A | The end mark of transport packet. |

The design of channel flag PID is temporarily reserved, which is used to realize multiple separate session channels in transport layer for the follow-up design.

The count identifier is used to guarantee the continuity of data transmission. That is to say, if the count values are recycling continuous from 0 to 250, the data of received transport layer packets is continuous; otherwise, the data is discontinuous.

# Business Layer Protocol

Data structure of second layer (business layer) protocol:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Head | cmd\_type | cmd\_sn | flag | Len | block\_list | Crc | Tail |

Description of each field:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Count** | **Range** | **Description** |
| Head | 2 | 0x04, 0x11 | start mark of the command |
| cmd\_type | 1 | [0,1] | request/respond |
| cmd\_sn | 4 | [0,0xFFFFFFFF] | Request mark (created by command requester and copied by command responder ) |
| flag | 4 | [1,1] | temporarily assigned as 1 |
| Len | 2 | [1,0xFF00] | Indicate the length of block\_list |
| block\_list | n | Struct (see explanation of “block” for details) | The list table of block |
| crc | 4 | [0,0xFFFFFFFF] | Check code (from cmd\_type to block\_list) |
| Tail | 2 | 0x09, 0x24 | end mark of the command |

For the protocol command type “cmd\_type”, the command packets transmitted by exterior to the receivers are set to 0, and the command packets transmitted by receivers are set to 1. The request mark “cmd\_sn” is used to match the received respond with the previous transmitted request. If the order of received respond cannot be the same as the order of transmitted request, this design is useful when multiple request configuration commands are sent continuously. The check code “crc” is used for checking data packet content of business layer. The check is neglected in this version and will be added in the follow-up version.

The length range of data transmitted by business layer is [1,0xFF00], namely, the maximal byte length of data packet is (0xFF00 + 19) bytes. However, maximal data length of transport layer is 250 bytes. So if the length of business layer data packet, which is in the process of transport layer data packet encapsulation, exceeds 250 bytes, the data packet needs to be subpackaged. The subpackage needs to guarantee the lengths of multiple consecutive packets are equal, except the last packet.

# Predefined Data Type

The predefined data types covered in this protocol:

|  |  |  |
| --- | --- | --- |
| **Predefined data type** | **Data type(C/C++)** | **Description** |
| HC\_BOOL | unsigned char | 0 false not 0 true |
| HC\_INT8 | char |  |
| HC\_PINT8 | char \* | A pointer to char |
| HC\_UINT8 | unsigned char |  |
| HC\_PUINT8 | unsigned char \* |  |
| HC\_INT16 | short |  |
| HC\_UINT16 | unsigned short |  |
| HC\_INT32 | int |  |
| HC\_UINT32 | unsigned int |  |

All the multi-byte basic data types (HC\_INT16, HC\_UINT16, HC\_INT32, HC\_UINT32, float and double) are written based on big endian storage mode during the collectively packing of protocols. That is to say, the high byte of data is stored at starting address (low address).

# Business Layer Command Explanation

## Command (cmd block)

Basic unit structure of business layer data content:

|  |  |  |
| --- | --- | --- |
| block\_id | len | value |

Explanation of block field:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Count** | **Range** | **Description** |
| block\_id | 2 | [0,0xFF00] | ID number of command |
| len | 2 | See explanation of “block\_id” for details | Length of command value |
| value | n |  | Command value, and its length is determined by len |

## Command Set (cmd block list)

The congregation of several commands is command set, which is the data content of business layer data packet and the complete description of the specific business layer content, whose structure is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Block 1 (Main)** | Block 2 | Block 3 | ... | Block n |

A complete command set (block list) is composed of a first command (main command), which indicates the type of command set content, and several commands with that type.

## Command ID and Command Type

The command ID corresponds to a piece of content of specific business area, and the command will partly or fully support the settings (SET), query (GET), reply (REPLY) and sending (SEND) of that content. From the point of view of specific business type, the command ID can be divided into several types, as shown in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Name of command module** | **Range** | **Description** |
| 1 | Command Information (HC\_CMD) | [1,1] | Describe the command content |
| 2 | COM (HC\_COM) | [0x0101,0x01FE] | COM communication related configuration command |
| 3 | System module(HC\_SYS) | [0x0301,0x03FE] | System related configuration command |
| 4 | GNSS module (HC\_GNSS) | [0x0401,0x05FE] | GNSS related configuration and output command |
| 5 | Radio module (HC\_RADIO) | [0x0701,0x07FE] | Radio related configuration command |
| 6 | WIFI module (HC\_WIFI) | [0x0A01,0x0AFE] | WIFI related configuration command |
| 7 | File module (HC\_FILE) | [0x0B01,0x0BFE] | System file configuration command |
| 8 | Network link (HC\_NETLINK) | [0x1101,0x11FE] | Network link related configuration command |
| 9 | MODEM module (HC\_MODEM) | [0x1401,0x14FE] | Mobile network module (2G/3G/4G) related configuration command |
| 10 | Datalink (HC\_DATALINK) | [0x1501,0x15FE] | Datalink related configuration command |

## Main Command ID

Main command ID is the ID of first command (main command) in the command set. The value of main command designates actions of follow-up command.

|  |  |  |  |
| --- | --- | --- | --- |
| Block\_id | Len | Value | Description |
| HC\_CMD\_ID\_CMD  [=0x0001] | 2 | cmd\_value : unsigned short | Behavior type of command |

There are three types of command: SET command, GET command and SEND command. SET command is used to configure parameters or send some configuration start action, such as configure Baudrate parameters of COM port, send command to turn off receiver, etc. GET command is used to request the acquisition of configuration parameters or status information. SEND command is the command content that sent by receiver at a certain frequency, like outputting position information or satellites status information, etc.

Value description

The definition of cmd\_value’s valid value is shown in the following table:

|  |  |
| --- | --- |
| **Cmd\_value** | **Description** |
| HC\_CMD\_COM\_SET [=0x0002] | Define settings behavior of COM communication related command |
| HC\_CMD\_COM\_GET [=0x0003] | Define query behavior of COM communication related command |
| HC\_CMD\_SYSTEM\_SET [=0x000A] | Define settings behavior of system related command |
| HC\_CMD\_SYSTEM\_GET [=0x000B] | Define query behavior of system related command |
| HC\_CMD\_GNSS\_SET [=0x000E] | Define settings behavior of GNSS related command |
| HC\_CMD\_GNSS\_GET [=0x000F] | Define query behavior of GNSS related command |
| HC\_CMD\_GNSS\_SEND [=0x0010] | Define initiative sending behavior of GNSS related data packet |
| HC\_CMD\_RADIO\_SET [=0x0016] | Define settings behavior of radio related command |
| HC\_CMD\_RADIO\_GET [=0x0017] | Define query behavior of radio related command |
| HC\_CMD\_WIFI\_SET [=0x0022] | Define settings behavior of WIFI related command |
| HC\_CMD\_WIFI\_GET [=0x0023] | Define query behavior of WIFI related command |
| HC\_CMD\_FILE\_SET [=0x0026] | Define settings behavior of file related command |
| HC\_CMD\_FILE\_GET [=0x0027] | Define query behavior of file related command |
| HC\_CMD\_3G\_SET [=0x002A] | Define settings behavior of mobile network module related command |
| HC\_CMD\_3G\_GET [=0x002B] | Define query behavior of mobile network module related command |
| HC\_CMD\_NETLINK\_SET [=0x002E] | Define settings behavior of network link related command |
| HC\_CMD\_NETLINK\_GET [=0x002F] | Define query behavior of network link related command |
| HC\_CMD\_DATALINK\_SET [=0x0032] | Define settings behavior of datalink related command |
| HC\_CMD\_DATALINK\_GET [=0x0033] | Define query behavior of datalink related command |

This command must be the first one of the command set, and the behavior of subsequent command action should be the same as that described by first command, that is to say, if the first command is SET command, the behavior of subsequent command action should be SET.

## Reply Command ID

The receiver will have unified replies to all the SET commands, and the reply command is defined as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Block\_id | Len | Value | Description |
| HC\_CMD\_ID\_CMD\_RESPONSE  [=0x0002] | 2 | response\_value : unsigned short | Completion status of SET command action: success, failure, overtime, command error |

Value description

The definition of cmd\_value’s valid value is shown in the following table:

|  |  |
| --- | --- |
| **Cmd\_value** | **Description** |
| HC\_RESPONSE\_STATUS\_OK[=1] | SET success |
| HC\_RESPONSE\_STATUS\_ERR [=2] | SET failure |
| HC\_RESPONSE\_STATUS\_OUTTIME[=3] | The response to command is overtime |
| HC\_RESPONSE\_STATUS\_CHECK\_ERR[=4] | Parameter check error |

## COM Related Command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Block\_id | Value len | | | | Description |
| REPLY | SET | GET | SEND |
| 1 | HC\_CMD\_ID\_COM\_BAUD  [=0x0103] | 2 | 2 | 0 | \_ | COM communication Baudrate |
| 2 | HC\_COM\_PORT  [=0x0104] | 2 | 2 | 2 | \_ | COM port number |
| 3 | HC\_CMD\_ID\_COM\_DIRECT  [=0x0108] | \_ | 2 | \_ | \_ | Communication pass-through settings |

The REPLY, SET, GET and SEND sub column of value len column in above table represent the value byte length defined by corresponding command when executes the REPLY (reply), SET (settings), GET (query) or SEND (initiatively sending) behavior respectively. The symbol “\_” indicates that the command does not support the behavior. The REPLY behavior is the reply to GET command by receiver. The descriptions in this chapter can apply to the follow-up chapters as well.

### HC\_CMD\_ID\_COM\_BAUD[=0x0103]

COM communication port Baudrate settings and query command is used to query and set the communication Baudrate of designated port, which cannot be used only and need to be used with HC\_COM\_PORT command together.

**[SET] & [REPLY] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Len** | **Description** |
| baud | HC\_UINT16 | 2 | Predefined enumeration value of Baudrate, please see the definition of HC\_COM\_BAUD\_ENUM |

Predefined enumeration value of Baudrate:

typedef enum

{

HC\_COM\_BAUD\_START = 0 ,

HC\_COM\_BAUD\_2400 =1 , // 2400

HC\_COM\_BAUD\_4800 =2 , // 4800

HC\_COM\_BAUD\_9600 =3 , // 9600

HC\_COM\_BAUD\_14400 =4 , // 14400

HC\_COM\_BAUD\_19200 =5 , // 19200

HC\_COM\_BAUD\_38400 =6 , // 38400

HC\_COM\_BAUD\_56K =7 , // 56000

HC\_COM\_BAUD\_57600 =8 , // 57600

HC\_COM\_BAUD\_115200 =9 , // 115200

HC\_COM\_BAUD\_128K =10 , // 128000

HC\_COM\_BAUD\_256K =11 , // 256000

HC\_COM\_BAUD\_512K =12 , // 512000

HC\_COM\_BAUD\_END

} HC\_COM\_BAUD\_ENUM;

### HC\_COM\_PORT[=0x0104]

Designate COM communication port command is used to query and set the related information of designated port, which cannot be used only and can be used with COM port configuration and query related command (such as HC\_CMD\_ID\_COM\_BAUD).

