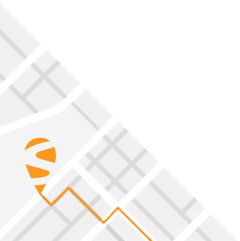
PolaRx5TR Reference Guide

Applicable to version 5.5.0 of the Firmware



PolaRx5TR Reference Guide

2022-08-19

Applicable to version 5.5.0 of the Firmware

Septentrio NV
Greenhill Campus, Interleuvenlaan 15i
3001 Leuven, Belgium

http://www.septentrio.com
Phone: +32 16 300 800
Fax: +32 16 221 640
🐦 @Septentrio

List of Contents

Scope

This document contains reference information about the receiver firmware.

Chapter 1 provides a set of step-by-step "how-to's" to help you find your way around the receiver's commands and logs.

Chapter 2 provides some background on the main algorithms running in the receiver and on the way to configure them.

Chapter 3 contains the complete description of the user command interface.

Chapter 4 contains the complete description of the SBF format.

Typographical Conventions

`abc`	User command name. Clicking a command name redirects to the full command description.
abc	Command argument name.
`abc`	Command replies.
	SBF block name or SBF field name. Clicking an SBF block name redirects to the full SBF block description.

List of Acronyms

Abbreviation	Description
AGC	Automatic Gain Control
API	Application Program Interface
ARP	Antenna Reference Point
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BeiDou	BeiDou Navigation System
BGD	Broadcast Group Delay
CA	Coarse Acquisition
CGGTTS	Common GPS GLONASS Time Transfer Standard
CMR	Compact Measurement Record
COG	Course Over Ground
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DGPS	Differential GPS
DHCP	Dynamic Host Configuration Protocol
DLL	Dynamically Linked Library
DNS	Domain Name Server
DOP	Dilution Of Precision
DVS	Data Validity Status
ECEF	Earth-Centered Earth-Fixed
EGNOS	European Geostationary Navigation Overlay System
ENU	East-North-Up
FTP	File Transfer Protocol

GEO	Geostationary Earth Orbiter
GLONASS	Global Orbiting Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GST	Galileo System Time
GUI	Graphical User Interface
HDOP	Horizontal DOP
HERL	Horizontal External Reliability Level
HMI	Hazardously Misleading Information
HPCA	HMI Probability Computation Algorithm
HPL	Horizontal Protection Level
HS	Health Status
ICD	Interface Control Document
IEEE	Institute of Electrical and Electronics Engineers
IERS	International Earth Rotation Service
IF	Intermediate Frequency
IGP	Ionospheric Grid Point
IGS	International GPS Service
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IODC	Issue of Data - Clock
IODE	Issue Of Data Ephemeris
IP	Internet Protocol
IRNSS	Indian Regional Navigational Satellite System
ITRF	International Terrestrial Reference Frame
ITRS	International Terrestrial Reference System
LBand	L-Band Receiver
L1	L1 carrier
L2	L2 carrier
L2C	L2C code
LED	Light Emitting Diode
LSB	Least Significant Bit
MDB	Minimum Detectable Bias

MIB	Management Information Base
MSB	Most Significant Bits
MT	Message Type
NATO	North Atlantic Treaty Organisation
NAV	Navigation
NavIC	Navigation with Indian Constellation
NAVSTAR	Navigation Satellite Timing And Ranging
NMEA	National Marine Electronics Association
OSNMA	Open Service Navigation Message Authentication
P	P(Y) code
P1	P1 code
P2	P2 code
PC	Phase Center
PDOP	Position DOP
PLL	Phase Locked Loop
PPP	Precise Point Positioning
PPS	Pulse Per Second
PRC	Pseudorange Correction
PRN	Pseudo Random Noise
PVT	Position, Velocity and Time
QZSS	Quasi-Zenith Satellite System
RAIM	Receiver Autonomous Integrity Monitoring
RINEX	Receiver Independent Exchange Format
RTCA	Radio Technical Commission for Aeronautics
RTCM	Radio Technical Commission for Maritime Services
RTK	Real-Time Kinematic
SBAS	Space-Based Augmentation System
SBF	Septentrio Binary Format
SIS	Signal In Space
SISA	Signal in Space Accuracy
SNMP	Simple Network Management Protocol
SV	Space Vehicle
SVID	Space Vehicle ID

TDOP	Time DOP
TOW	Time Of Week
UDRE	User Differential Range Error
UERE	User Equivalent Range Error
UHF	Ultra High Frequency
URA	User Range Accuracy
USB	Universal Serial Bus
UTC	Coordinated Universal Time
VDOP	Vertical DOP
VERL	Vertical External Reliability Level
VPL	Vertical Protection Level
VRS	Virtual Reference Station
WAAS	Wide Area Augmentation System
WGS84	World Geodetic System 1984
WN	Week Number
WNc	Week number
XERL	External Reliability Levels
XOR	Exclusive OR
XPL	Horizontal or Vertical Protection Level

Chapter 1

How To...

This chapter contains step-by-step instructions to help you with typical tasks. It will help you to familiarize yourself with the receiver commands without going into too much detail.

For a comprehensive description of the command set, refer to chapter 3. You can also click on any command or SBF block name in this manual to be redirected to the full description of that command or SBF block.

You can enter user commands in many different ways:

- Commands can be accessed graphically through menus in RxControl and in the web interface (see section 1.1.4).
- Using the Data Link program provided in the RxTools suite (or any suitable terminal emulation program), you can enter commands manually through one of the receiver input ports (see section 1.1). In this chapter, user commands are referred to by their full name for readability. When typing the command, you can always use the short mnemonic equivalent to save typing effort. For instance, instead of typing **`setCOMSettings`**, you can type **`scs`**.
- You can type commands or mnemonics in the console window of RxControl (menu *Tools* > *Expert Console*) or of the web interface (menu *Admin* > *Expert Console*).

⚠ Depending on the capabilities of your particular receiver (see section 1.33), some of the user commands, SBF blocks or communication interfaces described in this document may not be supported.

1.1 Connect to the Receiver

1.1.1 Via COM Ports

A simple way to communicate with the receiver is to connect one of its COM-ports to a COM-port of your host computer. You can use the provided COM cable for this purpose. The default COM-port settings are:

Parameter	Value
`baud rate`	`115200`
`data bits`	`8`
`parity`	`no`
`stop bits`	`1`
`flow control`	`none`

The baud rate can be modified at any time by using the **`setCOMSettings`** command.

RxControl and Data Link can communicate with the receiver over a COM-port connection: select *Serial Connection* option when opening the connection to the receiver.

1.1.2 Via USB

The Windows USB driver provided with your receiver emulates two virtual serial ports, which can be used as standard COM ports to access the receiver. The Windows USB diver can be installed through the RxTools software suite. On Linux, the standard Linux CDC-ACM driver is suitable. Most terminal emulation programs will make no distinction between virtual and native COM ports. Note that the port settings (baud rate, etc) for virtual serial ports are not relevant, and can be left in their default configuration in the terminal emulation program.

When connecting the USB cable to a Windows PC, a new drive appears in the file manager. This drive contains an installer for the USB driver. Running this installer is not needed if you have already installed the RxTools suite. A second drive is created when the receiver is configured as a USB mass-storage device, as explained in section 1.23.

1.1.3 Via a TCP/IP Port

TCP/IP connections allow remote control of the receiver and are potentially much faster than serial connections. Up to eight independent TCP/IP connections can be opened in parallel through port 28784 (the port number can be changed with the command **`setIPPortSettings`**).

RxControl and Data Link can communicate with remote receivers over a TCP/IP connection: select *TCP/IP Connection* option when opening the connection to the receiver.

TCP/IP connections can be made over the following interfaces.

1.1.3.1 Ethernet

Over the Ethernet interface, the receiver can be configured for dynamic or fixed IP address allocation. The default is dynamic address allocation, using the DHCP protocol. The hostname is `polarx5tr-xxxxxxx`, where *xxxxxxx* consists of the last seven digits of the serial number of the receiver.

Dynamic IP address allocation requires the availability of a DHCP server in your local network. In the absence of a DHCP server, or when a fixed IP address is desirable, it is possible to disable the DHCP client and use a fixed address. This is done using the **`setIPSettings`** command.

1.1.3.2 WiFi

1.1.3.2.1 Receiver as WiFi Access Point

By default, the receiver is configured in WiFi access point mode, with the SSID set to "`PolaRx5TR-xxxxxxx`" where *xxxxxxx* is the serial number. Encryption is disabled by default.

When you are connected to the receiver WiFi access point, the receiver can be reached at the fixed IP address `192.168.20.1`, or with the hostname `polarx5tr` (depending on your network configuration, you may need to use `polarx5tr.local` instead).

WiFi can be turned on and off with the **`setWiFiMode`** command, and the access point parameters (SSID, encryption, channel number, ...) can be adjusted with the **`setWiFiAccessPoint`** command.

1.1.3.2.2 Receiver as WiFi Client

It is possible to configure the receiver in WiFi client mode with the **`setWiFiMode`** command. In client mode, the receiver will attempt to connect to a reachable access point. The access point password must be entered with the command **`exeAddWiFiAccessPoint`** (this must be done only once). The list of access points can be obtained with the **`lstWiFiAccessPoints,all`** command.

In client mode, the receiver gets its IP address dynamically, and its hostname is `polarx5tr-xxxxxxx`.

1.1.3.3 Ethernet-over-USB

When an USB cable is connected, the receiver supports Ethernet-over-USB. The IP address allocated to the Ethernet-over-USB interface is `192.168.3.1`. That address cannot be changed, so that this feature is only to be used when a single receiver is connected to your computer.

By default, the receiver is not allowed to access the Internet over USB. This can be changed with the **`setUSBInternetAccess`** command. Note that this requires allowing Internet sharing on your computer. The procedure to do so depends on your operating system. For example, on Windows, it involves enabling *"Allow other network users to connect through this computer's Internet connection."* in the properties of the adapter providing Internet access. When Internet sharing is enabled, the receiver gets its IP address from a DHCP server on your computer. Depending on your computer's routing table, it may be that it is not reachable anymore at `192.168.3.1`.

1.1.3.4 Point-to-Point Link

The receiver incorporates a point-to-point protocol server, by which it can accept TCP/IP connections over a serial link.

Configuring the point-to-point server is done with the **`setPointToPoint`** command. For example, to set up a point-to-point communication over COM1, with the server (i.e. the receiver) having address 192.168.60.1 and the client having address 192.168.60.2, and using CHAP authentication with password “mypwd”, use this command:

```
setPointToPoint, P2PP1, Server, COM1, 192.168.60.2, 192.168.60.1,
CHAP, mypwd <CR>
```

If the client is a Linux computer, make sure the password is set in the `/etc/ppp/chap-secrets` file. For example, the contents of that file could be as follows:

```
# Secrets for authentication using CHAP
# client        server   secret                   IP addresses
*               *        mypwd                    192.168.60.1
```

Assuming that the serial cable is connected to the first serial port of your PC, and that the receiver's COM1 port is left in its default configuration (115200 baud and no hardware flow control), the client PPP daemon can be started with the following Linux command:

```
pppd /dev/ttyS0 115200 nocrtscts local
```

After a few seconds, the PPP link is established and it is possible to access the receiver at IP address 192.168.60.1.

1.1.4 Via a Web Browser

The receiver can be controlled and configured using a web browser. The hostname or fixed IP address is defined as explained in section 1.1.3.

For example, if your receiver's hostname is `polarx5tr-1234567`, simply use the following URL in your preferred web browser:

```
http://polarx5tr-1234567
```

or, for a secure connection:

```
https://polarx5tr-1234567
```

The `https` certificate (`.pem` file) can be uploaded through the *Communication* > *Web Server/TLS* menu of the web interface.

Most user commands described in section 3.2 can be accessed graphically from the web interface. You can also go to *Admin* > *Expert Control* > *Expert Console* to manually type ASCII commands and view replies.

By default, the web interface provides unrestricted read and write access to the receiver. This can be changed, as further explained in section 1.31 of this document.

Note that a lightweight (text only) version of the web interface is available by appending **`/lite`** to the URL, for example:

```
http://polarx5tr-1234567/lite
```

1.1.5 Connection Descriptors

Receiver connections are identified by their connection descriptor (CD). The different connection descriptors are shown in the table below. The three rightmost columns indicate the direction (input or output or both), and whether the connection can accept user command input.

CD	Description	In	Out	Cmd
`COMx`	one of the serial ports	•	•	•
`USBx`	one of the USB-device serial ports	•	•	•
`LOGx`	one of the log sessions. See section 1.15		•	
`IP1x`	one of the TCP/IP connections	•	•	•
`NTRx`	one of the NTRIP connections. Input in NTRIP client mode (section 1.10), output in NTRIP server mode (section 1.9)	•	•	
`IPSx`	one of the IP server connections. See section 1.12	•	•	•
`IPRx`	one of the IP receive connections. See section 1.13	•	•	•

1.2 Understand the Output of the Receiver

The receiver outputs proprietary and standardized messages. Each proprietary message begins with a two-character identifier, which identifies the message type.

Proprietary messages	First two characters
ASCII command replies and command error notification	`$R`
ASCII transmissions (e.g. periodic output of the status screen), terminated by a prompt. Two sub-types are defined: • `$TD` : ASCII display generated by the receiver; • `$TE` : event notification (e.g. receiver is shutting down).	`$T`
Formatted information blocks (e.g. formal command description)	`$-`
SNMP' binary command replies (Septentrio proprietary)	`$&`
Proprietary binary data (SBF)	`$@`

1.2.1 Proprietary Binary Output (SBF)

The binary messages conform to the SBF definition. The data are arranged in SBF blocks identified by block IDs. All the blocks begin with the SBF identifier `$@`. Please refer to section 4 for a description of the SBF format.

The benefit of SBF is completeness. This format should be your first choice if you wish to receive detailed information from the receiver.

The list of supported SBF messages can be found in appendix A

SBF Converter, provided in the RxTools package is an intuitive GUI which allows SBF conversion into e.g. RINEX, KML, GPX or ASCII.

1.2.2 BINEX

The list of supported BINEX messages can be found in appendix B

Section 1.4 explains how to stream BINEX records, and section 1.19 explains how to log BINEX files on the internal disk.

1.2.3 NMEA

The receiver can generate a set of approved NMEA sentences, which conform to the NMEA Standard (version 3.01(1) and version 4.10(2) are supported). The benefit of the NMEA format

(1) NMEA 0183, Standard for Interfacing Marine Electronic Devices, Version 3.01, National Marine Electronics Association, 2002

(2) NMEA 0183, Standard for Interfacing Marine Electronic Devices, Version 4.10, National Marine Electronics Association, 2012

is that it is standardized. Many electronic devices and software packages support NMEA. The drawback of NMEA is a relatively low level of detail.

NMEA output is configured with the **`setNMEAOutput`** command, and the NMEA version (3.01 or 4.10) is selected with the **`setNMEAVersion`** command.

The list of supported NMEA sentences can be found in appendix C.

1.2.4 RTCM and CMR

The receiver can operate as DGPS and/or RTK base station and output the corresponding RTCM or CMR messages. The instructions to set the receiver in base station mode can be found in section 1.6.

The list of supported RTCM and CMR messages can be found in appendix D.

1.3 Define an SBF Output Stream

As an example, this section explains how to use the command line interface to configure the receiver to output the `MeasEpoch` SBF block at 10 Hz, the `PVTCartesian` SBF block at 1 Hz, and the `GPSNav` block at its On-Change rate (see section 4.1.8 for more details on the SBF output rate). In this example, we will assume that these blocks must be output through the USB2 connection.

1. First make sure that the USB2 connection is configured for SBF output (this is the default). In case this is not so, you should invoke:
   ```
   setDataInOut,USB2,,+SBF <CR>
   ```

2. Scheduling SBF blocks for output is done by defining so-called "SBF streams". At least 10 SBF streams can be defined by the user. A stream consists of a set of SBF blocks that need to be output at a given rate through a given connection. By default, all streams are empty, and no SBF blocks are output. For our example, we will need to use two streams. Defining these SBF streams involves the **`setSBFOutput`** command:
   ```
   setSBFOutput,Stream1,USB2,MeasEpoch+GPSNav,msec100 <CR>
   setSBFOutput,Stream2,USB2,PVTCartesian,sec1 <CR>
   ```
   *Note that the rate specified with the **`setSBFOutput`** command (`msec100` or `sec1` above) only applies to the blocks that support a flexible output rate (see appendix A). The `GPSNav` block does not support flexible rate: it is always output at its “On-Change” rate regardless of the stream rate. For this reason, in the above example, we could equally have enabled `GPSNav` in `Stream2`.*

3. To stop outputting SBF on a given connection, you can either redefine or empty the corresponding streams:
   ```
   setSBFOutput,Stream1,USB2,none <CR>
   setSBFOutput,Stream2,USB2,none <CR>
   ```
   A second possibility is to disable all SBF messages on that connection:
   ```
   setDataInOut,USB2,,-SBF <CR>
   ```

Note that the **`exeSBFOnce`** command can be used to output a set of blocks once, instead of at regular interval. This is typically used to output all currently available satellite ephemerides at once. For example, the following command instructs the receiver to output all known GPS, GLONASS, Galileo and BeiDou ephemerides over USB2:

```
exeSBFOnce, USB2, GPS+GLO+GAL+BDS <CR>
```

This is a one-time action: the requested blocks are inserted in the stream, and then the normal flow of blocks as defined with **`setSBFOutput`** resumes. When logging the SBF stream for post-processing, it is a good practice to request all satellite ephemerides with the **`exeSBFOnce`** command when starting a new log file. Make sure however not to request measurement or PVT blocks with **`exeSBFOnce`** when these blocks are also enabled with **`setSBFOutput`** as it could cause the same epoch to be duplicated in the log file. Some post-processing tools may not work properly when the same epoch is repeated twice.

1.4 Define a BINEX Output Stream

As an example, this section explains how to use the command line interface to configure the receiver to output BINEX observation records (record 0x7F-05) at a 1-second interval through the USB2 connection, together with the decoded navigation records at their “On-Change” rate.

1. First make sure that BINEX output is enabled through the USB2 connection. BINEX is enabled by default through all connections, but in case the receiver is not in its default configuration, you should invoke:

   ```
   setDataInOut,USB2, ,+BINEX <CR>
   ```

2. Scheduling BINEX records for output is done by defining so-called "BINEX streams". Up to 16 BINEX streams can be defined by the user. A stream consists of a set of BINEX records that need to be output at a given rate through a given connection. By default, all streams are empty, and no BINEX records are output. For our example, we can use a single stream, defined as follows:

   ```
   setBINEXOutput,Stream1,USB2,Rec7F05+Rec01Nav,sec1 <CR>
   ```

   The above command defines a record interval of 1 second. Note that this interval is only applied to the observation record. Navigation records are always output at their OnChange interval. See also appendix B.

3. To stop outputting BINEX through a given connection, you can either redefine or empty the corresponding stream:

   ```
   setBINEXOutput,Stream1,USB2,none <CR>
   ```

1.5 Save the Configuration in Non-Volatile Memory

The receiver configuration includes all the user-selectable parameters, such as the elevation mask, the PVT mode, the COM port settings,...

By default, the receiver starts up in its factory default configuration. The factory defaults for each of the receiver parameters are underlined for each argument of each command in section 3.2

The current receiver configuration can be checked with the `lstConfigFile` command:

```
lstConfigFile, Current <CR>
```

At any time, it is possible to save the current configuration into non-volatile memory, in order to force the receiver to always start up in that configuration. To do so, the following command should be entered:

```
exeCopyConfigFile,Current,Boot <CR>
```

To revert to the default setting where the receiver starts in the default configuration, you should use:

```
exeCopyConfigFile,RxDefault,Boot <CR>
```

1.6 Configure the Receiver in DGPS/RTK-Base Mode

The receiver can generate and output DGPS and/or RTK corrections in the RTCM and CMR formats. The list of supported RTCM and CMR messages can be found in appendix D.

1.6.1 Static Base Station Mode

To configure the receiver in static base station mode, the following has to be done:

1. For accurate and repetitive absolute positioning, you must provide the accurate coordinates of the antenna reference point (ARP). The ARP usually corresponds to the center of the bottom of the antenna (see also section 2.5). For example, assuming the WGS84 position of the ARP is 50.5^{o}N, 4^{o}E and its altitude above the WGS84 ellipsoid is 100m, use:
   ```
   setStaticPosGeodetic,Geodetic1,50.5,4,100 <CR>
   setPVTMode,Static, ,Geodetic1 <CR>
   ```

   If you are only interested in accurate determination of the base-rover baseline, with the absolute position of the rover being of lesser importance, accurate positioning of the base station is not required, and you may simply let the receiver determine its fixed position autonomously ("auto-base" mode), by typing:
   ```
   setPVTMode,Static, ,auto <CR>
   ```

2. For RTCM 3.x, the antenna information in message types 1007, 1008 and 1033 can be specified using the `setAntennaOffset` command, with the serial number as sixth argument, and the antenna type (called "antenna descriptor" in RTCM) as fifth argument (see also section 2.5). For instance:
   ```
   setAntennaOffset,Main, , , ,"AT2775-54SW","5684" <CR>
   ```

3. Use the commands `setRTCMv2Interval`, `setRTCMv2IntervalObs`, `setRTCMv3Interval` or `setCMRv2Interval` to specify the message interval (default is one second for most messages). For instance, to change the interval at which RTCM 3.x message type 1033 is generated to 10 seconds, type:
   ```
   setRTCMv3Interval,RTCM1033,10 <CR>
   ```

4. Use the commands `setRTCMv2Formatting`, `setRTCMv3Formatting` or `setCMRv2Formatting` to specify the base station ID. If you are setting up multiple base stations, make sure to select a unique ID for each of them. For instance:
   ```
   setRTCMv3Formatting,496 <CR>
   ```

5. By default, the receiver is configured to output all RTCM and CMR messages necessary for DGPS and RTK operation. If necessary, the set of output messages can be specified with the commands `setRTCMv2Output`, `setRTCMv3Output` or `setCMRv2Output`. For instance, to output RTCM3.x messages 1006, 1033 and 1074 on COM2, use:
   ```
   setRTCMv3Output,COM2,RTCM1006+RTCM1033+RTCM1074 <CR>
   ```

   If you are using the RTCM3.x MSM messages (see appendix D), you can use the `setRTCMv3Formatting` command to configure the signal types that need to be encoded in MSM.

6. The RTCM stream can be output through any output connection listed in section 1.1.5. For instance, to enable RTCM 3.x output through COM2, use:

   ```
   setDataInOut,COM2, ,RTCMv3 <CR>
   ```

7. When sending differential corrections over a serial port, do not forget to specify the baud rate. For instance if the differential correction stream needs to be output on COM2 at 9600 baud, use:

   ```
   setCOMSettings,COM2,baud9600 <CR>
   ```

To stop transmitting RTCM messages, enter the following command:

```
setDataInOut,COM2, ,none <CR>
```

Note that, even in static mode, the receiver computes a PVT solution to estimate the clock bias.

1.7 Configure the Receiver in DGPS/RTK-Rover Mode

The receiver computes a DGPS and/or an RTK solution when it receives the relevant differential correction messages on one of its connections. The list of supported differential correction messages can be found in appendix D.

To configure the receiver in DGPS/RTK-rover mode, the following has to be done:

1. Make sure that at least one of the receiver connections is receiving differential corrections. Any input connection listed in section 1.1.5 is suitable. When using a serial connection, make sure to configure the baud rate to match the baud rate of the incoming RTCM stream. For instance if the incoming RTCM stream is received through COM2 at a baud rate of 9600 baud, use:
   ```
   setCOMSettings,COM2,baud9600 <CR>
   ```

2. The receiver automatically detects the format of the differential corrections (RTCM or CMR) and switches between standalone, DGPS or RTK modes according to the type of corrections it receives, provided these modes are enabled with the **`setPVTMode`** command (all modes are enabled by default).

Refer to sections 2.4.2 and 2.4.3 for further details on the DGPS and RTK positioning mode.

1.8 Use the SECORX Services

Your receiver is fully compatible with the SECORX services (see `http://www.septentrio.com/products/correction-services/secorx`) offering high-accuracy positioning without the need for a local base station.

In this section, we address the case where the primary positioning mode is RTK, and where the receiver uses the SECORX-D precise point positioning (PPP) service to continue outputting high-accuracy positions during RTK outages.

Configuring your receiver for RTK/SECORX-D positioning involves the following steps:

1. Make sure that both RTK and PPP positioning modes are enabled in your receiver. This is the default, but in case the receiver is not in its default configuration, you can enable RTK and PPP by issuing the following command:
   ```
   setPVTMode,Rover,+RTK+PPP <CR>
   ```

2. RTK positions are typically expressed in a regional datum which depends on your local RTK provider. Instead, SECORX-D PPP positions relate to a recent version of the global ITRF reference frame. To avoid coordinate jumps each time the PVT engine switches between RTK and PPP modes, the regional datum used by your RTK provider must be provided to the receiver. For example, in Europe, the RTK datum will often be ETRS89, and you would need to enter the following command:
   ```
   setGeodeticDatum, ETRS89 <CR>
   ```

3. Enable the automatic seeding of the PPP engine with RTK positions:
   ```
   setPPPAutoSeed,RTKFixed <CR>
   ```

1.9 Configure the Receiver in NTRIP Server Mode

In the example below, we show how to configure the receiver to send RTCM 3.x corrections to an NTRIP caster using the following parameters:

- NTRIP caster hostname: ntrip.example.com
- NTRIP caster port: 2101
- User name/password for basic authentication: USER / PASSWD
- Mount Point: LEUV1
- TLS: enabled and the caster is trusted by a public certification authority

1. Configure one of the NTRIP connections (see section 1.1.5) in server mode for sending data to the NTRIP caster. Here, we assume that the first NTRIP connection (`NTR1`) is free and can be used for that purpose:
   ```
   setNTRIPSettings,NTR1,Server,ntrip.example.com,2101,USER,PASSWD,LEUV1 <CR>
   ```

2. To enable TLS for `NTR1`, use:
   ```
   setNtripTlsSettings,NTR1,on,"" <CR>
   ```

3. By default, for RTCM 3.x, the receiver is configured to send message types 1004, 1006, 1012 ans 1033 at an interval of one second. This can be changed by using the **setRTCMv3Output** and **setRTCMv3Interval** commands. For instance, to change the interval of RTCM1033 to 10 seconds, use:
   ```
   setRTCMv3Interval,RTCM1033,10 <CR>
   ```

4. Enable the output of RTCM 3.x corrections on the `NTR1` connection:
   ```
   setDataInOut,NTR1, ,RTCMv3 <CR>
   ```

5. Closing the NTRIP connection is done with the following command:
   ```
   setNTRIPSettings,NTR1,off <CR>
   ```

See also section 1.6 for more information on configuring the receiver as a base station.

The NTRIP server can also send data to the built-in caster, by specifying “localhost” as hostname. Refer to section 1.11 for details.

1.10 Configure the Receiver in NTRIP Client Mode

In this section, we show how to configure the receiver to receive and use RTK corrections from an NTRIP caster. In the example below, the NTRIP caster and Mount Point details are as follows:

- NTRIP caster hostname: ntrip.example.com
- NTRIP caster port: 2101
- User name/password for basic authentication: USER / PASSWD
- Mount Point: LEUV1
- TLS: enabled and the caster is trusted by a public certification authority

1. Configure one of the NTRIP connections (see section 1.1.5) for communication with the NTRIP caster in client mode. Here, we assume that the first NTRIP connection (`NTR1`) is free and can be used for that purpose:
   ```
   setNTRIPSettings,NTR1,Client,ntrip.example.com,2101,USER,PASSWD,LEUV1 <CR>
   ```

2. To enable TLS for `NTR1`, use:
   ```
   setNtripTlsSettings,NTR1,on,"" <CR>
   ```

3. The receiver will automatically receive and decode the RTK corrections from the NTRIP caster and switch to RTK positioning mode, unless RTK is disabled with the **`setPVTMode`** command.

4. Closing the NTRIP connection is done with the following command:
   ```
   setNTRIPSettings,NTR1,off <CR>
   ```

The status of the NTRIP client connection is reported in the `NTRIPClientStatus` SBF block.

1.11 Use the Built-In NTRIP Caster

The receiver contains an NTRIP caster, which is able to broadcast local data streams originating from the receiver itself, or streams from any remote NTRIP server. The hostname or IP address of the built-in caster is as defined in section 1.1.3.

1.11.1 Broadcasting Local Streams

This section explains how to use the built-in NTRIP caster to broadcast a local stream generated by the receiver's own NTRIP server.

1. Define the mount point you want to use for streaming the data. For example, the following command enables the first mount point, gives it the name "MyMP", and specify that this mount point only accepts local streams:
   ```
   setNtripCasterMountPoints, MP1, on, MyMP, No <CR>
   ```

2. Define the data format. For example, if the mount point defined above is meant to stream RTCM v3.x corrections, use the following command:
   ```
   setNtripCasterMPFormat, MP1, RTCMv3 <CR>
   ```

3. Define the NTRIP client accounts. For example, the command below enables an NTRIP client connecting as user "u1" and with password "p1" to receive data from the first mount point:
   ```
   setNtripCasterUsers, User1, u1, p1, MP1 <CR>
   ```

4. Configure the local NTRIP server to send data to the mount point, as explained in section 1.9. To have the local NTRIP server send data to the built-in caster, the hostname has to be set to "localhost". For example, to send data to the mount point "MyMP" of the caster, use:
   ```
   setNTRIPSettings,NTR1,Server,localhost,,,,MyMP <CR>
   ```

5. Enable the built-in NTRIP caster:
   ```
   setNtripCasterSettings, on <CR>
   ```

1.11.2 Broadcasting Remote Streams

To configure the caster to broadcast a stream originating from a remote NTRIP server, follow the following steps.

1. Define the mount point. For example, the following command enables the first mount point, gives it the name "MyMP", and specifies the credentials that the remote NTRIP server will need to use in order to feed data to this mount point ("FeedUser" and "FeedPwd"):
   ```
   setNtripCasterMountPoints, MP1, on, MyMP, Yes, FeedUser,
   FeedPwd <CR>
   ```

2. Define the stream data format. For example, if the mount point defined above is meant to stream RTCM v3.x corrections, use the following command:

```
setNtripCasterMPFormat, MP1, RTCMv3 <CR>
```

3. Define the NTRIP client accounts. Up to five client accounts can be configured. For example, the command below enables an NTRIP client connecting as user “u1” and with password “p1” to receive data from the first mount point:
   ```
   setNtripCasterUsers, User1, u1, p1, MP1 <CR>
   ```

4. Enable the built-in NTRIP caster:
   ```
   setNtripCasterSettings, on <CR>
   ```

From now on, the NTRIP caster is ready to receive a data stream from a remote NTRIP server and to distribute it to NTRIP clients.

1.12 Configure an IP Server Port

In this example, we show how to configure the receiver such that any client connecting to TCP/IP port 28785 will receive the NMEA GGA message at a 1-second interval.

1. Configure one of the IP server connections (see section 1.1.5) to listen to port 28785. Here, we assume that the first IP server connection (`IPS1`) is free:
   ```
   setIPServerSettings,IPS1,28785,TCP <CR>
   ```

2. Output the GGA NMEA message to the `IPS1` connection, at a 1-Hz rate:
   ```
   setNMEAOutput,Stream1,IPS1,GGA,sec1 <CR>
   ```

3. Make sure that NMEA output is enabled on the `IPS1` connection. It is enabled by default, but in case your receiver is not in its default configuration, you should invoke:
   ```
   setDataInOut,IPS1,,+NMEA <CR>
   ```

A way to check the IP server functionality is to enter the URL **http://*polarx5tr-xxxxxxx*:28785** in your preferred web browser (replace *`polarx5tr-xxxxxxx`* by the hostname of your particular receiver). You should see the NMEA GGA message coming every second.

Note that up to eight clients can concurrently connect to the same IP server port.

The example above showed how to set up a TCP server. It is also possible to configure the receiver in UDP server mode. For example, to broadcast the GGA message to any UDP client listening to its port 28785, the command in step 1. above must be replaced by:

```
setIPServerSettings,IPS1,28785,UDP,255.255.255.255 <CR>
```

Conversely, the receiver can be configured to automatically receive data from an IP server. This is explained in section 1.13.

1.13 Configure an IP Receive Port

The receiver can be configured to automatically receive data (typically differential corrections) from an IP server. In this example, we show how to connect to an IP server having the hostname `MyServer` and using port 28786.

1. Configure one of the IP receive connections (see section 1.1.5) to listen to port 28786 of `MyServer`. Here, we assume that the first IP receive connection (`IPR1`) is free:
   ```
   setIPReceiveSettings,IPR1,28786,TCP2Way,MyServer <CR>
   ```

2. If the data stream from the IP server contains differential corrections in CMR or RTCM format, the receiver will automatically decode them and use them in the PVT processing.

3. To close the connection, enter the following command:
   ```
   setIPReceiveSettings,IPR1,0 <CR>
   ```

The TCP connection initiated by the receiver is bidirectional. Once the connection is established, the receiver accepts input data from the server (as shown above), but it can also send data to the server, or process user commands from the server.

The example showed how to set up a TCP connection with the server. The receiver can also listen to incoming UDP messages. In that case, the connection is unidirectional and the server address or hostname must not be specified. For example, to listen to UDP messages on port 28786, use the command:

```
setIPReceiveSettings,IPR1,28786,UDP <CR>
```

1.14 Manage Power Saving Mode

The receiver can put itself in low-power mode when its supply voltage drops below a user-configurable level. This can for example be used to prevent battery over-discharge when the receiver is powered from an external battery.

Power saving is configured with the **`setPowerThresholds`** and **`setStandbyMonitoring`** commands. When the supply voltage at the PWR connector drops below the threshold set with the **`setPowerThresholds`** command, the receiver enters low-power sleep mode for a duration given by **`setStandbyMonitoring`**. At the end of this period, the receiver wakes up and either resumes operation normally if the supply voltage is sufficient, or otherwise returns into sleep mode after about 30 seconds for a new period of time.

For example, to configure the receiver to sleep for one hour when the external supply voltage drops below 11.6V, use the following commands:

```
setStandbyMonitoring, 3600 <CR>

setPowerThresholds, 11.6 <CR>

exeCopyConfigFile, Current, Boot <CR>
```

The last command makes sure to save the configuration in non-volatile memory (see section 1.5), so that power monitoring remains active when the receiver wakes up.

During the sleep period, power-cycling the receiver will make it alive for at least 30 seconds before it goes back into sleep if the supply voltage is too low. This should give enough time to change the settings, or disable power saving if necessary.

Note that it is also possible to have the receiver sleep and wake up following a user-defined pattern, independently of the supply voltage. See the `ScheduledSleep` mode in the **`exePowerMode`** command.

1.15 Manage Log Sessions

Logging is done in multiple logging sessions referred to as `LOGx` (see also section 1.1.5). Each log session corresponds to a given directory on the disk (internal or USB drive).

For each session, the user can specify where to log (`DSK1`: internal disk, or `DSK2`: USB drive), the session name, how long to keep the log files, and the session priority.

Log sessions are configured with the **`setLogSession`** command. By default, logging is continuous, but it is also possible to configure a logging schedule, for example to log for one hour every day. See the **`setLogSessionSchedule`** command for details.

To prevent the disk from becoming full, log files are automatically deleted after the interval specified with the **`setLogSession`** command. Should the disk become full anyway, the receiver will either stop logging or attempt to remove old files according to the settings of the **`setDiskFullAction`** command. When attempting to remove old files, the receiver will first consider the log sessions with low priority. Refer to the description of the **`setDiskFullAction`** command for details.

It is possible to prevent auto-deletion of important files covering an event of interest (e.g. an Earthquake). The event of interest must be signaled to the receiver through one of its Event inputs (see section 1.28). Refer to the **`setPreserveOnEvent`** command for details.

Each session can log SBF files (see section 1.16), BINEX files (see 1.19), NMEA files (see 1.20), RTCM files (see 1.18), CGGTTS files (see 1.22) and RINEX files (see 1.17).

The log files can be automatically FTP-pushed to a remote server. A different FTP server can be configured for each log session, and different files types logged in the same session can be sent to different servers. Refer to section 1.25 to learn how to configure FTP push.

1.16 Log SBF Files

Enabling SBF logging on one of the log sessions involves the following steps:

1. Define the SBF naming convention and file duration in the log session of interest. By default, the receiver logs SBF blocks into a file named "`log.sbf`". You can specify any other file name, or you can select the IGS/RINEX naming convention, where the file name automatically changes every fifteen minutes, hour, six hours or day. File naming conventions can be selected independently for each log session. For instance, to let the receiver create daily files in the second log session (`LOG2`), use:
   ```
   setFileNaming,LOG1,IGS24H <CR>
   ```

   If the file name you selected already exists, the receiver will append new data at the end of the existing file.

2. Use the command **`setSBFOutput`** to define which SBF blocks need to be logged and at which interval (see also section 1.3). For instance, to log all SBF blocks necessary to build RINEX files in the `LOG2` session, with a data interval of 10 seconds, use:
   ```
   setSBFOutput,Stream1,LOG1,rinex,sec10 <CR>
   ```

3. Make sure that the log session is defined and enabled:
   ```
   setLogSession,LOG1,enabled <CR>
   ```

4. To stop SBF logging, disable the SBF stream as follows:
   ```
   setSBFOutput,Stream1,none <CR>
   ```
   or temporarily disable the log session:
   ```
   setLogSession,LOG1,disabled <CR>
   ```

Note that the maximum file size supported by the receiver is 4GBytes.