**[SET] & [GET] & [REPLY] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Len** | **Description** |
| port | HC\_UINT16 | 2 | Predefined enumeration value of COM port, please see the definition of HC\_COM\_PORT\_ENUM |

Predefined enumeration value of Baudrate:

typedef enum

{

HC\_COM\_PORT\_START = 0 ,

HC\_COM\_PORT\_1 =1 , // COM1

HC\_COM\_PORT\_2 =2 , // COM2

HC\_COM\_PORT\_3 =3, // COM3

HC\_COM\_PORT\_END

} HC\_COM\_PORT\_ENUM;

For example, the structure of corresponding block\_list when query the Baudrate of COM1 is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Block\_id** | **Len** | **Value** | **Description** |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_COM\_GET | COM information query |
| Block 2 | HC\_COM\_PORT | 2 | HC\_COM\_PORT\_1 | COM1 |
| Block 3 | HC\_COM\_BAUD | 0 | — | Query behavior |

The structure of block\_list in the reply data packet received by receiver is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Block\_id** | **Len** | **Value** | **Description** |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_COM\_GET | COM information query |
| Block 2 | HC\_COM\_PORT | 2 | HC\_COM\_PORT\_1 | COM1 |
| Block 3 | HC\_COM\_BAUD | 2 | HC\_COM\_BAUD\_9600 | Return the query value |

For example, the structure of corresponding block\_list when the Baudrate of COM1 is 38400 is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_COM\_SET | COM information settings |
| Block 2 | HC\_COM\_PORT | 2 | HC\_COM\_PORT\_1 | COM1 |
| Block 3 | HC\_COM\_BAUD | 2 | HC\_COM\_BAUD\_38400 | set as 38400 |

The reply to the execution result of SET command by receiver is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Block\_id** | **Len** | **Value** | **Description** |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_COM\_SET | COM information settings |
| Block 2 | HC\_CMD\_ID\_CMD\_RESPONSE | 2 | HC\_RESPONSE\_STATUS\_OK | Set successfully |

Note: The parametric commands, such as HC\_COM\_PORT, have no clear meaning if being used alone and should be used with relevant commands together to act as the defining parameters of query (GET) or settings (GET) command set.

### HC\_CMD\_ID\_COM\_DIRECT[=0x0108]

The settings of COM port communication pass-through connection can pass-through to GNSS board, internal radio and MCU. This command is mainly used for development debugging and pass-through to MCU by default.

**[SET] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| direct\_to | HC\_UINT16 | 2 | Predefined enumeration value of port pass-through, please see the definition of HC\_COM\_ DIRECT \_ENUM |

Predefined enumeration value of COM pass-through:

typedef enum

{

HC\_COM\_DIRECT\_SET\_START = 0 ,

HC\_COM\_DIRECT\_SET\_TO\_GNSS =1 , // pass-through to GNSS board

HC\_COM\_DIRECT\_SET\_TO\_RADIO =2 , // pass-through to internal radio

HC\_COM\_DIRECT\_SET\_TO\_MCU =3 , // pass-through to MCU

HC\_COM\_DIRECT\_SET\_END

} HC\_COM\_DIRECT\_ENUM;

## System Related Command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Block\_id | Value len | | | | Description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_SYSTEM\_POWER\_OFF  [=0x0303] | \_ | 0 | \_ | \_ | System power OFF |
| 2 | HC\_CMD\_ID\_SYSTEM\_POWER\_STATUS  [=0x0304] | 12 | \_ | 0 | \_ | System power status |
| 3 | HC\_CMD\_ID\_SYSTEM\_REBOOT  [=0x0305] | \_ | 0 | \_ | \_ | System reboot |
| 4 | HC\_CMD\_ID\_SYSTEM\_HARDWARE\_INFO  [=0x0306] | 36 | \_ | 0 | \_ | Hardware configuration information |
| 5 | HC\_CMD\_ID\_SYSTEM\_FIRMWARE\_VERSION  [=0x0307] | 100 | \_ | 0 | \_ | Firmware version |
| 6 | HC\_CMD\_ID\_SYSTEM\_ADMIN\_CMD  [=0x0308] | n | 2 | \_ | \_ | Reserved |
| 7 | HC\_CMD\_ID\_SYSTEM\_GSENSOR\_VALID  [=0x0309] | \_ | 3 | \_ | \_ | Compass and electric bubble calibration (Reserved) |
| 8 | HC\_CMD\_ID\_SYSTEM\_REGISTER\_CODE  [=0x030A] | 20 | 20 | 0 | \_ | System register code information |
| 9 | HC\_CMD\_ID\_SYSTEM\_REGISTER\_TIME  [=0x030B] | 5 | \_ | 0 | \_ | Register time |

### HC\_CMD\_ID\_SYSTEM\_POWER\_OFF[=0x0303]

Receiver system power OFF command, after the sending of this command, the receiver will be powered OFF immediately. This command only supports the SET behavior, the value is null. The structure of command set block\_list is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_COM\_SET | COM information setting |
| Block 2 | HC\_CMD\_ID\_SYSTEM\_POWER\_OFF | 0 | — | System power OFF |

### HC\_CMD\_ID\_SYSTEM\_POWER\_STATUS[=0x0304]

Receiver battery capacity query command, the value is NULL during the query. The length of value of the data packets returned by receiver is 12.

**[REPLY] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| power\_mode | HC\_UINT32 | 4 | 1 = Supplied by internal battery; 2 = Supplied by external battery |
| volt\_batt1 | float | 4 | The capacity of battery 1 (0-100), 100 indicates fully charged |
| volt\_batt2 | float | 4 | The capacity of battery 2 (0-100), 100 indicates fully charged |

typedef struct

{

HC\_UINT32 power\_mod;

float volt\_batt1;

float volt\_batt2;

} HC\_SYSTEM\_POWER\_STATUS\_STRUCT;

Query the receiver battery capacity, and the structure of command set block\_list is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_COM\_SET | COM information settings |
| Block 2 | HC\_CMD\_ID\_SYSTEM\_POWER\_STATUS | 0 | — | Query battery capacity |

### HC\_CMD\_ID\_SYSTEM\_REBOOT[=0x0305]

Receiver system reboot command, after the sending of this command, the receiver will be rebooted. This command only supports the SET behavior, the value is null. The structure of command set block\_list is shown as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_COM\_SET | COM information settings |
| Block 2 | HC\_CMD\_ID\_SYSTEM\_REBOOT | 0 | \_ | System reboot |

### HC\_CMD\_ID\_SYSTEM\_HARDWARE\_INFO[=0x0306]

Receiver system hardware configuration information query command, obtaining the receiver type, SN number, PN number, internal radio module type, internal wireless network module type, Bluetooth module type, etc.

**[REPLY] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| machine\_type | HC\_UINT32 | 4 | Device type identifier ID (have no specific definition for the moment) |
| sn | HC\_INT8[] | 7 | Device SN number |
| pn | HC\_INT8[] | 13 | Device PN number |
| radio\_type | HC\_UINT32 | 4 | Radio module type identifier (have no specific definition for the moment) |
| gprs\_type | HC\_UINT32 | 4 | Wireless network module type identifier (have no specific definition for the moment) |
| bt\_type | HC\_UINT32 | 4 | Bluetooth module type identifier (have no specific definition for the moment) |

typedef struct

{

HC\_UINT32 machine\_type;

HC\_INT8 sn[7];

HC\_INT8 pn[13];

HC\_UINT32 radio\_type;

HC\_UINT32 gprs\_type;

HC\_UINT32 bt\_type;

} HC\_SYSTEM\_HARDWARE\_INFO\_STRUCT;

### HC\_CMD\_ID\_SYSTEM\_FIRMWARE\_VERSION[=0x0307]

Receiver system firmware version information query command. It is used to query firmware version, firmware release date and internal version definition, etc.

**[REPLY] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| num | HC\_INT8[] | 50 | Firmware release version number |
| date | HC\_INT8[] | 50 | Firmware release date information |

typedef struct

{

HC\_UINT8 num[50];

HC\_UINT8 date[50];

} HC\_SYSTEM\_FIRMWARE\_VERSION\_STRUCT;

### HC\_CMD\_ID\_SYSTEM\_ADMIN\_CMD[=0x0308]

This command is reserved for the use of system development and debugging.

### HC\_CMD\_ID\_SYSTEM\_GSENSOR\_VALID[=0x0309]

This command is reserved for the calibration process control of electric bubble and compass internal integrated by the receiver, which is not supported by this version.

### HC\_CMD\_ID\_SYSTEM\_REGISTER\_CODE[=0x030A]

This is the receiver register code query and receiver registration command. During the receiver registration, namely, setting the new register code, the internal system will check whether the register code is successful or not, and then give the check response.

**[SET] & [REPLY] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| reg\_code | HC\_INT8[] | 20 | Receiver register code |

typedef struct

{

HC\_UINT8 reg\_code[20];

} HC\_REGISTER\_CODE\_STRUCT;

### HC\_CMD\_ID\_SYSTEM\_REGISTER\_TIME[=0x030B]

This is the receiver registration expire date query command, which only supports query (GET), but not supports settings (SET), its value will be updated and saved after the settings of new register code.

**[REPLY] value structure definition:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| expire | HC\_UINT8 | 1 | Indicate whether the register code is expire; 0 indicates unexpired, non 0 indicates expired |
| year | HC\_UINT16 | 2 | Valid year of registration |
| month | HC\_UINT8 | 1 | Valid month of registration |
| day | HC\_UINT8 | 1 | Valid day of registration |

typedef struct

{

HC\_UINT8 expire;

HC\_UINT16 year;

HC\_UINT8 month;

HC\_UINT8 day;

} HC\_REGISTER\_TIME\_STRUCT;

The receiver will judge the whether the registration is expired or not after it has been powered ON and searched the GPS time, before then the query returns 0 (unexpired) by default. If the registration is expired, all the SEND behavior will be stopped, which means there are no data sent out automatically and the network connection function is unavailable. The HC\_CMD\_ID\_SYSTEM\_REGISTER\_CODE command should be used to effectively register again for the normal usage of receiver.

## GNSS related command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order number | Block\_id | Value len | | | | Description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_GNSS\_FREQ  [=0x0403] | \_ | 2 | \_ | \_ | Data sending frequency |
| 2 | HC\_CMD\_ID\_GNSS\_IO\_ID  [=0x0404] | 2 | 2 | 2 | \_ | Data sending and reading I/O port |
| 3 | HC\_CMD\_ID\_GNSS\_DIFF\_TYPE  [=0x0405] | 2 | 2 | 0 | \_ | Systematical differential type |
| 4 | HC\_CMD\_ID\_GNSS\_DAT\_RAW  [=0x0406] | \_ | 2 | 0 | 4+n | GNSS raw observation data |
| 5 | HC\_CMD\_ID\_GNSS\_DAT\_EPH  [=0x0407] | \_ | 2 | 0 | 4+n | Ephemeris data |
| 6 | HC\_CMD\_ID\_GNSS\_DAT\_GGA  [=0x0408] | \_ | 2 | 0 | N | NEMA-0183  GPGGA |
| 7 | HC\_CMD\_ID\_GNSS\_DAT\_GSV  [=0x0409] | \_ | 2 | 0 | N | NEMA-0183  GPGSV |
| 8 | HC\_CMD\_ID\_GNSS\_DAT\_BLH  [=0x040a] | \_ | 2 | 0 | 28 |  |
| 9 | HC\_CMD\_ID\_GNSS\_DAT\_TIME  [=0x040b] | \_ | 2 | 0 | 8 |  |
| 10 | HC\_CMD\_ID\_GNSS\_DAT\_DOPS  [=0x040c] | \_ | 2 | 0 | 16 |  |
| 11 | HC\_CMD\_ID\_GNSS\_DAT\_SVTRACK  [=0x040d] | \_ | 2 | 0 | 8+n\*9 |  |
| 12 | HC\_CMD\_ID\_GNSS\_DAT\_PRE  [=0x040e] | \_ | 2 | 0 | 24 |  |
| 13 | HC\_CMD\_ID\_GNSS\_DAT\_BASE\_POSITION  [=0x040f] | \_ | 2 | 0 | 24 |  |
| 14 | HC\_CMD\_ID\_GNSS\_DAT\_POSINFO  [=0x0410] | \_ | 2 | 0 | 64 | Position information (coordinates, time, accuracy assessment) |
| 15 | HC\_CMD\_ID\_GNSS\_DAT\_GSENSOR  [=0x0411] | \_ | 2 | \_ | 27 | Electronic compass information output |
| 16 | HC\_CMD\_ID\_GNSS\_DAT\_GSA  [=0x0412] | \_ | 2 | 0 | N | NEMA-0183  GPGSA |
| 17 | HC\_CMD\_ID\_GNSS\_DAT\_GST  [=0x0413] | \_ | 2 | 0 | N | NEMA-0183  GPGST |
| 18 | HC\_CMD\_ID\_GNSS\_DAT\_RMC  [=0x0414] | \_ | 2 | 0 | N | NEMA-0183  GPRMC |
| 19 | HC\_CMD\_ID\_GNSS\_DAT\_GLL  [=0x0415] | \_ | 2 | 0 | N | NEMA-0183  GPGLL |
| 20 | HC\_CMD\_ID\_GNSS\_DAT\_VTG  [=0x0416] | \_ | 2 | 0 | N | NEMA-0183  GPVTG |
| 21 | HC\_CMD\_ID\_GNSS\_DAT\_ZDA  [=0x0417] | \_ | 2 | 0 | N | NEMA-0183  GPZDA |
| 22 | HC\_CMD\_ID\_GNSS\_DAT\_GRS  [=0x0418] | \_ | 2 | 0 | N | NEMA-0183  GPGRS |
| 23 | HC\_CMD\_ID\_GNSS\_DAT\_PJK  [=0x0419] | \_ | 2 | 0 | N | NEMA-0183  PTNL,PJK |
| 24 | HC\_CMD\_ID\_GNSS\_DAT\_PJT  [=0x041a] | \_ | 2 | 0 | N | NEMA-0183  PTNL,PJT |
| 25 | HC\_CMD\_ID\_GNSS\_DAT\_MAGNETOMETER  [=0x041b] | \_ | 2 | \_ | 14 | Magnetometer information |
| 26 | HC\_CMD\_ID\_GNSS\_PDOP\_MASK  [=0x0450] | 1 | 1 | 0 | \_ | PDOP mask |
| 27 | HC\_CMD\_ID\_GNSS\_ANGLE\_MASK  [=0x0451] | 1 | 1 | 0 | \_ | Elevation mask angle of searching stars |
| 28 | HC\_CMD\_ID\_GNSS\_BASE\_POSITION  [=0x0452] | 26 | 26 | 0 | \_ | Set base station mode and coordinates |
| 29 | HC\_CMD\_ID\_GNSS\_SEARCH\_RESET  [=0x0453] | \_ | 0 | \_ | \_ | GNSS reset |
| 30 | HC\_CMD\_ID\_GNSS\_UNLOGALL  [=0x0454] | \_ | 0 | \_ | \_ | Clear the data and output it |
| 31 | HC\_CMD\_ID\_GNSS\_SATELLITE\_ENABLE  [=0x0455] | 20 | 20 | 0 | \_ | Enable/Disable the specific satellite |
| 32 | HC\_CMD\_ID\_GNSS\_DATA\_CONFIG\_LIST  [=0x0456] | 2+n\*2 | \_ | 0 | \_ | Query port data and output configuration list |

### HC\_CMD\_ID\_GNSS\_FREQ[=0x0403]

Data sending frequency setting command. The command is meaningless when used alone, but it can be combined with any command that supports the SEND action in order to specify a variety of data on the same frequency to send data. This command is a obsolete design, so it can be substituted with SET which supports the command of SEND action. For more details, please refer to the interpretation of response command.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency of predefined enumeration values, please refer to the definition of HC\_DATA\_FRQ\_ENUM |

Frequency of predefined enumeration values：

typedef enum

{

HC\_DATA\_FRQ\_START = 0 ,

HC\_DATA\_FRQ\_OFF =1 ,

HC\_DATA\_FRQ\_ONCE =2 ,

HC\_DATA\_FRQ\_AUTO =3 ,

HC\_DATA\_FRQ\_50HZ =4 ,

HC\_DATA\_FRQ\_20HZ =5 ,

HC\_DATA\_FRQ\_10HZ =6 ,

HC\_DATA\_FRQ\_5HZ =7 ,

HC\_DATA\_FRQ\_2HZ =8 ,

HC\_DATA\_FRQ\_1HZ =9 ,

HC\_DATA\_FRQ\_2S =10 ,

HC\_DATA\_FRQ\_5S =11 ,

HC\_DATA\_FRQ\_10S =12 ,

HC\_DATA\_FRQ\_15S =13 ,

HC\_DATA\_FRQ\_30S =14 ,

HC\_DATA\_FRQ\_1M =15 ,

HC\_DATA\_FRQ\_5M =16 ,

HC\_DATA\_FRQ\_10M =17 ,

HC\_DATA\_FRQ\_END

} HC\_DATA\_FRQ\_ENUM;

### HC\_CMD\_ID\_GNSS\_IO\_ID[=0x0404]

Appointing GNSS data sending I/O port command. Unlike the HC\_COM\_PORT command which only represents the physical port on the device, the command even includes some virtual ports such as Bluetooth, WIFI, network and so on. Like the HC\_COM\_PORT command, this command should not be used alone, but it can be used to operate on inquiring or setting relevant designated ports only when combined with some commands such as HC\_CMD\_ID\_GNSS\_FREQ, HC\_CMD\_ID\_GNSS\_UNLOGALL, HC\_CMD\_ID\_GNSS\_DATA\_CONFIG\_LIST and so on.

**[SET] & [GET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** | **Len** | **Description** |
| IO\_ID | HC\_UINT16 | 2 | Data communication I/O port predefined enumerated values, please refer to the definition of HC\_GNSS\_IO\_ID\_ENUM |

Baud rate predefined enumerated values:

typedef enum

{

HC\_GNSS\_IO\_ID\_START = 0 ,

HC\_GNSS\_IO\_ID\_COM\_1 = 1 , //COM1

HC\_GNSS\_IO\_ID\_COM\_2 = 2 , //COM2

HC\_GNSS\_IO\_ID\_COM\_3 = 3 , //COM2

HC\_GNSS\_IO\_ID\_BT = 4 , //Bluetooth

HC\_GNSS\_IO\_ID\_RADIO = 5 , //inner radio

HC\_GNSS\_IO\_ID\_NETLINK\_ROVER = 6 , //inner network

HC\_GNSS\_IO\_ID\_NETLINK\_1 = 7 , //reserved

HC\_GNSS\_IO\_ID\_NETLINK\_2 = 8 , //reserved

HC\_GNSS\_IO\_ID\_NETLINK\_3 = 9 , //reserved

HC\_GNSS\_IO\_ID\_NETLINK\_4 = 10 , //reserved

HC\_GNSS\_IO\_ID\_NETLINK\_5 = 11 , //reserved

HC\_GNSS\_IO\_ID\_NETLINK\_6 = 12 , //reserved

HC\_GNSS\_IO\_ID\_NETSERVER\_1 = 13 , //reserved

HC\_GNSS\_IO\_ID\_NETSERVER\_2 = 14 , //reserved

HC\_GNSS\_IO\_ID\_NETSERVER\_3 = 15 , //reserved

HC\_GNSS\_IO\_ID\_NETSERVER\_4 = 16 , //reserved

HC\_GNSS\_IO\_ID\_WIFI = 17 , // WIFI

HC\_GNSS\_IO\_ID\_END

} HC\_GNSS\_IO\_ID\_ENUM;

For example, set Bluetooth I/O port like this: output the position information at a frequency of once per second and output GPGGA data at a frequency of once every 5 seconds, the command block\_list is structured as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_GNSS\_SET | GNSS information setting |
| Block 2 | HC\_CMD\_ID\_GNSS\_IO\_ID | 2 | HC\_GNSS\_IO\_ID\_BT | Setting Bluetooth I/O |
| Block 3 | HC\_CMD\_ID\_GNSS\_DAT\_POSINFO | 2 | HC\_DATA\_FRQ\_1HZ | Position information outputs at a frequency of once per second |
| Block 4 | HC\_CMD\_ID\_GNSS\_DAT\_SVTRACK | 2 | HC\_DATA\_FRQ\_5S | Satellite status information outputs at a frequency of once every 5 seconds |
| Block 5 | HC\_CMD\_ID\_GNSS\_DAT\_GGA | 2 | HC\_DATA\_FRQ\_5S | GPGGA data output at a frequency of once every 5 seconds |

For example, clear all the output data at the Bluetooth I/O port, that is to say, no more SEND any information actively. The command block\_list will be structured as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_GNSS\_SET | GNSS information setting |
| Block 2 | HC\_CMD\_ID\_GNSS\_IO\_ID | 2 | HC\_GNSS\_IO\_ID\_BT | Setting Bluetooth I/O |
| Block 3 | HC\_CMD\_ID\_GNSS\_UNLOGALL | 0 | \_ | Clear all the output |

In the above-mentioned block list, if not set block 2, that is to say, do not designate corresponding I/O port, then by default sending ports takes effect according to the current command.