Section 1.15 explains how to set up log sessions, section 1.25 shows how to automatically FTP push SBF files to a remote server and section 1.23 shows how to download log files.

1.17 Log RINEX Files

The receiver can log RINEX observation, navigation and meteo files on its internal disk. RINEX v2.11, 3.04, 3.05 and 4.00 are supported.

RINEX logging is configured independently for each log session.

Internal RINEX logging is typically configured as follows:

1. The information needed to generate the RINEX file names is taken from the **setMarkerParameters** command. For example, to set the station name designator (the first four characters of the RINEX file names) to "`LEUV`", use:
   ```
   setMarkerParameters, , , , LEUV <CR>
   ```

2. The fields in the RINEX observation header are specified with the **setMarkerParameters**, **setObserverParameters** and **setAntennaOffset** commands. For example, if the observer's name is "`MyName`" and its agency is "`MyAgency`", use the command:
   ```
   setObserverParameters, MyName, MyAgency <CR>
   ```

3. For static installations, the reference marker position to put in the "APPROX POSITION XYZ" header line can be defined with the *RefPos* argument of **setPVTMode**, with the marker-to-ARP offset being defined with **setAntennaOffset**. For example, assuming the WGS84 position of the ARP is 50.5^{o}N, 4^{o}E and its altitude above the WGS84 ellipsoid is 100m, and the ARP is 1.5 meters above the marker, use:
   ```
   setStaticPosGeodetic,Geodetic1,50.5,4,100 <CR>
   setPVTMode, , ,Geodetic1 <CR>
   setAntennaOffset,Main,0,0,1.5 <CR>
   ```

4. Use the **setRINEXLogging** command to configure the RINEX options: the file duration, the observation interval, etc. This is done independently for each log session. For example, to log daily RINEX files with the observation file containing only GPS L1CA data at a 30-s interval in the `LOG1` log session, use:
   ```
   setRINEXLogging, LOG1, Hour24, sec30, GPSL1CA <CR>
   ```

Note that the maximum file size supported by the receiver is 4GBytes.

Section 1.15 explains how to set up log sessions, section 1.25 shows how to automatically FTP push RINEX files to a remote server and section 1.23 shows how to download the RINEX files.

Instead of logging RINEX files inside the receiver, you can also convert an SBF file to RINEX using the `sbf2rin` program or the `SBFConverter` graphical tool.

1.18 Log RTCM-MSM Files

The receiver can log RTCM-MSM messages (a subset of the RTCM v3 messages) on its internal disk.

Internal RTCM-MSM logging is typically configured as follows:

1. The information needed to generate the RTCM-MSM file names is taken from the **`setMarkerParameters`** command. For example, to set the station name designator (the first four characters of the file names) to "`LEUV`", use:
   ```
   setMarkerParameters, , , , LEUV <CR>
   ```

2. The antenna information in message types 1008 or 1033 can be specified using the **`setAntennaOffset`** command, with the serial number as sixth argument, and the antenna type (called "antenna descriptor" in RTCM) as fifth argument (see also section 2.5). For instance:
   ```
   setAntennaOffset,Main, , , ,"AT2775-54SW","5684" <CR>
   ```

3. The reference position in message type 1006 is defined with the *RefPos* argument of **`setPVTMode`**. For example, assuming the WGS84 position of the ARP is 50.5^{o}N, 4^{o}E and its altitude above the WGS84 ellipsoid is 100m, use:
   ```
   setStaticPosGeodetic,Geodetic1,50.5,4,100 <CR>
   setPVTMode, , ,Geodetic1 <CR>
   ```

4. Use the **`setRTCMMSMLogging`** command to configure the RTCM-MSM logging options: the file duration, the observation interval, etc. This is done independently for each log session. For example, to log daily RTCM-MSM files in the first log session, with the files containing observations at a 30-s interval in MSM5 format, together with station- and ephemerides-related messages, use:
   ```
   setRTCMMSMLogging, LOG1, Hour24, off, MSM5+Nav+Station, sec30
   <CR>
   ```

Note that the maximum file size supported by the receiver is 4GBytes.

Section 1.15 explains how to set up log sessions, section 1.25 shows how to automatically FTP push RTCM files to a remote server and section 1.23 shows how to download the RTCM-MSM files.

1.19 Log BINEX Files

The receiver can log BINEX files on its internal disk. BINEX logging is configured independently for each log session.

Internal BINEX logging is typically configured as follows:

1. The BINEX file naming convention is inspired from the RINEX 2.11 file naming convention. The difference with RINEX is that the file type character is omitted and that the `.bnx` suffix is added. See the description of the **setBINEXLoggingParameters** command for details. The four-character station code must be specified with the **setMarkerParameters** command. For example, to set the station code to "`LEUV`", use:
   ```
   setMarkerParameters, , , , LEUV <CR>
   ```

2. Use the **setBINEXLoggingParameters** command to specify the BINEX file duration (fifteen minutes, one hour, six hours or one day). This is done independently for each log session. For example, to log daily BINEX files in the `LOG1` log session, use:
   ```
   setBINEXLoggingParameters, LOG1, Hour24 <CR>
   ```

3. Use the **setBINEXOutput** command to define which BINEX records need to be logged and at which interval (see also section 1.4). For instance, to log observation records (0x7F-05) at 10-second interval together with all decoded navigation records and metadata (0x00) at the beginning of the file, use:
   ```
   setBINEXOutput, Stream1, Rec7F05+Rec01Nav+Rec00, sec10 <CR>
   ```

4. Make sure that the log session is defined and enabled:
   ```
   setLogSession,LOG1,enabled <CR>
   ```

5. To stop BINEX logging, disable the BINEX stream as follows:
   ```
   setBINEXOutput,Stream1,none <CR>
   ```
   or temporarily disable the log session:
   ```
   setLogSession,LOG1,disabled <CR>
   ```

Note that the maximum file size supported by the receiver is 4GBytes.

Section 1.15 explains how to set up log sessions, section 1.25 shows how to automatically FTP push BINEX files to a remote server and section 1.23 shows how to download the BINEX files.

1.20 Log NMEA Files

Enabling NMEA logging on one of the log sessions involves the following steps:

1. Define the NMEA naming convention and file duration in the log session of interest. By default, the receiver logs NMEA into a file named "`log.nma`". You can specify any other file name, or you can select the IGS/RINEX naming convention, where the file name automatically changes every fifteen minutes, hour, six hours or day. File naming conventions can be selected independently for each log session. For instance, to let the receiver create daily files in the second log session (`LOG2`), use:
   ```
   setNMEALogging,LOG1,IGS24H <CR>
   ```
   If the file name you selected already exists, the receiver will append new data at the end of the existing file.

2. Use the command **`setNMEAOutput`** to define which NMEA sentences need to be logged and at which interval. For instance, to log the GGA sentences at an interval of 10 seconds in the `LOG2` session, use:
   ```
   setNMEAOutput,Stream1,LOG1,GGA,sec10 <CR>
   ```

3. Make sure that the log session is defined and enabled:
   ```
   setLogSession,LOG1,enabled <CR>
   ```

4. To stop NMEA logging, disable the NMEA stream as follows:
   ```
   setNMEAOutput,Stream1,none <CR>
   ```
   or temporarily disable the log session:
   ```
   setLogSession,LOG1,disabled <CR>
   ```

Note that the maximum file size supported by the receiver is 4GBytes.

Section 1.15 explains how to set up log sessions, section 1.25 shows how to automatically FTP push NMEA files to a remote server and section 1.23 shows how to download log files.

1.21 Estimate the Size of the Log Files

The table below compares the file size for different logging formats, with and without compression. It shows the disk space required to log all code and carrier phase measurements from all GNSS signals over one day at a 1-Hz rate.

The values have been measured in October 2020 in Belgium. They will change depending on the location and will also increase as more satellites are launched, but the table gives a good idea of the relative size between the different logging options.

Data format	File Size (uncompressed)	File Size (gzip-compressed)
SBF (Rinex group)	261 MB	186 MB
SBF (RinexMeas3 group)	70 MB	64 MB
RINEX (v3)	638 MB	64 MB (Hatanaka+gzip)
BINEX (7F-05)	143 MB	118 MB
RTCM-MSM (MSM4)	132 MB	96 MB

As can be seen, the smallest files are obtained when logging the RinexMeas3 SBF group (see the **`setSBFOutput`** command in section 1.16), or with the compressed Hatanaka format.

1.22 Log CGGTTS Files

The receiver can generate and log CGGTTS-v2E files for the GPS, GLONASS, Galileo and BeiDou constellations.

Internal CGGTTS logging is typically configured as follows:

1. Specify the antenna phase center position (not the ARP!) to be used for the CGGTTS file generation. This is done with the **`setCGGTTSPosition`** command. The specified position only affects the CGGTTS processing and has otherwise no effect on the receiver operation. For example, use:
   ```
   setCGGTTSPosition, all, manual, 4021429.16, 332877.37,
   4923031.62 <CR>
   ```

2. Configure all CGGTTS parameters and relevant electrical delays with the **`setTimeLabParameters`**, **`setCGGTTSTrackParameters`**, **`setINTDLY`**, **`setCABDLY`** and **`setREFDLY`** commands. For example, if the antenna+receiver electrical delay (INTDLY) at the GPSL1 frequency has been calibrated to be 30.2ns, use:
   ```
   setINTDLY, GPSL1, 30.2 <CR>
   ```

3. Enable CGGTTS logging. For example, to enable CGGTTS logging for the GPS and Galileo constellations in the `LOG1` log session, use:
   ```
   setCGGTTSLogging, LOG1, GPS+GALILEO <CR>
   ```

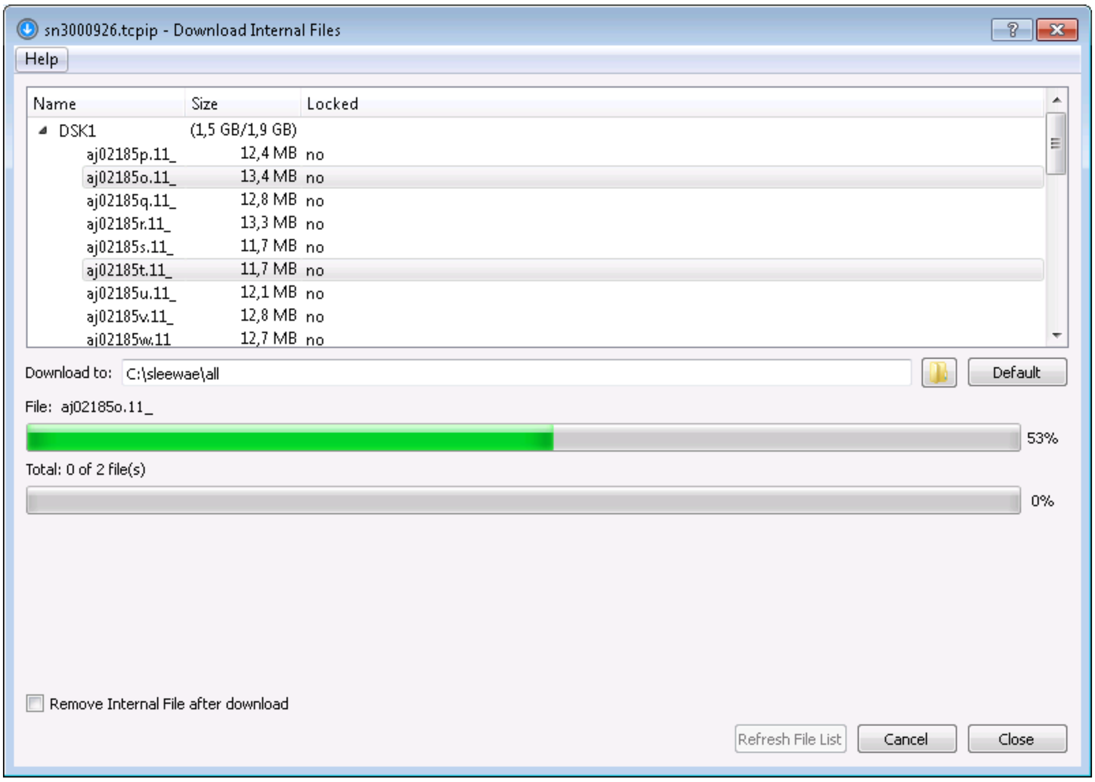
Section 1.15 explains how to set up log sessions, section 1.25 shows how to automatically FTP push CGGTTS files to a remote server and section 1.23 shows how to download the CGGTTS files.

Instead of logging CGGTTS files inside the receiver, you can also convert an SBF file to CGGTTS using the `sbf2cggtts` program or the `SBFConverter` graphical tool.

1.23 Download Log Files from the Receiver

There are different ways to download or delete files from the internal disk:

1. Using RxControl. Select *Logging* > *Download Internal Files* to download files to your computer, and *Logging* > *Remove Internal File* to remove a file from the internal disk.



sn3000926.tcpip - Download Internal Files

Name	Size	Locked
DSK1	(1,5 GB/1,9 GB)	
aj02185p.11_	12,4 MB	no
aj02185o.11_	13,4 MB	no
aj02185q.11_	12,8 MB	no
aj02185r.11_	13,3 MB	no
aj02185s.11_	11,7 MB	no
aj02185t.11_	11,7 MB	no
aj02185u.11_	12,1 MB	no
aj02185v.11_	12,8 MB	no
aj02185w.11	12,7 MB	no

Download to: C:\sleewae\all

File: aj02185o.11_

53%

Total: 0 of 2 file(s)

0%

☐ Remove Internal File after download

Figure 1-2: Download Internal Files from RxControl.

2. Using FTP or SFTP. The hostname or fixed IP address is defined as explained in section 1.1.3. For example, if your receiver's hostname is `polarx5tr-1234567`, you can type the following URL in your preferred web browser to open a session as anonymous user:
   `ftp://polarx5tr-1234567`

   User authentication for SFTP access can be done by entering a password or using an ssh public key, as defined with the **`setUserAccessLevel`** command.

   By default, anonymous users can download and delete files. This can be changed as explained in section 1.31.

3. Using `rsync`. If an `rsync` client is available on your computer, you can use `rsync` to download files or directories from the receiver. The hostname or IP address is the same as for (S)FTP, see above.

   For example, to download the contents of the `LOG1_TST/16134` folder of the internal disk (DSK1) to the current folder on your local computer, you could invoke `rsync` as follows:
   `rsync -r anonymous@polarx5tr-1234567:DSK1/SSN/LOG1_TST/16134 .`
   when prompted for a password, just press the enter key as no password is required for

anonymous accesses.

If the same command is issued again at a later stage, `rsync` will only transfer the deltas with respect to the files already present on the local machine, significantly reducing the number of bytes sent compared to retransmitting the entire files.

User authentication can be done by entering a password, or using an ssh public key, as defined with the **`setUserAccessLevel`** command.

By default, `rsync` is enabled for anonymous users. This can be changed with the **`setDefaultAccessLevel`** command.

4. Using the web interface (select the *Logging* tab).

5. Using a standard file browser and accessing the receiver as a removable drive (USB mass-storage device). This requires the USB cable to be connected to your computer, and the internal disk to be unmounted by the receiver so that it can be accessed by your computer's operating system. This is done using the **`exeManageDisk`** command. For example, to see the internal disk (DSK1) as a removable drive in your file browser, enter the command:

```
exeManageDisk,DSK1,Unmount <CR>
```

Internal logging is disabled on a disk that is unmounted. To resume logging by the receiver, the internal disk must be mounted again using the command:

```
exeManageDisk,DSK1,Mount <CR>
```

On Linux, it is needed to first eject the removable drive before entering this command.

1.24 CloudIt Workflow (beta)

⚠ CloudIt is a beta feature. The instructions in this “How to“ section are subject to changes.

1.24.1 Introduction

CloudIt offers an alternative to FTP log file submission from the PolaRx5TR receivers. CloudIt supports OAuth2 for authentication. The implementation is compliant with the specification of “The OAuth 2.0 Authorization Framework“.

Note that CloudIt only supports the “Authorization Code“ grant to comply with the security requirements and the authorization process for third party applications.

1.24.2 Server Setup

CloudIt expects the user to set up two servers, an authentication server and a resource server. The authentication server is used to verify if the receiver running CloudIt can access the resource server. The resource server stores the data files from the receiver.

In this section, the difference between the authentication and resource servers is used towards the network administrator. However inside the receiver configuration, this difference is invisible to the end user, as both the authentication and resource servers are, on the receiver, one Cloudit *Server*, such as in the **`setCloudItConfig`** command.

1.24.2.1 Authentication Server Setup

Keycloack is recommended as an Identity and Access Management platform for the authentication server, as CloudIt has been tested and validated using this tool (Keycloack.4.5.0). Other Access Management platforms were not tested but can offer similar possibilities. However, since most other tools offer more specific and/or proprietary implementations of the OAuth2 protocol, CloudIt might not (yet) be compatible with these.

Inside the authentication server it is required to create an application. At this level it should be possible to get a *ClientID* and a *ClientSecret* and to define the *Scope*. The *Scope* should, as an example, include the permission to write on your resource server.

The Access Management platform also provides the Authentication URL (identified in the receiver as *AuthURL*) and Token URL (identified in the receiver as *TokenURL*). The exact procedure on how to get this information depends on the Access Management platform setup.

These arguments will be used to configure CloudIt using **`setCloudItConfig`**.

Currently, CloudIt only supports the stateless access/refresh token authorization process. If the authorization server does not respond with both the access and refresh tokens, the authorization will fail.

The authorization server should be configured to accept authorization requests without a redirect_uri. However, the server will need to redirect all successful authorization requests to a web page that displays the authorization code, referenced in this document as *Authcode*. The user will use this code later in step 3 of the Receiver configuration of CloudIt.

1.24.2.2 Resource Server Setup

The resource server stores the data files from the receiver. This server should be configured to accept HTTP PUT requests only from authorized receivers. When authorized, the receiver will send HTTP PUT requests to the server to perform a `multipart` upload. The sent request is the following:

```
curl -H "Authorization: Bearer 654sdf542sdfkjsdf5484fgh74" \
     -F "path=/my/remote/directory" \
     -F "file=@/localpath/myfile.sbf;type=application/octet-stream" \
     "https://myresourceserver.com/api/upload/?uploadType=multipart")
```

Please note that the remote directory (where the file will be stored in the server) is sent via the *Path* argument. It is the responsibility of the server to interpret it and to manage, if needed, the creation of the folder. The remote directory, defined by the *Path*, can be set with the **`setSBFCloudIt`**, **`setRINEXCloudIt`**, **`setBINEXCloudIt`** or **`setRTCMMSMCloudIt`** commands depending on the type of files to upload. The *UploadURL* parameter defines the upload endpoint accepting the uploaded files. This parameter is required to configure the CloudIt server using the **`setCloudItConfig`** command.

1.24.3 Receiver Configuration of CloudIt

When the CloudIt authentication and resource servers are up and running, the PolaRx5TR can be configured following the steps below:

1. Set the CloudIt server parameters using **`setCloudItConfig`** command.
2. Use the **`lstAuthorizationLinkCloudIt`** command to get an authorization link. This link needs to be visited through a web browser where the user needs to login and authorize the receiver. When authorized, the user will be redirected to a web page displaying the authorization code on the page or in the URL.
3. Use **`exeAuthorizeCloudIt`** to authorize the receiver using the code given in step 2.

1.24.4 File Upload

When a log session is enabled, it is possible to upload the internally-logged files to a remote CloudIt server with the **`setSBFCloudIt`**, **`setBINEXCloudIt`**, **`setNMEACloudIt`**, **`setRTCMMSMCloudIt`** and **`setRINEXCloudIt`** commands depending on the file type. It is possible to upload each file type to a different server. The user can configure up to 8 servers.

For example, to configure the LOG2 log session to automatically upload RINEX files to the directory `mydata/rin/` on the remote server Cloud1, the following command needs to be entered:

```
setRINEXCloudIt, LOG2, Cloud1, mydata/rin/ <CR>
```

Note that the folder structure on the remote server will be similar to the folder structure on the receiver. For example, if you have a file in the receiver under `LOG2/19141`, the actual *Path* sent to the server in the HTTP upload request is: `mydata/rin/LOG2/19141`.

If the file transfer fails, the receiver will retry at a user-defined interval. See the description of the **setSBFCloudIt**, **setBINEXCloudIt**, **setNMEACloudIt**, **setRTCMMSMCloudIt** and **setRINEXCloudIt** commands for details.

Note: Within one log session and file type, it is possible to configure CloudIt and FTPPush (section 1.25) at the same time. However, if the user enables both features for the same log session and file type, only FTPPush will be processed.

1.25 FTP Push Log files

It is possible to configure the receiver to automatically send internally-logged files to a remote FTP server (FTP Push). This is done with the **`setSBFFTP`**, **`setBINEXFTP`**, **`setNMEAFTP`**, **`setRTCMMSMFTP`**, **`setCGGTTSFTP`** and **`setRINEXFTP`** commands depending on the file type. It is possible to configure the FTP server independently for each log session.

For example, to configure the `LOG2` log session to automatically FTP RINEX files to the directory `mydata/rin/`*`YYDDD`* (with *`YY`* and *`DDD`* the year and day-of-year) on the remote server `myftp.com`, with username `myname` and password `mypwd`, you would enter the following command:

```
setRINEXFTP, LOG2, myftp.com, mydata/rin/%y%j, myname, mypwd <CR>
```

FTP push will create the folder on the remote server if it does not exist yet.

If the file transfer fails, the receiver will retry at a user-defined interval. See the description of the **`setSBFFTP`**, **`setBINEXFTP`**, **`setNMEAFTP`**, **`setRTCMMSMFTP`** and **`setRINEXFTP`** commands for details.

Note: Within one log session and file type, it is possible to configure CloudIt (section 1.24) and FTPPush at the same time. However, if the user enables both features for the same log session and file type, only FTPPush will be processed.

For example, if the user issues the following commands: **`setSBFCloudIt, LOG1, Cloud1, mydata/rin/ <CR>`**

```
setSBFFTP, LOG1, myftp.com, mydata/rin/, myname, mypwd <CR>
```

the SBF files created in session LOG1 will only be pushed to the FTP server. If the user wants to upload the same files with CloudIt another log session with the same configuration needs to be configured.

1.26 Communicate with External Equipment

The receiver can send periodical queries to external equipment (such as a meteo sensor) connected to one of its serial ports, and log the replies from that sensor. In the following example, we show how to retrieve meteo data every 10 seconds from a meteo sensor connected to the receiver's COM2 port.

1. Tell the receiver which command to use to query the external sensor, and the interval at which this command must be sent to the sensor. For instance, for a MET3/MET4-compatible sensor, the command `*0100P9<CR><LF>` queries the meteo data. Assuming you want to get meteo data at a 10-second interval, enter the following command:
   `setPeriodicEcho, com2, A:*0100P9%%CR%%LF, sec10 <CR>`

2. Enable unformatted ASCII input on COM2 (to receive the replies from the meteo sensor):
   `setDataInOut,COM2,ASCIIIn <CR>`

The replies from the meteo sensor (containing the temperature, pressure and humidity) are available in the `ASCIIIn` SBF block and in BINEX record 0x7E-01.

You can convert an SBF file containing `ASCIIIn` SBF blocks to RINEX using the `sbf2rin` program or the `SBFConverter` graphical tool. Alternatively, you can log the RINEX meteo file directly on the receiver (see section 1.17). To be able to generate a RINEX file, the output of the meteo sensor must be formatted according to the NMEA XDR sentence.

1.27 Generate a "Pulse Per Second" Signal

The receiver is able to generate an x-pulse-per-second (xPPS) signal aligned with a selected GNSS system time, with UTC or with the internal receiver time.

The **`setPPSParameters`** command is used to set the xPPS parameters (rate, polarity, time system, ...). For instance, to synchronize the xPPS pulses with UTC and have one pulse every ten seconds, use:

```
setPPSParameters,sec10, , ,UTC <CR>
```

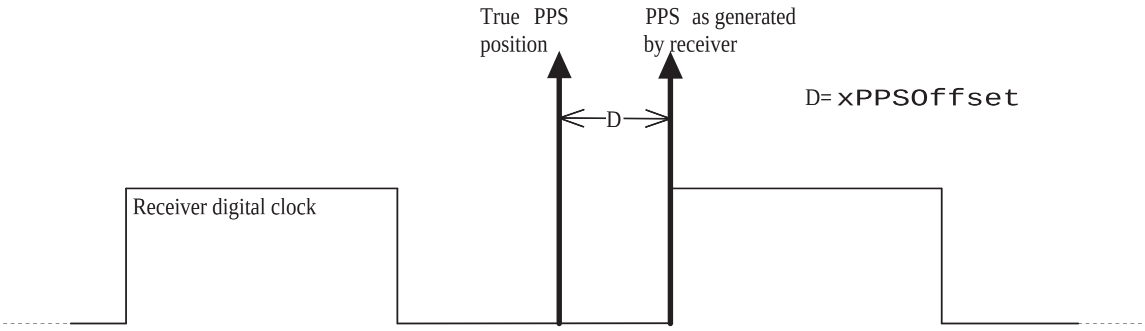


Figure 1-3: xPPS output granularity.

Although the position of the PPS pulse is computed accurately by the receiver, the actual pulse is generated at the nearest "tick" of the internal receiver digital clock, as illustrated in the figure above. This leaves an offset (noted "D" in the figure) between the true xPPS pulse and the one actually generated by the receiver. This offset can reach a few nanoseconds. It is available in real-time in the `xPPSOffset` SBF block.

To be able to align its xPPS output with the GNSS system time, the receiver needs a fresh estimate of the GNSS time from its PVT solution. If the last PVT solution is older than a prescribed timeout (set by the **`setPPSParameters`** command), no PPS pulse is generated. In addition, to align its PPS with UTC, the receiver needs to have received the UTC offset parameters from the satellite navigation messages. If these parameters are not available and the user has requested to align the xPPS with UTC, no xPPS pulse is generated.

1.28 Time Tag External Events

The receiver can time-tag electrical level transitions on its Event*X* inputs with an accuracy of 20ns.

By default, the receiver reacts on low-to-high transitions. You can use the **`setEventParameters`** command to react on falling edges instead:

```
setEventParameters,EventA,High2Low <CR>
```

Upon detection of a transition, the receiver can output the time and/or the position at the instant of the event (see for example the `ExtEvent` SBF block).

The following constraints must be observed to ensure proper event detection:

- There must be no more than 20 events in any interval of 100 milliseconds, all event pins considered.
- The minimum time between two events on the same Event*X* input must be at least 5ms.

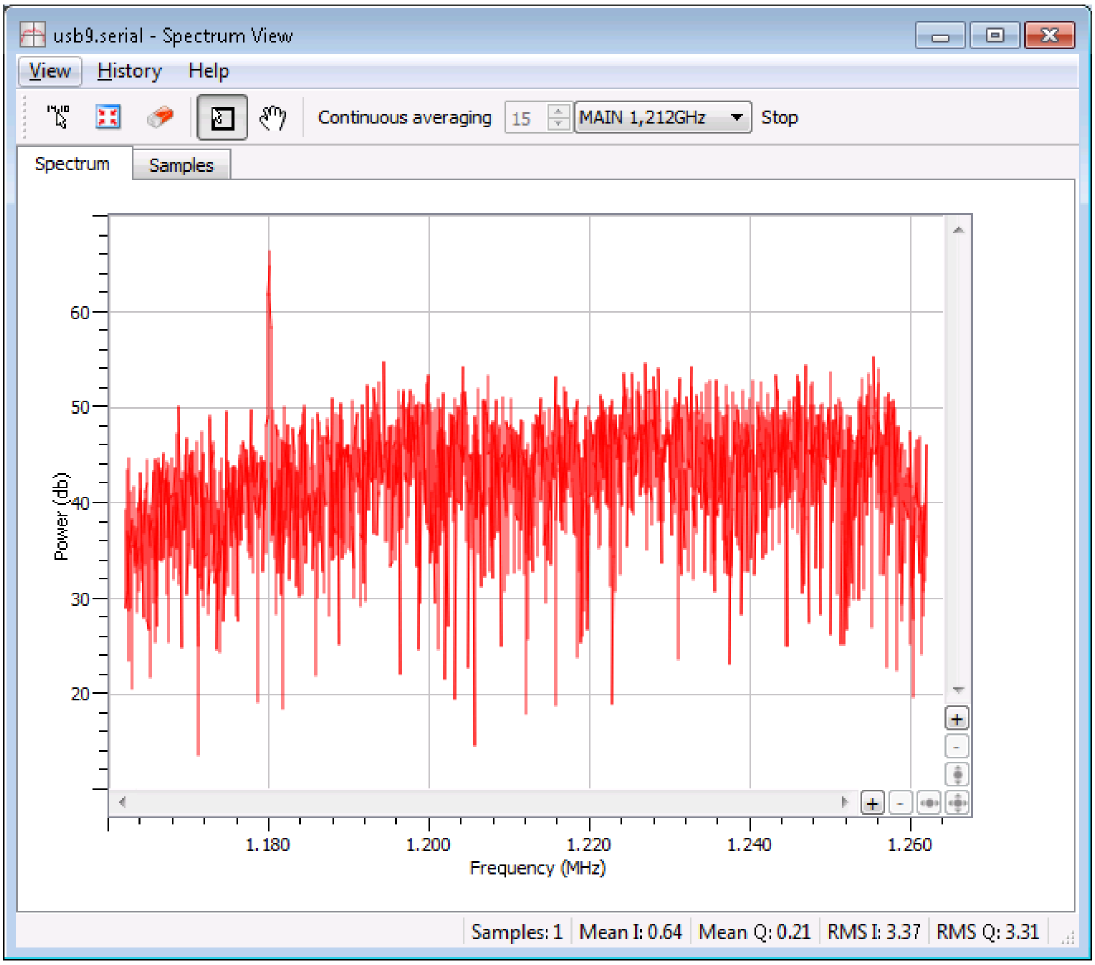
Missed events are flagged by the `MISSEDEVENT` bit in the `ReceiverStatus` SBF block.

The external event inputs can also be used to flag log files as “to be preserved”. If this is enabled, files logged during the occurrence of an external event are preserved from auto-deletion. Refer to the **`setPreserveOnEvent`** command for details.

1.29 Monitor the RF Spectrum

You can monitor the RF spectrum using the spectral analyzer in RxControl (go to the *View* > *Spectral View* menu) or in the web interface (go to the *GNSS* > *Spectrum* menu). This allows to detect the presence of interferences in the GNSS bands.

In the example shown below, a narrowband interference at 1180 MHz is clearly visible.



Samples: 1 | Mean I: 0.64 | Mean Q: 0.21 | RMS I: 3.37 | RMS Q: 3.31

Figure 1-4: Spectral Analyser functionality of RxControl.

The spectrum is computed from baseband samples taken at the output of the receiver's analog to digital converters. These samples are available to the users in the `BBSamples` SBF block.

1.30 Use Galileo OSNMA

With the Open Service Navigation Message Authentication (OSNMA) feature, Galileo satellites allow to verify the authenticity of navigation messages received from GNSS satellites, offering a powerful means to detect and counter spoofing attacks.

In Septentrio receivers, OSNMA is used to discard untrusted satellites from the PVT computation. Three operating modes are supported: *off* where OSNMA authentication is disabled, *loose* where satellites are included in the PVT if they are successfully authenticated or if their authentication status is unknown, and *strict* where only successfully-authenticated satellites are included in the PVT.

In *strict* mode, the number of satellites available to the PVT may be limited as OSNMA does not authenticate all visible satellites (e.g. only Galileo and GPS satellites depending on the OSNMA service status).

After enabling OSNMA (in *loose* or *strict* mode), it typically takes a few tens of seconds to a few minutes to authenticate messages. In *strict* mode, no PVT is computed during that time.

OSNMA is configured as follows:

1. Use the **`setGalOSNMAUsage`** command to enable OSNMA authentication. For example, to enable OSNMA in *loose* mode, use:
   ```
   setGalOSNMAUsage, loose <CR>
   ```

2. In strict mode, OSNMA authentication requires the availability of external time information. In loose mode, this is optional but recommended for enhanced security. The receiver can connect to an NTP time server for this purpose, as configured with the **`setNTPClient`** command. For example, to enable time retrieval from the default NTP server, use:
   ```
   setNTPClient, on, default
   ```

The receiver has been optimized for use with live Galileo signals. Necessary keys are embedded into the software. For example the Galileo Merkle Tree root key, needed to enable over-the-air reception of public keys, is known by the receiver, obviating the need for the user to provide it.

Refer to section 2.6 for further details.

1.31 Manage Users

When connecting to the receiver, users can remain "anonymous", or can log in using the **`login`** command. What anonymous users can do depends on the connection type. By default, anonymous users have full control of the receiver. This default configuration can be changed with the **`setDefaultAccessLevel`** command. For example, to prevent anonymous access to the web interface and to the FTP server, you would use:

```
setDefaultAccessLevel, none, none <CR>
```

To perform actions not allowed to anonymous users, you first need to authenticate yourself by entering a user name and a password through the **`login`** command. The list of user names and passwords and their respective access level is managed with the **`setUserAccessLevel`** command. Login fails if the provided user name or password is not in that list.

Logged-in users are granted one of the following access levels: "`User`" or "`Viewer`". The "`User`" level allows full control of the receiver, while the "`Viewer`" level only allows to view the configuration.

The following explains how to add or delete a user.

1. Check the current user list by entering the following command:
   ```
   getUserAccessLevel <CR>
   ```
   The reply to this command looks like:
   ```
   UserAccessLevel, User1, "admin", "R46NCG", User
   UserAccessLevel, User2, "", "", Viewer
   UserAccessLevel, User3, "", "", Viewer
   ...
   ```
2. In the example shown above, only one user is defined: `User1` with user name `admin`. For security reasons, the password shown here (`R46NCG`) is random and does not correspond to the actual password. It can be seen that the level of access of the `admin` user is "`User`": that particular user has full control of the receiver.
   To add a new user "`john`" with password "`abc123`" and to give full access to that user, select a free user index, e.g. `User2` in the above example, and type:
   ```
   setUserAccessLevel,User2,john,abc123,User <CR>
   ```
3. You can add up to eight users in this way. Deleting a user involves entering an empty string ("") as user name and password. For example, to delete the "`admin`" user from the above list, use:
   ```
   setUserAccessLevel,User1,"","" <CR>
   ```

The user list also applies to FTP, SFTP and `rsync` accesses. Users having the "`User`" access right are allowed to delete files from the internal disk via FTP, SFTP or `rsync`, while "`Viewer`" users can only download files.

1.32 Upgrade the Receiver

Upgrading the receiver is the process of installing a new GNSS firmware, a new permission file (see section 1.34) or a new antenna calibration file (see section 2.5).

⚠ In some cases, upgrading the GNSS firmware can clear the receiver configuration stored in non-volatile memory (see section 1.5). It is therefore advised to recheck the configuration after the upgrade.

⚠ Do not switch power off during the upgrade procedure.

⚠ Upgrading the receiver over a serial port can be very slow and it is recommended to upgrade using a faster connection whenever possible (USB, WiFi or Ethernet).

Septentrio upgrade files have the extension ".suf". There are several ways to upgrade the receiver:

1. By double clicking the ".suf" file. This should launch the RxUpgrade program.
2. By using the RxControl graphical interface (go to the *File* menu).
3. From the web interface (go to *Admin >Upgrade*). This requires to log in as a user with the "`User`" access level (see section 1.31).
4. By commanding the receiver to upgrade itself by fetching the upgrade file from a remote FTP server. This is done with the command **`exeFTPUpgrade`**.
5. By manually downloading upgrade files to the receiver. This upgrade procedure is explained below.

To manually upgrade the receiver, follow this procedure:

1. Reset the receiver into upgrade mode by entering the following command:
   `exeResetReceiver, Upgrade, none <CR>`

2. Wait till the receiver outputs the string: "Ready for SUF download ...". From that moment on, the receiver is waiting for an upgrade file to be downloaded. The file download must start within 200 seconds, otherwise the receiver will restart in normal mode.

3. Download the upgrade file to the receiver. Any of the receiver connections can be used. Make sure to send the file in binary mode, i.e. without changing its contents. During the download, the receiver outputs a progress indicator at regular interval.

4. At the end of the download, the receiver automatically executes the upgrade instructions and restarts with the new firmware version. You can check the firmware version by entering the following command:
   `lif,Identification <CR>`

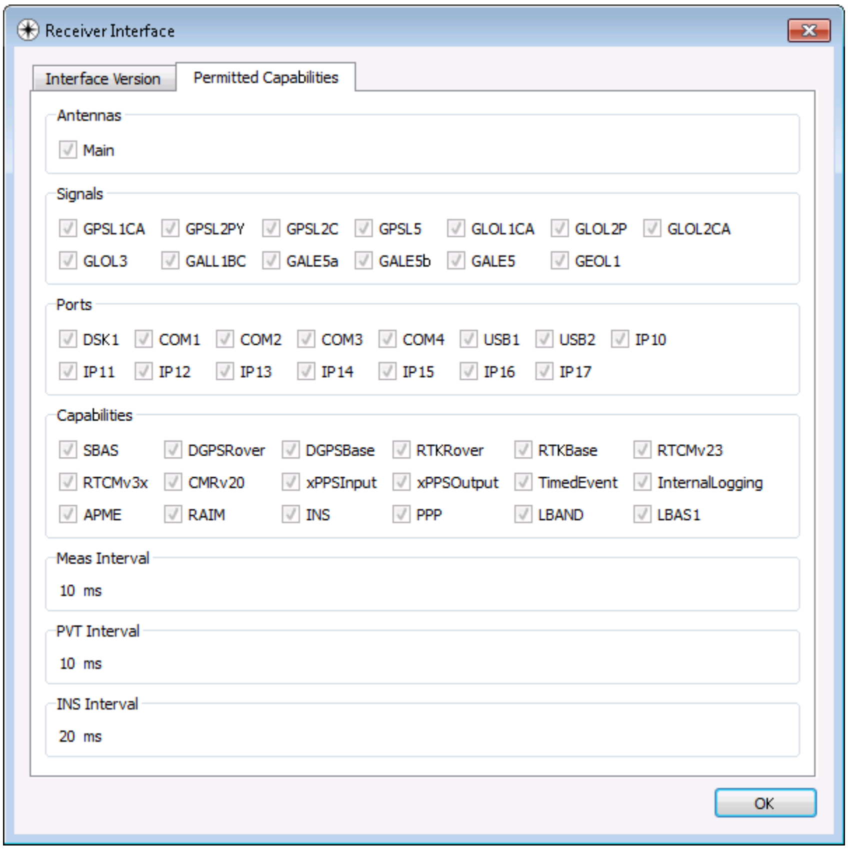
Before executing the upgrade instructions, the receiver checks the integrity of the downloaded file. If the file is corrupted, or is not a valid upgrade file, the receiver discards it and restarts in normal mode.

If the download is interrupted for any reason, the receiver will restart in normal mode after a timeout period of 200 seconds.

1.33 Check the Capabilities of your Receiver

The capabilities of your receiver are defined by the set of enabled features. The capabilities depend on the hardware, the current firmware version and the current set of permissions. Permissions are further explained in section 1.34.

The command **`getReceiverCapabilities`** lists the capabilities. You can also check them using the web interface (go to *Admin >About >Permitted Capabilities*) or RxControl (go to *Help >Receiver Interface* and select the *Permitted Capabilites* tab):



Antennas

☑ Main

Signals

☑ GPSL1CA ☑ GPSL2PY ☑ GPSL2C ☑ GPSL5 ☑ GLOL1CA ☑ GLOL2P ☑ GLOL2CA
☑ GLOL3 ☑ GALL1BC ☑ GALE5a ☑ GALE5b ☑ GALE5 ☑ GEOL1

Ports

☑ DSK1 ☑ COM1 ☑ COM2 ☑ COM3 ☑ COM4 ☑ USB1 ☑ USB2 ☑ IP10
☑ IP11 ☑ IP12 ☑ IP13 ☑ IP14 ☑ IP15 ☑ IP16 ☑ IP17

Capabilities

☑ SBAS ☑ DGPSRover ☑ DGPSBase ☑ RTKRover ☑ RTKBase ☑ RTCMv23
☑ RTCMv3x ☑ CMRv20 ☑ xPPSInput ☑ xPPSOutput ☑ TimedEvent ☑ InternalLogging
☑ APME ☑ RAIM ☑ INS ☑ PPP ☑ LBAND ☑ LBAS1

Meas Interval

10 ms

PVT Interval

10 ms

INS Interval

20 ms

Figure 1-5: Example of receiver capabilities.

1.34 Check or Change the Permission File

The permission file lists which optional features (such as GLONASS, Galileo, RTK, ...) are permitted on your receiver, for how long they are permitted and in which region they are permitted.

The permission file is stored in the receiver's non-volatile memory, and can be checked with the command **`lstInternalFile, Permissions`**, or with RxControl by clicking *Help >Receiver Permissions*.

Note that, for a given feature to be enabled in the receiver, it must be permitted and the hardware and firmware version must support it. See also section 1.33.

Each receiver is delivered with a permission file applicable to that receiver only. To enable new options, the user can order a new permission file to Septentrio, and install it on his/her receiver using the standard upgrade procedure (see section 1.32).

Chapter 2

Operation Details

This chapter describes the key processes implemented in the receiver and explains how they can be configured.

2.1 Channel Allocation and Signal Selection

The receiver automatically allocates satellites to tracking channels up to the limit of the number of channels. It is possible to override this automatic channel allocation by forcing a satellite to a given channel with the **`setChannelAllocation`** command. Also, a subset of satellites or a whole constellation can be disabled with the **`setSatelliteTracking`** command.

For each satellite, the receiver tries to track all signal types enabled with the **`setSignalTracking`** command. For example, if that command enables the GPSL1CA, GPSL2PY and GLOL1CA signals, GPS satellites will be tracked in dual-frequency mode (GPSL1CA and GPSL2PY) and GLONASS satellites will be tracked in single-frequency mode (GLOL1CA only). It is a good practice to only enable those signal types that are needed for your application to avoid wasting tracking channels.

2.2 Generation of Measurements

For each tracked GNSS signal, the receiver generates a "measurement set", mainly consisting of the following observables:

- a pseudorange in meters;
- a carrier phase in cycles;
- a Doppler in Hertz;
- a carrier-to-noise ratio in dB-Hz.

All data in a measurement set, and all measurement sets are taken at the same time, which is referred to as the "measurement epoch". All the measurement sets taken at a given measurement epoch are output in a `MeasEpoch` SBF block.

Several commands affect the way the receiver produces and outputs measurements:

- The **setHealthMask** command can be used to filter out measurements from unhealthy satellites: these measurements will not be used by the PVT algorithm, nor will they be included in the `MeasEpoch` SBF block.
- To further reduce the code measurement noise, the receiver can be ordered to smooth the pseudorange by the carrier phase. This technique, sometimes referred to as a "Hatch filtering", allows to reduce the pseudorange noise and multipath. It is controlled by the **setSmoothingInterval** command and is disabled by default.
- The **setMultipathMitigation** command can be used to enable or disable the mitigation of multipath errors on the pseudorange and carrier phase measurements. It is enabled by default.

For advanced applications or in-depth signal analysis, the `MeasExtra` SBF block contains various additional data complementing the `MeasEpoch` SBF block. Among other things, this block reports the multipath correction applied to the pseudorange (allowing one to recompute the original pseudorange), and the observable variances.

2.2.1 Pilot vs. Data Component

Most modern GNSS signals consist of two components: a so-called pilot component and a data component. For such signals, the measurements are based on the pilot component for optimal performance. In particular, the reported C/N_0 value is that of the pilot component only.

For all signals having a pilot and a data component, the table below indicates which component is tracked by Septentrio receivers. Note that your particular receiver model may not support all of these signals.

Signal	Signal component being used for measurement generation
GPS/QZSS L1C	L1C-P
GPS/QZSS L2C	L2C-L
GPS/QZSS L5	L5-Q
GLONASS L3	L3-Q
Galileo E1	E1-C
Galileo E6	E6-C or E6-B if E6-C is encrypted
Galileo E5a	E5a-Q
Galileo E5b	E5b-Q
Galileo E5AltBOC	E5AltBOC-Q
BeiDou B1C	B1C_pilot
BeiDou B2a	B2a_pilot
BeiDou B2b	B2b_I

See also the corresponding RINEX observation code in section 4.1.10.

2.3 Time Management

The receiver time is kept in two counters: the time-of-week counter in integer milliseconds (TOW) and the week number counter (WNc). TOW and WNc follow the GPS convention, i.e. WNc counts the number of complete weeks elapsed since January 6, 1980, and there are no leap seconds. The TOW and WNc counters are reported in all SBF blocks.

The synchronization of TOW and WNc involves the following steps:

- Upon powering up the receiver, TOW and WNc are assumed unknown, and set to a "Do-Not-Use value" in the SBF blocks.
- The transmission time-of-week and week number are decoded from the GNSS satellites (all constellations considered, not only GPS):
  - As soon as the first time-of-week is decoded, the TOW counter is initialized to within 20 ms of GPS time and starts counting. This is also the time when the receiver starts generating GNSS measurements (pseudoranges and carrier phases).
  - As soon as the week number is decoded (which can be either simultaneously with the time-of-week, or several seconds later), the WNc counter is set and starts counting.
- After the first position and time fix has been computed (for which measurements from at least 4 satellites are required), TOW is set to within X milliseconds of GPS time. This is done by introducing a jump of an integer number of milliseconds in the TOW counter. X is the maximal allowed offset between the receiver time and GPS time, and is set by the **`setClockSyncThreshold`** command (by default, X=0.5ms). This initial clock synchronization leads to a simultaneous jump in all the pseudorange and carrier phase measurements.

Note that in cases when the satellite visibility is obstructed, the receiver can also set its time from NMEA GPRMC or GPZDG sentences received on one of its serial ports. The time in the NMEA sentences must refer to the GPS time scale (and not to UTC), and the serial port receiving the NMEA sentences must be configured in ASCIIIn input mode with the **`setDataInOut`** command. This feature is only enabled when an external frequency and time pulse is provided on the REF IN and PPS IN connectors.

The synchronization level is given by three status bits (`TOWSET`, `WNSET` and `FINETIME`) available both in the `ReceiverTime` SBF block and the `ReceiverStatus` SBF block. Once the `FINETIME` bit is set, it remains set until the next reset of the receiver.

The receiver clock can be configured in free-running mode, or in steered mode using the command **`setClockSyncThreshold`**.

2.3.1 Free-Running Clock

In free-running mode, the receiver time slowly drifts with respect to GNSS time. The receiver continuously monitors this time offset: this is the clock bias term computed in the PVT solution, as provided in the `RxClkBias` field of the `PVTCartesian` and `PVTGeodetic` SBF blocks. A clock jump of an integer number of milliseconds is imposed on the receiver clock each time the clock bias exceeds X milliseconds by an absolute value (X is set by **`setClockSyncThreshold`**). This typically results in a saw-tooth profile similar to that shown in Figure 2-1. In this example, X=0.5ms and each time the clock bias becomes greater than 0.5ms, a jump of 1ms is applied.

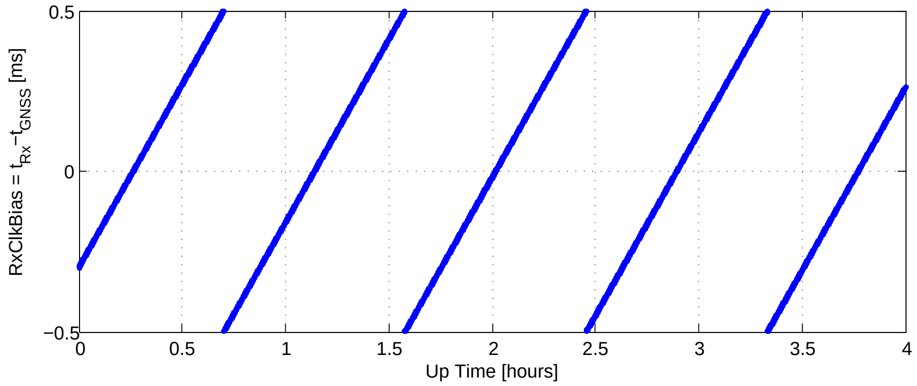


Figure 2-1: Example of the evolution of the receiver time offset with respect to the GNSS time in free-running mode.

Note that the clock bias is computed with respect to a particular GNSS time system (GPS, Galileo, BeiDou, ...) as set with the **`setTimingSystem`** command. The time offset between those systems is at the level of a few tens of nanoseconds only, and is ignored when applying the X-millisecond threshold.

When a receiver clock jump occurs, all measurements jump simultaneously. For example, a clock jump of 1ms will cause all the pseudoranges to jump by 0.001s * velocity_of_light = 299792.458m. The jump is applied on both the pseudoranges and the carrier phase measurements, and hence will not be seen on a code-minus-phase plot.

The cumulated clock jumps since the last reset of the receiver is reported in the `CumClkJumps` field of the `MeasEpoch` SBF block.

As can be seen in Figure 2-1, the initial clock bias is not necessarily zero, but it can take any value within -0.5ms and +0.5ms. This is the default configuration. You can use the second argument of the **`setClockSyncThreshold`** command to force the clock bias to be close to zero (typically <100ns) at startup. When this feature is enabled, a jump of a non-integer number of milliseconds is applied right after the first position fix, followed by a short loss of signal tracking.

2.3.2 Clock Steering

In steered mode, the receiver time is continuously steered to a GNSS time. The particular time realization (GPS, Galileo, BeiDou, ...) is generally irrelevant as the difference is very small. It can be selected with the **`setTimingSystem`** command if needed.

In the example of Figure 2-1, if the user would have enabled clock steering one hour after start up of the receiver, the clock bias would have been like in Figure 2-2 below.

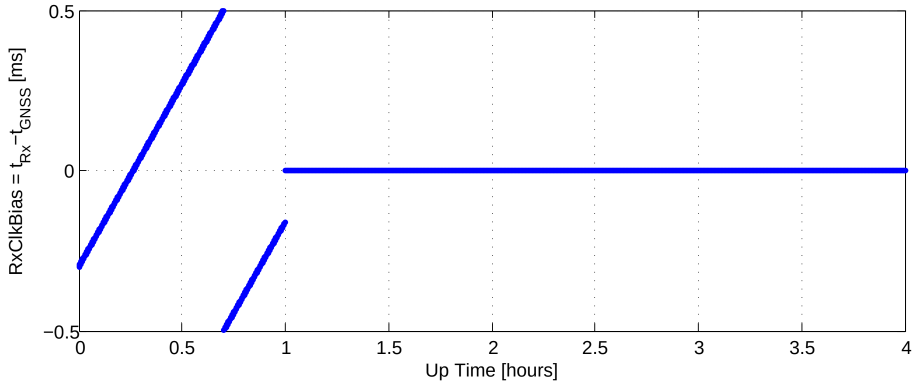


Figure 2-2: Effect of clock steering on the clock bias (clock steering enabled at an up time of 1 hour).

Clock steering accuracy is dependent on the satellite visibility, and it is recommended to only enable it under open-sky conditions.

Bit 3 of the `CommonFlags` field of the `MeasEpoch` SBF block indicates whether clock steering is active or not.

Note for the users of a GNSS constellation simulator:

When using a constellation simulator, and if the only simulated signals are the legacy L1 and L2 GPS and GLONASS signals, make sure to set the simulation time after December 31, 2019. The receiver time will be incorrect before that date.

For correct time determination, it is mandatory to reset the receiver before every (re)start of the simulation.

2.4 Computation of Position, Velocity, and Time (PVT Solution)

The receiver computes the position, velocity and time (PVT) based on the pseudoranges, the Doppler measurements and, if applicable, the differential corrections.

The availability of the PVT depends on:

- the number of available pseudoranges and Doppler measurements, equal to the number of tracked satellites, or a subset of them as specified by the **`setSatelliteUsage`** command;
- the number of valid sets of broadcast ephemerides, which are needed to compute the position, velocity, and clock bias for each tracked satellite;
- the number of valid sets of fast and long-term SBAS corrections and their age in the case of SBAS-aided positioning;
- the number of valid differential corrections and their age in the case of DGPS/RTK positioning.

A position fix requires a minimum of 4 tracked satellites with associated ephemerides. When a PVT solution is not available, PVT-related SBF blocks are still output with all the numeric fields set to Do-Not-Use values, and with the `Error` field set to indicate the source of the problem.

The accuracy of the PVT depends on:

- The signal level.
- The geometry of the satellite constellation expressed in the DOP values: these values indicate the ratio of positional errors to range errors and are computed on the basis of the error propagation theory. When the DOP is high, the accuracy of positioning will be low.
- The number of available satellites: the more satellites are available, the lower the DOP. Measurement redundancy also enables better outlier detection.
- Multipath errors on the pseudorange measurements: multipath errors can be largely attenuated by enabling the APME multipath mitigation method (see **setMultipathMitigation**) and/or using code smoothing (see **setSmoothingInterval**).
- The PVT mode as set by the **setPVTMode** command.
- The data available to compute ionospheric delays (see **setIonosphereModel**).
- The choice of the dynamics model: if the dynamics parameter set by the **setReceiverDynamics** command does not correspond to the actual dynamics of the receiver platform, the position estimation will be sub-optimal.

The a-posteriori accuracy estimate of the computed position is reported in the variance-covariance matrix, which comes in the `PosCovCartesian` and `PosCovGeodetic` SBF blocks. This accuracy estimate is based on the assumed measurement noise model and may differ from actual errors due to many external factors, most of all multipath.

2.4.1 SBAS Positioning

SBAS, which stands for ’Space Based Augmentation System’, enables differential operation over a large area with associated integrity information. System errors are computed from a dataset recorded over a continental area and disseminated via a geostationary satellite. The operation of SBAS is documented in the RTCA DO 229 standard. SBAS improves over DGPS corrections, in that it provides system corrections (ionosphere corrections and ephemeris long-term corrections) next to range corrections (the "fast corrections" in the DO 229 terminology).

The receiver provides an SBAS-aided position when it has sufficient satellites with at least fast and long-term corrections. The corrections are used as long as their applicability has not timed out. During the time-out interval the receiver applies correction degradation using the information received in message type (MT) 07 and 10.

By default, the receiver selects the SBAS satellite with the most SBAS corrections available. With the **setSBASCorrections** command, it is possible to force the receiver to use a particular SBAS satellite or a particular SBAS provider.

2.4.2 DGPS Positioning

DGPS (Differential GPS) is a pseudorange-based positioning technique where GNSS system errors are reduced by the use of range corrections. To work in DGPS rover mode, the receiver needs to receive differential corrections in the RTCM or CMR format.

Note on the RTCM v2.x corrections: the receiver takes the τ_{gd} parameter transmitted by the GPS satellites into account during the computation of the pseudorange corrections, as prescribed in v2.2 and v2.3 of the RTCM standard. The RTCM standard version 2.1 is ambiguous in this respect: it does neither prescribe nor discourage the use of τ_{gd}. The receiver can be configured in both modes using the command **`setRTCMv2Compatibility`**.

2.4.3 RTK Positioning

Real-Time Kinematic (RTK) is a carrier phase positioning method where the carrier phase ambiguities are estimated in a kinematic mode.

To work in RTK mode, the receiver requires the reception of RTK messages. Both the RTCM and the CMR message formats are supported. Multiple-base RTK is not supported: by default, the receiver selects the nearest base station if more than one base station is available.

In RTK mode, the absolute position is reported in the `PVTCartesian` or `PVTGeodetic` SBF blocks, and the baseline vector is reported in the `BaseVectorCart` and `BaseVectorGeod` SBF blocks.

2.4.3.1 Integer Ambiguities (RTK-fixed)

The key to high-accuracy carrier phase positioning is the fixing of the carrier phase integer ambiguities. Under normal circumstances the receiver will compute the integer ambiguities within several seconds and yield an RTK-fixed solution with centimeter-level accuracy. The less accurate pseudorange measurements will not be used. As long as no cycle slips or loss-of-lock events occurs, the carrier phase position is readily available.

RTK with fixed ambiguities is also commonly referred to as phase positioning using 'On-The-Fly' (OTF) ambiguity fixing. The RTK positioning engine of the receiver uses the LAMBDA method(1) developed at Delft University, department of Geodesy.

2.4.3.2 Floating Ambiguities (RTK-float)

When data availability is low (e.g. low number of satellites) or when the data are not of sufficient quality (high multipath), the receiver will not fix the carrier phase ambiguities to their integer value, but will keep them floating. At the start of the RTK-float convergence process, the position accuracy is equal to that of code-based DGPS. Over the course of several minutes the positional accuracy will converge from several decimeters to several centimeters as the floating ambiguities become more accurate.

2.4.4 Precise Point Positioning

Precise Point Positioning (PPP) provides high accuracy positioning without the need for a local base station. PPP uses precise satellite orbit and clock corrections computed by a global

---

(1) Teunissen, P.J.G., and C.C.J.M. Tiberius (1994) Integer least-squares estimation of the GPS phase ambiguities. Proceedings of International Symposium on Kinematic Systems in Geodesy, Geomatics and Navigation KIS'94, Banff, Canada, August 30-September 2, pp. 221-231.

network of reference stations and broadcast in real time by geostationary satellites in the L band.

2.4.4.1 PPP Seeding

PPP provides centimeter-level position accuracy, but suffers from a relatively long convergence time that can reach 15 to 20 minutes depending on the local multipath environment. The convergence time can be dramatically reduced by feeding the known position into the PPP engine. This process is referred to as PPP seeding, and the position fed into the PPP engine is called the PPP seed. The receiver supports two seeding modes:

Manual Seeding: in manual seeding, the user provides the accurate marker position (see section 2.5 for a definition of the marker position) to the PPP engine by using the **`exePPPSetSeedGeod`** command.

Automatic Seeding: the receiver can be configured to automatically seed the PPP engine from its current DPGS or RTK position. Automatic seeding is configured with the **`setPPPAutoSeed`** command.

2.4.4.2 PPP Datum Offset

By default, PPP positions are expressed in ITRS (which ITRF realization of ITRS depends on the PPP service provider), while RTK positions refer to the regional datum used by your RTK provider. To avoid coordinate jumps each time the PVT engine switches between RTK and PPP, and to ensure accurate seeding of the PPP engine from RTK, the regional datum must be provided with the **`setGeodeticDatum`** command.

Note that local RTK positions obtained after applying the datum transformation parameters transmitted in RTCM v3.x MT1021-1023 are never used for PPP seeding. In other words, the position reported in the `PosLocal` SBF block is not suitable for PPP seeding. See also section 2.4.6.

2.4.4.3 Tide Corrections

Since PPP is based on global satellite corrections, the PPP position would be sensitive to earth tide variations if no correction were applied. The receiver applies a tide correction based on the Sinko Earth tide model(2). All positions reported in the `PVTCartesian`, `PVTGeodetic` and `PosCart` SBF blocks are always tide-corrected.

2.4.5 Transition between PVT Modes

Whenever possible, the transitions from a more accurate PVT mode to a less accurate PVT mode are smooth. For example, when switching from RTK to DGPS mode, the position does not exhibit a sudden jump, but slowly degrades from RTK to DGPS accuracy.

(2) Sinko, J., A Compact Earth Tides Algorithm for WADGPS. Proceedings of ION GPS-95, Palm Springs, California, September 12-15, 1995, pp. 35-44.

2.4.6 Datum Transformation

By default the datum to which the coordinates refer depends on the positioning mode. For standalone, PPP and SBAS positioning for example, the coordinates refer to a global datum: WGS84 or ITRS. When using DGPS or RTK corrections from a DGPS/RTK provider, the coordinates usually refer to a regional datum (e.g. ETRS89 in Europe).

Recent realisations of WGS84 and ITRS are closely aligned and the difference can be neglected in most cases. The receiver considers them equivalent. However, regional datums may significantly differ from WGS84/ITRS, which may lead to coordinate jumps when switching between different positioning modes.

2.4.6.1 Transformation to Regional Datum

It is possible to avoid this datum shift by configuring the receiver to transform all coordinates to the regional datum used by the RTK base stations. This is done with the **`setGeodeticDatum`** command. The receiver knows the transformation parameters applicable to the most common datums (e.g. ETRS89 or NAD83), but user datums can also be defined with the **`setUserDatum`** command.

Coordinates in the `PVTCartesian` and `PVTGeodetic` SBF blocks refer to the datum selected in **`setGeodeticDatum`**. The datum can be checked by decoding the `Datum` field of these blocks.

2.4.6.2 Transformation to Local Datum

Sometimes it is needed to relate the coordinates to a local datum. Some RTK networks provide the necessary transformation and projection parameters as part of their RTCM stream, in message types 1021 to 1027.

The local geodetic coordinates (latitude, longitude and height) are reported in the `PosLocal` SBF block, and the plane grid coordinates (easting, northing, height) are reported in the `PosProjected` SBF block.

The following conditions must be met for the receiver to provide local coordinates from the information sent by the RTK network:

- the usage of RTCM v3.x MT1021-1027 must be enabled by the command **`setRTCMv3Usage`** (these messages are enabled by default).
- the complete set of datum transformation messages must have been received from the network. Plane grid coordinates are only available if the network supports one of the messages in the 1025-1027 range. Otherwise, only the local latitude, longitude and height are available.
- the position must be in the area of validity of the transformation parameters.
- to continue to get unbiased local coordinates when the positioning mode is not DGPS or RTK, the network regional datum must be set with the **`setGeodeticDatum`** command. See section 2.4.6.1.

2.5 Antenna Effects

To achieve the highest positioning precision, it is essential to take antenna effects into account.

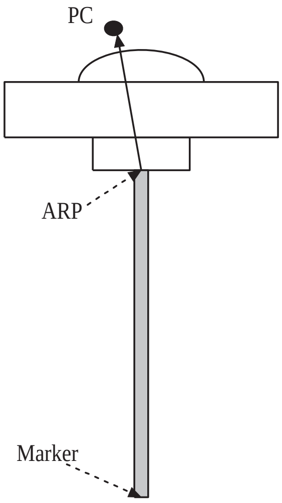


Figure 2-3: Antenna mount.

The GNSS measurements (pseudoranges and carrier phases observables) refer to a theoretical point in space called the phase center (noted PC in Figure 2-3). The position of this point is dependent on the elevation of the satellite and on the frequency band. It varies with time and it is different for the different GNSS frequency bands. The phase center variation can reach a few centimeters.

If no correction is applied, the computed position refers to an average phase center with no easy link with the antenna physical element. This average phase center fluctuates with time and cannot be used for accurate millimeter-level positioning.

For high-precision positioning, the GNSS measurements need to be corrected in such a way that they all refer to a common and stable point in space. That point is referred to as the antenna reference point (ARP). For convenience, it is usually selected at the center of the bottom surface of the antenna. PC to ARP calibration tables are available on Internet for a large number of geodetic-grade antennas. For example, the National Geodetic Survey (NGS) publishes calibration tables that can be downloaded from the following URL:
`https://www.ngs.noaa.gov/ANTCAL/`.
The antenna naming convention in such table is the one adopted by the IGS Central Bureau.

The receiver has a similar table in its non-volatile memory. This table can be upgraded following the standard upgrade procedure as described in section 1.32 (the upgrade file is named `ant_info.suf`).

2.5.1 Antenna Effects in Rover Mode

If the user specifies the type of his/her antenna using the **`setAntennaOffset`** command, the receiver compensates for the phase center variation in all rover positioning modes. If the antenna is not specified, or the antenna type is not present in the built-in antenna calibration file, the receiver cannot make the distinction between phase center and ARP, and the position accuracy is slightly degraded, especially in the height component.

The point to be positioned is the "marker" (see Figure 2-3). The offset between the ARP and the marker is a function of the antenna monumentation. It must be measured by the user and specified with the **`setAntennaOffset`** command.

The absolute position reported in the `PVTCartesian` and `PVTGeodetic` SBF blocks is always the marker position.

In DGPS or RTK modes, the receiver needs to know the type of antenna used at the base station in order to properly compensate for the phase center variation at the base. This information is typically included in the correction stream received from the base station.

The base-to-rover baseline coordinates in the `BaseVectorCart` and `BaseVectorGeod` SBF blocks is from ARP to ARP unless the receiver is not able to properly compensate for the phase center variation at base or rover. Refer to the description of the `BaseVectorCart` SBF blocks for details.

2.5.2 Antenna Effects in Base Mode

Phase center compensation always happens at the rover side. The base station sends uncompensated measurements in its differential correction messages, together with its ARP position and antenna type. The antenna type information allows the rover to apply the appropriate phase center compensation to the base measurements.

When setting up a base station, it is therefore important that the coordinates entered with the **`setStaticPosGeodetic`** or the **`setStaticPosCartesian`** commands refer to the ARP. The coordinates are encoded without change in the relevant differential correction messages. The antenna type must be provided with the **`setAntennaOffset`** command.

2.6 Galileo OSNMA

The receiver supports Galileo Open-Service Navigation Message Authentication (OSNMA), when enabled with the **`setGalOSNMAUsage`** command.

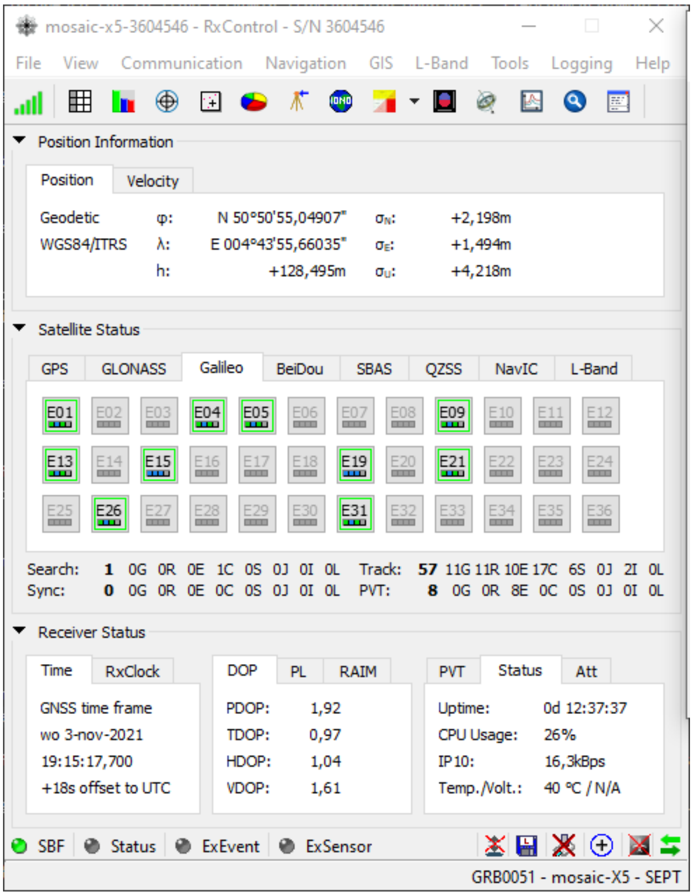
The authentication status is reported in the `GALAuthStatus` SBF block. The main steps involved in OSNMA authentication are summarized below.

1. The receiver may still need to gather initial information (e.g. public keys) from the Galileo satellites before being able to launch its authentication function. The initialization progress is reported in % in the `OSNMAStatus` field of `GALAuthStatus`. Once 100% is reached, all necessary information is available. The process of information gathering can take up to a few minutes, but is typically only needed once as the receiver stores the data in its non-volatile memory.

2. After initialization, the authentication process starts, indicated by the `OSNMAStatus` field of `GALAuthStatus` changing to *authenticating*. Roughly a minute after this transition, satellite authentication results are expected to become available. The `ActiveMask` and `AuthenticMask` fields indicate for which satellites authentication

is available and successful.

3. The authentication status gets reflected in the satellite status window of RxControl, as shown below.



Geodetic WGS84/ITRS
φ: N 50°50'55,04907" σN: +2,198m
λ: E 004°43'55,66035" σE: +1,494m
h: +128,495m σU: +4,218m

Search: 1 0G 0R 0E 1C 0S 0J 0I 0L Track: 57 11G 11R 10E 17C 6S 0J 2I 0L
Sync: 0 0G 0R 0E 0C 0S 0J 0I 0L PVT: 8 0G 0R 8E 0C 0S 0J 0I 0L

GNSS time frame
wo 3-nov-2021
19:15:17,700
+18s offset to UTC

PDOP: 1,92
TDOP: 0,97
HDOP: 1,04
VDOP: 1,61

Uptime: 0d 12:37:37
CPU Usage: 26%
IP10: 16,3kBps
Temp./Volt.: 40 °C / N/A

GRB0051 - mosaic-X5 - SEPT

Figure 2-4: Example OSNMA satellite status. The satellites marked with a green square are successfully authenticated.

2.6.1 Use of OSNMA in Simulated Scenarios

By default, OSNMA authentication is configured to work with live Galileo signals. In case the receiver is used in a simulated environment where the OSNMA public keys and Merkle Tree root key do not correspond with the live OSNMA keys, the following steps are needed to configure the OSNMA engine correctly:

1. Determine whether an NTP server (linked to the simulator) is available. If there is one, make the receiver aware of it using the **`setNTPClient`** command. If not, disable the NTP server connection using **`setNTPClient, off`**.

2. When Public Key Renewal (PKR) is not available, or to speed-up the authentication process, a user can manually introduce public keys using the **`setGalOSNMAPublicKeys`** command.

3. In case the PKR feature is needed, the appropriate Merkle Tree root key needs to be introduced using the second argument of the **`setGalOSNMAUsage`** command. Make

sure to enable the OSNMA operation at the same time (using either the *loose* or *strict* argument).

2.7 Receiver Autonomous Integrity Monitoring (RAIM)

The receiver features RAIM to ensure the integrity of the computed position solution, provided that sufficient satellites are available. The RAIM algorithm consists of three steps: detection, identification and adaptation, or shortly “D-I-A”(3):

- Detection : an overall model statistical test is performed to assess whether an integrity problem has occurred;
- Identification : statistical *w*-tests are performed on each individual measurement to assess whether it should be marked as an outlier;
- Adaptation : measurements marked as an outlier are removed from the position computation to restore the integrity of the position solution. This step is only applied if outliers have been detected in the detection step.

If the overall model statistical test fails, the RAIM module attempts to recover from the integrity failure by removing the responsible measurement(s) identified in the second step. As a consequence, the RAIM module will generally increase the continuity of integrity. The `IntegrityFlag` field of the `RAIMStatistics` SBF block reports an integrity failure if insufficient measurements remain after outlier removal (after several D-I-A steps), or if the overall model statistical test fails while no outliers can be identified. In the latter case the "sum of squared residuals too large" error is reported in the `Error` field of the PVT related SBF blocks.

The statistical tests assume an a-priori model of the measurement error probability distribution. As such, these tests can have the four classical outcomes in hypothesis testing, as shown in the table below (the letters A, B, C and D refer to the samples in Figure 2-5):

	no outlier	*outlier present*
outlier detected	False Alarm (type I error) A	Correct D
no outlier detected	Correct B	Missed Detection (type II error) C

The RAIM module makes a correct decision in two cases: an outlier present in the data is indeed detected, and no outlier is detected when none is present. However, when no outlier is present and the RAIM module declares an outlier is present, a false alarm is triggered. When an outlier remains undetected, a missed detection occurs.

---

(3) Baarda, W., A Testing Procedure For Use in Geodetic Networks, Netherlands Geodetic Commission, Publ. On Geodesy, Vol.2, no. 5, 1968

The probability computations are based on the assumption that the residuals are distributed as a Normal distribution (central if there is no outlier, and non-central if there is one), as illustrated in Figure 2-5.

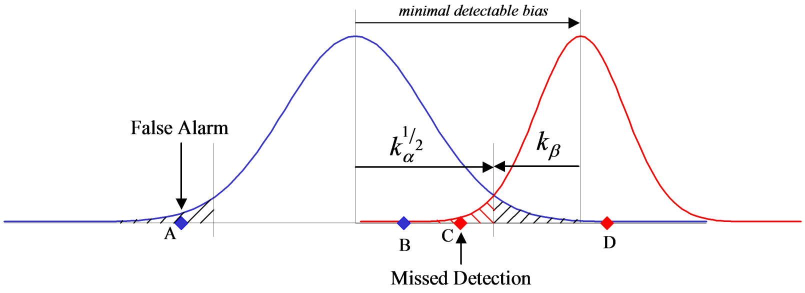


Figure 2-5: Statistical test outcomes.

Samples corresponding to the four test outcomes are represented in Figure 2-5: samples A and B are from the unbiased measurement distribution, while samples C and D are from a biased measurement distribution corresponding to an outlier. Since sample A is larger than the test threshold, it will be incorrectly flagged as an outlier (false alarm). Sample C is not detected as an outlier although it is part of the biased distribution (missed detection). The acceptable probability of false alarm and the probability of missed detection for the application must be determined and provided to the receiver. This is the purpose of the `setRAIMLevels` command.

2.7.1 Integrity Algorithm

Two kinds of statistical tests are performed: the detection step uses an *overall model* test to evaluate the integrity of the position solution as a whole, and the identification step uses the *w*-test (also known as "datasnooping") to evaluate the integrity of individual measurements. Depending on the positioning mode, the overall model test is computed for range, range-rate and/or phase measurements simultaneously, while the *w*-test is computed for each range, range rate and/or phase measurement individually. Both the overall model and the *w*-tests are of the *Generalized Likelihood Ratio Test* type.

The overall model test uses the weighted sum of the squared residuals as test statistic. This test statistic is distributed as a χ^2 distribution with r degrees of freedom, where r is the redundancy number equal to the number of satellites used in the position computation minus 4. The test reads:

$$\sigma^2 = \overline{e}^T Q_y^{-1} \overline{e} > \chi^2_\alpha(r, 0)$$

where:

- σ^2 is the overall model test statistic;
- $\overline{e}$ is the vector of residuals;
- Q_y is the variance-covariance matrix of the measurements;
- $\chi^2_\alpha(r, 0)$ is the test threshold yielding a probability α of false alarm.

The probability of false alarm of the overall model test is selectable by the user with the *ModelReliability* argument of the **`setRAIMLevels`** command.

If the overall model test statistic is lower than the test threshold, the test is passed and the integrity is guarantueed under the statistical assumptions specified by the **`setRAIMLevels`** command.

If the overall model test statistic is higher than the threshold, the test is rejected. In this case, the identification step will attempt to identify the measurement responsible for the rejection using the *w*-test discussed below. After removal of the responsible outlier(s), the overall model test statistic is recomputed to verify the integrity of the solution without the outlier present. This iterative process continues until either the overall model test along with the associated *w*-tests are accepted, or until the *w*-tests for each individual measurement are accepted with a rejected overall model test. In the latter case an integrity loss is declared; in the former case integrity is available. Note that under extreme circumstances the interactive D-I-A process can also halt due to insufficient available measurements for testing, after removal of outliers. In this case the "too many outliers" error is reported in the PVT related SBF blocks.