### HC\_CMD\_ID\_GNSS\_DIFF\_TYPE[=0x0405]

GNSS differential data type setting and query command, which is used to send differential data type when setting starting base station.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| diff\_type | HC\_UINT16 | 2 | Differential type predefined enumerated values, please refer to the definition of HC\_GNSS\_DIFF\_TYPE\_ENUM |

Differential type predefined enumerated values:

typedef enum

{

//used for pub and sub

HC\_GNSS\_DIFF\_TYPE\_START = 0 ,

HC\_GNSS\_DIFF\_TYPE\_NONE = 1 , // no

HC\_GNSS\_DIFF\_TYPE\_CMR = 2 , // CMR difference

HC\_GNSS\_DIFF\_TYPE\_CMR2 = 3 , // CMR+ difference

HC\_GNSS\_DIFF\_TYPE\_SCMR = 4 , // SCMR difference

HC\_GNSS\_DIFF\_TYPE\_RTCM = 5 , // RTCM V2.x difference

HC\_GNSS\_DIFF\_TYPE\_RTCMV3 = 6 , // RTCM V3.x difference

HC\_GNSS\_DIFF\_TYPE\_END

} HC\_GNSS\_DIFF\_TYPE\_ENUM;

### HC\_CMD\_ID\_GNSS\_DAT\_RAW[=0x0406]

GNSS raw packet setting and sending command. Set sending frequency by SET.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values, please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| len | HC\_UINT32 | 4 | Length of data |
| data | HC\_PUINT8 | len | GNSS raw observation data |

typedef struct

{

HC\_UINT32 len;

HC\_PUINT8 data;

} HC\_GNSS\_DAT\_BUFF\_STRUCT;

### HC\_CMD\_ID\_GNSS\_DAT\_EPH[=0x0407]

GNSS ephemeris packet setting and sending command. Set sending frequency by SET.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values, please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| len | HC\_UINT32 | 4 | Length of data |
| data | HC\_PUINT8 | len | GNSS ephemeris data |

typedef struct

{

HC\_UINT32 len;

HC\_PUINT8 data;

} HC\_GNSS\_DAT\_BUFF\_STRUCT;

### HC\_CMD\_ID\_GNSS\_DAT\_GGA[=0x0408]

This command is only used to set the sending frequency of GPGGA packet by combing with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPGGA packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values.  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions (NMEA-0183):**

$GPGGA,172814.0,3723.46587704,N,12202.26957864,W,2,6,1.2,18.893,M,-25.669,M,2.0,0031\*4F

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPGGA |
| 1 | UTC time of position fix |
| 2 | Latitude |
| 3 | Direction of latitude:  N: North  S: South |
| 4 | Longitude |
| 5 | Direction of longitude:  E: East  W: West |
| 6 | GPS Quality indicator:  0: Fix not valid  1: GPS fix  2: Differential GPS fix  4: Real Time Kinematic, fixed integers  5: Real Time Kinematic, float integers |
| 7 | Number of SVs in use, range from 00 through 12 |
| 8 | HDOP |
| 9 | Orthometric height (MSL please reference) |
| 10 | M: unit of measure for orthometric height is meters |
| 11 | Geoid separation |
| 12 | M: geoid separation is measured in meters |
| 13 | Age of differential GPS data record, Type 1 or Type 9. Null field when DGPS is not used. |
| 14 | please reference station ID, ranging from 0000 through 1023. A null field when any  please reference station ID is selected and no corrections are received. |
| 15 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_GSV[=0x0409]

This command is only used to set the sending frequency of GPGSV packet by combing with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPGSV packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values, please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions (NMEA-0183):**

$GPGSV,4,1,13,02,02,213,,03,-3,000,,11,00,121,,14,13,172,05\*67

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPGSV |
| 1 | Total number of messages of this type in this cycle |
| 2 | Message number |
| 3 | Total number of SVs visible |
| 4 | SV PRN number |
| 5 | Elevation, in degrees, 90° maximum |
| 6 | Azimuth, degrees from True North, 000° through 359° |
| 7 | SNR, 00–99 dB (null when not tracking) |
| 8-11 | Information about second SV, same format as fields 4 through 7 |
| 12-15 | Information about third SV, same format as fields 4 through 7 |
| 16-19 | Information about fourth SV, same format as fields 4 through 7 |
| 20 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_BLH[=0x040A]

GPS position longitude, latitude and elevation packets setting and sending command. Set sending frequency by SET.

**[SET] value structure definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| b | double | 8 | Latitude(radian) |
| l | double | 8 | longitude(radian) |
| h | double | 8 | height |
| type | HC\_UINT32 | 4 | GPS Quality indicator:  0: Fix not valid  1: GPS fix  2: Differential GPS fix  4: Real Time Kinematic, fixed integers  5: Real Time Kinematic, float integers |

typedef struct

{

double b;

double l;

double h;

HC\_UINT32 type;

} HC\_GNSS\_DAT\_BLH\_STRUCT;

### HC\_CMD\_ID\_GNSS\_DAT\_TIME[=0x040B]

GPS time packets setting and sending command. Set sending frequency by SET.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values，please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| week | HC\_UINT32 | 4 | GPS week |
| sec | HC\_UINT32 | 4 | Seconds less than a cycle |

typedef struct

{

HC\_UINT32 week;

HC\_UINT32 sec;

} HC\_GNSS\_DAT\_TIME\_STRUCT;

### HC\_CMD\_ID\_GNSS\_DAT\_DOPS[=0x040C]

GNSS PDOP、HDOP、VDOP and TDOP packets setting and sending command. Set sending frequency by SET.

**[SET] value structure definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| pdop | float | 4 | PDOP-Position dilution of precision |
| hdop | float | 4 | HDOP-Horizontal dilution of precision |
| vdop | float | 4 | VDOP-Vertical dilution of precision |
| tdop | float | 4 | TDOP-Time dilution of precision |

typedef struct

{

float pdop;

float hdop;

float vdop;

float tdop;

} HC\_GNSS\_DAT\_DOPS\_STRUCT;

### HC\_CMD\_ID\_GNSS\_DAT\_SVTRACK[=0x040D]

Satellite status information packets setting and sending command, including satellite PRN, elevation, azimuth, INR and so on. Set the sending frequency by SET.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values, please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | | Type | Len | Description |
| gps\_num | | HC\_UINT8 | 1 | GPS satellite number |
| glo\_num | | HC\_UINT8 | 1 | Glonass satellite number |
| bds\_num | | HC\_UINT8 | 1 | Compass satellite number |
| sbas\_num | | HC\_UINT8 | 1 | SBAS satellite number |
| galileo\_num | | HC\_UINT8 | 1 | Galileo satellite number |
| reserved1 | | HC\_UINT8 | 1 | Extension bit reserved |
| reserved2 | | HC\_UINT8 | 1 | Extension bit reserved |
| reserved3 | | HC\_UINT8 | 1 | Extension bit reserved |
| Sv1 | prn | HC\_UINT8 | 1 | PRN |
| type | HC\_UINT8 | 1 | 0:GPS 1:GLONASS 2:COMPASS 3:SPAS 4:Galileo  Please refer to the definition of GALAXY\_TYPE |
| elevation | HC\_UINT8 | 1 | Elevating angle |
| azimuth | HC\_UINT16 | 2 | Azimuth angle |
| L1 | HC\_UINT8 | 1 | INR L1(BDS B1) |
| L2 | HC\_UINT8 | 1 | INR L2(BDS B2) |
| L5 | HC\_UINT8 | 1 | INR L5(BDS B3) |
| use | HC\_UINT8 | 1 | Use positioning solution or not  0 no 1 yes |
| Sv2 | | SV\_INFO | 9 | Status information of the second satellite  Please refer to the definition of SV\_INF |
| .... | | | | |
| Svn | | SV\_INFO | 9 | Status information of the number n satellite  n=gps\_num+glo\_num+bds\_num+sbas\_num+galileo\_num |

typedef struct

{

HC\_UINT8 gps\_num;

HC\_UINT8 glo\_num;

HC\_UINT8 bds\_num;

HC\_UINT8 sbas\_num;

HC\_UINT8 galileo\_num;

HC\_UINT8 reserved1; // Extension bit reserved

HC\_UINT8 reserved2; // Extension bit reserved

HC\_UINT8 reserved3; // Extension bit reserved

SV\_INFO sv\_info[]; // SV\_INFO

} HC\_GNSS\_DAT\_SVTRACK\_STRUCT;

typedef struct

{

HC\_UINT8 prn;

HC\_UINT8 type; // GALAXY\_TYPE

HC\_UINT8 elevation;

HC\_UINT16 azimuth;

HC\_UINT8 L1;

HC\_UINT8 L2;

HC\_UINT8 L5;

HC\_UINT8 use; // 0 no 1 yes

} SV\_INFO;

typedef enum

{

GALAXY\_GPS = 0, // GPS

GALAXY\_GLO, // Glonass

GALAXY\_BDS, // Compass

GALAXY\_SBAS, // SBAS

GALAXY\_GALILEO, // Galileo

} GALAXY\_TYPE;

### HC\_CMD\_ID\_GNSS\_DAT\_PRE[=0x040E]

GNSS position accuracy packet setting and sending command. Set the sending frequency by SET.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| vpre | double | 8 | Vertical precision |
| hpre | double | 8 | Horizontal precision |
| rms | double | 8 | Position RMS |

typedef struct

{

double vpre;

double hpre;

double rms;

} HC\_GNSS\_DAT\_PRE\_STRUCT;

### HC\_CMD\_ID\_GNSS\_DAT\_BASE\_POSITION[=0x040F]

GNSS base coordinate packet setting and sending command. Set the sending frequency by SET.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values:  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| x | double | 8 | WGS84 space rectangular coordinate X |
| y | double | 8 | WGS84 space rectangular coordinate Y |
| z | double | 8 | WGS84 space rectangular coordinate Z |

typedef struct

{

double x;

double y;

double z;

} HC\_GNSS\_POS\_3D\_STRUCT;

### HC\_CMD\_ID\_GNSS\_DAT\_POSINFO[=0x0410]

GNSS position packet setting and sending command, including position coordinate, position time, position accuracy, differential delay and so on. Set the sending frequency by SET. This command is the combination and encapsulation of three commands including HC\_CMD\_ID\_GNSS\_DAT\_BLH, HC\_CMD\_ID\_GNSS\_DAT\_TIME and HC\_CMD\_ID\_GNSS\_DAT\_PRE.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| b | double | 8 | Latitude(radian) |
| l | double | 8 | longitude(radian) |
| h | double | 8 | height |
| type | HC\_UINT32 | 4 | GPS Quality indicator:  0: Fix not valid  1: GPS fix  2: Differential GPS fix  4: Real Time Kinematic, fixed integers  5: Real Time Kinematic, float integers |
| week | HC\_UINT32 | 4 | GPS week |
| sec | HC\_UINT32 | 4 | Seconds less than a cycle |
| vpre | double | 8 | Vertical precision |
| hpre | double | 8 | Horizontal precision |
| rms | double | 8 | Position RMS |
| lag | HC\_INT32 | 4 | Differential delay (second) |

typedef struct

{

HC\_GNSS\_DAT\_BLH\_STRUCT pos;

HC\_GNSS\_DAT\_TIME\_STRUCT time;

HC\_GNSS\_DAT\_PRE\_STRUCT pre;

HC\_INT32 lag;

} hc\_gnss\_dat\_posinfo\_struct;

### Hc\_cmd\_id\_gnss\_dat\_gsensor[=0x0411]

Design reserved.