For the evaluation of the *w*-test statistic, the following inequality is verified:

$$-k_{\alpha}^{1/2} < w_i = \frac{e_i}{\sigma_{e_i}} < +k_{\alpha}^{1/2}$$

where:

- w_i is the *w*-test statistic for the *i*th satellite;
- e_i is the residual for the *i*th satellite;
- σ_{e_i} is the standard deviation of the residual for the *i*th satellite;
- $k_{\alpha}^{1/2}$ is the test threshold yielding a probability α of false alarm.

The probability of false alarm of the *w*-test is selectable by the user with the *Pfa* argument of the **`setRAIMLevels`** command.

The test threshold is computed by the receiver with the assumption that the *w*-test statistic is distributed as a Normal distribution. For instance, if *Pfa* is set to 10%, residuals larger than 1.64 sigma are flagged as outliers. If *Pfa* is 0.01% the threshold will be 3.89.

2.7.2 Internal and External Reliability Levels

To assess the impact of undetected measurement errors on the computed position, the minimal detectable bias (MDB) in the range domain is computed and propagated to the position domain.

The MDB describes the internal reliability of the corresponding *w*-test. It is a measure of the range error that can be detected with a given probability of missed detection. It is computed as follows for each satellite (neglecting the probability that the biased measurement falls on the left-hand side of the non-biased distribution shown in Figure 2-5):

$$MDB_i = \sigma_{y_i} \left(\frac{\lambda_0}{(1 - \frac{\sigma^2_{\widehat{y}_i}}{\sigma^2_{y_i}})} \right)^{1/2}$$

where:

- σ_{y_i} is the standard deviation of the range measurement of the *i*th satellite;
- $\sigma_{\widehat{y}_i}$ is the standard deviation of the estimator for the (measured) range of the *i*th satellite;
- λ_0 is the non-centrality parameter, which depends upon the probability of false alarm of the *w*-test and the probability of missed detection.

The user can select the probability of missed detection acceptable for his/her application with the *Pmd* argument of the **`setRAIMLevels`** command.

The external reliability is defined as the influence of a model error of size MDB on the user position. It is computed by propagating the MDB for each satellite to the position domain, taking the satellite geometry into account. The receiver computes a distinct external reliability level (XERL) for the horizontal and the vertical components (referred to as HERL and VERL respectively). These values should be compared to the alarm threshold of your specific application in order to verify if the position solution is adequate for that application.

Detailed results of the RAIM algorithm are available in the `RAIMStatistics` and the `PVTResiduals` SBF blocks and in the `GBS` NMEA message.

Chapter 3

Command Line Reference

3.1 Command Line Interface Outline

The receiver outputs a prompt when it is ready to accept a user command. The prompt is of the form:

```
CD>
```

where `CD` is the connection descriptor of the current connection, e.g. COM1 (see section1.1.5).

The prompt indicates the termination of the processing of a given command. When sending multiple commands to the receiver, it is necessary to wait for the prompt between each command.

Sometimes a connection is not configured to accept user commands, for example because it is put into differential correction input mode. A way to force a connection to accept commands is to send a succession of ten “S” characters to that connection and then to press the enter key (`SSSSSSSSSS<CR>`). See also the description of the **`setDataInOut`** command.

3.1.1 Command Types

Most commands fall into one of the following categories:

`set`-commands to change one or more configuration parameters;
`get`-commands to get the current value of one or more configuration parameters;
`exe`-commands to initiate some action;
`lst`-commands to retrieve the contents of internal files or list the commands.

Each **`set`**-command has its **`get`**-counterpart, but the opposite is not true. For instance, the **`setNMEAOutput`** command has a corresponding **`getNMEAOutput`**, but **`getReceiverCapabilities`** has no **`set`**-counterpart. Each **`exe`**-command also has its **`get`**-counterpart which can be used to retrieve the parameters of the last invocation of the command.

3.1.2 Command Line Syntax

Each ASCII command line consists of a command name optionally followed by a list of arguments and terminated by <CR>, <LF> or <CR><LF> character(s) usually corresponding to pressing the "Enter" key on the keyboard.

To minimize typing effort when sending commands by hand, the command name can be replaced by its 3-5 character mnemonic. For instance, **`grc`** can be used instead of **`getReceiverCapabilities`**.

The receiver is case insensitive when interpreting a command line.

The maximum length of any ASCII command line is 2000 characters.

For commands requiring arguments, the comma "," must be used to separate the arguments from each other and from the command's name. Any number of spaces can be inserted before and after the comma.

Each argument of a **`set`**-command corresponds to a single configuration parameter in the receiver. Usually, each of these configuration parameters can be set independently of the others, so most of the **`set`**-command's arguments are optional. Optional arguments can be omitted but if omitted arguments are followed by non-omitted ones, a corresponding number of commas must be entered. Omitted arguments always keep their current value.

3.1.3 Command Replies

The reply to ASCII commands always starts with "$R" and ends with <CR><LF> followed by the prompt corresponding to the connection descriptor you are connected to.

The following types of replies are defined for ASCII commands:

- For comment lines (user input beginning with "#") or empty commands (just pressing "Enter"), the receiver replies with the prompt.

  ```
  COM1> # This is a comment!  <CR>
  COM1>
  ```

- For invalid commands, the reply is an error message, always beginning with the keyword "$R?" followed by an error message.

- For all valid **`set`**-, **`get`**- and **`exe`**-commands, the first line of the reply is an exact copy of the command as entered by the user, preceded with "$R:". One or more additional lines are printed depending on the command. These lines report the configuration of the receiver after execution of the command.

  ```
  COM1>setNMEAOutput, stream1, com1, GGA, sec1 <CR>
  $R: setNMEAOutput, stream1, com1, GGA, sec1
    NMEAOutput, stream1, com1, GGA, sec1
  COM1>
  ```

  For commands which reset or halt the receiver (e.g. **`exeResetReceiver`**), the reply is terminated by "`STOP>`" instead of the standard prompt, to indicate that no further command can be entered.

- For all valid **`lst`**-commands, the first line of the reply is an exact copy of the command as entered by the user, preceded with "$R;". The second line is a pseudo-prompt "––––>" and the remaining of the reply is a succession of formatted blocks, each of them starting with "$–– BLOCK".

ASCII replies to **`set`**-, **`get`**- and **`exe`**-commands, including the terminating prompt, are atomic: they cannot be broken by other messages from the receivers. For the **`lst`**-commands, the replies may consist of several atomic formatted blocks which can be interleaved with other output data. If more than one formatted block is output for a lst-command, each of the intermediate blocks is terminated with a pseudo-prompt "––––>". The normal prompt will only be used to terminate the last formatted block of the reply so that one single prompt is always associated with one command.

3.1.4 Command Syntax Tables

All ASCII commands are listed in section 3.2. Each command is introduced by a compact formal description of it called a "syntax table". Syntax tables contain a complete list of arguments with their possible values and default settings when applicable.

The conventions used in syntax tables are explained below by taking a fictitious **`setCommandName`** command as example. The syntax table for that command is:

scn gcn	**setCommandName** **getCommandName**	***Cd*** *Cd*	*Distance*	*Time*	*Message (120)*	*Password (40)*	*Mode*	*PRN*
		+ Com1 + Com2 all	-20.00 … 0.00 … 20.00 m	1 … 50 sec	Unknown		on off	none + G01 ... G32 + S120 ... S138 + SBAS + GPS all

GUI: Navigation > Receiver Operation > Example

The associated **`set`**- and **`get`**-commands are always described in pairs, and the same holds for the associated **`exe`**- and **`get`**-commands. The command name and its equivalent 3-5 character mnemonic are printed in the first two columns. The list of arguments for the set- and get-commands is listed in the first and second row respectively. In our example, **`setCommandName`** can accept up to 6 arguments and **`getCommandName`** only accepts one argument. Mandatory arguments are printed in bold face. Besides the mandatory arguments, at least one of the optional arguments must be provided in the command line.

The list of possible values for each argument is printed under each of them. Default values for optional arguments are underlined.

The link printed in blue under the syntax table shows under which GUI menu the command can be found.

The fictitious command above contains all the possible argument types:

- *Cd* serves as an index for all following arguments. This can be noticed by the possibility to use this argument in the **`get`**-command. This argument is mandatory in the **`set`**-command. The accepted values are `COM1`, `COM2` and `all`, corresponding to the first or second serial ports, or to both of them respectively. The "+" sign before the first two values indicates that they can be combined to address both serial ports in the same command.

Examples: `COM1, COM1+COM2, all` (which is actually an alias for `COM1+COM2`).

- *Distance* is a number between `-20` and `20` with a default value of `0`, and up to 2 decimal digits. An error is returned if more digits are provided. The "m" indicates that the value is expressed in meters. Note that this "m" should not be typed when entering the command.

  Examples: `20, 10.3, -2.34`

- *Time* is a number between `1` and `50`, with no decimal digit (i.e. this is an integer value). This value is expressed in seconds.

  Examples: `1, 10`

- *Message* is a string with a maximum length of 120 characters. The default value of that argument is "`Unknown`". When spaces must to be used, the string has to be put between quotes and these enclosing quotes are not considered part of the string. The list of allowed characters in strings is:

  ```
  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
  !#%@()*+-./:;<=>?[\]^_`{|}~
  ```

  Example: "`Hello World!`"

- *Password* is a password argument with a maximum length of 20 characters (40/2). Password arguments are always named *Password* or *Key*. Only half of the total password length is available to the user, the other half being reserved by the system. Passwords are obfuscated by the receiver so that they cannot be read back in command replies. In addition to the characters above (see the *Message* argument), special characters are allowed in passwords using the corresponding escape sequence:
  - Type `%%DQ` to obtain `"`
  - Type `%%SQ` to obtain `'`
  - Type `%%DL` to obtain `$`
  - Type `%%AM` to obtain `&`
  - Type `%%CM` to obtain `,`

  Example: "`ab%%AM123`"

- *Mode* is a range of individual values that cannot be combined (they are not preceded by a "+" sign). Either `off` or `on` can be selected for that argument and the default value is `on`.

  Example: `on`

- *PRN* is a range of values that can be combined together with the "`+`" sign. The default value `GPS` is an alias for `G01+G02+ ... +G32`, `SBAS` is an alias for `S120+ ... +S138` and `all` an alias for `GPS+SBAS`. A "`+`" sign can be set before the argument to indicate to add the specified value(s) to the current list. If the value "`none`" is supported (which is the case in this example), a "`-`" sign can be set before the argument to remove the specified value(s) from the current list. It is possible to add or remove multiple values at once by "adding" or "subtracting" them with the "`+`" or "`-`" operator. However, "`+`" and "`-`" can never be combined in a single argument.

  Examples: `G01+G02, +G03, GPS+S120, +G04+G05, -S122-S123, -GPS`

3.2 Command Definitions

3.2.1 Receiver Administration

lai	**lstAntennaInfo**	***Antenna***								
		Overview Main [antenna name]								

Use this command with the argument *Antenna* set to `Overview` to get a list of all antenna names for which the receiver knows the phase center variation parameters (see section 2.5).

Use this command with the argument *Antenna* set to one of the antenna names returned by **`lstAntennaInfo, Overview`** to retrieve the complete phase center variation parameters for that particular antenna. Do not forget to enclose the name between double quotes if it contains whitespaces.

Using the values `Main` will return the phase center variation parameters corresponding to the main antenna type as specified in the command **`setAntennaOffset`** .

Examples

```
COM1>  lai, Overview <CR>
$R; lai, Overview
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AntennaInfo version="0.1">
  <Antenna ID="AERAT1675_29    NONE"/>
  <Antenna ID="AERAT2775_150   NONE"/>
  <Antenna ID="AERAT2775_159       "/>
  <Antenna ID="AERAT2775_159   SPKE"/>
  <Antenna ID="AERAT2775_160       "/>
  ...
  <Antenna ID="TRM_R8_GNSS         "/>
</AntennaInfo>
COM1>

COM1>  lai, "AERAT2775_159 SPKE"<CR>
$R; lai,"AERAT2775_159   SPKE"
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AntennaInfo version="0.1">
  <Antenna ID="AERAT2775_159   SPKE"/>
    <L1>
      <offset north="0.4" east="0.1" up="77.2"/>
      <phase elevation="90" value="0.0"/>
      <phase elevation="85" value="-0.2"/>
      ...
      <phase elevation=" 5" value="0.0"/>
      <phase elevation=" 0" value="0.0"/>
    </L2>
</AntennaInfo>
COM1>
```

help	**lstCommandHelp**	***Action (255)***								
		Overview								

Use this command to retrieve a short description of the ASCII command-line interface.

When invoking this command with the `Overview` argument, the receiver returns the list of all supported **`set`**-, **`get`**- and **`exe`**-commands. The **`lstCommandHelp`** command can also be called with any supported **`set`**-, **`get`**- or **`exe`**-command (the full name or the mnemonic) as argument.

The reply to this command is free-formatted and subject to change in future versions of the receiver's software. This command is designed to be used by human users. When building software applications, it is recommended to use the formal **`lstMIBDescription`**.

Examples

```
COM1>  help, Overview <CR>
$R; help, Overview
$-- BLOCK 1 / 0
MENU: communication
  GROUP: ioSelection
    sdio, setDataInOut
    gdio, getDataInOut
...
COM1>
```

```
COM1>  help, getReceiverCapabilities <CR>
$R; help, getReceiverCapabilities
... Here comes a description of getReceiverCapabilities ...
COM1>
```

```
COM1>  help, grc <CR>
$R; help, grc
... Here comes a description of getReceiverCapabilities ...
COM1>
```

lcf	lstConfigFile	*File*								
		Current Boot RxDefault User1 User2								

Use this command to list the contents of a configuration file. A configuration file contains the list of user commands needed to bring the receiver from factory default to a certain non-default configuration.

The following configuration files are available:

File	Description
`Current`	The current configuration.
`Boot`	The configuration that is loaded at boot time, after a power cycle or after a hard reset (see also the **`exeResetReceiver`** command).
`RxDefault`	The default configuration.
`User1`	A user-defined configuration.
`User2`	A user-defined configuration.

See also the related **`exeCopyConfigFile`** command to learn how to manage configuration files.

Example

```
COM1>  smp, TestMarker <CR>
$R: smp, TestMarker
  MarkerParameters, "TestMarker"
COM1>  lcf, Current <CR>
$R; lcf, Current
$-- BLOCK 1 / 1
  setMarkerParameters, "TestMarker"
COM1>
```

eccf gccf	**exeCopyConfigFile** **getCopyConfigFile**	***Source***	***Target***							
		Current Boot User1 User2 RxDefault	Current Boot User1 User2							

RxControl: File > Copy Configuration

Use this command to manage the configuration files. See the **`lstConfigFile`** command for a description of the different configuration files.

With this command, the user can copy configurations files into other configuration files. For instance, copying the `Current` file into the `Boot` file makes that the receiver will always boot in the current configuration.

Examples

To save the current configuration in the `Boot` file, use:

```
COM1> eccf, Current, Boot <CR>
$R: eccf, Current, Boot
  CopyConfigFile, Current, Boot
COM1>
```

To load the configuration stored in `User1`, use:

```
COM1> eccf, User1, Current <CR>
$R: eccf, User1, Current
  CopyConfigFile, User1, Current
COM1>
```

seth geth	**setEthernetMode** **getEthernetMode**	*Enable*								
		off on								

RxControl: Communication > Network Settings > General

Use this command to turn the Ethernet interface on or off.

Before turning Ethernet off, make sure that the receiver will still be accessible through another interface (serial, USB,...). This is especially important for remote receivers. It will not be possible to access the receiver over Ethernet after invoking the **`setEthernetMode, off`** command.

Example

```
COM1>  seth, on<CR>
$R: seth, on
  EthernetMode, on
COM1>
```

efup gfup	**exeFTPUpgrade** **getFTPUpgrade**	***Server (32)***	***Path (64)***	***Login (12)***	***Password (24)***					
				anonymous						

RxControl: File > Upgrade Receiver using FTP

Use this command to upgrade the receiver by fetching the upgrade file from an FTP server. The arguments specify the FTP server, the path to the upgrade file (.SUF format), and the login and password to use.

This procedure always resets the receiver, even if the upgrade file does not exist.

Before resetting, the receiver broadcasts a "`$TE ResetReceiver`" message to all active communication ports, to inform all users of the imminent reset.

After a reset, the user may have to adapt the communication settings of his/her terminal program as they may be reset to their default values.

Example

```
COM1> efup, myftp.com, /tst.suf, user, password<CR>
$R: efup, myftp.com, /tst.suf, user, password
  FTPUpgrade, "myftp.com", "/tst.suf", "user",
     "I301I5B8DG8E7QTT6RZT7IQ"
STOP>
$TE ResetReceiver Upgrade
STOP>
```

sgpf ggpf	**setGPIOFunctionality** **getGPIOFunctionality**	***GPPin*** *GPPin*	*Mode*	*Input*	*Output*					
		+GP1 +GP2 all	Output	none	LevelLow LevelHigh					

RxControl: Navigation > Receiver Operation > GPIO

Use these commands to define/inquire the functionality assigned to every GPIO pin.

Currently, only the output pins (`GPx`) can be controlled by this command, and the *Mode* and *Input* arguments can only take the values `Output` and `none` respectively. The argument *Output* sets the electrical level to be applied to the pin specified in *GPPin*.

In housed products, the number of GPIO pins configurable by this command is larger than the number of GPIO pins available to the user. The extra pins are used for internal purposes, and their settings should not be modified. Please refer to the Hardware Manual or the User Manual of your product to check which GPIO pins are available.

Example

To set the signal on GP2 to a logical 1, use:

```
COM1>  sgpf, GP2, Output, , LevelHigh <CR>
$R: sgpf, GP2, Output, , LevelHigh
  GPIOFunctionality, GP2, Output, none, LevelHigh
COM1>
```

lif	lstInternalFile	*File*								
		Permissions Identification Debug Error SisError DiffCorrError SetupError LBAS1Access LBAS1Subscr IPParameters RxMessages								

Use this command to retrieve the contents of one of the receiver internal files:

File	Description
`Permissions`	List of permitted options in your receiver.
`Identification`	Information about the different components being part of the receiver (e.g. serial number, firmware version, etc.).
`Debug`	Program flow information that can help support engineers to debug certain issues.
`Error`	Last internal error reports.
`SisError`	Last detected signal-in-space anomalies.
`DiffCorrError`	Last detected anomalies in the incoming differential correction streams.
`SetupError`	Last detected anomalies in the receiver setup.
`LBAS1Access`	LBAS1 (L-Band Augmentation Service 1) Access information.
`LBAS1Subscr`	LBAS1 (L-Band Augmentation Service 1) Subscription status.
`IPParameters`	Hostname, MAC and IP addresses, DNS addresses, netmask and gateway.
`RxMessages`	Event log from the receiver. This is the list of the recent event log messages. These messages are also available in the `RxMessage` SBF block.

Example

```
COM1>  lif, Permissions <CR>
$R; lif, Permissions
---->
$-- BLOCK 1 / 1
... here follows the permission file ...
COM1>
```

slm glm	**setLEDMode** **getLEDMode**	*GPLED*								
		DIFFCORLED LOGLED								

RxControl: Navigation > Receiver Operation > GPIO

Use this command to define/inquire the blinking mode of the General Purpose LED(s).

The different LED blinking modes are described in the Hardware Manual or in the User Manual of your receiver.

Example

```
COM1>  slm, DIFFCORLED <CR>
$R: slm, DIFFCORLED
  LEDMode, DIFFCORLED
COM1>
```

lmd	**lstMIBDescription**	***File (255)***								
		Overview SBFTable								

Use this command to retrieve the ASN.1-compliant syntax of the user command interface. The name of the command refers to the MIB (Management Information Base), which holds the whole receiver configuration. There is a one-to-one relationship between the formal MIB description and the ASCII command-line interface for all **`exe`**-, **`get`**- and **`set`**-commands.

When the value `Overview` is used, the general syntax of the interface is returned. With the value `SBFTable`, the receiver will output the list of supported SBF blocks and whether they can be output at a user-selectable rate or not. The **`lstMIBDescription`** command can also be called with every supported **`set`**-, **`get`**- or **`exe`**-command (the full name or the mnemonic) as argument.

No formal description of the **`lst`**-commands can be retrieved with **`lstMIBDescription`**.

Examples

```
COM1>  lmd, Overview <CR>
$R; lmd, Overview
... Here comes the generic command syntax ...
COM1>

COM1>  lmd, grc <CR>
$R; lmd, grc
... Here comes the description of getReceiverCapabilities ...
COM1>
```

grc	getReceiverCapabilities									

RxControl: Help > Receiver Interface > Permitted Capabilities

Use this command to retrieve the so-called "capabilities" of your receiver. The first returned value is the list of supported antenna(s), followed by the list of supported signals, the list of available communication ports and the list of enabled features.

The three values at the end of the reply line correspond to the default measurement interval, the default PVT interval and the default integrated INS/GNSS interval respectively. This is the interval at which the corresponding SBF blocks are output when the `OnChange` rate is selected with the **`setSBFOutput`** command. These values are expressed in milliseconds.

Each of the above-mentioned lists contain one or more of the elements in the tables below.

Antennas	Description
`Main`	The receiver’s main antenna.

Signals	Description
`GPSL1CA`	GPS L1 C/A signal.
`GPSL1PY`	GPS L1 P(Y) signal.
`GPSL2PY`	GPS L2 P(Y) signal.
`GPSL2C`	GPS L2 C signal.
`GPSL5`	GPS L5 signal.
`GPSL1C`	GPS L1C signal.
`GLOL1CA`	GLONASS L1 C/A signal.
`GLOL1P`	GLONASS L1 P signal.
`GLOL2P`	GLONASS L2 P signal.
`GLOL2CA`	GLONASS L2 C/A signal.
`GLOL3`	GLONASS L3 signal.
`GALL1BC`	Galileo L1 BC signal.
`GALE6BC`	Galileo E6 BC signal.
`GALE5a`	Galileo E5a signal.
`GALE5b`	Galileo E5b signal.
`GALE5`	Galileo E5 AltBOC signal.
`GEOL1`	SBAS L1 C/A signal.
`GEOL5`	SBAS L5 signal.
`BDSB1I`	BeiDou B1I signal.
`BDSB2I`	BeiDou B2I signal.
`BDSB3I`	BeiDou B3I signal.
`BDSB1C`	BeiDou B1C signal.
`BDSB2a`	BeiDou B2a signal.

Signals (Continued)	**Description**
`BDSB2b`	BeiDou B2b signal.
`QZSL1CA`	QZSS L1 C/A signal.
`QZSL2C`	QZSS L2 C signal.
`QZSL5`	QZSS L5 signal.
`QZSL6`	QZSS L6 signal.
`QZSL1C`	QZSS L1C signal.
`QZSL1S`	QZSS L1S signal.
`QZSL5S`	QZSS L5S signal.
`QZSL1CB`	QZSS L1 C/B signal.
`LBAND`	MSS L-Band signal.
`NAVICL5`	NavIC/IRNSS L5 signal.

ComPorts	**Description**
`COM1`	Serial port 1.
`COM2`	Serial port 2.
`COM3`	Serial port 3.
`COM4`	Serial port 4.
`USB1`	USB-device virtual serial port 1.
`USB2`	USB-device virtual serial port 2.
`IP10`	TCP/IP port 1.
`IP11`	TCP/IP port 2.
`IP12`	TCP/IP port 3.
`IP13`	TCP/IP port 4.
`IP14`	TCP/IP port 5.
`IP15`	TCP/IP port 6.
`IP16`	TCP/IP port 7.
`IP17`	TCP/IP port 8.
`NTR1`	NTRIP port 1.
`NTR2`	NTRIP port 2.
`NTR3`	NTRIP port 3.
`IPS1`	IP Server port 1.
`IPS2`	IP Server port 2.
`IPS3`	IP Server port 3.
`IPS4`	IP Server port 4.
`IPS5`	IP Server port 5.
`IPR1`	IP Receive port 1.
`IPR2`	IP Receive port 2.
`IPR3`	IP Receive port 3.

Capabilities	Description
`SBAS`	Positioning with SBAS corrections.
`DGPSRover`	Positioning with DGPS corrections.
`DGPSBase`	Generation of DGPS corrections.
`RTKRover`	Positioning with RTK corrections.
`RTKBase`	Generation of RTK corrections.
`RTCMv23`	Generation/decoding of RTCM v2.3 corrections.
`RTCMv3x`	Generation/decoding of RTCM v3.x corrections.
`CMRv20`	Generation/decoding of CMR v2.0 corrections.
`TimeSync`	Internal clock synchronisation to external PPS signal.
`xPPSOutput`	Generation of xPPS output signal.
`TimedEvent`	Accurate time mark of event signals.
`InternalLogging`	Internal logging.
`APME`	A-Posteriori Multipath Estimator.
`RAIM`	Receiver Autonomous Integrity Monitoring.
`PPPLand`	PPP through Wide Area Augmentation Service for land based use.
`LBAS1L`	L Band Augmentation data Service 1 Land only.
`MeasAv`	Measurement availability.
`IM`	Interference mitigation.
`WiFi`	WiFi.
`Bat`	Battery.

Example

```
COM1>  grc <CR>
$R: grc
  ReceiverCapabilities, Main, GPSL1CA+GEOL1, COM1+COM2+COM3+COM4+
     USB1+USB2,
  APME+SBAS, 100, 100, 100
COM1>
```

gri	**getReceiverInterface**	*Item*								
		+ RxName + SNMPLanguage + SNMPVersion all								

RxControl: Help > Receiver Interface > Interface Version

Use this command to retrieve the version of the receiver command-line interface. The reply to this command is a subset of the reply returned by the **`lstInternalFile, Identification`** command.

Example

```
COM1>  gri <CR>
$R: gri
  ReceiverInterface, RxName, AsteRx1
  ReceiverInterface, SNMPLanguage, English
  ReceiverInterface, SNMPVersion, 20060308
COM1>
```

era gra	**exeRegisteredApplications** **getRegisteredApplications**	***Cd*** *Cd*	***Application (12)***							
		+ COM1 + COM2 + COM3 + COM4 + USB1 + USB2 + IP10 ... IP17 all	Unknown							

RxControl: Communication > Registration

Use these commands to define/inquire the name of the application that is currently using a given connection descriptor (*Cd* - see 1.1.5).

Registering an application name for a connection does not affect the receiver operation, and is done on a voluntary basis. Application registration can be useful to developers of external applications when more than one application is to communicate with the receiver concurrently. Whether or not this command is used, and the way it is used is up to the developers of external applications.

Example

```
COM1>  era, com1, MyApp <CR>
$R: era, com1, MyApp
  RegisteredApplications, COM1, "MyApp"
  RegisteredApplications, COM2, "Unknown"
  RegisteredApplications, COM3, "Unknown"
  RegisteredApplications, USB1, "Unknown"
  RegisteredApplications, USB2, "Unknown"
COM1>
```

erst grst	**exeResetReceiver** **getResetReceiver**	***Level***	***EraseMemory***							
		Soft Hard Upgrade	none + Config + PVTData + SatData + BaseStations + WiFiAccessPoints + HTTPSCertificate + SISAuthData all							

RxControl: File > Reset Receiver

Use this command to reset the receiver and to erase some previously stored data. The first argument specifies which level of reset you want to execute:

Level	Description
`Soft`	This is a reset of the receiver's firmware. After a few seconds, the receiver will restart operating in the same configuration as before the command was issued, unless the "`Config`" value is specified in the second argument.
`Hard`	This is similar to a power off/on sequence. After hardware reset, the receiver will use the configuration saved in the boot configuration file.
`Upgrade`	Set the receiver into upgrade mode. After a few seconds, the receiver is ready to accept an upgrade file (SUF format) from any of its connections.

The second argument specifies which part of the non-volatile memory should be erased during the reset. The following table contains the possible values for the *EraseMemory* argument:

EraseMemory	Description
`Config`	The receiver's configuration is reset to the factory default, with the following exceptions. The IP settings set by the **setIPSettings** and **setIPPortSettings** commands keep their value. The UHF table set by the **setUHFChannelTable** command is maintained. After reset, the `Current` and `Boot` configuration files are erased (see the **exeCopyConfigFile** command), but the `User1` and `User2` configuration files are kept unchanged.
`PVTData`	The latest computed PVT data stored in non-volatile memory is erased.
`SatData`	All satellite navigation data (ephemeris, almanac, ionosphere parameters, UTC, ...) stored in non-volatile memory is erased.
`BaseStations`	All base stations stored in non-volatile memory are erased.

EraseMemory (Continued)	Description
`WiFiAccessPoints`	The list of known WiFi access points is erased (see the **`exeAddWiFiAccessPoint`** command).
`HTTPSCertificate`	Remove current HTTPS certificate. It will be replaced by a self-signed certificate at boot.
`SISAuthData`	Remove stored OSNMA data (PKR - Public Keys, floating KROOT)

Before resetting, the receiver broadcasts a "`$TE ResetReceiver`" message to all active communication ports, to inform all users of the imminent reset.

After a reset, the user may have to adapt the communication settings of his/her terminal program as they may be reset to their default values.

Example

```
COM1>  erst, soft, none <CR>
$R: erst, soft, none
  ResetReceiver, Soft, none
STOP>
$TE ResetReceiver Soft
STOP>
```

suia guia	**setUSBInternetAccess** **getUSBInternetAccess**	*Enable*								
		off on								

RxControl: Communication > Network Settings > General

Use this command to enable or disable outgoing network access over USB.

If enabled, the receiver will attempt to obtain an IP address via DHCP over the USB connection. It will then be able to access the Internet through that connection, allowing, for example, to communicate with a NTRIP server. Note that this requires that Internet sharing is enabled on the computer attached to the receiver.

The IP address assigned to the receiver can be retrieved from the **`lstInternalFile, IPParameters`** command.

See also section 1.1.3.3.

Example

```
COM1>  suia, on<CR>
$R: suia, on
  USBInternetAccess, on
COM1>
```

3.2.2 Standby and Sleep Configuration

epwm gpwm	**exePowerMode** **getPowerMode**	***Mode***								
		ScheduledSleep StandBy								

RxControl: File > Power Mode > Shut Down

Use this command to set the receiver in sleep or standby mode, in which it consumes only a fraction of its normal operational power.

When in standby mode, the receiver can be awoken by sending the appropriate signal to one of its input pins (see the receiver Hardware Manual or User Manual for details).

With the `ScheduledSleep` option, the receiver automatically sleeps and wakes up at regular intervals. This functionality is controlled by the **`setWakeUpInterval`** command.

Upon waking up, the receiver applies the configuration that is stored in the boot configuration file (see the **`lstConfigFile`** command).

Before entering standby mode, the receiver broadcasts a "`$TE PowerMode`" message to all active communication ports, to inform all users of the imminent halt.

Example

```
COM1>  epwm, Standby <CR>
$R: epwm, Standby
  PowerMode, StandBy
STOP>
$TE PowerMode Standby
STOP>
```

spth gpth	**setPowerThresholds** **getPowerThresholds**	*ExtPowerSupply*								
		0.0 . . . 30.0 V								

RxControl: File > Power Mode > Monitoring

This command sets the supply voltage level at which the receiver enters power-saving mode. It is to be used in conjunction with the **`setStandbyMonitoring`** command.

When the supply voltage (e.g. from an external battery) drops below the threshold set with the *ExtPowerSupply* argument, the receiver enters low-power sleep mode for a duration given by the **`setStandbyMonitoring`** command. At the end of this duration, the receiver wakes up and either resumes operation normally if the supply voltage is sufficient, or otherwise returns into sleep mode.

Each time the receiver wakes up, it applies the boot configuration. For the receiver to wake up in the current configuration, it has to be saved in the boot configuration with the **`exeCopyConfigFile`** command. When the condition to go to sleep is met at the time the **`setPowerThresholds`** command is entered, the user has 30 seconds to save the configuration before the receiver goes to sleep.

When the *ExtPowerSupply* argument of this command is zero, or when the *StandbyPeriod* argument of **`setStandbyMonitoring`** is zero, the voltage threshold is not monitored.

Note that only the voltage of the main PWR connector is monitored. If the receiver is powered over Ethernet (PoE), the voltage monitoring is not applicable.

Example

If you want the receiver to go to sleep for an hour each time the external power supply drops below 11.4V, use:

```
COM1>  ssm, 3600 <CR>
$R: ssm, 3600
  StandbyMonitoring, 3600
COM1>  spth, 11.4 <CR>
$R: spth, 11.4
  PowerThresholds, 11.4
COM1>  eccf, Current, Boot <CR>
$R: eccf, Current, Boot
  CopyConfigFile, Current, Boot
COM1>
```

ssm gsm	**setStandbyMonitoring** **getStandbyMonitoring**	*StandbyPeriod*								
		0 . . . 15724800 s								

RxControl: File > Power Mode > Monitoring

This command is to be used in conjunction with the **`setPowerThresholds`** command. It specifies the sleep duration when the supply voltage drops below the level set with **`setPowerThresholds`**.

When the supply voltage (e.g. from an external battery) drops below the threshold set with **`setPowerThresholds`**, the receiver enters low-power sleep mode for a duration given by the *StandbyPeriod* argument. At the end of this duration, the receiver wakes up and either resumes operation normally if the supply voltage is sufficient, or otherwise returns into sleep after about 30 seconds for a new *StandbyPeriod* duration.

This power-monitoring and -saving feature is enabled if both the *StandbyPeriod* argument of this command and the *ExtPowerSupply* argument of the **`setPowerThresholds`** command are different from zero.

Note that power monitoring can be enabled together with the scheduled sleep feature (see the **`exePowerMode`** command). In that case, the scheduled-sleep pattern will be interrupted for at least *StandbyPeriod* when the supply voltage drops under the threshold.

Example

If you want the receiver to go to sleep for an hour each time the external power supply drops below 11.4V, use:

```
COM1>  ssm, 3600 <CR>
$R: ssm, 3600
  StandbyMonitoring, 3600
COM1>  spth, 11.4 <CR>
$R: spth, 11.4
  PowerThresholds, 11.4
COM1>  eccf, Current, Boot <CR>
$R: eccf, Current, Boot
  CopyConfigFile, Current, Boot
COM1>
```

swui gwui	**setWakeUpInterval** **getWakeUpInterval**	*WakeUpTime (30)*	*AwakeDuration*	*RepetitionPeriod*						
		2000-01-01 00:00:00	0 . . . 604800 s	0 . . . 604800 s						

RxControl: File > Power Mode > Scheduling

This command can be used to set up an automatic receiver awake/sleep pattern. It is possible to order the receiver to wake up at a given time, for a certain period, and/or at regular intervals. A possible application is keeping fast time-to-first-fix even after days in sleep mode. This can be done by waking up the receiver every few hours for a few minutes, such that it can regularly refresh its ephemerides.

The *WakeUpTime* argument defines the epoch when the receiver should automatically wake up the first time. It also serves as reference epoch for the *RepetitionPeriod* argument. It refers to the GPS time scale. The format of the *WakeUpTime* argument is "YYYY-MM-DD hh:mm:ss".

The *AwakeDuration* argument defines the period for which the receiver should stay awake. If this argument is set to 0 (the default value), the receiver will remain awake indefinitely.

The *RepetitionPeriod* can be used to repeat the awake/sleep pattern at regular interval. *RepetitionPeriod* should be at least 5 seconds longer than *AwakeDuration* to allow a minimum sleep time of 5 seconds between awake periods. If *RepetitionPeriod* is set to a value smaller than *AwakeDuration*, the repetition functionality is disabled.

Be aware that the receiver must know the time to automatically go into sleep mode: if no antenna is connected to the receiver or if not enough satellites could be tracked after boot, the receiver will continue operating beyond its prescribed awake duration, and only possibly enter sleep mode at the next scheduled "go-to-sleep" epoch, if any.

To force the receiver to go into sleep mode immediately, use the command **`exePowerMode, ScheduledSleep`** instead.

If interaction with the receiver is needed during the sleep period, the user can always force the receiver to wake up by hardware means. See the receiver Hardware Manual or User Manual for details. When maintenance is done, the user should put the receiver back in sleep mode by typing **`exePowerMode, ScheduledSleep`**. This does not perturb the awake/sleep pattern: the receiver will continue to automatically wake up at the next wake-up epoch.

Each time the receiver wakes up, it applies the boot configuration. For the receiver to wake up in the current configuration, it has to be saved in the boot configuration with the **`exeCopyConfigFile`** command. In particular, when using the repetitive awake/sleep pattern, the command **`exeCopyConfigFile,current,boot`** must be entered before the first time that the receiver enters sleep mode. This is to make sure that the awake/sleep pattern configuration is not lost during the sleep periods.

Examples

If you want the receiver waking up on December 31, 2012 at 23h00 for 2 hours, use:

```
COM1>  swui, "2012-12-31 23:0:0", 7200 <CR>
$R: swui, "2012-12-31 23:0:0", 7200
  WakeUpInterval, "2012-12-31 23:00:00", 7200, 0
COM1>
```

If you want to set up an automatic wake up every day at midnight for 1 hour, use:

```
COM1>  swui, , 3600, 86400 <CR>
$R: swui, , 3600, 86400
  WakeUpInterval, "2000-01-01 00:00:00", 3600, 86400
COM1>  eccf, Current, Boot <CR>
$R: eccf, Current, Boot
  CopyConfigFile, Current, Boot
COM1>
```

3.2.3 User Management

lcu	**lstCurrentUser**									

Use this command to check which user is currently logged in on this port, if any. See also the **`login`** command.