### Hc\_cmd\_id\_gnss\_dat\_gsa[=0x0412]

This command can only be used to set the sending frequency of GPGSA packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPGSA packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | type | len | description |
| freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definitions of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions（NMEA-0183）：**

$GPGSA,<1>,<2>,<3>,<3>,,,,,<3>,<3>,<3>,<4>,<5>,<6>\*<7><CR><LF>

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPGSA |
| 1 | Mode 1, M = manual, A = automatic |
| 2 | Mode 2, Fix type, 1 = not available, 2 = 2D, 3 = 3D |
| 3 | PRN number, 01 through 32, of satellite used in solution, up to 12 transmitted |
| 4 | PDOP-Position dilution of precision, 0.5 through 99.9 |
| 5 | HDOP-Horizontal dilution of precision, 0.5 through 99.9 |
| 6 | VDOP-Vertical dilution of precision, 0.5 through 99.9 |
| 7 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_GST[=0x0413]

This command can only be used to set the sending frequency of GPGST packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPGST packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions（NMEA-0183）：**

$GPGST,172814.0,0.006,0.023,0.020,273.6,0.023,0.020,0.031\*6A

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPGST |
| 1 | UTC of position fix |
| 2 | RMS value of the pseudo range residuals; includes carrier phase residuals during  Periods of RTK(float) and RTK(fixed) processing |
| 3 | Error ellipse semi-major axis 1 sigma error, in meters |
| 4 | Error ellipse semi-minor axis 1 sigma error, in meters |
| 5 | Error ellipse orientation, degrees from true north |
| 6 | Latitude 1 sigma error, in meters |
| 7 | Longitude 1 sigma error, in meters |
| 8 | Height 1 sigma error, in meters |
| 9 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_RMC[=0x0414]

This command can only be used to set the sending frequency of GPRMC packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPRMC packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions（NMEA-0183）：**

$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W\*6A

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPRMC |
| 1 | UTC of position fix |
| 2 | Status  A=active/valid  V=Void (Navigation receiver warning). This is output to indicate something is wrong. |
| 3 | Latitude |
| 4 | Longitude |
| 5 | Speed over the ground in knots |
| 6 | Track angle in degrees (True) |
| 7 | Date |
| 8 | Magnetic variation in degrees |
| 9 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_GLL[=0x0415]

This command can only be used to set the sending frequency of GPGLL packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPGLL packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions（NMEA-0183）：**

$GPGSA,A,3,3,6,27,19,9,14,21,22,18,15,,,2.1,1.0,1.8\*03

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPGLL |
| 1 | Latitude in dd mm,mmmm format (0-7 decimal places) |
| 2 | Direction of latitude N: North S: South |
| 3 | Longitude in ddd mm,mmmm format (0-7 decimal places) |
| 4 | Direction of longitude E: East W: West |
| 5 | UTC of position in hhmmss.ss format |
| 6 | Fixed text "A" shows that data is valid |
| 7 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_VTG[=0x0416]

This command can only be used to set the sending frequency of GPVTG packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPVTG packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions (NMEA-0183):**

$GPVTG,,T,,M,0.00,N,0.00,K\*4E

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPVTG |
| 1 | Track made good (degrees true) |
| 2 | T: track made good is relative to true north |
| 3 | Track made good (degrees magnetic) |
| 4 | M: track made good is relative to magnetic north |
| 5 | Speed, in knots |
| 6 | N: speed is measured in knots |
| 7 | Speed over ground in kilometers/hour (kph) |
| 8 | K: speed over ground is measured in kph |
| 9 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_ZDA[=0x0417]

This command can only be used to set the sending frequency of GPZDA packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPZDA packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions（NMEA-0183）：**

$GPZDA,172809,12,07,1996,00,00\*45

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPZDA |
| 1 | UTC time |
| 2 | Day, ranging between 01 and 31 |
| 3 | Month, ranging between 01 and 12 |
| 4 | Year |
| 5 | Local time zone offset from GMT, ranging from 00 through ±13 hours |
| 6 | Local time zone offset from GMT, ranging from 00 through 59 minutes |
| 7 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_GRS[=0x0418]

This command can only be used to set the sending frequency of GPGRS packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The GPGRS packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions（NMEA-0183）：**

$GPGRS,220320.0,0,-0.8,-0.2,-0.1, -0.2,0.8,0.6,,,,,,,\*55

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $GPZDA |
| 1 | UTC time of GGA position fix |
| 2 | Residuals  0: Residuals used to calculate position given in the matching GGA line  1: Residuals recomputed after the GGA position was computed |
| 3-14 | Range residuals for satellites used in the navigation solution, in meters |
| 15 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_PJK[=0x0419]

This command can only be used to set the sending frequency of PTNL,PJK packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The PTNL,PJK packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions (NMEA-0183):**

$PTNL,PJK,010717.00,081796,+732646.511,N,+1731051.091,E,1,05,2.7,EHT-28.345,M\*7C

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $PTNL,PJK |
| 1 | UTC time of position fix |
| 2 | Date |
| 3 | Northing, in meters |
| 4 | Direction of Northing will always be N (North) |
| 5 | Easting, in meters |
| 6 | Direction of Easting will always be E (East) |
| 7 | GPS Quality indicator:  0: Fix not available or invalid  1: Autonomous GPS fix  2: RTK float solution  3: RTK fix solution  4: Differential, code phase only solution (DGPS)  5: SBAS solution – WAAS, EGNOS, MSAS  6: RTK Float 3D network solution  7: RTK Fixed 3D network solution  8: RTK Float 2D network solution  9: RTK Fixed 2D network solution  10: omnistar HP/XP solution  11: omnistar VBS solution |
| 8 | Number of satellites in fix |
| 9 | PDOP of fix |
| 10 | Ellipsoidal height of fix |
| 11 | M: ellipsoidal height is measured in meters |
| 12 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_PJT[=0x041a]

This command can only be used to set the sending frequency of PTNL,PJT packets by combining with the command called HC\_CMD\_ID\_GNSS\_FREQ. The PTNL,PJT packets are grouped and transmitted from the receiver according to NMEA-0183 protocol and analyses are totally based on NMEA-0183 protocol.

**[SET] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | HC\_UINT16 | 2 | Frequency predefined enumerated values,  Please refer to the definition of HC\_DATA\_FRQ\_ENUM |

**[REPLY] structure definitions（NMEA-0183）：**

$PTNL,PJT,NAD83(Conus),California Zone 4 0404,\*51

|  |  |
| --- | --- |
| Field | Description |
| 0 | Message ID $PTNL,PJT |
| 1 | Coordinate system name (can include multiple words) |
| 2 | Project name (can include multiple words) |
| 3 | The checksum data, always begins with \* |

For more details, please refer to NMEA-0183 protocol.

### HC\_CMD\_ID\_GNSS\_DAT\_MAGNETOMETER[=0x041b]

Design reserved.

### HC\_CMD\_ID\_GNSS\_PDOP\_MASK[=0x0450]

GNSS PDOP mask setting and query command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Pdop\_mask | HC\_UINT8 | 1 | PDOP mask, range of values 0-100 |

### HC\_CMD\_ID\_GNSS\_ANGLE\_MASK[=0x0451]

GNSS elevation mask angle of searching stars setting and query command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Angle\_mask | HC\_UINT8 | 1 | Elevation mask angle of searching stars (degree) |

### HC\_CMD\_ID\_GNSS\_BASE\_POSITION[=0x0452]

GNSS base station coordinate and base station boot mode setting and query command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Mode | HC\_UINT16 | 2 | Base station boot mode：  1 do not start base station (rover station) now, the base station coordinate(x,y,z) can be ignored  2 start base station automatically  3 start base station manually |
| X | Double | 8 | WGS84 space rectangular coordinate system X |
| Y | Double | 8 | WGS84 space rectangular coordinate system Y |
| Z | Double | 8 | WGS84 space rectangular coordinate system Z |

Typedef struct

{

HC\_UINT16 mode; // HC\_GNSS\_BASE\_MODE\_ENUM

HC\_GNSS\_POS\_3D\_STRUCT pos;

} HC\_GNSS\_BASE\_SET\_STRUCT;

typedef enum

{

HC\_GNSS\_BASE\_MODE\_START = 0,

HC\_GNSS\_BASE\_MODE\_NONE,

HC\_GNSS\_BASE\_MODE\_AUTO,

HC\_GNSS\_BASE\_MODE\_MANUAL,

HC\_GNSS\_BASE\_MODE\_END,

} HC\_GNSS\_BASE\_MODE\_ENUM;

### HC\_CMD\_ID\_GNSS\_SEARCH\_RESET[=0x0453]

Receiver’s motherboard restart setting command. The receiver will search the star again after restarting. The command block\_list is constructed as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_GNSS\_SET | GNSS information setting |
| Block 2 | HC\_CMD\_ID\_GNSS\_SEARCH\_RESET | 0 | \_ | Reboot the motherboard |

### HC\_CMD\_ID\_GNSS\_UNLOGALL[=0x0454]

Clear port data output setting command. If the designated port is set to clear data, the command needs to be combined with the HC\_CMD\_ID\_GNSS\_IO\_ID command. Under the circumstance of no specified ports, the default is that the port sending commands currently is set. The command block\_list is constructed as followed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_GNSS\_SET | GNSS information setting |
| Block 2 | HC\_CMD\_ID\_GNSS\_UNLOGALL | 0 | \_ | Clear data output |

### HC\_CMD\_ID\_GNSS\_SATELLITE\_ENABLE[=0x0455]

Design reserved.

### HC\_CMD\_ID\_GNSS\_DATA\_CONFIG\_LIST[=0x0456]

Designated port’s output data type and frequency information query and setting command. It needs to be combined with the HC\_CMD\_ID\_GNSS\_IO\_ID command. Under the circumstance of no specified ports, the default is that the port sending commands currently is set.

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| IO\_ID | HC\_UINT16 | 2 | Port number, please refer to the definition of HC\_GNSS\_IO\_ID\_ENUM |
| Freq\_list | HC\_UINT16[] | N\*2 | Output frequency of corresponding data type, please refer to the definition of HC\_DATA\_FRQ\_ENUM.  Array index of corresponding data type, please refer to the definition of HC\_DATA\_INDEX\_ID\_ENUM |

typedef enum

{

HC\_DATA\_INDEX\_ID\_START = 0 ,

HC\_DATA\_INDEX\_ID\_NONE = 1,

HC\_DATA\_INDEX\_ID\_CMR = 2,

HC\_DATA\_INDEX\_ID\_CMR2 = 3,

HC\_DATA\_INDEX\_ID\_SCMR = 4,

HC\_DATA\_INDEX\_ID\_RTCM = 5,

HC\_DATA\_INDEX\_ID\_RTCMV3 = 6,

HC\_DATA\_INDEX\_ID\_RAW = 7,

HC\_DATA\_INDEX\_ID\_EPH = 8,

HC\_DATA\_INDEX\_ID\_GGA = 9,

HC\_DATA\_INDEX\_ID\_GSV = 10,

HC\_DATA\_INDEX\_ID\_BLH = 11,

HC\_DATA\_INDEX\_ID\_TIME = 12,

HC\_DATA\_INDEX\_ID\_DOP = 13,

HC\_DATA\_INDEX\_ID\_SV\_TRACK = 14,

HC\_DATA\_INDEX\_ID\_BASE\_POSITION = 15,

HC\_DATA\_INDEX\_ID\_VOLT = 16,

HC\_DATA\_INDEX\_ID\_PRE = 17,

HC\_DATA\_INDEX\_ID\_POSINFO = 18,

HC\_DATA\_INDEX\_ID\_GSENSOR = 19,

HC\_DATA\_INDEX\_ID\_RAW2 = 20, //for U370

HC\_DATA\_INDEX\_ID\_EPH2 = 21, //for U370

HC\_DATA\_INDEX\_ID\_GSA = 22,

HC\_DATA\_INDEX\_ID\_GST = 23,

HC\_DATA\_INDEX\_ID\_RMC = 24,

HC\_DATA\_INDEX\_ID\_GLL = 25,

HC\_DATA\_INDEX\_ID\_VTG = 26,

HC\_DATA\_INDEX\_ID\_ZDA = 27,

HC\_DATA\_INDEX\_ID\_GRS = 28,

HC\_DATA\_INDEX\_ID\_PJK = 29,

HC\_DATA\_INDEX\_ID\_PJT = 30,

HC\_DATA\_INDEX\_ID\_MAGNETOMETER = 31,

HC\_DATA\_INDEX\_ID\_END

} HC\_DATA\_INDEX\_ID\_ENUM;