Example

```
COM1>  lcu <CR>
$R! lstCurrentUser
  Not logged in.
COM1>  login, admin, admin <CR>
$R! LogIn
  User admin logged in.
COM1>  lcu <CR>
$R! lstCurrentUser
  Logged in as admin.
COM1>
```

sdal gdal	setDefaultAccessLevel getDefaultAccessLevel	*Web*	*FileTransfer*	*Ip*	*Com*	*Usb*				
		none Viewer User	none Viewer User	none Viewer User	none Viewer User	none Viewer User				

RxControl: File > User Management

This command defines what an anonymous user is authorized to do when connected to the receiver. An anonymous user is one who has not logged in with the **`login`** command.

The anonymous authorization level can be set independently for the different interfaces.

For all arguments except *FileTransfer*, setting the authorization level to `User` grants full control of the receiver to the anonymous user connected through the corresponding connection. The `Viewer` level allows the anonymous user to view the receiver configuration without changing it (i.e. to only issue **`get`**-commands). `none` prevents anonymous users from viewing or changing the configuration.

For the *FileTransfer* argument, `Viewer` means that the anonymous user is allowed to download log files from the receiver using FTP, SFTP or `rsync`, but not to delete them. `User` means that the anonymous user can both download and delete files, and `none` disables anonymous accesses.

To perform actions not allowed to anonymous users, you first need to authenticate yourself by entering a *UserName* and *Password* through the **`login`** command.

See also the commands **`setUserAccessLevel`** to learn how to define user accounts.

This command does not change the status of existing connections. For example, for *Com* or *Usb* connections, it will only take effect after a reset.

Example

```
COM1>  sdal, none, Viewer, Viewer, Viewer, Viewer<CR>
$R: sdal, none, Viewer, Viewer, Viewer, Viewer
  DefaultAccessLevel, none, Viewer, Viewer, Viewer, Viewer
COM1>
```

login	**LogIn**	*UserName (16)*	*Password (32)*							

Use this command to authenticate yourself. When initially connecting to the receiver, a user is considered "anonymous". The level of control granted to anonymous users is defined by the command **`setDefaultAccessLevel`**.

To perform actions not allowed to anonymous users, you need to authenticate yourself by entering a *UserName* and *Password* through the **`login`** command.

The list of user names and passwords and their respective access level can be managed with the **`setUserAccessLevel`** command. Login fails if the provided *UserName* or *Password* is not in that list.

The **`logout`** command returns to unauthenticated (anonymous) access. The **`lstCurrentUser`** command can be invoked to find out which user is logged in on the current port.

It is not necessary to log out before logging in as a different user.

Examples

To log in as user "admin" with password "admin", use

```
COM1>  login, admin, admin <CR>
$R! LogIn
  User admin logged in.
COM1>
```

Logging in with a wrong username or password gives an error:

```
COM1>  login, foo, foo <CR>
$R? LogIn: Wrong username or password!
COM1>
```

If the user does not have sufficient access right, some commands may give an error:

```
COM1>  sso, Stream1, COM1, MeasEpoch, sec1 <CR>
$R? SBFOutput: Not authorized!
COM1>
```

logout	**LogOut**									

Use this command to return to anonymous access. It is the reverse of `login`.

Example

The following sequence of commands logs in as user "admin" with password "admin", reconfigures SBF output, and logs out again:

```
COM1>  login, admin, admin <CR>
$R! LogIn
  User admin logged in.
COM1>  sso, Stream1, COM1, PVTCartesian, sec1 <CR>
$R: sso, Stream1, COM1, PVTCartesian, sec1
  SBFOutput, Stream1, COM1, PVTCartesian, sec1
COM1>  logout <CR>
$R! LogOut
  User admin logged out.
COM1>
```

sual gual	**setUserAccessLevel** **getUserAccessLevel**	***UserID*** *UserID*	*UserName (16)*	*Password (32)*	*UserLevel*	*SSHKey (232)*				
		+ User1 ... User8 all			Viewer User					

RxControl: File > User Management

Use these commands to manage the user accounts and their access rights on the receiver. Up to eight user accounts can be defined (`User1` to `User8`).

Each user is identified with a *UserName* and *Password*, and has a certain level of acces (*UserLevel*). If *UserLevel* is `User`, the user has full control of the receiver. If it is `Viewer`, the user can only issue **`get`**-commands.

The *SSHKey* argument can be used to associate an SSH public key to a user. ECDSA, Ed25519 and RSA PEM-encoded (base64) public keys conforming to RFC 4716 are supported. The number of bits in the key must be such that the corresponding base64 public key does not exceed 232 characters. RSA keys need to be at least 1024 bits long. Whenever possible, ECDSA or Ed25519 keys are recommended for enhanced security.
When an SSH key is defined with the *SSHKey* argument, a user can download log files using SFTP or `rsync` without the need for entering a password, provided the matching private key is known by the key agent running on his machine.

To delete an user account, use the empty string "" as *UserName* and *Password*.

Note that the receiver encrypts the password so that it cannot be read back with the command **`getUserAccessLevel`**.

Example

```
COM1>  sual, User3, Mildred, mypwd, Viewer, AAAAE2VjZH ...
   a9YSdPMw==<CR>
$R: sual, User3, Mildred, mypwd, Viewer, AAAAE2VjZH ...  a9YSdPMw==
  UserAccessLevel, User3, Mildred, mypwd, Viewer, AAAAE2VjZH ...
     a9YSdPMw==
COM1>
```

3.2.4 Tracking and Measurement Generation

sca gca	setChannelAllocation getChannelAllocation	***Channel*** *Channel*	*Satellite*	*Search*	*Doppler*	*Window*				
		+ Ch01 ... Ch80 all	auto G01 ... G32 F01 ... F14 E01 ... E36 S120 ... S158 C01 ... C63 J01 ... J07 I01 ... I14	auto manual	-50000 . . . 0 . . . 50000 Hz	1 . . . 16000 . . . 100000 Hz				

RxControl: Navigation > Advanced User Settings > Channel Allocation

Use these commands to define/inquire the satellite-to-channel allocation of the receiver.

The action of the **`setChannelAllocation`** command is to force the allocation of a particular satellite on the set of channels identified with the *Channel* argument, thereby overruling the automatic channel allocation performed by the receiver. It is possible to allocate the same satellite to more than one channel. If you assign a satellite to a given channel, any other channel that was automatically allocated to the same satellite will be stopped and will be reallocated.

The values `Gxx`, `Exx`, `Fxx`, `Cxx`, `Ixx`, `Jxx` and `Sxxx` for the *Satellite* argument represent GPS, Galileo, GLONASS, BeiDou, NavIC/IRNSS, QZSS and SBAS satellites respectively. For GLONASS, the frequency number (with an offset of 8) should be provided, and not the slot number (hence the "F"). Setting the *Satellite* argument to `auto` brings the channel back in auto-allocation mode.

The user can specify the Doppler window in which the receiver has to search for the satellite. This is done by setting the *Search* argument to `manual`. In that case, the *Doppler* and *Window* arguments can be provided: the receiver will search for the signal within an interval of *Window* Hz centred on *Doppler* Hz. The value to be provided in the *Doppler* argument is the expected Doppler at the GPS L1 carrier frequency (1575.42MHz). This value includes the geometric Doppler and the receiver and satellite frequency biases. Specifying a Doppler window can speed up the search process in some circumstances. A satellite already in tracking that falls outside of the prescribed window will remain in tracking.

If *Search* is set to `auto`, the receiver applies its usual search procedure, as it would do for auto-allocated satellites, and the *Doppler* and *Window* arguments are ignored.

Be aware that this command may disturb the normal operation of the receiver and is intended only for expert-level users.

Note that, when manually allocating a large number of channels to a single constellation, it is possible that some channels are left idle if the receiver does not have enough tracking hardware of the type required for the selected constellation.

Examples

```
COM1>  sca, Ch05, G01 <CR>
$R: sca, Ch05, G01
  ChannelAllocation, Ch05, G01, auto, 0, 16000
COM1>
```

```
COM1>  gca, Ch05 <CR>
$R: gca, Ch05
  ChannelAllocation, Ch05, G01, auto, 0, 16000
COM1>
```

scm gcm	**setCN0Mask** **getCN0Mask**	***Signal*** *Signal*	*Mask*								
		+ GPSL1CA + Reserved1 + Reserved2 + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL1P + GLOL2P + GLOL2CA + GLOL3 + GALL1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL6 + QZSL1C + QZSL1S + QZSL5S + QZSL1CB + NAVICL5 all	0 . . . 10 . . . 60 dB-Hz								

RxControl: Navigation > Receiver Operation > Masks

Use these commands to define/inquire the carrier-to-noise ratio mask for the generation of measurements. The receiver does not generate measurements for those signals of which the C/N_0 is under the specified mask, and does not include these signals in the PVT computation. However, it continues to track these signals and to decode and use the navigation data as long as possible, regardless of the C/N_0 mask.

The mask can be set independently for each of the signal types supported by the receiver, except for the GPS P-code, of which the mask is fixed at 1 dB-Hz (this is because of the codeless tracking scheme needed for GPS P-code).

Examples

```
COM1>  scm, GEOL1, 30 <CR>
$R: scm, GEOL1, 30
  CN0Mask, GEOL1, 30
COM1>

COM1>  gcm, GEOL1 <CR>
$R: gcm, GEOL1
  CN0Mask, GEOL1, 30
COM1>
```

smm gmm	**setMultipathMitigation** **getMultipathMitigation**	*Code*	*Carrier*							
		off on	off on							

RxControl: Navigation > Receiver Operation > Tracking and Measurements > Multipath

Use these commands to define/inquire whether multipath mitigation is enabled or not.

The arguments *Code* and *Carrier* enable or disable the A-Posteriori Multipath Estimator (APME) for the code and carrier phase measurements respectively. APME is a technique by which the receiver continuously estimates the multipath error and corrects the measurements accordingly.

This multipath estimation process slightly increases the thermal noise on the pseudoranges. However, this increase is more than compensated by the dramatic decrease of the multipath noise.

Examples

```
COM1>  smm, on, off <CR>
$R: smm, on, off
  MultipathMitigation, on, off
COM1>

COM1>  gmm <CR>
$R: gmm
  MultipathMitigation, on, off
COM1>
```

sst gst	**setSatelliteTracking** **getSatelliteTracking**	*Satellite*								
		none +G01 ... G32 +R01 ... R30 +E01 ... E36 +S120 ... S158 +C01 ... C63 +J01 ... J07 +I01 ... I14 +GPS +GLONASS +GALILEO +SBAS +BEIDOU +QZSS +NAVIC all								

RxControl: Navigation > Advanced User Settings > Tracking > Satellite Tracking

Use these commands to define/inquire which satellites are allowed to be tracked by the receiver. It is possible to enable or disable a single satellite (e.g. `G01` for GPS PRN1), or a whole constellation. `Gxx`, `Exx`, `Rxx`, `Cxx`, `Ixx`, `Jxx` and `Sxxx` refer to a GPS, Galileo, GLONASS, BeiDou, NavIC/IRNSS, QZSS or SBAS satellite respectively. GLONASS satellites must be referenced by their slot number in this command.

For a satellite to be effectively tracked by the receiver, make sure that at least one of its signals is enabled in the **`setSignalTracking`** command.

A satellite which is disabled by this command is not considered anymore in the automatic channel allocation mechanism, but it can still be forced to a given channel, and tracked, using the **`setChannelAllocation`** command.

Tracking a satellite does not automatically mean that the satellite will be included in the PVT computation. The inclusion of a satellite in the PVT computation is controlled by the **`setSatelliteUsage`** command.

Examples

To only enable the tracking of GPS satellites, use:

```
COM1> sst, GPS <CR>
$R: sst, GPS
  SatelliteTracking, G01+G02+G03+G04+G05+G06+G07+G08+G09+G10+G11
  +G12+G13+G14+G15+G16+G17+G18+G19+G20+G21+G22+G23+G24+G25+G26+G27
  +G28+G29+G30+G31+G32
COM1>
```

To add all SBAS satellites in the list of satellites to be tracked, use:

```
COM1> sst, +SBAS <CR>
$R: sst, +SBAS
  SatelliteTracking, G01+G02+G03+G04+G05+G06+G07+G08+G09+G10+G11
  +G12+G13+G14+G15+G16+G17+G18+G19+G20+G21+G22+G23+G24+G25+G26+G27
  ...
COM1>
```

To remove SBAS PRN120 from the list of allowed satellites, use:

```
COM1>  sst, -S120 <CR>
$R: sst, -S120
  SatelliteTracking, G01+G02+G03+G04+G05+G06+G07+G08+G09+G10+G11
  +G12+G13+G14+G15+G16+G17+G18+G19+G20+G21+G22+G23+G24+G25+G26+G27
  ...
COM1>
```

snt gnt	**setSignalTracking** **getSignalTracking**	*Signal*								
		+GPSL1CA +GPSL1PY +GPSL2PY +GPSL2C +GPSL5 +GPSL1C +GLOL1CA +GLOL1P +GLOL2P +GLOL2CA +GLOL3 +GALL1BC +GALE6BC +GALE5a +GALE5b +GALE5 +GEOL1 +GEOL5 +BDSB1I +BDSB2I +BDSB3I +BDSB1C +BDSB2a +BDSB2b +QZSL1CA +QZSL2C +QZSL5 +QZSL6 +QZSL1C +QZSL1S +QZSL5S +QZSL1CB +NAVICL5 +GPS +GLONASS +GALILEO +SBAS +BEIDOU +QZSS +NAVIC all								

RxControl: Navigation > Advanced User Settings > Tracking > Signal Tracking

Use these commands to define/inquire which signals are allowed to be tracked by the receiver. The signals can be addressed individually, or all signals from a constellation can be addressed at once. For example, `GALILEO` is an alias for all Galileo signals.

Note that some signals can only be enabled together with other signals:

- enabling `GPSL1PY` has no effect unless `GPSL1CA` and `GPSL2PY` are enabled as well;
- enabling `GPSL2PY` has no effect unless `GPSL1CA` is enabled as well;
- enabling `GLOL1P` has no effect unless `GLOL1CA` is enabled as well;
- enabling `GLOL2P` has no effect unless `GLOL2CA` is enabled as well;
- enabling `GLOL3` has no effect unless `GLOL1CA` is enabled as well;
- enabling `QZSL6` has no effect unless `QZSL1CA` or `QZSL1CB` is enabled as well.

Invoking this command causes all tracking loops to stop and restart.

Examples

To configure the receiver in a single-frequency L1 GPS+SBAS mode, use:

```
COM1>  snt, GPSL1CA+GEOL1 <CR>
$R: snt, GPSL1CA+GEOL1
  SignalTracking, GPSL1CA+GEOL1
COM1>

COM1>  gnt <CR>
$R: gnt
  SignalTracking, GPSL1CA+GEOL1
COM1>
```

ssi gsi	**setSmoothingInterval** **getSmoothingInterval**	***Signal*** *Signal*	*Interval*	*Alignment*								
		+GPSL1CA +GPSL2PY +GPSL2C +GPSL5 +GPSL1C +GLOL1CA +GLOL1P +GLOL2P +GLOL2CA +GLOL3 +GALL1BC +GALE6BC +GALE5a +GALE5b +GALE5 +GEOL1 +GEOL5 +BDSB1I +BDSB2I +BDSB3I +BDSB1C +BDSB2a +BDSB2b +QZSL1CA +QZSL2C +QZSL5 +QZSL6 +QZSL1C +QZSL1S +QZSL5S +QZSL1CB +NAVICL5 all	0 . . . 1000 s	0 . . . 1000 s								

RxControl: Navigation > Receiver Operation > Tracking and Measurements > Smoothing

Use these commands to define/inquire the code measurement smoothing interval.

The *Interval* argument defines the length of the smoothing filter that is used to smooth the code measurements by the carrier phase measurements. It is possible to define a different interval for each signal type. If *Interval* is set to 0, the code measurements are not smoothed. The smoothing interval can vary from 1 to 1000 seconds.

To prevent transient effect from perturbing the smoothing filter, smoothing is disabled during the first ten seconds of tracking, i.e. when the lock time is lower than 10s. Likewise, the smoothing effectively starts with a delay of 10 seconds after entering the **`setSmoothingInterval`** command.

Code smoothing allows reducing the pseudoranges noise and multipath. It has no influence on the carrier phase and Doppler measurements. The smoothing filter has an incremental effect; the noise of the filtered pseudoranges will decrease over time and reach its minimum after *Interval* seconds. For some applications, it may be necessary to wait until this transient effect is over before including the measurement in the PVT computation. This is the purpose of the *Alignment* argument. If *Alignment* is not set to 0, measurements taken during the first *Alignment*+10 seconds of tracking will be discarded. The effective amount of *Alignment* is never larger than *Interval*, even if the user sets it to a larger value.

Examples

```
COM1>  ssi, GPSL1CA, 300 <CR>
$R: ssi, GPSL1CA, 300
  SmoothingInterval, GPSL1CA, 300, 0
COM1>

COM1>  gsi, GPSL1CA <CR>
$R: gsi, GPSL1CA
  SmoothingInterval, GPSL1CA, 300, 0
COM1>
```

stlp gtlp	**setTrackingLoopParameters** **getTrackingLoopParameters**	***Signal*** *Signal*	*DLLBandwidth*	*PLLBandwidth*	*MaxTpDLL*	*MaxTpPLL*	*Adaptive*			
		+GPSL1CA +Reserved1 +Reserved2 +GPSL2C +GPSL5 +GPSL1C +GLOL1CA +GLOL1P +GLOL2P +GLOL2CA +GLOL3 +GALL1BC +GALE6BC +GALE5a +GALE5b +GALE5 +GEOL1 +GEOL5 +BDSB1I +BDSB2I +BDSB3I +BDSB1C +BDSB2a +BDSB2b +QZSL1CA +QZSL2C +QZSL5 +Reserved3 +QZSL1C +QZSL1S +QZSL5S +QZSL1CB +NAVICL5 all	0.01 . . . 0.25 . . . 5.00 Hz	1 . . . 15 . . . 100 Hz	1 . . . 100 . . . 500 ms	1 . . . 10 . . . 200 ms	off on			

RxControl: Navigation > Advanced User Settings > Tracking > Tracking Loop Parameters

Use these commands to define/inquire the tracking loop parameters for each individual signal type.

The *DLLBandwidth* and *PLLBandwidth* arguments define the single-sided DLL and PLL noise bandwidth, in Hz.

The *MaxTpDLL* argument defines the maximum DLL pre-detection time, in millisecond. The actual pre-detection time applied by the receiver (*TpDLL*) depends on the presence of a pilot component. For signals having a pilot component (e.g. GPS L2C), *TpDLL* = *MaxTpDLL*. For signals without pilot component (e.g. GPS L1CA), *TpDLL* is the largest divider of the symbol duration smaller than or equal to *MaxTpDLL*.

The *MaxTpPLL* argument defines the maximal PLL pre-detection time, in millisecond. The actual pre-detection time in the receiver (*TpPLL*) is computed in the same way as indicated for the *MaxTpDLL* argument.

Setting the *Adaptive* argument to `on` allows the receiver to dynamically change the loop parameters in order to optimize performance in specific conditions.

After entering this command, all active tracking loops stop and restart with the new settings.

This command should only be used by expert users who understand the consequences of modifying the default values. In some circumstances, changing the tracking parameters may result in the impossibility for the receiver to track a specific signal, or may significantly increase the processor load. It is recommended that the product of *TpPLL* (in milliseconds) and *PLLBandwidth* (in Hz) be kept between 100 and 200.

Note that decreasing the pre-detection times increases the load on the processor.

Example

```
COM1>  stlp, GPSL1CA, 0.20, 12, , , off <CR>
$R: stlp, GPSL1CA, 0.20, 12, , , off
  TrackingLoopParameters, GPSL1CA, 0.20, 12, 100, 10, off
COM1>
```

3.2.5 Frontend and Interference Mitigation

sam gam	**setAGCMode** **getAGCMode**	***Band*** *Band*	*Mode*	*Gain*						
		+ L1 + L2L5 + E6 all	auto frozen manual	0 . . . 35 . . . 70 dB						

RxControl: Navigation > Advanced User Settings > Frontend and Interference Mitigation > Frontend Settings

Use these commands to define/inquire the operation mode of the Automatic Gain Control (AGC) in the receiver frontend. The AGC is responsible for amplifying the input RF signal to an appropriate level.

By default (*Mode* is set to `auto`), the AGC automatically adjusts its gain in function of the input signal power. In `frozen` mode, the AGC gain is kept constant at its current value (after a ten-second stabilisation period) and does not follow any subsequent variation of the input signal power. In `manual` mode, the user can set the gain to a fixed value specified by the *Gain* argument. The *Gain* argument is ignored in `auto` and `frozen` modes.

The first argument (*Band*) specifies for which frequency band the settings apply.

Example

```
COM1>  sam, all, frozen <CR>
$R: sam, all, frozen
  AGCMode, L1, frozen, 30
COM1>
```

sbbs gbbs	**setBBSamplingMode** **getBBSamplingMode**	*Mode*								
		BeforeIM AfterIM								

RxControl: Navigation > Advanced User Settings > Frontend and Interference Mitigation > Frontend Settings

Use this command to configure the baseband samples (ADC samples) logged in the `BBSamples` SBF block.

The following sampling modes are defined:

Mode	Description
`BeforeIM`	The samples in the `BBSamples` SBF block are taken before interference mitigation (see the **`setNotchFiltering`** command). All frequency bands are sampled in turn.
`AfterIM`	The samples in the `BBSamples` SBF block are taken after interference mitigation (see the **`setNotchFiltering`** command). All frequency bands are sampled in turn.

Example

```
COM1>  sbbs, BeforeIM <CR>
$R: sbbs, BeforeIM
  BBSamplingMode, BeforeIM
COM1>
```

snf gnf	**setNotchFiltering** **getNotchFiltering**	***Notch*** *Notch*	*Mode*	*CenterFreq*	*Bandwidth*					
		+ Notch1 + Notch2 + Notch3 all	auto off manual	1100.000 . . . 2550.000 MHz	30 . . . 1600 kHz					

RxControl: Navigation > Advanced User Settings > Frontend and Interference Mitigation > Frontend Settings

Use these commands to set the position of the notch filter(s) in the receiver's frontend. Notch filters are used to cancel narrowband interferences.

The *Mode* argument is used to enable or disable the notch filter specified in the first argument. When set to `auto`, the receiver performs automatic detection of the region of the spectrum affected by interference if any. In `manual` mode, the user forces a certain region of the spectrum to be blanked by the notch filter. That region must be specified by the arguments *CenterFreq* and *Bandwidth*. *Bandwidth* is the double-sided bandwidth centered at *CenterFreq*. Specifying a region outside of a GNSS band has no effect.

In some cases, changing the operating mode of the notch filters (i.e. modifying the *Mode* argument) can cause the tracking loops to reset.

Example

```
COM1>  snf, Notch1, manual, 1227.0, 30<CR>
$R: snf, Notch1, manual, 1227.0, 30
  NotchFiltering, Notch1, manual, 1227.000, 30
COM1>
```

swbi gwbi	**setWBIMitigation** **getWBIMitigation**	*Mode*								
		off on								

RxControl: Navigation > Advanced User Settings > Frontend and Interference Mitigation > Frontend Settings

Use this command to enable or disable the mitigation of wideband interferences, including swept-frequency or pulsed interferences. When enabled (argument *Mode* set to `on`), the interference mitigation is done automatically and can be monitored with the `RFStatus` SBF block.

Invoking this command causes all tracking loops to stop and restart.

Example

```
COM1>  swbi, on<CR>
$R: swbi, on
  WBIMitigation, on
COM1>
```

3.2.6 Navigation Filter

sao	**setAntennaOffset**	***Antenna***	*DeltaE*	*DeltaN*	*DeltaU*	*Type (20)*	*SerialNr (20)*	*SetupID*		
gao	**getAntennaOffset**	*Antenna*								
		+ Main all	-1000.0000 . . . 0.0000 . . . 1000.0000 m	-1000.0000 . . . 0.0000 . . . 1000.0000 m	-1000.0000 . . . 0.0000 . . . 1000.0000 m	Unknown	Unknown	0 . . . 255		

RxControl: Navigation > Receiver Setup > Antennas

Use these commands to define/inquire the parameters that are associated with the antenna connected to your receiver.

The arguments *DeltaE*, *DeltaN* and *DeltaU* are the offsets of the antenna reference point (ARP, see section 2.5) with respect to the marker, in the East, North and Up (ENU) directions respectively, expressed in meters. All absolute positions reported by the receiver are marker positions, obtained by subtracting this offset from the ARP. The purpose is to take into account the fact that the antenna may not be located directly on the surveying point of interest.

Use the argument *Type* to specify the type of your antenna. For best positional accuracy, it is recommended to select a type from the list returned by the command **`lstAntennaInfo, Overview`**. This is the list of antennas for which the receiver can compensate for phase center variation (see section 2.5). If *Type* does not match any entry in the list returned by **`lstAntennaInfo, Overview`**, the receiver will assume that the phase center variation is zero at all elevations and frequency bands, and the position will not be as accurate. If the antenna name contains whitespaces, it has to be enclosed between double quotes. For proper name matching, it is important to keep the exact same number of whitespaces and the same case as the name returned by **`lstAntennaInfo, Overview`**.

The argument *SerialNr* is the serial number of your particular antenna. It may contain letters as well as digits (do not forget to enclose the string between double quotes if it contains whitespaces).

The argument *SetupID* is the antenna setup ID as defined in the RTCM standard. It is a parameter for use by the service provider to indicate the particular reference station-antenna combination. The number should be increased whenever a change occurs at the station that affects the antenna phase center variations. Setting *SetupID* to zero means that the values of a standard model type calibration should be used. The value entered for this argument is used to set the setup ID field in the message type 23 of RTCM2.3, and in message types 1007, 1008 and 1033 of RTCM3. It has otherwise no effect on the receiver operation.

Example

```
COM1>  sao, Main, 0.1, 0.0, 1.3, "AERAT2775_159 SPKE", 5684, 0<CR>
$R: sao, Main, 0.1, 0.0, 1.3, "AERAT2775_159   SPKE", 5684, 0
  AntennaOffset, Main, 0.1000, 0.0000, 1.3000, "AERAT2775_159
     SPKE", 5684, 0
COM1>
```

sdca gdca	**setDiffCorrMaxAge** **getDiffCorrMaxAge**	*DGPSCorr*	*RTKCorr*	*PPPCorr*	*Iono*					
		0.0 . . . 400.0 . . . 3600.0 s	0.0 . . . 20.0 . . . 3600.0 s	0.0 . . . 360.0 . . . 3600.0 s	0.0 . . . 600.0 . . . 3600.0 s					

RxControl: Navigation > Positioning Mode > PPP and Differential Corrections

Use these commands to define/inquire the maximum age acceptable for a given differential correction type. A correction is applied only if its age (aka latency) is under the timeout specified with this command and if it is also under the timeout specified with the *MaxAge* argument of the **setDiffCorrUsage** command. In other words, the command **setDiffCorrUsage** sets a global maximum timeout value, while the command **setDiffCorrMaxAge** can force shorter timeout values for certain correction types.

The argument *DGPSCorr* defines the timeout of the range corrections when the PVT is computed in DGPS mode.

The argument *RTKCorr* defines the timeout of the base station code and carrier phase measurements when the PVT is computed in RTK mode.

The argument *PPPCorr* defines the timeout of the wide-area satellite clock and orbit corrections used in PPP mode (only applicable if your receiver supports PPP positioning mode).

The argument *Iono* defines the timeout of the ionospheric corrections (such as transmitted in RTCM2.x MT15) used in DGPS PVT mode.

If the timeout is set to 0, the receiver will never apply the corresponding correction.

Note that this command does not apply to the corrections transmitted by SBAS satellites. For these corrections, the receiver always applies the timeout values prescribed in the DO229 standard.

Example

```
COM1>  sdca, 10 <CR>
$R: sdca, 10
  DiffCorrMaxAge, 10.0, 20.0, 300.0, 300.0
COM1>
```

sdcu gdcu	**setDiffCorrUsage** **getDiffCorrUsage**	*Mode*	*MaxAge*	*BaseSelection*	*BaseID*					
		LowLatency	0.1 . . . 3600.0 s	auto manual	0 . . . 4095					

RxControl: Navigation > Positioning Mode > PPP and Differential Corrections

Use these commands to define/inquire the usage of incoming differential corrections in DGPS or RTK rover mode.

The *Mode* argument defines the type of differential solution that will be computed by the receiver. If `LowLatency` is selected, the PVT is computed at the moment local measurements of the receiver are available and the most recently received differential corrections are extrapolated to the current time.

The *MaxAge* argument defines the maximum age of the differential corrections to be considered valid. *MaxAge* applies to all types of corrections (DGPS, RTK, satellite orbit, etc), except for those received from a SBAS satellite. See also the command **`setDiffCorrMaxAge`** to set different maximum ages for different correction types.

The *BaseSelection* argument defines how the receiver should select the base station(s) to be used. If `auto` is selected and the receiver is in DGPS-rover mode, it will use all available base stations. If `auto` is selected and the receiver is in RTK-rover mode, it will automatically select the nearest base station. If `manual` is selected, the receiver will only use the corrections from the base station defined by the *BaseID* argument (in both DGPS and RTK modes).

Example

```
COM1>  sdcu, LowLatency, 5.0, manual, 1011<CR>
$R: sdcu, LowLatency, 5.0, manual, 1011
  DiffCorrUsage, LowLatency, 5.0, manual, 1011
COM1>
```

sem gem	**setElevationMask** **getElevationMask**	***Engine*** *Engine*	*Mask*							
		+ Tracking + PVT all	-90 . . . 0 . . . 90 deg							

RxControl: Navigation > Receiver Operation > Masks

Use these commands to set or get the elevation mask in degrees. There are two masks defined: a tracking mask and a PVT mask.

Satellites under the tracking elevation mask are not tracked, and therefore there is no measurement, nor navigation data available from them. The tracking elevation mask does not apply to SBAS satellites: SBAS satellites are generally used to supply corrections and it is undesirable to make the availability of SBAS corrections dependent on the satellite elevation. The tracking elevation mask does not apply to satellites that are manually assigned with the **`setChannelAllocation`** command.

Satellite under the PVT mask are not included in the PVT solution, though they still provide measurements and their navigation data is still decoded and used. The PVT elevation mask do apply to the SBAS satellites: the ranges to SBAS satellites under the elevation mask are not used in the PVT, but the SBAS corrections are still decoded and potentially used in the PVT.

Although possible, it does not make sense to select a higher elevation mask for the tracking than for the PVT, as, obviously, a satellite which is not tracked cannot be included in the PVT.

The mask can be negative to allow the receiver to track satellites below the horizon. This can happen in case the receiver is located at high altitudes or if the signal is refracted through the atmosphere.

Example

```
COM1>  sem, PVT, 15<CR>
$R: sem, PVT, 15
  ElevationMask, PVT, 15
COM1>
```

sgu ggu	**setGeoidUndulation** **getGeoidUndulation**	*Mode*	*Undulation*							
		auto manual	-250.000 . . . 0.000 . . . 250.000 m							

RxControl: Navigation > Receiver Operation > Position > Earth Models

Use these commands to define/inquire the geoid undulation at the receiver position. The geoid undulation specifies the local difference between the geoid and the WGS84 ellipsoid.

If *Mode* is set to `auto`, the receiver computes the geoid undulation with respect to the WGS84 ellipsoid using the model defined in 'Technical Characteristics of the NAVSTAR GPS, NATO, June 1991', regardless of the datum specified with the **`setGeodeticDatum`** command. In `auto` mode, the *Undulation* argument is ignored.

The geoid undulation is included in the `PVTCartesian` and the `PVTGeodetic` SBF blocks and in the NMEA position messages.

Examples

```
COM1>  sgu, manual, 25.3 <CR>
$R: sgu, manual, 25.3
  GeoidUndulation, manual, 25.3
COM1>
```

```
COM1>  ggu <CR>
$R: ggu
  GeoidUndulation, manual, 25.3
COM1>
```

shm ghm	**setHealthMask** **getHealthMask**	***Engine*** *Engine*	*Mask*							
		+ Tracking + PVT all	off on							

RxControl: Navigation > Receiver Operation > Masks

Use these commands to define/inquire whether measurements (pseudoranges, carrier phases...) should be produced for unhealthy satellite signals (i.e. signals for which the unhealthy flag is set in the satellite navigation message), and whether these measurements should be included in the PVT solution.

If *Mask* is `on` for the `Tracking` engine, GNSS measurements are only generated for healthy signals or when the health is unknown. Signals flagged unhealthy remain internally tracked and their navigation data is still decoded and processed, but the corresponding measurements are discarded.

If *Mask* is `on` for the `PVT` engine, measurements from unhealthy signals or from signals of which the health is unknown are not included in the PVT. Setting this mask to `off` must be done with caution: including a non-healthy signal in the PVT computation may lead to unpredictable behaviour of the receiver.

Examples

To track unhealthy satellites/signals, use:

```
COM1>  shm, Tracking, off <CR>
$R: shm, Tracking, off
  HealthMask, Tracking, off
COM1>

COM1>  ghm <CR>
$R: ghm
  HealthMask, Tracking, off
  HealthMask, PVT, on
COM1>
```

sim gim	**setIonosphereModel** **getIonosphereModel**	*Model*								
		auto off Klobuchar SBAS MultiFreq KlobucharBeiDou								

RxControl: Navigation > Receiver Operation > Position > Atmosphere

Use these commands to define/inquire the type of model used to correct ionospheric errors in the PVT computation. The following models are available:

Model	Description
`auto`	With this selection, the receiver will, based on the available information, automatically select the best model on a satellite to satellite basis.
`off`	The receiver will not correct measurements for the ionospheric delay. This may be desirable if the receiver is connected to a GNSS signal simulator.
`Klobuchar`	This model uses the parameters as transmitted by the GPS satellites to compute the ionospheric delays.
`SBAS`	This model complies with the DO229 standard: it uses the near real-time ionospheric delays transmitted by the SBAS satellites in MT18 and MT26. If no such message has been received, the `Klobuchar` model is selected automatically.
`MultiFreq`	This model uses a combination of measurements on different carriers to accurately estimate ionospheric delays. It requires the availability of at least dual-frequency measurements.
`KlobucharBeiDou`	This model uses the parameters as transmitted by the BeiDou satellites to compute the ionospheric delays.

Unless the model is set to `auto`, the receiver uses the same model for all satellites, e.g. if the `Klobuchar` model is requested, the Klobuchar parameters transmitted by GPS satellites are used for all tracked satellites, regardless of their constellation.

If not enough data is available to apply the prescribed model to a given satellite (for instance if only single-frequency measurements are available and the model is set to `MultiFreq`), the satellite in question will be discarded from the PVT. Under most circumstances, it is recommended to leave the model to `auto`.

Examples

To disable the compensation for ionospheric delays, use:

```
COM1>  sim, off <CR>
$R: sim, off
  IonosphereModel, off
COM1>
```

```
COM1>  gim <CR>
$R: gim
  IonosphereModel, off
COM1>
```

scls gcls	**setL6CLASSource** **getL6CLASSource**	*Satellite*	*Message*							
		auto none J01 ... J07	L6D L6E							

RxControl: Navigation > Positioning Mode > QZSS CLAS Configuration

This command sets the QZSS satellite from which the L6 signal will be tracked and CLAS corrections will be decoded. If the *Satellite* argument is set to `auto`, the receiver automatically selects the optimal QZSS satellite.

QZSS satellites transmit two message streams on the L6 carrier (L6D and L6E). The second argument selects the message stream that the receiver needs to decode.

Example

```
COM1>  scls, J03, L6D<CR>
$R: scls, J03, L6D
  L6CLASSource, J03, L6D
COM1>
```

smv gmv	**setMagneticVariance** **getMagneticVariance**	*Mode*	*Variation*							
		auto manual	-180.0 . . . 0.0 . . . 180.0 deg							

RxControl: Navigation > Receiver Operation > Position > Earth Models

Use these commands to define the magnetic variation (a.k.a. magnetic declination) at the current position. The magnetic variation specifies the local offset of the direction to the magnetic north with respect to the geographic north. The variation is positive when the magnetic north is east of the geographic north.

By default (the argument *Mode* is set to `auto`), the receiver automatically computes the variation according to the 12th generation of the International Geomagnetic Reference Field (IGRF) model, using the IGRF2015 coefficients corrected for the secular variation.

Note that the magnetic variation is used solely in the generation of NMEA messages.

Examples

```
COM1>  smv, manual, 1.1 <CR>
$R: smv, manual, 1.1
  MagneticVariance, manual, 1.1
COM1>
```

```
COM1>  gmv <CR>
$R: gmv
  MagneticVariance, manual, 1.1
COM1>
```

snrc gnrc	**setNetworkRTKConfig** **getNetworkRTKConfig**	*NetworkType*								
		auto VRS								

RxControl: Navigation > Positioning Mode > PPP and Differential Corrections

Use these commands to define/inquire the type of the RTK network providing the differential corrections.

In most cases, it is recommended to leave the *Type* argument to `auto` to let the receiver autodetect the network type. For some types of VRS networks (especially for those having long baselines between the base stations), optimal performance is obtained by forcing the type to `VRS`.

Example

```
COM1>  snrc, VRS <CR>
$R: snrc, VRS
  NetworkRTKConfig, VRS
COM1>
```

spas gpas	**setPPPAutoSeed** **getPPPAutoSeed**	*Mode*								
		none + DGPS + RTKFixed all								

RxControl: Navigation > Positioning Mode > PPP and Differential Corrections

Use this command to specify which position mode is allowed to be used as a seed for the PPP engine.

If both `RTKFixed` and `DGPS` modes are enabled, the receiver gives priority to `RTKFixed` seeding over `DGPS` seeding.

In any case, a manual seed entered with the command **`exePPPSetSeedGeod`** overrules any automatic seeding.