## Radio related command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order Number | Block\_id | Value len | | | | Description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_RADIO\_FREQ  [=0x0703] | 4 | 4 | 0 | \_ |  |
| 2 | HC\_CMD\_ID\_RADIO\_POWER  [=0x0704] | 4 | 4 | 0 | \_ |  |
| 3 | HC\_CMD\_ID\_RADIO\_STEPPER  [=0x0709] | 4 | 4 | 0 | \_ |  |
| 4 | HC\_CMD\_ID\_RADIO\_AIR\_BAUD  [=0x070b] | 4 | 4 | 0 | \_ |  |
| 5 | HC\_CMD\_ID\_RADIO\_PROTOCOL  [=0x070c] | 2 | 2 | 0 | \_ |  |
| 6 | HC\_CMD\_ID\_RADIO\_INFOLIST  [=0x070D] | 26 | \_ | \_ | \_ |  |
| 7 | HC\_CMD\_ID\_RADIO\_POWER\_ON  [=0x070E] | \_ | 0 | \_ | \_ |  |
| 8 | HC\_CMD\_ID\_RADIO\_POWER\_OFF  [=0x070F] | \_ | 0 | \_ | \_ |  |
| 9 | HC\_CMD\_ID\_RADIO\_AUTO\_POWER\_ON  [=0x0710] | 1 | 1 | 0 | \_ |  |
| 10 | HC\_CMD\_ID\_RADIO\_MODULE\_TYPE  [=0x0711] | 2 | \_ | 0 | \_ |  |
| 11 | HC\_CMD\_ID\_RADIO\_POWER\_STATUS  [=0x0712] | 2 | \_ | 0 | \_ |  |

### HC\_CMD\_ID\_RADIO\_FREQ[=0x0703]

Radio frequency query and setting command.

**[SET] & [REPLY] value structure definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Freq | Float | 4 | Internal radio frequency values |

### HC\_CMD\_ID\_RADIO\_POWER[=0x0704]

Radio power query and setting command.

**[SET] & [REPLY] value structure definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Power | HC\_UINT32 | 4 | Internal radio power values |

### HC\_CMD\_ID\_RADIO\_STEPPER[=0x0709]

Radio stepper query and setting command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Power | HC\_UINT32 | 4 | Internal radio steppers |

### HC\_CMD\_ID\_RADIO\_AIR\_BAUD[=0x070b]

Radio air baud rate query and setting command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Air\_baud | HC\_UINT32 | 4 | Internal radio air baud rate |

### HC\_CMD\_ID\_RADIO\_PROTOCOL[=0x070c]

Radio protocol query and setting command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Protocol | HC\_UINT16 | 2 | Radio protocol type,  Please refer to the definition of HC\_RADIO\_PROTOCOL\_ENUM |

Typedef enum

{

HC\_RADIO\_PROTOCOL\_START = 0 ,

HC\_RADIO\_PROTOCOL\_HUACE = 1,

HC\_RADIO\_PROTOCOL\_TRANSPARENT = 2,

HC\_RADIO\_PROTOCOL\_TT450 = 3,

HC\_RADIO\_PROTOCOL\_SATEL\_3AS = 4,

HC\_RADIO\_PROTOCOL\_PCC\_4FSK = 5,

HC\_RADIO\_PROTOCOL\_PCC\_GMSK = 6,

HC\_RADIO\_PROTOCOL\_GMSK = 7,

HC\_RADIO\_PROTOCOL\_END

} HC\_RADIO\_PROTOCOL\_ENUM;

### HC\_CMD\_ID\_RADIO\_INFOLIST[=0x070D]

Design reserved.

### HC\_CMD\_ID\_RADIO\_POWER\_ON[=0x070E]

Power on the radio and open the setup commands.The command block\_list is structured as followed：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_RADIO\_SET | Radio information settings |
| Block 2 | HC\_CMD\_ID\_RADIO\_POWER\_ON | 0 | \_ | Turn on the radio |

### HC\_CMD\_ID\_RADIO\_POWER\_OFF[=0x070f]

Power off the radio and close the setup commands. The command block\_list is structured as followed：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_RADIO\_SET | Radio information settings |
| Block 2 | HC\_CMD\_ID\_RADIO\_POWER\_OFF | 0 | \_ | Turn off the radio |

### HC\_CMD\_ID\_RADIO\_AUTO\_POWER\_ON[=0x0710]

Turn on the radio, whether it will power on automatically query and setting command.

**[SET] & [REPLY] value structure definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| auto\_power on | HC\_UINT8 | 1 | Zero not open after the boot  Nonzero open automatically after the boot |

### HC\_CMD\_ID\_RADIO\_MODULE\_TYPE[=0x0711]

Radio module type query commands.According to the radio type ID, determinate the protocol ,baud rate ,power,stepper and other characteristic information that the radio supports.

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Module\_type | Hc\_uint16 | 2 | Radio module type:  1 no radio module  2 chc receive-only  3 chc transceiver  4 lfs radio  5 satel radio  Please refer to the definition of hc\_radio\_module\_type\_enum |

typedef enum

{

HC\_RADIO\_MODULE\_TYPE\_START = 0 ,

HC\_RADIO\_MODULE\_TYPE\_NONE = 1 ,

HC\_RADIO\_MODULE\_TYPE\_HC\_RX = 2 ,

HC\_RADIO\_MODULE\_TYPE\_HC\_RXTX = 3 ,

HC\_RADIO\_MODULE\_TYPE\_LFS = 4 ,

HC\_RADIO\_MODULE\_TYPE\_SATEL = 5 ,

HC\_RADIO\_MODULE\_TYPE\_END = 6 ,

} HC\_RADIO\_MODULE\_TYPE\_ENUM;

### HC\_CMD\_ID\_RADIO\_POWER\_STATUS[=0x0712]

Radio module power status query command.

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Power\_status | Hc\_uint16 | 2 | Radio power status:  1 opening  2 having opened  3 having closed  Please refer to the definitions of hc\_radio\_power\_status\_enum |

typedef enum

{

HC\_RADIO\_POWER\_STATUS\_START = 0 ,

HC\_RADIO\_POWER\_STATUS\_ING = 1 ,

HC\_RADIO\_POWER\_STATUS\_ON = 2 ,

HC\_RADIO\_POWER\_STATUS\_OFF = 3 ,

HC\_RADIO\_POWER\_STATUS\_END = 4 ,

} HC\_RADIO\_POWER\_STATUS\_ENUM;

## WIFI related commands

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order number | Block\_id | Value len | | | | Description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_WIFI\_PARA\_AP  [=0x0A03] | 42 | 42 | 0 | \_ |  |
| 2 | HC\_CMD\_ID\_WIFI\_MODE  [=0x0A04] | 2 | 2 | 0 | \_ |  |
| 3 | HC\_CMD\_ID\_WIFI\_OFF  [=0x0A05] | \_ | 0 | \_ | \_ |  |
| 4 | HC\_CMD\_ID\_WIFI\_ON  [=0x0A06] | \_ | 0 | \_ | \_ |  |
| 5 | HC\_CMD\_ID\_WIFI\_AUTO\_OPEN  [=0x0A07] | 1 | 1 | 0 | \_ |  |
| 6 | HC\_CMD\_ID\_WIFI\_POWER\_STATUS  [=0x0A08] | 2 | \_ | 0 | \_ |  |

### HC\_CMD\_ID\_WIFI\_PARA\_AP[=0x0A03]

WIFI mode AP parameter setting and query command.The setting command will lead to the restarting of WIFI mode and the existing connection will break down.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Ssid | Hc\_uint8[] | 20 | Wifi service set identifier |
| Passwd\_type | Hc\_uint16 | 2 | Password type  1 wpa-psk  2 wpa2-psk  Please refer to the definition of hc\_wifi\_passwd\_type\_enum |
| Passwd | Hc\_uint8[] | 20 | Password |

typedef struct

{

HC\_UINT8 ssid[20];

HC\_UINT16 passwd\_type; // HC\_WIFI\_PASSWD\_TYPE\_ENUM

HC\_UINT8 passwd[20];

} HC\_WIFI\_PARA\_AP\_STRUCT;

typedef enum

{

HC\_WIFI\_PASSWD\_TYPE\_START = 0 ,

HC\_WIFI\_PASSWD\_TYPE\_WPA = 1 ,

HC\_WIFI\_PASSWD\_TYPE\_WPA2 = 2 ,

HC\_WIFI\_PASSWD\_TYPE\_END = 3 ,

} HC\_WIFI\_PASSWD\_TYPE\_ENUM;

### HC\_CMD\_ID\_WIFI\_MODE[=0x0A04]

WIFI mode query and setting command.The mode includes AP mode and Client mode.Aiming at the achievement of the present version of I80 receiver, the mode will be set to AP mode before going out and does not support modification.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Wifi\_mode | HC\_UINT16 | 2 | WIFI mode：  1 Access Point  2 Client |

### HC\_CMD\_ID\_WIFI\_OFF[=0x0A05]

WIFI off setting command. The command block\_list is structured as followed：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_WIFI\_SET | WIFI information setting |
| Block 2 | HC\_CMD\_ID\_WIFI\_OFF | 0 | \_ | Turn off WIFI |

### HC\_CMD\_ID\_WIFI\_ON[=0x0A06]

WIFI on setting command. The command block\_list is structured as followed：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_WIFI\_SET | WIFI information setting |
| Block 2 | HC\_CMD\_ID\_WIFI\_ON | 0 | \_ | Turn on WIFI |

### HC\_CMD\_ID\_WIFI\_AUTO\_OPEN[=0x0A07]

Whether it will open automatically after turning on WIFI query and setting command .

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| auto\_power on | HC\_UINT8 | 1 | Zero not open after the boot  Nonzero open automatically after the boot |

### HC\_CMD\_ID\_WIFI\_POWER\_STATUS[=0x0A08]

WIFI module power status query command.

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Power\_status | Hc\_uint16 | 2 | Radio power status:  1 opening  2 having opened  3 having closed  Please refer to the definition of hc\_wifi\_power\_status\_enum |

typedef enum

{

HC\_WIFI\_POWER\_STATUS\_START = 0 ,

HC\_WIFI\_POWER\_STATUS\_ING = 1 ,

HC\_WIFI\_POWER\_STATUS\_ON = 2 ,

HC\_WIFI\_POWER\_STATUS\_OFF =3 ,

HC\_WIFI\_POWER\_STATUS\_END = 4 ,

} HC\_WIFI\_POWER\_STATUS\_ENUM;

## File record related command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order number | Block\_id | Value len | | | | Description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_FILE\_RECORD\_ON  [=0x0B03] | \_ | 0 | \_ | \_ |  |
| 2 | HC\_CMD\_ID\_FILE\_RECORD\_OFF  [=0x0B04] | \_ | 0 | \_ | \_ |  |
| 3 | HC\_CMD\_ID\_FILE\_RECORD\_AUTO\_OPEN  [=0x0B05] | 1 | 1 | 0 | \_ |  |
| 4 | HC\_CMD\_ID\_FILE\_RECORD\_PARA  [=0x0B06] | 43 | 43 | 0 | \_ |  |
| 5 | HC\_CMD\_ID\_FILE\_RECORD\_STATUS  [=0x0B07] | 2 | \_ | 0 | \_ |  |
| 6 | HC\_CMD\_ID\_FILE\_RECORD\_PPK\_CMD  [=0x0B08] | \_ | 4+n | \_ | \_ |  |

### HC\_CMD\_ID\_FILE\_RECORD\_ON[=0x0B03]

Turn on the receiver’s file record setting command.The command block\_list is structured as followed：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_FILE\_SET | File record information setting |
| Block 2 | HC\_CMD\_ID\_FILE\_RECORD\_ON | 0 | \_ | Turn on file record |

### HC\_CMD\_ID\_FILE\_RECORD\_OFF[=0x0B04]

Turn off the receiver’s file record setting command. The command block\_list is structured as followed：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_FILE\_SET | File record information setting |
| Block 2 | HC\_CMD\_ID\_FILE\_RECORD\_OFF | 0 | \_ | Turn off file record |

### HC\_CMD\_ID\_FILE\_RECORD\_AUTO\_OPEN[=0x0b05]

Whether it will open file record automatically after the boot query and setting command .

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Auto\_record | HC\_UINT8 | 1 | Zero not record automatically after the boot  Nonzero record automatically after the boot |

### HC\_CMD\_ID\_FILE\_RECORD\_PARA[=0x0B06]

Receiver’s file record parameter query and setting command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| name | HC\_UINT8[] | 20 | File name (prefix), the specific generated whole file name consists of the prefix and suffix which contains data and time information |
| tm\_start\_en | HC\_BOOL | 1 | Whether to open the timing recording function |
| tm\_start\_h | HC\_UINT32 | 4 | Timing record moment hour(0-23) |
| tm\_start\_m | HC\_UINT32 | 4 | Timing record moment minute (0-59) |
| tm\_length | HC\_UINT32 | 4 | Recording length (minute) |
| rolling | HC\_BOOL | 1 | When the storage space is full, it will record whether the storage will rollback |
| hcn\_flg | HC\_BOOL | 1 | Whether to record static files with the format of hcn (at least one choice from hcn and rinex) |
| rinex\_flg | HC\_BOOL | 1 | Whether to record static files with the format of rinex |
| frq | HC\_UINT16 | 2 | Frequency of static recording . With regard to i80, currently the highest frequency is 1Hz, please refer to the definition of HC\_DATA\_FRQ\_ENUM |
| internal | HC\_BOOL | 1 | Whether to store in the internal flash storage space |
| ftp\_push\_channel | HC\_UINT32 | 4 | Ftp push channel. N72 receiver’s file recording will push the data to the server by the recording channel which is connected to a specific server |

typedef struct

{

HC\_UINT8 name[20];

HC\_BOOL tm\_start\_en;

HC\_UINT32 tm\_start\_h;

HC\_UINT32 tm\_start\_m;

HC\_UINT32 tm\_length;

HC\_BOOL rolling;

HC\_BOOL hcn\_flg;

HC\_BOOL rinex\_flg;

HC\_UINT16 frq; // HC\_DATA\_FRQ\_ENUM

HC\_BOOL internel; // if store in internel flash

HC\_UINT32 ftp\_push\_channel; //0 or 1 or 2 or 3

} HC\_FILE\_RECORD\_CONFIG\_STRUCT;

### HC\_CMD\_ID\_FILE\_RECORD\_STATUS[=0x0B07]

Receiver’s file record status query command.

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Record\_status | Hc\_uint16 | 2 | File record status:  1 being recording  2 stop recording  3 error executing file  Please refer to the definition of hc\_file\_record\_status\_enum |

typedef enum

{

HC\_FILE\_RECORD\_GLOBAL\_STATUS\_START = 0,

HC\_FILE\_RECORD\_GLOBAL\_STATUS\_ON,

HC\_FILE\_RECORD\_GLOBAL\_STATUS\_OFF,

HC\_FILE\_RECORD\_GLOBAL\_STATUS\_ERR,

HC\_FILE\_RECORD\_GLOBAL\_STATUS\_END

} HC\_FILE\_RECORD\_STATUS\_ENUM;

### HC\_CMD\_ID\_FILE\_RECORD\_PPK\_CMD[=0x0B08]

Design reserved.

## Netlink related command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order number | Block\_id | Value len | | | | description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_NETLINK\_INDEX  [=0x1102] | 4 | 4 | 0 | \_ |  |
| 2 | HC\_CMD\_ID\_NETLINK\_IP\_ADDR  [=0x1103] | 57 | 57 | 0 | \_ |  |
| 3 | HC\_CMD\_ID\_NETLINK\_SERVER\_TYPE  [=0x1104] | 2 | 2 | 0 | \_ |  |
| 4 | HC\_CMD\_ID\_NETLINK\_STATUS  [=0x1105] | 8 | \_ | 0 | \_ |  |
| 5 | HC\_CMD\_ID\_NETLINK\_SOURCE\_LIST  [=0x1106] | 4+n | \_ | 0 | \_ |  |
| 6 | HC\_CMD\_ID\_NETLINK\_OPEN  [=0x1107] | \_ | 0 | \_ | \_ |  |
| 7 | HC\_CMD\_ID\_NETLINK\_CLOSE  [=0x1108] | \_ | 0 | \_ | \_ |  |
| 8 | HC\_CMD\_ID\_NETLINK\_AUTO\_OPEN  [=0x1109] | 1 | 1 | 0 | \_ |  |
| 9 | HC\_CMD\_ID\_NETLINK\_DATA\_SOURCE  [=0x110A] | 20 | 20 | 0 | \_ |  |
| 10 | HC\_CMD\_ID\_NETLINK\_ACCOUNT  [=0x110B] | 70 | 70 | 0 | \_ |  |

### HC\_CMD\_ID\_NETLINK\_INDEX[=0x1102]

Design reserved.

### HC\_CMD\_ID\_NETLINK\_IP\_ADDR[=0x1103]

Netlink IP address setting and query command, including setting and inquiring server’s IP address, port number, domain name settings and so on.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Ip | Hc\_uint32 | 4 | Ip network byte order, please refer to the structure definition of winsock [in\_addr](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.chs/winsock/winsock/in_addr_2.htm) |
| Port | Hc\_uint16 | 2 | Web service port number |
| Use\_domain | Hc\_uint8 | 1 | Need domain name or not:  0 do not use domain name and the content in domain field can be ignored  1 use domain name domain |
| Domain | Hc\_uint8[] | 50 | Network domain name |

### HC\_CMD\_ID\_NETLINK\_SERVER\_TYPE[=0x1104]

Netlink server type setting and query command. The command aims to set and inquire network protocols connected to the server.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | description |
| Serv\_type | Hc\_uint16 | 2 | The type of web service  1 Ntrip server  2 Ntrip caster  3 Apis rover  4 Apis base  5 TCP direct  6 UDP direct  please refer to the definition of HC\_NETLINK\_SERVER\_TYPE\_ENUM |

typedef enum

{

HC\_NETLINK\_SERVER\_START = 0 ,

HC\_NETLINK\_SERVER\_CORS\_SERVER = 1 ,

HC\_NETLINK\_SERVER\_CORS\_CASTER = 2 ,

HC\_NETLINK\_SERVER\_APIS\_ROVER = 3 ,

HC\_NETLINK\_SERVER\_APIS\_BASE = 4 ,

HC\_NETLINK\_SERVER\_TCP = 5 ,

HC\_NETLINK\_SERVER\_UDP = 6 ,

hc\_netlink\_server\_end = 7 ,

} hc\_netlink\_server\_type\_enum;

### Hc\_cmd\_id\_netlink\_status[=0x1105]

Netlink status query command.

**[reply] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| link\_status | HC\_UINT16 | 2 | Netlink status:  1 not connected (initial state)  2 connecting to network  3 requesting to login  4 network login successfully  please refer to the definition of HC\_NETLINK\_STATUS\_ENUM |
| link\_cnt | HC\_UINT32 | 4 | Number of connection retries |
| link\_err | HC\_UINT16 | 2 | Code of network connection error:  0 no error  0xFF01 failed in authentication of user name and password  (1-150) linux system’s kernel definition error code |

typedef struct

{

HC\_UINT16 link\_status; //HC\_NETLINK\_STATUS\_ENUM

HC\_UINT32 link\_cnt;

HC\_UINT16 link\_err;

} HC\_NETLINK\_STATUS\_STRUCT;

typedef enum

{

HC\_NETLINK\_START = -1 ,

HC\_NETLINK\_UNCONNECTED = 1 ,

HC\_NETLINK\_CONNECTING = 2 ,

HC\_NETLINK\_LOGING = 3 ,

HC\_NETLINK\_LOGED = 4 ,

HC\_NETLINK\_END = 5 ,

} HC\_NETLINK\_STATUS\_ENUM;

typedef enum

{

HC\_NETLINK\_ERR\_START = -1,

HC\_NETLINK\_ERR\_NOERR = 0,

HC\_NETLINK\_ERR\_UNAUTHORIZED = 0xFF01,

HC\_NETLINK\_ERR\_END

} HC\_NETLINK\_ERR\_ENUM;

When the underlying network is not ready, wireless network fails to dial up and the receiver starts the network connection automatically after the boot or the network connection is set by the HC\_CMD\_ID\_NETLINK\_OPEN command, the network will be in the condition of connecting, returning the relevant network error codes and keeping trying to connect, then returning the connection times. The error code -0xFF01(user name and password authentication failure) is the defined value concerted by CHC. The code 1-150 is the defined value of linux system kernel. For more details, please refer to the definition of linux system kernel.

### HC\_CMD\_ID\_NETLINK\_SOURCE\_LIST[=0x1106]

CORS source list query command.