Before enabling seeding from DGNSS or RTK, make sure that the DGNSS/RTK datum is specified with the **`setGeodeticDatum`** command.

Example

```
COM1>  spas, RTKFixed <CR>
$R: spas, RTKFixed
  PPPAutoSeed, RTKFixed
COM1>
```

epss gpss	**exePPPSetSeedGeod** **getPPPSetSeedGeod**	***Latitude***	***Longitude***	***Altitude***	***Datum***					
		-90.000000000 . . . 0.000000000 . . . 90.000000000 deg	-180.000000000 . . . 0.000000000 . . . 180.000000000 deg	-1000.0000 . . . 0.0000 . . . 30000.0000 m	WGS84 ETRS89 NAD83 NAD83_PA NAD83_MA GDA94 GDA2020 User1 User2 Other					

RxControl: Navigation > Receiver Initialization > PPP Set Seed

Use this command to seed the PPP engine with the specified position. The geodetic coordinates in the *Latitude*, *Longitude* and *Altitude* arguments are that of the marker, and not of the ARP. The argument *Longitude* is positive to the east of Greenwich.

The datum to which the coordinates refer must be specified with the *Datum* argument:

Datum	Description
`WGS84`	WGS84 or ITRFxx (the receiver does not make a distinction between them)
`ETRS89`	European ETRS89 (ETRF2000 realization)
`NAD83`	NAD83(2011), North American Datum (2011)
`NAD83_PA`	NAD83(PA11), North American Datum, Pacific plate (2011)
`NAD83_MA`	NAD83(MA11), North American Datum, Marianas plate (2011)
`GDA94`	GDA94(2010), Geocentric Datum of Australia (2010)
`GDA2020`	GDA2020, Geocentric Datum of Australia 2020
`User1`	First user-defined datum. The corresponding transformation parameters must be specified by the **`setUserDatum`** and **`setUserDatumVel`** commands, while the corresponding ellipsoid must be defined by the **`setUserEllipsoid`** command.
`User2`	Second user-defined datum
`Other`	Datum not in the list or unknown

The receiver applies the necessary transformations to ITRF (the datum used by the PPP engine) automatically. If the *Datum* argument is set to `Other`, no datum transformation is applied.

It is the user's responsibility to ensure that the specified marker position is centimeter-accurate and that the datum is properly specified. Inaccurate seeding will result in a long convergence time, and/or in an incorrect estimate of the variance/covariance matrix of the PPP solution.

At the moment when the command is entered, the receiver must be static. To prevent gross errors, the command is ignored when the seed significantly differs from the current position computed by the receiver, or when the current velocity is not zero.

In the event that the command is issued when the receiver is already in PPP mode, the PPP filter is reset and re-seeded.

Example

```
COM1>  epss, 4.5, 3.568, 0.1, WGS84<CR>
$R: epss, 4.5, 3.568, 0.1, WGS84
  PPPSetSeedGeod, 4.500000000, 3.568000000, 0.1000, WGS84
COM1>
```

spm gpm	**setPVTMode** **getPVTMode**	*Mode*	*RoverMode*	*RefPos*						
		Static Rover	+ StandAlone + SBAS + DGPS + RTKFloat + RTKFixed + PPP + RTK all	auto Geodetic1 Cartesian1						

RxControl: Navigation > Positioning Mode > PVT Mode

Use these commands to define/inquire the main PVT mode of the receiver. The argument *Mode* specifies the general positioning mode. If `Rover` is selected, the receiver assumes that it is moving and it computes the best PVT allowed by the *RoverMode* argument. If `Static` is selected, the receiver assumes that it is static and it reports a constant position. The static position can be specified by the user or computed by the receiver, according to the settings of the *RefPos* argument.

The argument *RoverMode* specifies the allowed PVT modes when the receiver is operating in `Rover` mode. Different modes can combined with the "+" operator. Refer to section 2.4 for a description of the PVT modes. The value `RTK` is an alias for `RTKFloat+RTKFixed`. When more than one mode is enabled in *RoverMode*, the receiver automatically selects the mode that provides the most accurate solution with the available data.

The *RefPos* argument defines the reference position of the antenna ARP. This is the position that is encoded in the RINEX header (after application of the marker-ARP offset specified with the **`setAntennaOffset`** command) and in the relevant RTCM and CMR differential correction messages.

If *RefPos* is set to `Geodetici` or `Cartesiani`, the fixed ARP coordinates must be provided by the user with the **`setStaticPosCartesian`** or the **`setStaticPosGeodetic`** commands.

If *RefPos* is set to `auto`, the reference ARP position is computed by the receiver. In rover mode, the reference position is not fixed. This is the setting that must be used on moving platforms.

Examples

```
COM1>  spm, Rover, StandAlone+RTK <CR>
$R: spm, Rover, StandAlone+RTK
  PVTMode, Rover, StandAlone+RTK, auto
COM1>
```

To set up a fixed base station at a known location, use the following:

```
COM1>  sspg, Geodetic1, 50.5209, 4.4245, 113.3 <CR>
$R: sspg, Geodetic1, 50.5209, 4.4245, 113.3
  StaticPosGeodetic, Geodetic1, 50.52090000, 4.42450000, 113.3000
COM1>  spm, Static, , Geodetic1 <CR>
$R: spm, Static, , Geodetic1
  PVTMode, Static, StandAlone+RTK, Geodetic1
COM1>
```

srl grl	**setRAIMLevels** **getRAIMLevels**	*Mode*	*Pfa*	*Pmd*	*Reliability*					
		off on	-12 . . . -4 . . . -1	-12 . . . -4 . . . -1	-12 . . . -3 . . . -1					

RxControl: Navigation > Receiver Operation > Position > Integrity

Use these commands to define/inquire the parameters of the Receiver Autonomous Integrity Monitoring (RAIM) algorithm in rover PVT mode.

The *Mode* argument acts as an on/off switch: it determines whether RAIM is active or not.

The *Pfa* argument sets the probability of false alarm of the w-test used in the "identification" step of the RAIM algorithm. Increasing this parameter increases the integrity but may reduce the availability of the positional solution.

The *Pmd* argument sets the probability of missed detection, which the receiver uses to compute the Minimal Detectable Bias and hence the XERL values.

The *Reliability* argument sets the probability of false alarm of the Overall Model test used in the "detection" step of the RAIM algorithm.

The value to be provided in the *Pfa*, *Pmd* and *Reliability* arguments are the base-10 logarithms of the desired probabilities. For instance, if you want a probability of false alarm of 1e-6, you have to set the *Pfa* argument to -6.

Note that this command has no effect when the receiver is configured in static PVT mode with the **`setPVTMode, Static`** command. In static PVT mode, the RAIM algorithm uses fixed parameters that cannot be changed by the user.

Examples

To configure the receiver outlier detection with a probability of 1e-4 (0.01%) that a false alarm will be raised (type I error), a probability of 1e-4 (0.01%) that an outlier will be missed (type II error) and an Overall Model probability of false alarm of 1e-6 (0.0001%), use:

```
COM1>  srl, on, -4, -4, -6 <CR>
$R: srl, on, -4, -4, -6
  RAIMLevels, on, -4, -4, -6
COM1>
```

To disable the outlier detection, use:

```
COM1>  srl, off <CR>
$R: srl, off
  RAIMLevels, off, -4, -4, -6
COM1>
```

srd grd	**setReceiverDynamics** **getReceiverDynamics**	*Level*	*Motion*							
		Max High Moderate Low	Static Quasistatic Pedestrian Automotive EarthquakeMon Unlimited							

RxControl: Navigation > *Receiver Operation* > *Position* > *Motion*

Use these commands to set the type of dynamics the GNSS antenna is subjected to.

The *Level* argument sets the balance between noise and dynamics in the GNSS measurements and the PVT solution. If rapid displacements (such as those caused by shocks, drops, and oscillations) need to be detected, the *Level* argument can be set to `High`. In that case, high-frequency motion becomes visible at the expense of an increase in the noise. Conversely, if noise reduction is important, the *Level* argument can be set to `Low`. It is generally recommended to keep the default value (`Moderate`), where the receiver will sense the dynamics and adapt itself accordingly. The `Max` level is a special setting, for which the navigation filter is disabled and the receiver computes epoch-by-epoch independent PVT solutions. Note that the `Max` level can lead to a longer latency of the PVT output.

The *Motion* argument defines the general characteristics of the receiver motion, such as the expected speed, rotation, and vibration level. This can help the receiver to optimize certain parameters for your application.

For example, when selecting the `Pedestrian` motion, signal reacquisition will be faster after a long outage, as the position is known not to have changed much.

Motion	Description
`Static`	Fixed base and reference stations.
`Quasistatic`	Low speed, limited area motion typical of surveying applications.
`Pedestrian`	Low speed (<7m/s) motion. E.g. pedestrians, low-speed land vehicles, ...
`Automotive`	Medium speed (<50m/s) motion. E.g. passenger cars, rail vehicles. This setting is generally adequate for most applications.
`EarthquakeMon`	Specific setting for earthquake monitoring, where the position is mostly static, but can suddenly change during an earthquake.
`Unlimited`	Unconstrained motion.

Example

```
COM1>  srd, High, Automotive<CR>
$R: srd, High, Automotive
  ReceiverDynamics, High, Automotive
COM1>
```

ernf grnf	**exeResetNavFilter** **getResetNavFilter**	***Level***								
		+ PVT + AmbRTK all								

RxControl: Navigation > Receiver Initialization > Reset Navigation Filter

Use this command to reset the different navigation filters in the receiver. The user can reset each navigation filter independently or together with the value `all`.

The following values for *Level* are defined:

Level	**Description**
`PVT`	Reset the whole PVT filter such that all previous positioning information is discarded, including the RTK ambiguities and the INS/GNSS integration filter when applicable.
`AmbRTK`	Only reset the ambiguities used in RTK positioning to float status.

Example

```
COM1>  ernf, PVT <CR>
$R: ernf, PVT
  ResetNavFilter, PVT
COM1>
```

ssu gsu	**setSatelliteUsage** **getSatelliteUsage**	*Satellite*									
		none +G01 ... G32 +R01 ... R24 +R25 +R26 +R27 +R28 +R29 +R30 +E01 ... E36 +S120 ... S158 +C01 ... C63 +J01 ... J07 +GPS +GLONASS +GALILEO +SBAS +BEIDOU +QZSS all									

RxControl: Navigation > Advanced User Settings > PVT > Satellite Usage

Use these commands to define/inquire which satellites are allowed to be included in the PVT computation. It is possible to enable or disable a single satellite (e.g. `G01` for GPS PRN1), or a whole constellation. `Gxx`, `Exx`, `Rxx`, `Cxx` and `Sxxx` refer to a GPS, Galileo, GLONASS, BeiDou and SBAS satellite respectively. GLONASS satellites must be referenced by their slot number in this command.

Examples

To only use GPS measurements in the PVT computation, use:

```
COM1>  ssu, GPS <CR>
$R: ssu, GPS
  SatelliteUsage, G01+G02+G03+G04+G05+G06+G07+G08+G09+G10+G11
  +G12+G13+G14+G15+G16+G17+G18+G19+G20+G21+G22+G23+G24+G25+G26
  +G27+G28+G29+G30+G31+G32
COM1>
```

To add the usage of SBAS measurements in the PVT, use:

```
COM1>  ssu, +SBAS <CR>
$R: ssu, +SBAS
  SatelliteUsage, G01+G02+G03+G04+G05+G06+G07+G08+G09+G10+G11
  +G12+G13+G14+G15+G16+G17+G18+G19+G20+G21+G22+G23+G24+G25+G26
  ...
COM1>
```

To remove the measurement of one satellite from the PVT, use:

```
COM1>  ssu, -S120 <CR>
$R: ssu, -S120
  SatelliteUsage, G01+G02+G03+G04+G05+G06+G07+G08+G09+G10+G11
  +G12+G13+G14+G15+G16+G17+G18+G19+G20+G21+G22+G23+G24+G25+G26
  ...
CM1>
```

ssbc gsbc	**setSBASCorrections** **getSBASCorrections**	*Satellite*	*SISMode*	*NavMode*	*DO229Version*					
		auto EGNOS WAAS MSAS GAGAN SDCM S120 ... S158	Test Operational	EnRoute PrecApp MixedSystems	auto DO229C					

RxControl: Navigation > Positioning Mode > SBAS Corrections

Use these commands to define/inquire the details on the usage of SBAS data in the PVT computation. This command does not define whether SBAS corrections are to be used or not in the PVT (this is done by the **`setPVTMode`** command), but it specifies how these corrections should be used.

The *Satellite* argument defines the provider of SBAS corrections, being either an individual satellite or a satellite system. If `EGNOS`, `WAAS`, `MSAS`, `GAGAN` or `SDCM` is selected, the receiver restricts the automatic selection of a satellite to those that are part of the EGNOS, WAAS, MSAS, GAGAN or SDCM system. When `auto` is selected for the *Satellite* argument, the receiver will automatically select a satellite on the basis of the location of the receiver and on the availability of SBAS corrections.

The *SISMode* argument defines the interpretation of a "Do Not Use for Safety Applications" message. It depends on the SBAS service selection with the **`setSBASService`** command.

For the DO229 service: When set to `Operational`, the receiver will discard all SBAS corrections received from a satellite upon reception of a MT00 from that satellite. Note that MT02 content encoded in a MT00 message will be interpreted by the receiver as a MT02 message: only MT00 with all '0' symbols will be interpreted as a true "Do Not Use for Safety Applications". When the argument *SISMode* is set to `Test`, the receiver will ignore the reception of a "Do Not Use for Safety Applications" message. This provides the possibility to use a signal from a SBAS system in test mode.

For the DFMC service: When set to `Operational`, the receiver will discard all SBAS corrections received from a satellite upon reception of a MT00 from that satellite. When the argument *SISMode* is set to `Test`, the receiver will ignore the reception of a "Do Not Use for Safety Applications" message. This provides the possibility to use a signal from a SBAS system in test mode. In this case the MT00 message content will be interpreted by the receiver as integrity data (MT34, MT35 or MT36).

The SBAS standards, which have their origin in aviation, make a distinction between two positioning applications: en-route and precision approach. The choice between both applications influences the length of the interval during which the SBAS corrections are valid. During a precision approach the validity of the data is much shorter. The receiver can operate in both modes, which is controlled by the *NavMode* argument.

In `EnRoute` or in `PrecApp` mode, the receiver only uses the satellite systems for which SBAS corrections are available. For best positioning accuracy, it is typically preferable to include all satellites in the position computation, even if they are not corrected by SBAS. This is achieved by the `MixedSystems` mode.

The *DO229Version* argument can be used to specify which version of the DO 229 standard to conform to.

Example

To force the receiver to use corrections from PRN 122 and ignore message MT00:

```
COM1>  ssbc, S122, Test <CR>
$R: ssbc, S122, Test
  SBASCorrections, S122, Test, MixedSystems, auto
COM1>
```

sssc gssc	**setSBASService** **getSBASService**	***Service***								
		DO229 DFMC								

RxControl: Navigation > Positioning Mode > SBAS Corrections

Use this command to define/inquire the SBAS service to be used in the PVT computation.

If `DO229` is selected, the SBAS corrections provided on the L1 signal are used (only for GPS satellites). If `DFMC` is selected, the Dual-Frequency Multi-Constellation (DFMC) corrections provided on L5 are used.

Example

```
COM1>  sssc, DO229<CR>
$R: sssc, DO229
  SBASService, DO229
COM1>
```

snu gnu	**setSignalUsage** **getSignalUsage**	*PVT*	*NavData*								
		+GPSL1CA	+GPSL1CA								
		+GPSL1PY	+GPSL1PY								
		+GPSL2PY	+GPSL2PY								
		+GPSL2C	+GPSL2C								
		+GPSL5	+GPSL5								
		+GPSL1C	+GPSL1C								
		+GLOL1CA	+GLOL1CA								
		+GLOL1P	+GLOL1P								
		+GLOL2P	+GLOL2P								
		+GLOL2CA	+GLOL2CA								
		+GLOL3	+GLOL3								
		+GALL1BC	+GALL1BC								
		+GALE6BC	+GALE6BC								
		+GALE5a	+GALE5a								
		+GALE5b	+GALE5b								
		+GALE5	+GALE5								
		+GEOL1	+GEOL1								
		+GEOL5	+GEOL5								
		+BDSB1I	+BDSB1I								
		+BDSB2I	+BDSB2I								
		+BDSB3I	+BDSB3I								
		+BDSB1C	+BDSB1C								
		+BDSB2a	+BDSB2a								
		+BDSB2b	+BDSB2b								
		+QZSL1CA	+QZSL1CA								
		+QZSL2C	+QZSL2C								
		+QZSL5	+QZSL5								
		+QZSL6	+QZSL6								
		+QZSL1C	+QZSL1C								
		+QZSL1S	+QZSL1S								
		+QZSL5S	+QZSL5S								
		+QZSL1CB	+QZSL1CB								
		+NAVICL5	+NAVICL5								
		all	all								

RxControl: Navigation > Advanced User Settings > PVT > Signal Usage

Use these commands to define/inquire which signal types are used by the receiver.

The *PVT* argument lists the signals that can be used by the PVT. Removing a signal from the list will disable the usage of the corresponding range, phase & Doppler measurements in the PVT computation.

The *NavData* argument lists the signals for which the receiver is allowed to decode the navigation message. Removing a signal from the list will disable further decoding of the corresponding navigation data (ephemeris, ionosphere parameters ...). Beware that data decoded in the past will still be used. Past data can be erased with the **`exeResetReceiver, hard, SatData+PVTData`** command.

Example

To force the receiver to only use the L1 GPS C/A measurements and navigation information in the PVT solution, use:

```
COM1>  snu, GPSL1CA, GPSL1CA <CR>
$R: snu, GPSL1CA, GPSL1CA
  SignalUsage, GPSL1CA, GPSL1CA
COM1>
```

sspc gspc	**setStaticPosCartesian** **getStaticPosCartesian**	***Position*** *Position*	*X*	*Y*	*Z*	*Datum*				
		+ Cartesian1 all	-8000000.0000 . . . 0.0000 . . . 8000000.0000 m	-8000000.0000 . . . 0.0000 . . . 8000000.0000 m	-8000000.0000 . . . 0.0000 . . . 8000000.0000 m	WGS84 ETRS89 NAD83 NAD83_PA NAD83_MA GDA94 GDA2020 User1 User2 Other				

RxControl: Navigation > Positioning Mode > PVT Mode

Use these commands to define/inquire a set of Cartesian coordinates. This command should be used in conjunction with the **setPVTMode** command to specify a reference position. The Cartesian coordinates in the *X*, *Y* and *Z* arguments must refer to the antenna reference point (ARP), and not to the marker.

The argument *Datum* specifies the datum to which the coordinates refer. When the PVT engine is in static mode (**setPVTMode, Static** command), the specified datum is reflected in the `Datum` field of the position-related SBF blocks (e.g. `PVTCartesian`). Note that the receiver does not apply any datum transformation to the *X*, *Y* and *Z* coordinates. In particular, the coordinates are encoded without change into the relevant differential correction messages.

Datum	**Description**
`WGS84`	WGS84 or ITRFxx (the receiver does not make a distinction between them)
`ETRS89`	European ETRS89 (ETRF2000 realization)
`NAD83`	NAD83(2011), North American Datum (2011)
`NAD83_PA`	NAD83(PA11), North American Datum, Pacific plate (2011)
`NAD83_MA`	NAD83(MA11), North American Datum, Marianas plate (2011)
`GDA94`	GDA94(2010), Geocentric Datum of Australia (2010)
`GDA2020`	GDA2020, Geocentric Datum of Australia 2020
`User1`	First user-defined datum. The corresponding ellipsoid must be defined with the **setUserEllipsoid** command.
`User2`	Second user-defined datum
`Other`	Datum not in the list or unknown

Example

To set up a static base station in Cartesian coordinates:

```
COM1> sspc, Cartesian1, 4019952.028, 331452.954, 4924307.458 <CR>
$R: sspc, Cartesian1, 4019952.028, 331452.954, 4924307.458
  StaticPosCartesian, Cartesian1, 4019952.0280, 331452.9540,
      4924307.4580, WGS84
COM1> spm, Static, , Cartesian1 <CR>
$R: spm, Static, , Cartesian1
```

```
  PVTMode, Static, StandAlone+SBAS+DGPS+RTKFloat+RTKFixed,
     Cartesian1
COM1>
```

sspg gspg	**setStaticPosGeodetic** **getStaticPosGeodetic**	***Position*** *Position*	*Latitude*	*Longitude*	*Altitude*	*Datum*				
		+ Geodetic1 all	-90.000000000 . . . 0.000000000 . . . 90.000000000 deg	-180.000000000 . . . 0.000000000 . . . 180.000000000 deg	-1000.0000 . . . 0.0000 . . . 30000.0000 m	WGS84 ETRS89 NAD83 NAD83_PA NAD83_MA GDA94 GDA2020 User1 User2 Other				

RxControl: Navigation > *Positioning Mode* > *PVT Mode*

Use these commands to define/inquire a set of geodetic coordinates. This command should be used in conjunction with the **`setPVTMode`** command to specify a reference position. The geodetic coordinates in the *Latitude*, *Longitude* and *Altitude* arguments must refer to the antenna reference point (ARP), and not to the marker.

The argument *Datum* specifies the datum to which the coordinates refer. See the **`setStaticPosCartesian`** command for a short description of the supported datums.

Example

To set up a static base station in geodetic coordinates:

```
COM1> sspg, Geodetic1, 50.86696443, 4.71347657, 114.880 <CR>
$R: sspg, Geodetic1, 50.86696443, 4.71347657, 114.880
  StaticPosGeodetic, Geodetic1, 50.86696443, 4.71347657, 114.8800,
     WGS84
COM1> spm, Static, , Geodetic1 <CR>
$R: spm, Static, , Geodetic1
  PVTMode, Static, StandAlone+SBAS+DGPS+RTKFloat+RTKFixed,
     Geodetic1
COM1>
```

stm gtm	**setTroposphereModel** **getTroposphereModel**	*ZenithModel*	*MappingModel*							
		off Saastamoinen MOPS	Niell MOPS							

RxControl: Navigation > Receiver Operation > Position > Atmosphere

Use these commands to define/inquire the type of model used to correct tropospheric errors in the PVT computation.

The *ZenithModel* parameter indicates which model the receiver uses to compute the dry and wet delays for radio signals at 90 degree elevation. The modelled zenith tropospheric delay depends on assumptions for the local air total pressure, the water vapour pressure and the mean temperature. The following zenith models are defined:

ZenithModel	Description
`off`	The measurements will not be corrected for the troposphere delay. This may be desirable if the receiver is connected to a GNSS signal simulator.
`Saastamoinen`	Saastamoinen, J. (1973). "Contributions to the theory of atmospheric refraction". In three parts. Bulletin Geodesique, No 105, pp. 279-298; No 106, pp. 383-397; No. 107, pp. 13-34.
`MOPS`	Minimum Operational Performance Standards for Global Positioning/Wide Area Augmentation System Airborne Equipment RTCA/DO-229C, November 28, 2001.

The `Saastamoinen` model uses user-provided values of air temperature, total air pressure referenced to the Mean Sea Level and relative humidity (see **`setTroposphereParameters`** command) and estimates actual values adjusted to the receiver height.

The MOPS model neglects the user-provided values and instead assumes a seasonal model for all the climatic parameters. Local tropospheric conditions are estimated based on the coordinates and time of the year.

The use of the `Saastamoinen` model can be recommended if external information on temperature, pressure, humidity is available. Otherwise it is advisable to rely on climate models.

The zenith delay is mapped to the current elevation for each satellite using the requested *MappingModel*. The following mapping models are defined:

MappingModel	Description
`Niell`	Niell, A.E. (1996). Global Mapping Functions for the atmosphere delay at radio wavelengths, Journal of Geophysical Research, Vol. 101, No. B2, pp. 3227-3246.
`MOPS`	Minimum Operational Performance Standards for Global Positioning/Wide Area Augmentation System Airborne Equipment RTCA/DO-229C, November 28, 2001.

Examples

```
COM1>  stm, MOPS, MOPS <CR>
$R: stm, MOPS, MOPS
  TroposhereModel, MOPS, MOPS
COM1>

COM1>  gtm <CR>
$R: gtm
  TroposhereModel, MOPS, MOPS
COM1>
```

stp gtp	**setTroposphereParameters** **getTroposphereParameters**	*Temperature*	*Pressure*	*Humidity*						
		-100.0 ... 15.0 ... 100.0 degC	800.00 ... 1013.25 ... 1500.00 hPa	0 ... 50 ... 100 %						

RxControl: Navigation > Receiver Operation > Position > Atmosphere

Use these commands to define/inquire the climate parameters to be used when the zenith troposphere is estimated using the Saastamoinen model (see the **`setTroposphereModel`** command).

The troposphere model assumes the climate parameters to be valid for a receiver located at the Mean Sea Level (MSL). If you want to use your receiver with a weather station, you have to convert the measured *Temperature*, *Pressure* and *Humidity* to MSL.

Example

```
COM1>  stp, 25, 1013, 60<CR>
$R: stp, 25, 1013, 60
  TroposphereParameters, 25.0, 1013.00, 60
COM1>
```

3.2.7 Authentication

lopk	**lstGalOSNMAPublicKeys**									

Use this command to retrieve the list of applicable OSNMA public keys.

The list contains user-defined public keys, as introduced with **`setGalOSNMAPublicKeys`** command, possibly updated with new keys provided through the Galileo OSNMA protocol (over the air).

This command is very similar to the command **`getGalOSNMAPublicKeys`**, the only difference being that the latter only reports the list of user-defined public keys.

Example

```
COM1>  lstGalOSNMAPublicKeys <CR>
$R; lstGalOSNMAPublicKeys
---->
$-- BLOCK 1 / 1
  GalOSNMAPublicKeys, Key0, ""
  GalOSNMAPublicKeys, Key1, "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE+
     Q2wvmvfdQg1sQF6OmCEy8skCSiu79vBnRrKmaPpCJnaMOOvm26Us6ELhebL+
     q75MAyWAXJjlyRZZwp68gSAHw=="
...
COM1>
```

sopk gopk	**setGalOSNMAPublicKeys** **getGalOSNMAPublicKeys**	***ID*** *ID*	***Key (233)***							
		+ Key0 ... Key15 all								

RxControl: Navigation > Advanced User Settings > Galileo OSNMA > Public Keys

Public keys for live OSNMA operation are hardcoded in the receiver, as they are not expected to change frequently. In order to have the OSNMA function operate with live signals, the user is not required to input any keys.

However, in simulated environments, different keys may be needed and they can be provided with this command. OSNMA defines 16 different keys which can be provided individually with the *ID* and *Key* arguments.

The format of the *Key* argument is equivalent to PEM (Private Mail Enhanced), a Base64 encoded certificate, but without the "—BEGIN—" header and "—END—" footer.

⚠ If the keys provided with this command do not correspond to the Galileo keys, the receiver will not be able to authenticate live Galileo messages. Make sure to delete all user-selected keys (e.g. with the **`setGalOSNMAPublicKeys,all,""`** command) when leaving the simulated environment.

Example

```
COM1>  sopk, Key2,
   ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEnAtF3t3kbYx6tH80MEIuis+
   HtLdGNGU8Cj8kUesPfc/OEbNRcbedY5iQHsc+t5bEN0GV6gkLIp0= <CR>
$R: sopk, Key2,
   ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEnAtF3t3kbYx6tH80MEIuis+
   HtLdGNGU8Cj8kUesPfc/OEbNRcbedY5iQHsc+t5bEN0GV6gkLIp0=
  GalOSNMAPublicKeys, Key2,
     "ME4wEAYHKoZIzj0CAQYFK4EEACEDOgAEnAtF3t3kbYx6tH80MEIuis+
     HtLdGNGU8Cj8kUesPfc/OEbNRcbedY5iQHsc+t5bEN0GV6gkLIp0="
COM1>
```

sou gou	**setGalOSNMAUsage** **getGalOSNMAUsage**	*Mode*	*MTRoot (65)*							
		off loose strict								

RxControl: Navigation > Advanced User Settings > Galileo OSNMA > Settings

Use this command to configure the OSNMA processing of the receiver.

By default, the *Mode* argument is set to `off` and the authentication function is switched off. No OSNMA authentication checking is being performed and no authentication results presented.

In `loose` mode, authentication results are reported in the `GALAuthStatus` SBF block and used for those satellites supported by OSNMA. In case authentication fails for a particular satellite, it is excluded from the PVT. In all other cases (authentication successful or unknown), it is used in the PVT.

In `strict` mode, only proven authentic satellites are included in the PVT. Satellites for which authentication is not available (e.g. BeiDou satellites) or which have not been verified yet are excluded. The reported PVT solution is solely based on authenticated satellites.

Another difference between `loose` and `strict` modes is the usage of an NTP time server. In `loose` mode, NTP access is optional. In `strict` mode, it is mandatory. See also the **`setNTPClient`** command.

The *MTRoot* argument allows users to specify the root of the Merkle Tree used to validate new public keys. The format is an hexademical string representing the bits of the key. This is only needed in simulated environments. When using the receiver for live OSNMA operation, no Merkle Root key should be specified (the argument should be left blank).

Example

```
COM1>  sou, strict <CR>
$R: sou, strict
  GalOSNMAUsage, strict, ""
COM1>
COM1>  sou, loose,
   AB20435CB64A837C8EACC6FF427416433DFE715F1607D361236A2C163B9E0A5A
   <CR>
$R: sou, loose,
   "AB20435CB64A837C8EACC6FF427416433DFE715F1607D361236A2C163B9E0A5A
   "
  GalOSNMAUsage, loose,
     "AB20435CB64A837C8EACC6FF427416433DFE715F1607D361236A2C163B9E0A5A
     "
COM1>
```

3.2.8 Datum Definition

sgd ggd	**setGeodeticDatum** **getGeodeticDatum**	*TargetDatum*								
		WGS84 ETRS89 NAD83 NAD83_PA NAD83_MA GDA94 GDA2020 Default User1 User2								

RxControl: Navigation > Receiver Operation > Position > Datum

Use this command to define the datum to which you want the coordinates to refer.

TargetDatum	**Description**
`WGS84`	Equivalent to `Default`
`ETRS89`	European ETRS89 (ETRF2000 realization)
`NAD83`	NAD83(2011), North American Datum (2011)
`NAD83_PA`	NAD83(PA11), North American Datum, Pacific plate (2011)
`NAD83_MA`	NAD83(MA11), North American Datum, Marianas plate (2011)
`GDA94`	GDA94(2010), Geocentric Datum of Australia (2010)
`GDA2020`	GDA2020, Geocentric Datum of Australia 2020
`Default`	Default datum, which depends on the positioning mode as explained below.
`User1`	First user-defined datum. The corresponding transformation parameters must be specified by the **`setUserDatum`** and **`setUserDatumVel`** commands, while the corresponding ellipsoid must be defined by the **`setUserEllipsoid`** command.
`User2`	Second user-defined datum

By default (argument *TargetDatum* set to `Default`), the datum depends on the positioning mode. For standalone and SBAS positioning, the coordinates refer to a global datum: WGS84 or ITRF (recent realisations of WGS84 and ITRF are closely aligned and the receiver considers them equivalent). When using PPP corrections, the coordinates refer to ITRF. When using DGNSS or RTK corrections from a regional DGNSS/RTK provider, the coordinates usually refer to a regional datum (e.g. ETRS89 in Europe or NAD83 in North America).

With this command, the user can select the datum the coordinates should refer to. In case you are using corrections from a regional DGNSS/RTK provider, the datum to be specified here must be the datum used by your correction provider.

When a non-default datum is selected, the receiver transforms the coordinates obtained in standalone, SBAS and PPP modes to the specified datum. Positions obtained using local or regional DGNSS/RTK corrections are not transformed, as it is assumed that the selected datum is the one used by the DGNSS/RTK provider.

In the current firmware version, the `WGS84` value for the *TargetDatum* argument has no effect, but it is kept for backwards compatibility reasons. Setting *TargetDatum* to `WGS84` is equivalent to setting it to `Default`.

Example

```
COM1>  sgd, ETRS89 <CR>
$R: sgd, ETRS89
  GeodeticDatum, ETRS89
COM1>
```

sud gud	**setUserDatum** **getUserDatum**	***Datum*** *Datum*	*Tx*	*Ty*	*Tz*	*Rx*	*Ry*	*Rz*	*D*	
		+ User1 + User2 all	-2000000.00 . . . 0.00 . . . 2000000.00 mm	-2000000.00 . . . 0.00 . . . 2000000.00 mm	-2000000.00 . . . 0.00 . . . 2000000.00 mm	-100.0000 . . . 0.0000 . . . 100.0000 mas	-100.0000 . . . 0.0000 . . . 100.0000 mas	-100.0000 . . . 0.0000 . . . 100.0000 mas	-100.00000 . . . 0.00000 . . . 100.00000 ppb	

RxControl: Navigation > Receiver Operation > Position > Datum

Use these commands to define datum transformation parameters from the global WGS84/ITRF datum to the user datum identified by the first argument.

The receiver applies the linearized form of the Helmert similarity transformation. The coordinates in WGS84/ITRF are transformed to the user datum using the following formula:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{User} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix} + \begin{pmatrix} D+1 & -R_z & R_y \\ R_z & D+1 & -R_x \\ -R_y & R_x & D+1 \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{WGS84/ITRF}$$

where T_x, T_y and T_z are the three translation components, R_x, R_y and R_z are the rotation angles and D is the scale factor. Note that the rotation angles are expressed in radians in the above formula, but they must be provided in milliarcsecond (1 mas = $2\pi/360/3600000$ radians) in the arguments of the command. The sign convention corresponds to that of the IERS Conventions (2010), Technical Note No. 36.

The time derivative of the transformation parameters can be specified with the command **`setUserDatumVel`**.

For the receiver to apply the transformation parameters, the corresponding user datum must be selected in the **`setGeodeticDatum`** command.

Example

```
COM1>  sud, User1, 52.1, 49.3, -58.5, 0.891, 5.390, -8.712,
    1.34<CR>
$R: sud, User1, 52.1, 49.3, -58.5, 0.891, 5.390, -8.712, 1.34
  UserDatum, User1, 52.10, 49.30, -58.50, 0.8910, 5.3900, -8.7120,
     1.34000
COM1>
```

sudv gudv	**setUserDatumVel** **getUserDatumVel**	***Datum*** *Datum*	*TxVel*	*TyVel*	*TzVel*	*RxVel*	*RyVel*	*RzVel*	*DVel*	*RefYear*
		+ User1 + User2 all	-2000.00 . . . 0.00 . . . 2000.00 mm/yr	-2000.00 . . . 0.00 . . . 20000.00 mm/yr	-2000.00 . . . 0.00 . . . 2000.00 mm/yr	-10.0000 . . . 0.0000 . . . 10.0000 mas/yr	-10.0000 . . . 0.0000 . . . 10.0000 mas/yr	-10.0000 . . . 0.0000 . . . 10.0000 mas/yr	-1.00000 . . . 0.00000 . . . 1.00000 ppb/yr	1900.00 . . . 2000.00 . . . 2100.00 yr

RxControl: Navigation > Receiver Operation > Position > Datum

Use these commands to define the time derivative of the seven datum transformation parameters defined with the **`setUserDatum`** command.

For instance, *TxVel* is the yearly change of the X-translation component. At the epoch specified with *RefYear* (in decimal years), the X-translation component is *Tx* as defined in **`setUserDatum`**. One year later, the X-translation component is *Tx*+*TxVel*, etc.

Refer to the **`setUserDatum`** command for a description of the datum transformation formula implemented in the receiver.

Example

```
COM1>  sudv, User1, 0.1, 0.1, -1.8, 0.081, 0.49, -0.792, 0.08,
   2000<CR>
$R: sudv, User1, 0.1, 0.1, -1.8, 0.081, 0.49, -0.792, 0.08, 2000
  UserDatumVel, User1, 0.10, 0.10, -1.80, 0.0810, 0.4900, -0.7920,
     0.08000, 2000.00
COM1>
```

sue gue	**setUserEllipsoid** **getUserEllipsoid**	***Datum*** *Datum*	*A*	*Invf*						
		+ User1 + User2 all	6300000.000 . . . 6378137.000 . . . 6400000.000 m	290.000000000 . . . 298.257223563 . . . 305.000000000						

RxControl: Navigation > Receiver Operation > Position > Datum

Use these commands to define the ellipsoid associated with the `User1` or `User2` datum.

a is the reference ellispoid semi-major axis and *Invf* is the inverse of the flattening.

See also the **`setGeodeticDatum`** and the **`setUserDatum`** commands.

Example

```
COM1>  sue, User1, 6378388, 297 <CR>
$R: sue, User1, 6378388, 297
  UserEllipsoid, User1, 6378388.000, 297.000000000
COM1>
```

3.2.9 Timing and Time Management

scst gcst	**setClockSyncThreshold** **getClockSyncThreshold**	*Threshold*	*StartupSync*							
		ClockSteering usec500 msec1 msec2 msec3 msec4 msec5	off on							

RxControl: Navigation > Receiver Operation > Timing

Use these commands to define/inquire the maximum allowed offset between the receiver internal clock and the system time defined by the **setTimingSystem** command.

If the argument `ClockSteering` is selected, the receiver internal clock is continuously steered to the system time to within a couple of nanoseconds. Clock steering accuracy is dependent on the satellite visibility, and it is recommended to only enable it under open-sky conditions.