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Len | HC\_UINT32 | 4 | The length of packet in source list |
| Dat | HC\_UINT8[] | Len | Source list information（Ntrip protocol） |

typedef struct

{

HC\_UINT32 len;

HC\_PUINT8 dat ;

} HC\_CORS\_SOURCE\_LIST\_STRUCT;

Inquiring source list, block\_list should be structured as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_NETLINK\_GET | Netlink information query |
| Block 2 | HC\_CMD\_ID\_NETLINK\_SOURCE\_LIST | 2 | \_ | Inquiring source list |

If the query fails because of the connection status or signal status, there will be a unified overtime reply.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_NETLINK\_GET | Netlink information query |
| Block 2 | HC\_CMD\_ID\_CMD\_RESPONSE | 2 | HC\_RESPONSE\_STATUS\_OUTTIME | Timeout |

### HC\_CMD\_ID\_NETLINK\_OPEN[=0x1107]

Netlink setting command. The netlink setting command block\_list is structured as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_NETLINK\_SET | Netlink information setting |
| Block 2 | HC\_CMD\_ID\_NETLINK\_OPEN | 0 | \_ | Connecting to the network |

### HC\_CMD\_ID\_NETLINK\_CLOSE[=0x1108]

Close network setting command. The closing network setting command block\_list is structured as followed:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_NETLINK\_SET | Netlink informaion setting |
| Block 2 | HC\_CMD\_ID\_NETLINK\_CLOSE | 0 | \_ | Disconnecting the network |

### HC\_CMD\_ID\_NETLINK\_AUTO\_OPEN[=0x1109]

Whether the network will be connected automatically after the boot status query and setting command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Auto\_connect | HC\_UINT8 | 1 | Zero not connect to network automatically  Nonzero connect to network automatically |

### HC\_CMD\_ID\_NETLINK\_DATA\_SOURCE[=0x110a]

Network CORS connection data source query and setting command. The data source information will be selected according to the CORS server.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Src\_name | HC\_INT8[] | 20 | Data source information, the string is ended by ‘\0’ |

### HC\_CMD\_ID\_NETLINK\_ACCOUNT[=0x110b]

Netlink user name and password information query and setting command.

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Name | HC\_INT8[] | 50 | User name, the string is ended by ‘\0’ |
| Password | HC\_INT8[] | 20 | Password, the string is ended by ‘\0’ |

## Mobile network related command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order number | Block\_id | Value len | | | | Description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_3G\_DIAL\_PARA  [=0x1403] | 100 | 100 | 0 | \_ |  |
| 2 | HC\_CMD\_ID\_3G\_DIAL\_STATUS  [=0x1404] | 8 | \_ | 0 | \_ |  |
| 3 | HC\_CMD\_ID\_3G\_DIAL\_ON  [=0x1405] | \_ | 0 | \_ | \_ |  |
| 4 | HC\_CMD\_ID\_3G\_DIAL\_OFF  [=0x1406] | \_ | 0 | \_ | \_ |  |
| 5 | HC\_CMD\_ID\_3G\_POWER\_ON  [=0x1407] | \_ | 0 | \_ | \_ |  |
| 6 | HC\_CMD\_ID\_3G\_POWER\_OFF  [=0x1408] | \_ | 0 | \_ | \_ |  |
| 7 | HC\_CMD\_ID\_3G\_AUTO\_DIAL  [=0x1409] | 1 | 1 | 0 | \_ |  |
| 8 | HC\_CMD\_ID\_3G\_AUTO\_POWER\_ON  [=0x140A] | 1 | 1 | 0 | \_ |  |
| 9 | HC\_CMD\_ID\_3G\_POWER\_STATUS  [=0x140B] | 2 | \_ | 0 | \_ |  |
| 10 | HC\_CMD\_ID\_3G\_SIGNAL\_INTENSITY  [=0x140C] | 5 | \_ | 0 | \_ |  |
| 11 | HC\_CMD\_ID\_3G\_BAND\_MODE  [=0x140D] | 2 | 2 | 0 | \_ |  |

### HC\_CMD\_ID\_3G\_DIAL\_PARA[=0x1403]

Wireless network module network dialing parameters setting and query command。

**[SET] & [REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Init\_string | Hc\_uint8[] | 20 | Initial string |
| Dial\_string | Hc\_uint8[] | 20 | Dial string |
| Account | Hc\_uint8[] | 20 | Account |
| Password | Hc\_uint8[] | 20 | Password |
| Apn | Hc\_uint8[] | 20 | Access point name |

typedef struct

{

HC\_INT8 init\_string[20];

HC\_INT8 dial\_string[20];

HC\_INT8 account[20];

HC\_INT8 password[20];

HC\_INT8 apn[20];

} HC\_MODEM\_DIAL\_PARA\_STRUCT;

### HC\_CMD\_ID\_3G\_DIAL\_STATUS[=0x1404]

Wireless network module dialing status query command.

**[REPLY] value structure definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | description |
| Dial\_status | Hc\_uint16 | 2 | Wireless network dialing status：  1 dialing successfully  2 dialing disconnected  3 dialing unsuccessfully  error code: dial\_err  4 being dialing  please refer to the definition of HC\_MODEM\_DIAL\_STATUS\_ENUM |
| Dial\_cnt | Hc\_uint32 | 4 | Number of dialing retry |
| Dial\_err | Hc\_uint16 | 2 | Netlink error code:  0 no errror  1 SIM card test failed |

typedef struct

{

HC\_UINT16 dial\_status; // HC\_MODEM\_DIAL\_STATUS\_ENUM

HC\_UINT32 dial\_cnt;

HC\_UINT16 dial\_err; // HC\_MODEM\_DIAL\_ERR\_NUM\_ENUM

} HC\_MODEM\_DIAL\_STATUS\_STRUCT;

typedef enum

{

HC\_MODEM\_DIAL\_STATUS\_START = 0 ,

HC\_MODEM\_DIAL\_STATUS\_DAIL\_ON = 1 , //dial ok

HC\_MODEM\_DIAL\_STATUS\_DAIL\_OFF = 2 , //closed

HC\_MODEM\_DIAL\_STATUS\_DAIL\_FAIL = 3 , //dial fail

HC\_MODEM\_DIAL\_STATUS\_DAIL\_ING = 4 , //dialing

HC\_MODEM\_DIAL\_STATUS\_END

} HC\_MODEM\_DIAL\_STATUS\_ENUM;

typedef enum

{

HC\_MODEM\_DIAL\_ERR\_NUM\_START = 0 ,

HC\_MODEM\_DIAL\_ERR\_NUM\_SIM\_ERR = 1 ,

HC\_MODEM\_DIAL\_ERR\_NUM\_END = 2 ,

} HC\_MODEM\_DIAL\_ERR\_NUM\_ENUM;

### HC\_CMD\_ID\_3G\_DIAL\_ON[=0x1405]

Wireless network module dial on setting command. The command block\_list is structured as followed：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_3G\_SET | Wireless network information setting |
| Block 2 | HC\_CMD\_ID\_3G\_DIAL\_ON | 0 | \_ | Wireless network dial on |

### HC\_CMD\_ID\_3G\_DIAL\_OFF[=0x1406]

Wireless network dial off setting command. The command block\_list is structured as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_3G\_SET | Wireless network information setting |
| Block 2 | HC\_CMD\_ID\_3G\_DIAL\_OFF | 0 | \_ | Wireless network dial off |

### HC\_CMD\_ID\_3G\_POWER\_ON[=0x1407]

Wireless network module power on setting command. The command block\_list is structured as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_3G\_SET | Wireless network information setting |
| Block 2 | HC\_CMD\_ID\_3G\_POWER\_ON | 0 | \_ | Module power on |

### HC\_CMD\_ID\_3G\_POWER\_OFF[=0x1408]

Wireless network module power off setting command. The command block\_list is constructed as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order number | Block\_id | Len | Value | Description |
| Block 1 | HC\_CMD\_ID\_CMD | 2 | HC\_CMD\_3G\_SET | Wireless network information setting |
| Block 2 | HC\_CMD\_ID\_3G\_POWER\_OFF | 0 | \_ | Module power off |

### HC\_CMD\_ID\_3G\_AUTO\_DIAL[=0x1409]

Whether the wireless network module will dial automatically after the boot status query and setting command.

**[SET] & [REPLY] value construction definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Auto\_dial | HC\_UINT8 | 1 | Zero not dial automatically after the boot  Nonzero dial automatically after the boot |

### HC\_CMD\_ID\_3G\_AUTO\_POWER\_ON[=0x140a]

Whether the wireless network module will power on automatically after the boot status query and setting command.

**[set] & [reply] value construction definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Auto\_poweron | Hc\_uint8 | 1 | Zero not power on automatically after the boot  Nonzero power on automatically after the boot |

### Hc\_cmd\_id\_3g\_power\_status[=0x140b]

Wireless network module power status query command.

**[reply] value construction definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| power\_status | HC\_UINT16 | 2 | Wireless network module power status ：  1 powering on (during the initialization）  2 the module has been powered on  3 the module has been powered off  please refer to the definition of HC\_MODEM\_POWER\_STATUS\_ENUM |

typedef enum

{

HC\_MODEM\_POWER\_STATUS\_START = 0 ,

HC\_MODEM\_POWER\_STATUS\_ING = 1 ,

HC\_MODEM\_POWER\_STATUS\_POWER\_ON = 2 ,

HC\_MODEM\_POWER\_STATUS\_POWER\_OFF = 3 ,

HC\_MODEM\_POWER\_STATUS\_END = 4 ,

} HC\_MODEM\_POWER\_STATUS\_ENUM;

### HC\_CMD\_ID\_3G\_SIGNAL\_INTENSITY[=0x140C]

Wireless network signal intensify query command.

**[REPLY] value construction definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Snr | Hc\_uint16 | 2 | Wireless network signal snr |
| Grade | Hc\_uint8 | 1 | Wireless network signal strength grade (0-10) |

typedef struct

{

HC\_INT32 snr;

HC\_UINT8 grade;

} HC\_MODEM\_SIGNAL\_INTENSITY\_STRUCT;

### HC\_CMD\_ID\_3G\_BAND\_MODE[=0x140D]

Wireless network preferred type setting and query command.

**[REPLY] value construction definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Band\_mode | Hc\_uint16 | 2 | Preferred wireless network:  1 use 2g network only  2 use 3g network only  3 choose 2g or 3g network freely according to the signal intensify |

typedef enum

{

Hc\_modem\_band\_mode\_statr = 0 ,

hc\_modem\_band\_mode\_2g\_only = 1 ,

hc\_modem\_band\_mode\_3g\_only = 2 ,

hc\_modem\_band\_mode\_2g\_3g\_auto = 3 ,

hc\_modem\_band\_mode\_end

} hc\_modem\_band\_mode\_enum;

## Datalink related command

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Order number | Block\_id | Value len | | | | Description |
| **REPLY** | **SET** | **GET** | **SEND** |
| 1 | HC\_CMD\_ID\_DATALINK\_MODE  [=0x1503] | 2 | 2 | 0 | \_ |  |
| 2 | HC\_CMD\_ID\_DATALINK\_DATA  [=0x1504] | \_ | 4+n | \_ | \_ |  |

### HC\_CMD\_ID\_DATALINK\_MODE[=0x1503]

Design reserved.

### HC\_CMD\_ID\_DATALINK\_DATA[=0x1504]

Differential data import command. When you get the differential data by connecting PDA network to CORS servers, the command will be used to transmit the data to the GNSS motherboard and the receiver will judge which type of the differential data is used according to the differential data that has been imported.

**[SET] value construction definitions：**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Len | Description |
| Len | HC\_UINT32 | 4 | Length of the differential data packet |
| Dat | HC\_UINT8[] | Len | The differential data packet |

CHC - Shanghai HuaCe Navigation Technology Ltd.

Building C, NO. 599 Gaojing Road,

Qingpu District, 201702 Shanghai, China

Tel: +86 21 542 60 273

Fax: +86 21 649 50 963

Email: [sales@chcnav.com](mailto:sales@chcnav.com) | [support@chcnav.com](mailto:support@chcnav.com)

Website: [www.chcnav.com](http://www.chcnav.com)