If any other argument is selected, the internal clock is left free running. Synchronization with the system time is done through regular millisecond clock jumps. More specifically, when the receiver detects that the time offset is larger than *Threshold*, it initiates a clock jump of an integer number of milliseconds to re-synchronise its internal clock with the system time. These clock jumps have no influence on the generation of the xPPS pulses: the xPPS pulses are always maintained within a few nanoseconds from the requested time, regardless of the value of the *Threshold* argument.

The *StartupSync* argument can be used to force the receiver to start with a small clock bias value (less than 100ns). This setting will only have effect at the next reset or reboot of the receiver. So, when enabling startup synchronization (*StartupSync* set to `on`), do not forget to save the configuration in the boot configuration file with the **exeCopyConfigFile** command.
This argument has no effect when clock steering is enabled or when a PPS IN signal is available.

Refer to section 2.3 for a more detailed description of the time keeping in your receiver.

Example

```
COM1>  scst, msec1, on<CR>
$R: scst, msec1, on
  ClockSyncThreshold, msec1, on
COM1>
```

sep gep	**setEventParameters** **getEventParameters**	***Event*** *Event*	*Polarity*	*Delay*						
		+ EventA + EventB all	Low2High High2Low	-500.000000 . . . 0.000000 . . . 500.000000 ms						

RxControl: Navigation > Receiver Operation > Timing

Use these commands to define/inquire the polarity of the electrical transition on which the receiver will react on its `Event` input(s). The polarity of each event pin can be set individually or simultaneously by using the value `all` for the *Event* argument.

The command also allows defining a time delay for each event. This can be handy when the electrical transition at the event pin is not synchronous with the actual event that needs to be timed. For example, if the electrical transition occurs 100 milliseconds prior to the actual event of interest, the *Delay* argument must be set to 100. *Delay* is positive when the event of interest occurs after the electrical transition, and negative otherwise.

The event time (corrected by the specified delay) is available in the `ExtEvent` SBF block. The position at that time is available in the `ExtEventPVTCartesian` and `ExtEventPVTGeodetic` SBF blocks. Beware that, when using large *Delay* values in high-dynamics conditions, the position accuracy may degrade.

Example

```
COM1>  sep, EventA, High2Low, 10<CR>
$R: sep, EventA, High2Low, 10
  EventParameters, EventA, High2Low, 10.000000
COM1>
```

snc gnc	**setNtpClient** **getNtpClient**	*Mode*	*Server (40)*							
		on off	default							

RxControl: Navigation > Receiver Operation > Timing

Use this command to configure the retrieval of the time from an external NTP time server. Getting the time from an external source can help the receiver to perform some sanity checks on the signal received from the GNSS satellites. It is fully optional, except in case of strict OSNMA operation (see the **`setGalOSNMAUsage`** command).

If *Mode* is `on`, the receiver will attempt to get the current time from the NTP server specified with the *Server* argument.

The *Server* argument accepts a host name or a raw IP address. If set to "`default`", a server is automatically selected by the receiver.

Accessing the NTP server requires the receiver to have access to the Internet. If the receiver is not connected to a network, the access fails with no error message.

Example

```
COM1>  snc, on, pool.ntp.org<CR>
$R: snc, on, pool.ntp.org
  NtpClient, on, pool.ntp.org
COM1>
```

sntp gntp	**setNTPServer** **getNTPServer**	*Enable*								
		off on								

RxControl: Navigation > Receiver Operation > Timing

Use this command to enable or disable the built-in NTP (Network Time Protocol) server.

When enabled, the NTP server accepts UDP timestamp requests on port number 123.

Example

```
COM1>  sntp, on<CR>
$R: sntp, on
  NTPServer, on
COM1>
```

spps gpps	**setPPSParameters** **getPPSParameters**	*Interval*	*Polarity*	*Delay*	*TimeScale*	*MaxSyncAge*	*PulseWidth*			
		off msec10 msec20 msec50 msec100 msec200 msec250 msec500 sec1 sec2 sec4 sec5 sec10 sec30 sec60	Low2High High2Low	-1000000.00 . . . 0.00 . . . 1000000.00 ns	GPS Galileo BeiDou GLONASS UTC RxClock	0 . . . 60 . . . 3600 s	0.001 . . . 5.000 . . . 1000.000 ms			

RxControl: Navigation > Receiver Operation > Timing

Use these commands to define/inquire the parameters of the x-pulse-per-second (xPPS) output. Refer to section 1.27 for additional information on the xPPS functionality.

The *Interval* argument specifies the time interval between the pulses. A special value "`off`" is defined to disable the xPPS signal.

The *Polarity* argument defines the polarity of the xPPS signal.

The *Delay* argument can be used to compensate for the overall signal delays in the system (including antenna, antenna cable and xPPS cable). Setting *Delay* to a higher value causes the xPPS pulse to be generated earlier. For example, if the antenna cable is replaced by a longer one, the overall signal delay could be increased by, say, 20 ns. If *Delay* is left unchanged, the xPPS pulse will come 20 ns too late. To re-synchronize the xPPS pulse, *Delay* has to be increased by 20 ns.

The xPPS pulses are aligned with the time system set with the *TimeScale* argument. `RxClock` corresponds to the receiver time scale. When setting *TimeScale* to `RxClock`, the xPPS pulses are synchronous with the internal measurement epochs.

The xPPS timing information is derived primarily from the satellites of the system selected with the *TimeScale* argument. If there is no satellite from that system available, the receiver will use information from other constellations to maintain the continuity of the pulses. If all satellite signals are blocked, the xPPS pulses will continue to be generated for a duration set with the *MaxSyncAge* argument. If *MaxSyncAge* is set to 0, or if *TimeScale* is set to `RxClock`, this timeout is disabled.

The *PulseWidth* argument sets the duration of the PPS pulse. The maximum duration of the pulse equals the PPS interval minus 100 microseconds, even if the requested *PulseWidth* is longer than that.

Example

```
COM1>  spps, sec1, Low2High, 23.40, GPS, 60, 0.1<CR>
$R: spps, sec1, Low2High, 23.40, GPS, 60, 0.1
  PPSParameters, sec1, Low2High, 23.40, GPS, 60, 0.100
COM1>
```

srom grom	**setREFOUTMode** **getREFOUTMode**	*Enable*	*Disciplined*	*FreqOffset*						
		off on	off on	-1.000 . . . 0.000 . . . 1.000 ppm						

RxControl: Navigation > Advanced User Settings > Receiver Clock

Use this command to configure the 10-MHz REF OUT frequency reference output of the receiver.

The *Enable* argument enables or disables the frequency reference output at the REF OUT connector.

The *Disciplined* argument enables or disables the disciplining of the frequency reference to GNSS time. When enabled, the frequency reference at the REF OUT connector is locked to GNSS time scale. This locking process can take a few minutes. When lock is achieved, the REFOUT_LOCKED bit is set in the `RxState` field of the `ReceiverStatus` SBF block.

Instead of locking the frequency to the GNSS time scale, a small offset can be forced with the *FreqOffset* argument. In that case, the REFOUT_LOCKED bit is never set.

Note that the *Disciplined* and the *FreqOffset* arguments have no effect when the receiver is fed with an external frequency reference through the REF IN connector.

Example

```
COM1>  srom, on, off, 0.0<CR>
$R: srom, on, off, 0.0
  REFOUTMode, on, off, 0.000
COM1>
```

sts gts	**setTimingSystem** **getTimingSystem**	*System*								
		Galileo GPS BeiDou auto								

RxControl: Navigation > Receiver Operation > Timing

Use these commands to define/inquire the reference time system for the computation of the receiver clock bias.

As part of the PVT computation, the receiver determines the offset between its own time (receiver time) and the time of the GNSS system specified with the *System* argument. This offset is reported in the `RxClkBias` field of the `PVTCartesian` and `PVTGeodetic` SBF blocks.

⚠ Note that at least one satellite of the selected system must be visible and tracked by the receiver. Otherwise no PVT will be computed.

When the *System* argument is set to `auto`, the receiver automatically selects the GNSS system according to the availability of satellites. This is the recommended setting.

Example

```
COM1>  sts, GPS<CR>
$R: sts, GPS
  TimingSystem, GPS
COM1>
```

3.2.10 Time Transfer Configuration

scdy gcdy	**setCABDLY** **getCABDLY**	*CABDLY*								
		-1000.000 . . . 0.000 . . . 1000.000 ns								

RxControl: Logging > Internal CGGTTS Logging > CGGTTS Parameters

This command sets the electrical delay in the cable from the antenna to the receiver. This delay is commonly referred to as X_C in the time-transfer literature.

The delay provided by this command is only used for the CGGTTS processing, and does not otherwise affect the receiver operation.

Example

```
COM1>  scdy, 36.411<CR>
$R: scdy, 36.411
  CABDLY, 36.411
COM1>
```

scpo gcpo	**setCGGTTSPosition** **getCGGTTSPosition**	***SatSystem*** *SatSystem*	*Mode*	*X*	*Y*	*Z*				
		+ GPS + GLONASS + GALILEO + BEIDOU all	manual	-8000000.0000 . . . 0.0000 . . . 8000000.0000 m	-8000000.0000 . . . 0.0000 . . . 8000000.0000 m	-8000000.0000 . . . 0.0000 . . . 8000000.0000 m				

RxControl: Logging > Internal CGGTTS Logging > CGGTTS Parameters

Use this command to specify the antenna coordinates applicable to the CGGTTS processing. Contrary to other commands dealing with the antenna position, the (X, Y, Z) coordinates refer to the mean antenna phase center and not to the ARP. If needed, if is possible to enter different coordinates for the different GNSS systems.

The specified coordinates are copied to the CGGTTS file header and are used in the computation of the various fields of the CGGTTS file. They otherwise do not affect the receiver operation.

Example

```
COM1>  scpo, all, manual, 4021429.16, 332877.37, 4923031.62<CR>
$R: scpo, all, manual, 4021429.16, 332877.37, 4923031.62
  CGGTTSPosition, all, manual, 4021429.1600, 332877.3700,
     4923031.6200
COM1>
```

scgt gcgt	**setCGGTTSTrackParameters** **getCGGTTSTrackParameters**	*ElMask*	*TrackSchedule*							
		-90 . . . 10 . . . 90 deg	BIPM Contiguous							

RxControl: Logging > Internal CGGTTS Logging > CGGTTS Parameters

This command configures some parameters of the CGGTTS computation.

The *ElMask* argument specifies the elevation mask to be used when computing the CGGTTS data. This mask is only used in the context of CGGTTS. The global receiver elevation mask is set with the **`setElevationMask`** command.

The scheduling of the 13-minute CGGTTS tracks is defined with the *TrackSchedule* argument. By default, the conventional BIPM scheduling is applied (13-minute tracks starting at pre-defined epochs with 3-minute idle intervals between tracks). In `Contiguous` mode, the CGGTTS parameters are computed every 13 minutes without idle interval, starting directly when CGGTTS logging is enabled. This non-standard mode provides a faster update rate of the time-transfer parameters.

Example

```
COM1>  scgt, 15, BIPM<CR>
$R: scgt, 15, BIPM
  CGGTTSTrackParameters, 15, BIPM
COM1>
```

sidy gidy	**setINTDLY** **getINTDLY**	***Frequency*** *Frequency*	*INTDLY*							
		+GPSL1 +GPSL2 +GLOL1 +GLOL2 +GALE1 +GALE5a +BDSB1I +BDSB2I all	-1000.000 . . . 0.000 . . . 1000.000 ns							

RxControl: Logging > Internal CGGTTS Logging > CGGTTS Parameters

This command sets the electrical delays (INTDLY) applicable to the receiver and antenna. *INTDLY* is the sum of the delay in the antenna and in the receiver, commonly referred to as X_S and X_R in the literature. These delays are frequency dependent and are typically determined through a calibration campaign.

The delays provided by this command are only used for the CGGTTS processing, and do not otherwise affect the receiver operation.

Example

```
COM1>  sidy, BDSB1I, 3.65<CR>
$R: sidy, BDSB1I, 3.65
  INTDLY, BDSB1I, 3.650
COM1>
```

spim gpim	**setPPSINMonitor** **getPPSINMonitor**	*Mode*								
		off on								

RxControl: Navigation > Receiver Operation > Timing

By default, the receiver uses the PPS IN signal only at startup for initial synchronization. During operation, PPS IN is ignored and the receiver time is kept in sync by counting the cycles of the 10-MHz REF IN reference. It is assumed that the phase of PPS IN remains constant with respect to the 10-MHz reference, or in other words, that there are exactly 10 millions of cycles of the 10-MHz reference between each PPS IN pulse.

In some cases, this assumption is not valid, for example when the PPS IN phase can jump independently of the 10 MHz reference. When enabling PPS IN monitoring with the **`setPPSINMonitor`** command, the receiver continuously monitors the phase of the PPS IN signal. In case of jump, the receiver resets and resynchronizes to the new PPS IN phase. This ensures that the receiver time is always aligned with PPS IN. The jump detection threshold is set to about one nanosecond.

Example

```
COM1>  spim, on<CR>
$R: spim, on
  PPSINMonitor, on
COM1>
```

spin gpin	**setPPSINParameters** **getPPSINParameters**	*XOCompensation*								
		off auto								

RxControl: Navigation > Receiver Operation > Timing

Use this command to enable or disable automatic compensation of the delay between the PPS IN pulse at the PPS IN connector and the receiver internal time reference. This delay is commonly referred to as X_{O} in the literature.

When delay compensation is enabled (the *XOCompensation* argument is set to `auto`), X_{O} can be considered zero in all time transfer applications.

When delay compensation is disabled, X_{O} is not zero and must be measured by external means.

Refer to the User Manual for more details on the usage of the PPS IN input signal.

Example

```
COM1>  spin, off<CR>
$R: spin, off
  PPSINParameters, off
COM1>
```

srdy grdy	**setREFDLY** **getREFDLY**	*REFDLY*								
		-1000.000 . . . 0.000 . . . 1000.000 ns								

RxControl: Logging > Internal CGGTTS Logging > CGGTTS Parameters

This command sets the electrical delay between the laboratory clock and the receiver internal time reference. This delay is commonly referred to as $X_P + X_O$ in the literature. If the PPS IN internal delay compensation is enabled with the **`setPPSINParameters`** command, this is the delay in the cable from the clock to the PPS IN connector of the receiver (X_P).

See also the User Manual for a description of the various delays relevant to time transfer.

The delay provided by this command is only used for the CGGTTS processing, and does not otherwise affect the receiver operation.

Example

```
COM1>  srdy, 23.4<CR>
$R: srdy, 23.4
  REFDLY, 23.400
COM1>
```

stmp gtmp	**setTimeLabParameters** **getTimeLabParameters**	*LabName (40)*	*LabID (2)*	*RxID (2)*	*CalID (30)*	*CalRevDate (30)*	*TimeRef (30)*	*Comment (80)*		

RxControl: Logging > Internal CGGTTS Logging > CGGTTS Parameters

Use this command to specify the value of various fields of the CGGTTS file header and file name.

LabName is the name of the time laboratory, corresponding to the "LAB" field of the CGGTTS header.

LabID is the 2-character code assigned to the laboratory, used in the CGGTTS file name.

RxID is the 2-character receiver identification, used in the CGGTTS file name.

CalID is the calibration ID, corresponding to the "CAL_ID" field of the CGGTTS header.

CalRevDate is the revision date of the CGGTTS header data, corresponding to the "REV DATE" field of the CGGTTS header.

TimeRef is the identifier of the time reference, corresponding to the "REF" field of the CGGTTS header.

Comment is any comment, which will be added to the "COMMENTS" field of the CGGTTS header.

Example

```
COM1>  stmp, MyLab, L1, R1, MyCal, 2017-12-31, UTC(MyLab), This is
   a comment<CR>
$R: stmp, MyLab, L1, R1, MyCal, 2017-12-31, UTC(MyLab), This is a
   comment
  TimeLabParameters, MyLab, L1, R1, MyCal, 2017-12-31, UTC(MyLab),
     This is a comment
COM1>
```

3.2.11 Station Settings

smp gmp	**setMarkerParameters** **getMarkerParameters**	*MarkerName (60)*	*MarkerNumber (2*	*MarkerType (20)*	*StationCode (10)*	*MonumentIdx*	*ReceiverIdx*	*CountryCode (3)*		
		SEPT	Unknown	Unknown		0 . . . 9	0 . . . 9			

RxControl: Navigation > Receiver Setup > Station Settings

Use these commands to define/inquire the marker and station parameters.

The set of allowed characters for the *MarkerName* argument and for the *StationCode* argument is limited to:

`_0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`

The *StationCode* argument is the site code associated to the station (typically four characters). *MonumentIdx* can be used to identify the monument when there are multiple monuments at the same station. *ReceiverIdx* can be used to identify the receiver when there are multiple receivers at the same monument. A three-letter ISO country code can be specified with the *CountryCode* argument.

If internal logging is enabled in one of the `IGS` file naming modes, the file name depends on the settings of the **setMarkerParameters** command. Refer to the description of the logging commands (**setFileNaming**, **setBINEXLoggingParameters**, **setNMEALogging**, **setRTCMMSMLogging**, **setRINEXLogging**) for details.

The parameters set by this command are copied into the `ReceiverSetup` SBF block, which defines the file name and the header contents when converting SBF files into RINEX with the `sbf2rin` program.

Example

```
COM1>  smp, Test, 356, GEODETIC, TST1, 0, 0, BEL<CR>
$R: smp, Test, 356, GEODETIC, TST1, 0, 0, BEL
  MarkerParameters, Test, 356, GEODETIC, TST1, 0, 0, BEL
COM1>
```

soc goc	**setObserverComment** **getObserverComment**	*Comment (120)*								
		Unknown								

RxControl: Navigation > Receiver Setup > Station Settings

Use these commands to define/inquire the content of the `Comment` SBF block.

Examples

```
COM1>  soc, "Data taken with choke ring antenna"<CR>
$R: soc, "Data taken with choke ring antenna"
  ObserverComment, "Data taken with choke ring antenna"
COM1>

COM1>  goc <CR>
$R: goc
  ObserverComment, "Data taken with choke ring antenna"
COM1>
```

sop gop	**setObserverParameters** **getObserverParameters**	*Observer (20)*	*Agency (40)*							
		Unknown	Unknown							

RxControl: Navigation > Receiver Setup > Station Settings

Use these commands to define/inquire the observer name or ID, and his/her agency. These parameters are copied in the `ReceiverSetup` SBF block and in the header of RINEX observation files.

The length of the arguments complies with the RINEX format definition.

Examples

```
COM1>  sop, TestObserver, TestAgency <CR>
$R: sop, TestObserver, TestAgency
  ObserverParameters, "TestObserver", "TestAgency"
COM1>

COM1>  gop <CR>
$R: gop
  ObserverParameters, "TestObserver", "TestAgency"
COM1>
```

3.2.12 General Input/Output

scs	**setCOMSettings**	***Cd***	*Rate*	*DataBits*	*Parity*	*StopBits*	*FlowControl*			
gcs	**getCOMSettings**	*Cd*								
		+ COM1	baud1200	bits8	No	bit1	none			
		+ COM2	baud2400				RTS|CTS			
		+ COM3	baud4800							
		+ COM4	baud9600							
		all	baud19200							
			baud38400							
			baud57600							
			baud115200							
			baud230400							

RxControl: Communication > COM Port Settings

Use these commands to define/inquire the communication settings of the receiver's COM ports. By default, all COM ports are set to a baud rate of 115200 baud, using 8 data-bits, no parity, 1 stop-bit and no flow control.

Depending on your receiver hardware, it may be that not all COM ports support flow control. Please refer to the receiver Hardware Manual or User Manual to check which COM ports are equipped with the RTS/CTS lines.

When modifying the settings of the current connection, make sure to also modify the settings of your terminal emulation program accordingly.

Example

```
COM1>  scs, COM1, baud19200, bits8, No, bit1, RTS|CTS<CR>
$R: scs, COM1, baud19200, bits8, No, bit1, RTS|CTS
  COMSettings, COM1, baud19200, bits8, No, bit1, RTS|CTS
COM1>
```

scda gcda	**setCrossDomainWebAccess** **getCrossDomainWebAccess**	*Mode*								
		off on								

RxControl: Communication > Network Settings > General

This command enables or disables true open access across domain boundaries according to the CORS specification (Cross-Origin Resource Sharing).

Setting the *Mode* argument to `on` enables the cross-domain access to the receiver web server, and as such it allows external client applications (e.g. your own web application) to access receiver data via HTTP requests. Please contact Septentrio support for additional information on the receiver's JavaScript libraries.

Example

```
COM1>  scda, on <CR>
$R: scda, on
  CrossDomainWebAccess, on
COM1>
```

sdcm gdcm	**setDaisyChainMode** **getDaisyChainMode**	***DC*** *DC*	*Mode*							
		+DC1 +DC2 all	Raw ASCII							

RxControl: Communication > Input/Output Selection

Use this command to define how data is transferred in a daisy chain configured with the **`setDataInOut`** command.

By default (*Mode* is `Raw`), incoming bytes are transferred in small chunks from the input to the output connector.

In some cases, it is preferred to transmit complete ASCII strings at once. This can be done by configuring the daisy chain in ASCII mode. A string is considered complete when a carriage-return and/or a line-feed character is received.

Example

```
COM1>  sdcm, DC1, ASCII<CR>
$R: sdcm, DC1, ASCII
  DaisyChainMode, DC1, ASCII
COM1>
```

sdio gdio	**setDataInOut** **getDataInOut**	***Cd*** *Cd*	*Input*	*Output*	*Show*					
		+COM1	none	none	(off)					
		+COM2	CMD	+RTCMv2	(on)					
		+COM3	RTCMv2	+RTCMv3	(waiting)					
		+COM4	RTCMv3	+CMRv2						
		+USB1	CMRv2	+SBF						
		+USB2	DC1	+NMEA						
		+IP10 ... IP17	DC2	+ASCIIDisplay						
		+NTR1	RTCMV	+DC1						
		+NTR2	ASCIIIN	+DC2						
		+NTR3	auto	+BINEX						
		+IPS1		+LBandBeam1						
		+IPS2		+LBandBeam2						
		+IPS3								
		+IPS4								
		+IPS5								
		+IPR1								
		+IPR2								
		+IPR3								
		+LOG1 ... LOG8								
		all								

RxControl: Communication > Input/Output Selection

Use these commands to define/inquire the type of data that the receiver should accept/send on a given connection descriptor (*Cd* - see 1.1.5).

The *Input* argument is used to tell the receiver how to interpret incoming bytes on the connection *Cd*. If a connection is to be used for receiving user commands or differential corrections in RTCM or CMR format, it is recommended to leave it in the default `auto` input mode. In this mode, the receiver automatically detects the input format.

It is also possible to set the input format explicitly. `CMD` means that the connection is to be used for user command input exclusively. `RTCMv2`, `RTCMv3` and `CMRv2` can be used to manually select the differential correction format, overriding the auto detection. `RTCMV` is the LBAS1-proprietary differential correction stream decoded from L-Band. `ASCIIIN` is used for connections receiving free-formatted ASCII messages, e.g. from an external meteo sensor.

In `auto` mode, the receiver automatically detects the `CMD`, `RTCMv2`, `RTCMv3`, `RTCMV`, or `CMRv2` formats. The other input formats must be specified explicitly.

A connection that is not configured in `CMD` mode or `auto` mode will be blocked for user commands. There are two ways to re-enable the command input on a blocked connection. The first way is to reconfigure the connection by entering the command **`setDataInOut`** from another connection. The second way is to send the "escape sequence" consisting of a succession of ten "S" characters to the blocked connection within a time interval shorter than 5 seconds.

A connection that is configured in `auto` mode will initially accept user commands and differential corrections. However, as soon as differential corrections have been detected, the connection is blocked for user commands until the escape sequence is received.

The *Output* argument is used to select the types of data allowed as output. The receiver supports outputting different data types on the same connection. The `ASCIIDisplay` is a textual report of the tracking and PVT status at a fixed rate of 1Hz. It can be used to get a quick overview of the receiver operation. The `LBandBeami` option is to output the stream of bytes decoded from an L-Band beam (not available for some proprietary streams).

When opening an `IPxx` connection, the *Input* and *Output* modes are always reset to their default value.

`DC1` and `DC2` represent two internal pipes that can be used to create a daisy-chain. Set the *Input* argument to `DCi` to connect the input of pipe *i* to the specified connection. Set the *Output* argument to `DCi` to connect the output of pipe *i* to the specified connection. The daisy-chain can operate in binary or ASCII mode, as configured with the **`setDaisyChainMode`** command.

After the *Cd*, *Input* and *Output* arguments, an extra read-only *Show* argument will be returned in the command reply. This last argument can take the value `on`, `off` or `waiting`, depending on whether the connection descriptor is open, close, or waiting for a connection.

The *Input* argument is ignored for output-only connections, and the *Output* argument is ignored for input-only connections. See section 1.1.5 for details.

Note that not all input connections can accept user commands, check section 1.1.5 for details.

Examples

```
COM1>  sdio, COM1, CMD <CR>
$R: sdio, COM1, CMD
  DataInOut, COM1, CMD, SBF, (on)
COM1>
```

On receivers that have three COM ports, to set up a two-way daisy-chain between COM2 and COM3, i.e. to have all incoming bytes from COM2 redirected to COM3 and all incoming bytes from COM3 redirected to COM2, enter the following commands from a connection different than COM2 and COM3:

```
COM1>  sdio, COM2, DC1, DC2 <CR>
$R: sdio, COM2, DC1, DC2
  DataInOut, COM2, DC1, DC2, (on)
COM1>  sdio, COM3, DC2, DC1 <CR>
$R: sdio, COM3, DC2, DC1
  DataInOut, COM3, DC2, DC1, (on)
COM1>
```

sdds gdds	**setDynamicDNS** **getDynamicDNS**	*Provider*	*UserName (40)*	*Password (40)*	*Hostname (40)*	*Bind*				
		off dyndns.org no-ip.com				auto Ethernet WiFi				

RxControl: Communication > Network Settings > DynDNS

This command configures the built-in dynamic DNS client (DynDNS or DDNS).

Before using DynDNS, you will need to create an account and define a hostname for your receiver at one of the supported DynDNS providers. The list of supported providers is shown below.

Provider	**Description**
`off`	DynDNS disabled
`dyndns.org`	dyndns.org
`no-ip.com`	no-ip.com

The *Provider* argument specifies your DynDNS provider, the *UserName* and *Password* arguments specify the account credentials at this provider, and the *Hostname* is the full hostname associated to your receiver. Setting the *Provider* to `off` disabled the DynDNS functionality.

On receivers with multiple active network interfaces, the receiver registers the public IP address of the interface with the highest priority (Ethernet first, then WiFi), unless a specific interface is forced with the *Bind* argument.

The receiver checks every 2 minutes if the public IP address has changed, and updates the DynDNS server if needed. In addition a forced DynDNS server update is performed every 30 days. This is done to prevent the expiration of a DynDNS entry.

The DynDNS settings configured by this command are applied immediately and are kept upon a power cycle and even after a reset to factory default (see command **`exeResetReceiver`**).

Note that this command is not shown in the output of the **`lstConfigFile`** command.

Example

```
COM1>  sdds, dyndns.org, Bart, MyPwd, rx1.dyndns-free.com, auto<CR>
$R: sdds, dyndns.org, Bart, MyPwd, rx1.dyndns-free.com, auto
  DynamicDNS, dyndns.org, Bart, MyPwd, rx1.dyndns-free.com, auto
COM1>
```

eecm gecm	**exeEchoMessage** **getEchoMessage**	***Cd***	***Message (242)***	***EndOfLine***						
		COM1	A:Unknown	none						
		COM2		+CR						
		COM3		+LF						
		COM4		all						
		USB1								
		USB2								
		IP10 ... IP17								
		IPS1								
		IPS2								
		IPS3								
		IPS4								
		IPS5								
		IPR1								
		IPR2								
		IPR3								
		DC1								
		DC2								

RxControl: Communication > Output Settings > Echo Message

Use this command to send a message to one of the connections of the receiver.

The *Message* argument defines the message that should be sent on the *Cd* port. If the given message starts with "`A:`", the remainder of the message is considered an ASCII string that will be forwarded without changes to the requested connection. If the given message starts with "`H:`", the remainder of the message is considered a hexadecimal representation of a succession of bytes to be sent to the requested connection. In this case, the string should be a succession of 2-character hexadecimal values separated by a single whitespace.

Make sure to enclose the string between double quotes if it contains whitespaces. The maximum length of the *Message* argument (including the `A:` or `H:` prefix) is 242 characters.

The *EndOfLine* argument defines which end-of-line character should be sent after the message. That argument is ignored when the *Message* argument starts with `H:`.

To send a message at a regular interval instead of once, use the command **`setPeriodicEcho`**.

When the *Cd* argument is `DC1` or `DC2`, the message is injected into one of the internal daisy-chain pipes. See the **`setDataInOut`** command for details.

Examples

To send the string "Hello world!" to COM2, use:

```
COM1> eecm, COM2, "A:Hello world!", none <CR>
$R: eecm, COM2, "A:Hello world!", none
  EchoMessage, COM2, "A:Hello world!", none
COM1>
```

To send the same string, the following command can also be used:

```
COM1> eecm, COM2, "H:48 65 6C 6C 6F 20 77 6F 72 6C 64 21", none
    <CR>
$R: eecm, COM2, "H:48 65 6C 6C 6F 20 77 6F 72 6C 64 21", none
  EchoMessage, COM2, "H:48 65 6C 6C 6F 20 77 6F 72 6C 64 21", none
COM1>
```

shs ghs	**setHttpsSettings** **getHttpsSettings**	*Protocol*								
		+HTTP +HTTPS all								

RxControl: Communication > Network Settings > Security

This command can be used to enable or disable HTTP and/or HTTPS access to the receiver. By default, both HTTP and HTTPS are enabled.

Secure HTTP access requires the user to provide a certificate to the receiver. This is done by navigating to the "Communication > Web Server/TLS" page of the web interface, and uploading a `.pem` file containing the certificate. By default, if no user-provided certificate is available, the receiver will use a self-signed certificate instead. The user-provided certificate can be erased with **`exeResetReceiver, hard, HTTPSCertificate`** command, reverting to the self-signed certificate.

Note that the HTTPS certificate is also applicable to the built-in NTRIP caster in TLS mode. See also the **`setNtripCasterSettings`** command.

Example

```
COM1>  shs, HTTP<CR>
$R: shs, HTTP
  HttpsSettings, HTTP
COM1>
```

sipf gipf	**setIPFiltering** **getIPFiltering**	*Mode*	*AddrList (200)*								
		off on									

RxControl: Communication > Network Settings > Security

Use this command to configure the IP filtering functionality. When IP filtering is enabled, only the specified IP addresses are allowed to connect to the receiver.

By default, IP filtering is off (the *Mode* argument is `off`) and the receiver accepts connections from any IP address.

When enabling IP filtering (*Mode* set to `on`), the *AddrList* argument must contain a whitespace-separated list of IP addresses (IPv4) allowed to connect to the receiver. Only IP addresses are allowed here, not hostnames. To enable a whole range of IP addresses, a netmask can also be specified using the so-called "slash notation", where the IP address is followed by a forward slash (/) and the subnet mask number from 0 to 32.

After entering the command, existing IP connections are kept active, but any new connection from a non-allowed IP address will be rejected.

Note that IP filtering is not applicable to the WiFi interface when the receiver is configured in access point mode.

Example

```
COM1>  sipf, on, 192.168.0.7 192.168.2.0/24<CR>
$R: sipf, on, 192.168.0.7 192.168.2.0/24
  IPFiltering, on, 192.168.0.7 192.168.2.0/24
COM1>
```

sipk gipk	**setIPKeepAlive** **getIPKeepAlive**	*Enable*	*IdleTime*	*Interval*	*MaxCount*					
		off on	15 . . . 18000 s	1 . . . 3600 s	1 . . . 15 . . . 3600					

RxControl: Communication > Network Settings > General

Use these commands to configure the TCP/IP KeepAlive mechanism.

When enabled, the receiver sends periodic KeepAlive messages over idle IP and IPS connections to solicit a response from the other end. The *IdleTime* argument sets the idle time after which the first KeepAlive is sent. If the connection stays idle (no response received), other KeepAlive messages are sent every *Interval* until *MaxCount* KeepAlives have been sent, after which the connection is closed.

KeepAlive messages are only sent when the connection is idle and no data is being transfered.

⚠ Keep in mind that when the *Interval* is set to a low number on a high latency connection, this might cause unnecessary traffic on the connection as well as premature disconnection.

Example

```
COM1>  sipk, on, 20, 20, 20<CR>
$R: sipk, on, 20, 20, 20
  IPKeepAlive, on, 20, 20, 20
COM1>
```

sipp gipp	**setIPPortSettings** **getIPPortSettings**	*Command*	*FTPControl*							
		1 . . . 28784 . . . 65535	1 . . . 21 . . . 65535							

RxControl: Communication > Network Settings > General

Use these commands to define/inquire the port numbers where the receiver listens for incoming TCP/IP connections.

The *Command* argument defines the port where the receiver listens for user commands.

The *FTPControl* argument defines the FTP control port number.

The IP port numbers configured by this command keep their value upon a power cycle and even after a reset to factory default (see command **`exeResetReceiver`**).

Note that this command is not shown in the output of the **`lstConfigFile`** command.

When selecting a port number, make sure to avoid conflicts with other services (for example select a different port than in the **`setIPServerSettings`** and the **`setIPReceiveSettings`** commands).

Example

```
COM1>  sipp, 12345, 21<CR>
$R: sipp, 12345, 21
  IPPortSettings, 12345, 21
COM1>
```

sirs girs	**setIPReceiveSettings** **getIPReceiveSettings**	***Cd*** *Cd*	*Port*	*Mode*	*TCPAddress (40)*					
		+ IPR1 + IPR2 + IPR3 all	0 . . . 65535	TCP2Way UDP	0.0.0.0					

RxControl: Communication > Network Settings > General

This command configures the "IP receive" ports (IPR).

When *Mode* is set to `TCP2Way`, the receiver connects to the specified port of a server of which the IP address or hostname is provided in the *TCPAddress* argument. It then receives all data sent by this server on that port. The TCP2Way connection is bidirectional, and it is possible to send data to the server or to process commands from the server.

When *Mode* is set to `UDP`, the receiver listens for incoming UDP messages on its port identified by the *Port* argument. In `UDP` mode, the *TCPAddress* argument is ignored. Note that, contrary to the TCP connection, the UDP connection is unidirectional.

If *Port* is set to 0, the corresponding IPR connection is disabled.

This command is the counterpart of the **`setIPServerSettings`** command. **`setIPServerSettings`** configures the sender side of the communication, while **`setIPReceiveSettings`** configures the receiver side.

⚠ When selecting a port number, make sure to avoid conflicts with other services (for example select a different port than in the **`setIPPortSettings`** command).

Example

```
COM1>  sirs, IPR1, 28785, TCP2Way, 192.168.10.5<CR>
$R: sirs, IPR1, 28785, TCP2Way, 192.168.10.5
  IPReceiveSettings, IPR1, 28785, TCP2Way, 192.168.10.5
COM1>
```

siss giss	**setIPServerSettings** **getIPServerSettings**	***Cd*** *Cd*	*Port*	*Mode*	*UDPAddress (200)*					
		+ IPS1 + IPS2 + IPS3 + IPS4 + IPS5 all	0 . . . 65535	TCP UDP TCP2Way	255.255.255.255					

RxControl: Communication > Network Settings > General

By default (*Mode* set to `TCP`), this command defines the TCP/IP port where the receiver's IP Servers (IPS) listen for incoming TCP/IP connections. When a client connects to an IPS port, all output data specified for that port are streamed to the client.

In `TCP` mode, the IPS port is unidirectional: it only sends data and incoming bytes are discarded. The `TCP2Way` mode is the same as the `TCP` mode, except that the receiver will also process input data (such as user commands or differential corrections). An IPS port configured in `TCP2Way` mode can only accept a single client at a time.

When *Mode* is set to `UDP` and *UDPAddress* is set to `255.255.255.255`, the IPS works in UDP broadcast mode. In that mode, the IPS data stream is delivered to any host on the local network listening to the IP port specified by the *Port* argument.

When *Mode* is set to `UDP` and *UDPAddress* contains a whitespace-separated list of IP addresses or hostnames, the IPS data stream is only delivered to the specified hosts. Remember to enclose the *UDPAddress* argument between double quotes when it contains whitespaces.

Use the **`setDataInOut`** command and the various output setting commands (e.g. **`setNMEAOutput`**) to define the data stream to be output by the IPS connections. Note that the UDP implementation is meant to be used with small data volumes and low update rates. It is the user's responsibility to only enable short messages at low rate when using UDP, in order to prevent throughput degradation of the network.

It is possible to configure some IPS connections in UDP mode, and others in TCP mode. The *UPDAddress* argument is ignored in TCP mode.

⚠ When selecting a port number, make sure to avoid conflicts with other services (for example select a different port than in the **`setIPPortSettings`** command).

All IPS connections must use different ports. Set the *Port* argument to 0 to disable an IPS connection.

Example

```
COM1> siss, IPS1, 28785, UDP, 255.255.255.255<CR>
$R: siss, IPS1, 28785, UDP, 255.255.255.255
  IPServerSettings, IPS1, 28785, UDP, 255.255.255.255
COM1>
```

sips gips	**setIPSettings** **getIPSettings**	*Mode*	*IP (16)*	*Netmask (16)*	*Gateway (16)*	*Domain (63)*	*DNS1 (16)*	*DNS2 (16)*	*MTU*	
		DHCP Static	0.0.0.0	255.255.255.0	0.0.0.0		0.0.0.0	0.0.0.0	0 . . . 1500	

RxControl: Communication > Network Settings > General

Use these commands to define the IP (Internet Protocol) settings of the receiver's Ethernet port. By default, the receiver is configured to use DHCP.

In `Static` mode, the receiver will not attempt to request an address via DHCP. It will use the specified IP address, netmask, gateway, domain name and DNS. *DNS1* is the primary DNS, and *DNS2* is the backup DNS. The arguments *IP*, *Netmask*, *Gateway*, *Domain*, *DNS1*, and *DNS2* are ignored in `DHCP` mode.

In `Static` mode, the value of *MTU* is used as the MTU of the Ethernet port. When *MTU* is set to 0, the receiver will use the default MTU. In `DHCP` mode, setting *MTU* to 0 will result in using the MTU supplied by the DHCP server. When set to any other value, *MTU* will override the DHCP Server once the link is established.

The IP settings configured by this command keep their value upon a power cycle and even after a reset to factory default (see command **`exeResetReceiver`**).

Note that this command is not shown in the output of the **`lstConfigFile`** command.

The command **`getIPSettings`** cannot be used to get the current IP address assigned to the receiver by the DHCP server. The current IP address can be retrieved from the command **`lstInternalFile, IPParameters`**, or from the `IPStatus` SBF block.

Example

```
COM1>  sips, Static, 192.168.1.123, 255.255.252.0, 192.168.1.255,
   mydomain.local, 192.168.100.3, 192.168.100.4, 1500<CR>
$R: sips, Static, 192.168.1.123, 255.255.252.0, 192.168.1.255,
   mydomain.local, 192.168.100.3, 192.168.100.4, 1500
  IPSettings, Static, 192.168.1.123, 255.255.252.0, 192.168.1.255,
     mydomain.local, 192.168.100.3, 192.168.100.4, 1500
COM1>
```

spe	**setPeriodicEcho**	***Cd***	*Message (201)*	*Interval*						
gpe	**getPeriodicEcho**	*Cd*								
		+COM1	A:Unknown	off						
		+COM2		once						
		+COM3		msec100						
		+COM4		msec200						
		all		msec500						
				sec1						
				sec2						
				sec5						
				sec10						
				sec15						
				sec30						
				sec60						
				min2						
				min5						
				min10						
				min15						
				min30						
				min60						

RxControl: Communication > Output Settings > Periodic Echo message

Use this command to periodically send a message to one of the connections of the receiver.

The *Message* argument defines the message that should be sent on the *Cd* port. If the given message starts with "`A:`", the remainder of the message is considered an ASCII string that will be forwarded to the requested connection. All occurrences of the `%%CR` character sequence are replaced by a single carriage return character (ASCII code 13d) and all occurrences of the `%%LF` character sequence are replaced by a single line feed character (ASCII code 10d). If the *Message* argument starts with "`H:`", the remainder of the message is considered a hexadecimal representation of a succession of bytes to be sent to the requested connection. In this case, the string should be a succession of 2-character hexadecimal values separated by a single whitespace.

Make sure to enclose the string between double quotes if it contains whitespaces. The maximum length of the *Message* argument (including the `A:` or `H:` prefix) is 201 characters.

The *Interval* argument defines the interval at which the message should be sent.

To send a message only once, set *Interval* to `once`. The only difference with the command **`exeEchoMessage`** is that **`exeEchoMessage`** cannot be stored in the boot configuration file, while **`setPeriodicEcho`** can. This can be used to output a message once at each reset or reboot. The third example below shows how to do this.

Examples

To send the string "Hello!<CR><LF>" to COM2 every minute, use:

```
COM1> spe, COM2, "A:Hello!%%CR%%LF", sec60 <CR>
$R: spe, COM2, "A:Hello!%%CR%%LF", sec60
  PeriodicEcho, COM2, "A:Hello!%%CR%%LF", sec60
COM1>
```

The same can be achieved with the following command:

```
COM1> spe, COM2, "H:48 65 6C 6C 6F 21 0D 0A", sec60 <CR>
$R: spe, COM2, "H:48 65 6C 6C 6F 21 0D 0A", sec60
```

```
  PeriodicEcho, COM2, "H:48 65 6C 6C 6F 21 0D 0A", sec60
COM1>
```

To let the receiver output the string "Hello!<CR><LF>" to COM2 at each reset, use the following command sequence:

```
COM1>  spe, COM2, "A:Hello!%%CR%%LF", once <CR>
$R: spe, COM2, "A:Hello!%%CR%%LF", once
  PeriodicEcho, COM2, "A:Hello!%%CR%%LF", once
COM1>  eccf, Current, Boot <CR>
$R: eccf, Current, Boot
  CopyConfigFile, Current, Boot
COM1>
```

sp2p gp2p	**setPointToPoint** **getPointToPoint**	***SessionID*** *SessionID*	*Mode*	*Cd*	*ClientIP (20)*	*ServerIP (20)*	*Auth*	*Password (40)*	*ConnectTimeout*	*ActivityTimeout*
		+ P2PP1 all	Off Server	COM1 COM2 COM3 COM4	192.168.50.2	192.168.50.1	None PAP CHAP		60 . . . 300 s	10 . . . 600 . . . 32000 s

RxControl: Communication > Network Settings > P2P Protocol

The receiver features a Point-to-Point Protocol (P2PP) server, which emulates an IP link over a serial port. To avoid confusion with Precise Point Positioning, this feature is referred to as P2PP in Septentrio receivers. This command configures the P2PP server.

In the current version, the receiver implements a single P2PP server, and the first argument (*ServerID*) can only take the value `P2PP1`.

The *Mode* argument enables the P2PP server (it is disabled by default).

Cd sets the COM port to be used for the point-to-point communication. The baud rate and hardware flow control must be configured with the **`setCOMSettings`** command.

ClientIP sets the IP address that will be given to the client (i.e. your local computer) when a connection is established.

ServerIP sets the IP address that will be given to the server (i.e. the receiver) when a connection is established.

Auth determines whether the client needs to authenticate itself when establishing the connection. `PAP` will use Password Authentication Protocol and `CHAP` will use Challenge Handshake Authentication Protocol. When authentication is enabled, the *Password* argument sets the password that will need to be supplied.

ConnectTimeout sets the maximum time, in seconds, that a connection attempt may consume before being refused.

ActivityTimeout sets the maximum time, in seconds, that a connection may be idle (no data transfer) before it is disconnected.

When a timeout occurs, the receiver will shut down the P2PP server and restart it. When a server is enabled, and the configuration is correct, the receiver will start the P2PP server within a maximum of 30 seconds.

Example

```
COM1>  sp2p, P2PP1, Off, COM1, 255.255.255.255, 255.255.255.255,
   CHAP, P@ssw0rd1, 60, 600<CR>
$R: sp2p, P2PP1, Off, COM1, 255.255.255.255, 255.255.255.255, CHAP,
    P@ssw0rd1, 60, 600
  PointToPoint, P2PP1, Off, COM1, 255.255.255.255, 255.255.255.255,
      CHAP, P@ssw0rd1, 60, 600
COM1>
```

spfw gpfw	**setPortFirewall** **getPortFirewall**	***Interface*** *Interface*	*OpenPorts*	*PortList (100)*						
		+ Ethernet + WiFi all	none default all PortList							

RxControl: Communication > Network Settings > Security

Use this command to configure the receiver firewall, i.e. to specify the list of IP ports which are open to receive data.

The list of open ports can be specified independently for all network interfaces. The default (*OpenPorts* is set to `default`) depends on the interface, as follows:

Interface	Description
`Ethernet`	By default, all ports open.
`WiFi`	By default, all ports open. Note that the WiFi settings only applies when the receiver is configured as WiFi client (see 1.1.3.2.2). The WiFi firewall is disabled when the receiver is the access point.

It is possible to close all ports (*OpenPorts* is `none`), to open all ports (*OpenPorts* is `all`), or to manually specify a list of ports to open (*OpenPorts* is `PortList`). In the latter case, the list of port numbers needs to be specified in the *PortList* argument. The different port numbers must be separated by whitespaces. The *PortList* argument is ignored if *OpenPorts* is not set to `PortList`.

Example

```
COM1>  spfw, Ethernet, PortList, "21 80 28784"<CR>
$R: spfw, Ethernet, PortList, "21 80 28784"
  PortFirewall, Ethernet, PortList, "21 80 28784"
COM1>
```

3.2.13 NTRIP Settings

snmp	**setNtripCasterMountPoints**	***MountPointID***	*Enable*	*MPName (32)*	*ExtServer*	*UserName (20)*	*Password (40)*	*ClientAuth*		
gnmp	**getNtripCasterMountPoints**	*MountPointID*								
		+ MP1	off		No			none		
		+ MP2	on		Yes			basic		
		+ MP3								
		all								

RxControl: Communication > NTRIP Settings > NTRIP Caster settings

This command defines the general characteristics of the mount points available on the built-in NTRIP caster. The caster supports up to three mount points.

The *Enable* argument disables or disables a mount point. The *MPName* argument is the mount point name, as it will appear in the stream record of the caster source table. Make sure to give each enabled mount point a different name.

The *ExtServer* argument defines if the mount point is allowed to receive a stream from a remote NTRIP server (argument set to `Yes`), or if only local streams are allowed, i.e. streams originating from the receiver's own NTRIP server. The *UserName* and *Password* arguments are the credentials needed for the remote server to feed data. These arguments are ignored if *ExtServer* is set to `No`.

The *ClientAuth* argument defines the mount point client access protection. If set to `none`, all clients will be able to connect without providing credentials.

Note that the caster is reset each time a setting is changed with this command.

Example

```
COM1>  snmp, MP1, on, MyMP, Yes, MyUser, MyPwd, basic<CR>
$R: snmp, MP1, on, MyMP, Yes, MyUser, MyPwd, basic
  NtripCasterMountPoints, MP1, on, MyMP, Yes, MyUser, MyPwd, basic
COM1>
```

smpf gmpf	**setNtripCasterMPFormat** **getNtripCasterMPFormat**	***MountPointID*** *MountPointID*	*Format*	*ManualFt (30)*	*FtDetails (100)*					
		+ MP1 + MP2 + MP3 all	RTCMv2 RTCMv3 CMR NMEA BINEX RAW manual							

RxControl: Communication > NTRIP Settings > NTRIP Caster settings

Use this command to define the format of the streams available on the caster mount points.

The *Format* argument defines the stream format, as will be reported in the `<format>` field of the sourcetable STR records. It is possible to select one of the predefined formats, or to enter a user-defined format. The latter is done by setting the *Format* argument to `manual` and by providing the format string with the *ManualFt* argument. The *ManualFt* argument is ignored when *Format* is not set to `manual`.

The *FtDetails* argument sets the contents of the `<format-details>` field of the sourcetable STR records.

When you need a comma in the *ManualFt* or *FtDetails* argument, use the "`%%CM`" escape sequence. Do not forget to enclose the string between double quotes if it contains whitespaces.

Note that the caster is reset each time a stream format is changed with this command.

⚠ When feeding a stream to the caster, make sure that the format of the stream corresponds to the settings in this command.

Example

```
COM1>  smpf, MP1, manual, RAW%%CMNMEA, "SBF (1s)%%CM NMEA (5s)"<CR>
$R: smpf, MP1, manual, RAW%%CMNMEA, "SBF (1s)%%CM NMEA (5s)"
  NtripCasterMPFormat, MP1, manual, RAW%%CMNMEA, "SBF (1s)%%CM NMEA
       (5s)"
COM1>
```

sncs gncs	**setNtripCasterSettings** **getNtripCasterSettings**	*Mode*	*Port*	*Identifier (100)*	*TlsPort*				
		off on	0 . . . 2101 . . . 65535	default	0 . . . 2102 . . . 65535				

RxControl: Communication > NTRIP Settings > NTRIP Caster settings

Use this command to enable and configure the built-in NTRIP caster.

The *Port* argument specifies on which IP port the caster can be accessed in "unsecure" mode, and the *TlsPort* specifies on which TLS port the caster can be accessed. The TLS certificate is the same as the https certificate. See the **`setHttpsSettings`** command for details.

Note that, if *Port* and *TlsPort* are equal, TLS is disabled.

The *Identifier* argument is a free text that can be used to describe the caster. If *Identifier* is set to the string "default", it is replaced by the receiver name and serial number. This text will appear in the "Identifier" field of the caster record in the NTRIP source table.

Example

```
COM1>  sncs, on, 2101, default, 2102<CR>
$R: sncs, on, 2101, default, 2102
  NtripCasterSettings, on, 2101, default, 2102
COM1>
```

sncu gncu	**setNtripCasterUsers** **getNtripCasterUsers**	***UserID*** *UserID*	*UserName (20)*	*Password (40)*	*MountPoints*	*MaxClients*				
		+ User1 + User2 + User3 + User4 + User5 all			none + MP1 + MP2 + MP3 all	1 . . . 10				

RxControl: Communication > NTRIP Settings > NTRIP Caster settings

This command defines the user accounts (user name and password) that clients can use to connect to the built-in NTRIP caster. The password must contain at least one character (empty passwords are not supported). Up to five user accounts can be defined.

The *MounPoints* argument defines the list of mount points allowed for a given user account.

The caster can accept up to 10 concurrent client connections in total. The *MaxClients* argument can be used to limit the number of clients that are allowed to concurrently connect using a particular account.

To delete a user account, enter this command with an empty *UserName* argument.

Note that the caster is reset each time a user account is added, deleted or modified with this command.

Example

```
COM1>  sncu, User1, MyUser, MyPwd, all, 1<CR>
$R: sncu, User1, MyUser, MyPwd, all, 1
  NtripCasterUsers, User1, MyUser, MyPwd, all, 1
COM1>
```

snts gnts	**setNtripSettings** **getNtripSettings**	***Cd*** *Cd*	*Mode*	*Caster (40)*	*Port*	*UserName (20)*	*Password (40)*	*MountPoint (32)*	*Version*	*SendGGA*
		+NTR1 +NTR2 +NTR3 all	off Server Client		0 . . . 2101 . . . 65535				v1 v2	auto off sec1 sec5 sec10 sec60

RxControl: Communication > NTRIP Settings > NTRIP Server/Client settings

Use this command to specify the parameters of the NTRIP connection referenced by the *Cd* argument.

The *Mode* argument specifies the type of NTRIP connection. In `Server` mode, the receiver is sending data to an NTRIP caster. In `Client` mode, the receiver gets data from the NTRIP caster. Set *Mode* to `off` to disable the connection.

Caster is the hostname or IP address of the NTRIP caster to connect to. To send data to the built-in NTRIP caster, use "localhost" for the *Caster* argument. *Port*, *UserName*, *Password* and *MountPoint* are the IP port number, the user name, the password and the mount point to be used when connecting to the NTRIP caster. The default NTRIP port number is 2101. Note that the receiver encrypts the password so that it cannot be read back with the command **`getNtripSettings`**.

The *Version* argument specifies which version of the NTRIP protocol to use (v1 or v2).

The *SendGGA* argument specifies whether or not to send NMEA GGA messages to the NTRIP caster, and at which rate. In `auto` mode (the default), the receiver automatically sends GGA messages if requested by the caster. This argument is ignored in NTRIP server mode.

Example

```
COM1>  snts, NTR1, Client, ntrip.com, 2101, USER, PWD, MP1, v2,
   auto<CR>
$R: snts, NTR1, Client, ntrip.com, 2101, USER, PWD, MP1, v2, auto
  NtripSettings, NTR1, Client, ntrip.com, 2101, USER, PWD, MP1, v2,
      auto
COM1>
```

lnst	**lstNTRIPSourceTable**	***Caster (40)***	*Port*							
			0 . . . 2101 . . . 65535							

Use this command to retrieve the source table from the specified NTRIP caster.

Caster is the hostname or IP address of the NTRIP caster to connect to, and *Port* is the IP port number. The default NTRIP port number is 2101.

Example

```
COM1>  lnst, ntripcaster <CR>
$R; lnst, ntripcaster
---->
$-- BLOCK 1 / 0 C
HTTP/1.1 200 OK
Ntrip-Version: Ntrip/2.0
Ntrip-Flags: st_filter,st_auth,st_match,st_strict
Server: NTRIP Caster/2.0.15
...
$-- BLOCK 1 / 0 C
ENDSOURCETABLE
COM1>
```

sntt gntt	**setNtripTlsSettings** **getNtripTlsSettings**	***Cd*** *Cd*	*Enable*	*Fingerprint (96)*						
		+NTR1 +NTR2 +NTR3 all	off on							

RxControl: Communication > NTRIP Settings > NTRIP Server/Client settings

Use this command to enable or disable TLS on the NTRIP connection referenced by the *Cd* argument.

If the caster's certificate is known by a publicly-trusted certification authority (CA), the receiver will authenticate it by following the usual CA chain of trust, starting from root certificates present on the receiver. In that case, the *Fingerprint* argument should be left empty. If the NTRIP caster uses a self-signed certificate or a certificate only known by a private CA, its SHA-256 fingerprint must be provided. The self-signed certificate of the built-in caster (localhost) is present in the trust-zone of the receiver, and so the *Fingerprint* argument can be left empty for NTRIP client or server connections to localhost.

The examples below show different formats for the *Fingerprint* argument.

Examples

```
COM1>  sntt, NTR1, on, ""<CR>
$R: sntt, NTR1, on, ""
  NtripTlsSettings, NTR1, on, ""
COM1>
```

Lower and upper case characters are allowed:

```
COM1>  sntt, NTR1, on, Aa:Bb:56:78:90:12:  ...  78:90:12:34 <CR>
$R: sntt, NTR1, on, Aa:Bb:56:78:90:12: ... 78:90:12:34
  NtripTlsSettings, NTR1, on, "AA:BB:56:78:90:12: ... 78:90:12:34"
COM1>
```

When using whitespaces as delimiter, do not forget to enclose the fingerprint in double quotes:

```
COM1>  sntt, NTR1, on, "Aa Bb 56 78 90 12 ...  78 90 12 34"<CR>
$R: sntt, NTR1, on, "Aa Bb 56 78 90 12 ... 78 90 12 34"
NtripTlsSettings, NTR1, on, "AA:BB:56:78:90:12: ... 78:90:12:34"
COM1>
```

It is also allowed to leave out the delimiters in the fingerprint:

```
COM1>  sntt, NTR1, on, AaBb56789012 ...  78901234 <CR>
$R: sntt, NTR1, on, AaBb56789012 ... 78901234
  NtripTlsSettings, NTR1, on, "AA:BB:56:78:90:12: ... 78:90:12:34"
COM1>
```

3.2.14 WiFi Settings

eawa gawa	**exeAddWiFiAccessPoint** **getAddWiFiAccessPoint**	***SSID (32)***	***Key (40)***							

RxControl: Communication > WiFi Settings > Client Add Network

Use this command to add a WiFi access point to the list of known access points, or to modify the password of a known access point. The *SSID* argument is the identifier of the access point and the *Key* argument is the password needed to connect to the access point.

This command must be entered for all access points that the receiver is to connect to. For open access points that do not require a key, set *Key* to "".

When the receiver is configured in WiFi client mode with the **`setWiFiMode`** command, it will check if a known access point is reachable and automatically connect to it.

By default, if multiple WiFi access points are reachable, the last one that was added with the **`exeAddWiFiAccessPoint`** command has the priority. Use the **`exeManageWiFiAccessPoint`** to overrule this and manually define the preferred access point.

The command permanently adds the WiFi access point to the list of known networks. The list is persistent across power cycles. The **`exeManageWiFiAccessPoint`** command can be used to remove an access point from the list (i.e. to forget an access point), and the **`exeResetReceiver, hard, WiFiAccessPoints`** command can be used to erase the complete list.

Use the command **`lstWiFiAccessPoints`** to get a list of the known and/or reachable WiFi access points.

Example

```
COM1>  eawa, MyWiFi, 12345678<CR>
$R: eawa, MyWiFi, 12345678
  AddWiFiAccessPoint, MyWiFi, "7VBC0NULTV6I"
COM1>
```

emwa gmwa	**exeManageWiFiAccessPoint** **getManageWiFiAccessPoint**	***SSID (32)***	***Action***							
			Promote Remove							

RxControl: Communication > WiFi Settings > Client Manage Network

Use this command to remove a WiFi access point from the list of known access points, or to promote it as preferred access point.

If the *Action* argument is `Promote`, the access point identified by the *SSID* argument is given the highest priority in case multiple known access points are reachable.

If the *Action* argument is `Remove`, the access point identified by the *SSID* argument is removed from the list of known access points. This will prevent the receiver from connecting to this access point until it is re-enabled with the **`exeAddWiFiAccessPoint`** command.

Example

```
COM1>  emwa, MyWiFi, Promote<CR>
$R: emwa, MyWiFi, Promote
  ManageWiFiAccessPoint, MyWiFi, Promote
COM1>
```

swfa gwfa	**setWiFiAccessPoint** **getWiFiAccessPoint**	*SSID (32)*	*EncryptionType*	*Key (40)*	*Channel*	*SSIDActual (32)*				
		default	none WPA2	password	1 . . . 6 . . . 11	model-sn				

RxControl: Communication > WiFi Settings > Access Point

Use this command to configure the WiFi access point.

The SSID is the name of your receiver in the WiFi network. By default, the SSID is the receiver model name followed by its serial number. Use the *SSID* argument to assign your receiver a different WiFi access point name. Using the reserved keyword "default" reverts to the default SSID.

By default, WiFi encryption is turned off. The encryption type and password can be specified with the second and third arguments.

The *Channel* argument sets the WiFi frequency channel to be used. It is not necessary to change this value unless you notice interference problems with another nearby WiFi device.

The last argument (*SSIDActual*) is read-only. It shows the actual SSID. It is a copy of the *SSID* argument except when the *SSID* argument is set to "default".

Example

```
COM1>  swfa, , WPA2, password<CR>
$R: swfa, , WPA2, password
  WiFiAccessPoint, default, WPA2, "7VBC0NULTV6I", 6, off, "RxName
     -123456"
COM1>
```

lwa	**lstWiFiAccessPoints**	***Type***								
		+ Known + Reachable all								

Use this command to list the known and/or the reachable WiFi access points.

The following information is provided for each access points (AP): the SSID, the signal level in dBm (only if AP is in reach), the security type, the current status (`Connected`, `Known` or `Unknown`), and, for known APs, the access point priority (P1 for highest priority). A "known" AP is an AP that has been defined with the **`exeAddWiFiAccessPoint`** command.

The *Type* argument defines the contents of the list:

Type	Description
`Known`	List of known access points.
`Reachable`	List of the access points that are currently in reach of the receiver.

Example

```
COM1>  lwa, all <CR>
$R; lwa, all
---->
$-- BLOCK 1 / 0 C
"TestAP",-62,Unsecured,Unknown,None
"MyAP",-75,Unsecured,Connected,P2
"guest",-78,WPA,Unknown,None
COM1>
```

swfm gwfm	**setWiFiMode** **getWiFiMode**	*Enable*	*Mode*							
		off on	AccessPoint Client							

RxControl: Communication > WiFi Settings > General

Use this command to turn WiFi on and off, and to specify in which WiFi mode the receiver should operate (client or access point).

Example

```
COM1>  swfm, off<CR>
$R: swfm, off
  WiFiMode, off, AccessPoint
COM1>
```

3.2.15 NMEA Configuration

enoc gnoc	**exeNMEAOnce** **getNMEAOnce**	***Cd***	***Messages***							
		COM1 COM2 COM3 COM4 USB1 USB2 IP10 ... IP17 NTR1 NTR2 NTR3 IPS1 IPS2 IPS3 IPS4 IPS5 IPR1 IPR2 IPR3 LOG1 ... LOG8	+ALM +DTM +GBS +GGA +GLL +GNS +GRS +GSA +GST +GSV +HDT +RMC +ROT +VTG +ZDA +LLQ +RBP +RBV +RBD +AVR +GGK +GFA +GGQ +LLK +GMP +TFM +SNC +SRX +SDI +SPW							

RxControl: Communication > Output Settings > NMEA Output Once

Use this command to output a set of NMEA messages on a given connection. This command differs from the related **`setNMEAOutput`** command in that it instructs the receiver to output the specified messages only once, instead of at regular intervals.

The *Cd* argument defines the connection descriptor (see 1.1.5) on which the message(s) should be output and the *Messages* argument defines the list of messages that should be output. Refer to appendix C for a short description of the NMEA sentences.

Please make sure that the connection specified by *Cd* is configured to allow NMEA output (this is the default for all connections). See the **`setDataInOut`** command.

Example

To output the receiver position on COM1, use:

```
COM1>  enoc, COM1, GGA <CR>
$R: enoc, COM1, GGA
  NMEAOnce, COM1, GGA
COM1>
```

sno	**setNMEAOutput**	***Stream***	*Cd*	*Messages*	*Interval*				
gno	**getNMEAOutput**	*Stream*							
		+Stream1 ... Stream10 all	none COM1 COM2 COM3 COM4 USB1 USB2 IP10 ... IP17 NTR1 NTR2 NTR3 IPS1 IPS2 IPS3 IPS4 IPS5 IPR1 IPR2 IPR3 LOG1 ... LOG8	none +ALM +DTM +GBS +GGA +GLL +GNS +GRS +GSA +GST +GSV +HDT +RMC +ROT +VTG +ZDA +LLQ +RBP +RBV +RBD +PUMRD +AVR +GGK +GFA +GGQ +LLK +GMP +TXTbase +TFM +SNC +SRX +SDI +SPW	off OnChange msec10 msec20 msec40 msec50 msec100 msec200 msec500 sec1 sec2 sec5 sec10 sec15 sec30 sec60 min2 min5 min10 min15 min30 min60				

RxControl: Communication > Output Settings > NMEA Output > NMEA Output Intervals

Use this command to output a set of NMEA messages on a given connection at a regular interval. The *Cd* argument defines the connection descriptor (see 1.1.5) on which the message(s) should be output and the *Messages* argument defines the list of messages that should be output. Refer to appendix C for a short description of the NMEA sentences.

This command is the counterpart of the **`setSBFOutput`** command for NMEA sentences. Please refer to the description of that command for a description of the arguments.

Examples

To output GGA at 1Hz and RMC at 10Hz on COM1, use:

```
COM1>  sno, Stream1, COM1, GGA, sec1 <CR>
$R: sno, Stream1, COM1, GGA, sec1
  NMEAOutput, Stream1, COM1, GGA, sec1
COM1>  sno, Stream2, COM1, RMC, msec100 <CR>
$R: sno, Stream2, COM1, RMC, msec100
  NMEAOutput, Stream2, COM1, RMC, msec100
COM1>
```

To get the list of NMEA messages currently output, use:

```
COM1>  gno <CR>
$R: gno
```

```
  NMEAOutput, Stream1, COM1, GGA, sec1
  NMEAOutput, Stream2, COM1, RMC, msec100
  NMEAOutput, Stream3, none, none, off
  NMEAOutput, Stream4, none, none, off
  NMEAOutput, Stream5, none, none, off
  NMEAOutput, Stream6, none, none, off
  NMEAOutput, Stream7, none, none, off
  NMEAOutput, Stream8, none, none, off
  NMEAOutput, Stream9, none, none, off
  NMEAOutput, Stream10, none, none, off
COM1>
```

snp gnp	**setNMEAPrecision** **getNMEAPrecision**	*NrExtraDigits*	*Compatibility*	*LocalDatum*	*MinStdDev*					
		0 . . . 3	Nominal Mode1 Mode2	off only	0.000 . . . 0.001 . . . 1.000 m					

RxControl: Communication > Output Settings > NMEA Output > Customize

Use these commands to define/inquire the number of extra digits in the latitude, longitude and altitude reported in NMEA sentences and to tune certain sentences to be compatible with third-party applications that are not fully compliant with the NMEA 0183 standard.

When *NrExtraDigits* is 0, the latitude and longitude are reported in degrees with 5 decimal digit, and altitude is reported in meters with 2 decimal digit. These default numbers of digits lead to a centimeter-level resolution of the position. To represent RTK positions with their full precision (millimeter-level), it is recommended to set *NrExtraDigits* to 2.

Note that increasing the number of digits (setting *NrExtraDigits* to a non-zero value) may cause the NMEA standard to be broken, as the total number of characters in a sentence may end up exceeding the prescribed limit of 82.

When setting the argument *Compatibility* to `Mode1`, the GPS Quality Indicator in GGA sentences is set to the value "2: Differential GPS" for all non-standalone positioning modes, the Mode Indicator in GNS sentences is set to "D: Differential" for all non-standalone positioning modes, and the Course Over Ground in the VTG sentences is not a null field for stationary receivers.

When setting the argument *Compatibility* to `Mode2`, the Course Over Ground in the VTG sentences is not a null field for stationary receivers.

The *LocalDatum* argument specifies whether transformation parameters sent out by the RTK service provider should be applied or not in NMEA sentences GGA and GNS. If *LocalDatum* is `off`, the transformation parameters are not applied, and the coordinates in GGA and GNS correspond to the coordinates in the `PVTGeodetic` SBF block. If *LocalDatum* is `only` and the relevant transformation parameters have been received, the coordinates are transformed to the local datum and correspond to the `PosLocal` SBF block. If the transformation parameters are not available, the coordinates are untransformed (in particular, the datum setting of the **`setGeodeticDatum`** command has no effect). The TFM proprietary NMEA sentence can be used to determine which transformation has been applied.

The *MinStdDev* argument defines the minimum standard deviation values that can be encoded in the GST sentence. If an actual standard deviation is below the value provided in *MinStdDev*, the value in *MinStdDev* is encoded instead.

Example

```
COM1>  snp, 2, Mode2, off, 0.05<CR>
$R: snp, 2, Mode2, off, 0.05
  NMEAPrecision, 2, Mode2, off, 0.050
COM1>
```

snti gnti	**setNMEATalkerID** **getNMEATalkerID**	*TalkerID*								
		auto GP GN								

RxControl: Communication > Output Settings > NMEA Output > Customize

Use these commands to define/inquire the "Device Talker" for NMEA sentences. The device talker allows users to identify the type of equipment from which the NMEA sentence was issued.

When the *TalkerID* argument is set to `auto`, the talker will depend on the type of solution that is output. For a GNSS solution , "GN" is used if satellites from multiple constellations are used, "GP" for a GPS only solution, "GA" for a Galileo only solution, and "BD" for a BeiDou only solution.

Note that the command is ignored for the NMEA sentences where it would conflict with the standard. For example, the GSV sentence reporting the GPS visibility will always have its device talker set to "GP" regardless of the **`setNMEATalkerID`** command.

Example

```
COM1>  snti, GP <CR>
$R: snti, GP
  NMEATalkerID, GP
COM1>
```

snv gnv	**setNMEAVersion** **getNMEAVersion**	*Version*								
		v3x v4x								

RxControl: Communication > Output Settings > NMEA Output > Customize

Use this command to set the NMEA version the receiver should comply with.

If `v3x` is selected, the NMEA sentences are formatted according to the 3.01 version of the standard. If `v4x` is selected, system ID, signal ID and navigational status fields are added in some sentences according to version 4.11 of the NMEA standard.

Example

```
COM1>  snv, v4x<CR>
$R: snv, v4x
  NMEAVersion, v4x
COM1>
```

3.2.16 SBF Configuration

smrf gmrf	**setMeas3MaxRefInterval** **getMeas3MaxRefInterval**	*MaxIntrvl*								
		OnlyRef msec500 sec1 sec5 sec10 sec30 sec60								

RxControl: Communication > Output Settings > SBF Output > Customize

Use this command to define at which interval to encode reference epochs (i.e. full measurements) into the `Meas3Ranges` SBF block. Refer to the description of the `Meas3Ranges` block for a definition of the reference epochs.

When generating the `Meas3Ranges` SBF block, the receiver automatically selects an optimal interval between reference epochs. This command allows the user to define the maximum interval allowed for his application. Setting the *MaxIntrvl* argument to `OnlyRef` forces the receiver to only encode reference epochs in `Meas3Ranges`. This will allow to decode all epochs independently of any previous epoch, at the expense of a larger average SBF block size.

Example

```
COM1>  smrf, OnlyRef<CR>
$R: smrf, OnlyRef
  Meas3MaxRefInterval, OnlyRef
COM1>
```

ssgp gsgp	**setSBFGroups** **getSBFGroups**	***Group*** *Group*	*Messages*								
		+ Group1 + Group2 + Group3 + Group4 all	none [SBF List] + Measurements + Meas3 + RawNavBits + GPS + GLO + GAL + GEO + BDS + QZS + NavIC + PVTCart + PVTGeod + PVTExtra + Time + Events + DiffCorr + Status + LBand + Advanced + PostProcess + Rinex + RinexMeas3 + Support								

RxControl: Communication > Output Settings > SBF Groups

Use these commands to define/inquire user-defined groups of SBF blocks that can be re-used in the **`exeSBFOnce`** and the **`setSBFOutput`** commands. The purpose of defining groups is to ease the typing effort when the same set of SBF blocks are to be addressed regularly.

The list of supported SBF blocks [SBF List] can be found in appendix A.

A number of predefined groups of SBF blocks are available (such as `Measurements`). See the command **`setSBFOutput`** for a description of these predefined groups.

Example

To output the messages `MeasEpoch`, `PVTCartesian` and `DOP` as one group on COM1 at a rate of 1 Hz, you could use the following sequence of commands:

```
COM1>  ssgp, Group1, MeasEpoch+PVTCartesian+DOP <CR>
$R: ssgp, Group1, MeasEpoch+PVTCartesian+DOP
  SBFGroups, Group1, MeasEpoch+PVTCartesian+DOP
COM1>  sso, Stream1, COM1, Group1, sec1 <CR>
$R: sso, Stream1, COM1, Group1, sec1
  SBFOutput, Stream1, COM1, Group1, sec1
COM1>
```

esoc gsoc	**exeSBFOnce** **getSBFOnce**	***Cd***	***Messages***							
		COM1 COM2 COM3 COM4 USB1 USB2 IP10 ... IP17 NTR1 NTR2 NTR3 IPS1 IPS2 IPS3 IPS4 IPS5 IPR1 IPR2 IPR3 LOG1 ... LOG8	[SBF List] +Measurements +Meas3 +GPS +GLO +GAL +GEO +BDS +QZS +NavIC +PVTCart +PVTGeod +PVTExtra +Time +Status +LBand +UserGroups +Advanced +PostProcess +Rinex +RinexMeas3 +Support							

RxControl: Communication > Output Settings > SBF Output Once

Use this command to output a set of SBF blocks on a given connection. This command differs from the related **`setSBFOutput`** command in that it instructs the receiver to output the specified SBF blocks only once, instead of at regular intervals.

The *Cd* argument defines the connection descriptor (see 1.1.5) on which the message(s) should be output and the *Messages* argument defines the list of messages that should be output. The list of SBF blocks [SBF List] supported by the **`exeSBFOnce`** command can be found in appendix A.

Make sure that the connection specified by *Cd* is configured to allow SBF output (this is the default for all connections). See also the **`setDataInOut`** command.

Predefined groups of SBF blocks (such as `Measurements`) can be addressed in the *Messages* argument. These groups are defined in the table below.

When using this command to output a block that is also scheduled with the **`setSBFOutput`** command, the block will be sent twice. Note that this can cause duplicate measurement or PVT epochs in the SBF stream.

Messages	**Description**
`Measurements`	+MeasEpoch +MeasExtra +EndOfMeas
`Meas3`	+Meas3Ranges +Meas3CN0HiRes +Meas3Doppler +Meas3PP +Meas3MP
`GPS`	+GPSNav +GPSCNav +GPSAlm +GPSIon +GPSUtc
`GLO`	+GLONav +GLOAlm +GLOTime
`GAL`	+GALNav +GALAlm +GALIon +GALUtc +GALGstGps
`GEO`	+GEONav +GEOAlm
`BDS`	+BDSAlm +BDSNav +BDSCNav2 +BDSIon +BDSUtc

Messages (Continued)	Description
`QZS`	+QZSNav +QZSAlm
`NavIC`	+NavICLNav
`PVTCart`	+PVTCartesian +PosCovCartesian +VelCovCartesian +BaseVectorCart
`PVTGeod`	+PVTGeodetic +PosCovGeodetic +VelCovGeodetic +BaseVectorGeod +PosLocal
`PVTExtra`	+DOP +PVTSatCartesian +PVTResiduals +RAIMStatistics +GEOCorrections +PVTSupport +PVTSupportA +EndOfPVT
`Time`	+ReceiverTime
`Status`	+SatVisibility +ChannelStatus +ReceiverStatus +InputLink +OutputLink +IPStatus +NTRIPClientStatus +NTRIPServerStatus +WiFiAPStatus +WiFiClientStatus +CosmosStatus +PowerStatus +QualityInd +DiskStatus +LogStatus +RFStatus +DynDNSStatus +P2PPStatus
`LBand`	+LBandTrackerStatus +LBAS1DecoderStatus +LBAS1Messages +LBandBeams
`UserGroups`	+Group1 +Group2 +Group3 +Group4
`Advanced`	+SystemInfo
`PostProcess`	+MeasEpoch +MeasExtra +GPSNav +GPSIon +GPSUtc +GLONav +GLOTime +GALNav +GALIon +GALUtc +GALGstGps +GEONav +BDSNav +BDSIon +BDSUtc +QZSNav +ReceiverSetup +Commands
`Rinex`	+MeasEpoch +GPSNav +GPSCNav +GPSIon +GPSUtc +GLONav +GALNav +GALUtc +GALGstGps +GEONav +BDSNav +BDSCNav2 +QZSNav +NavICLNav +PVTGeodetic +ReceiverSetup +Comment
`RinexMeas3`	+Meas3Ranges +GPSNav +GPSCNav +GPSIon +GPSUtc +GLONav +GALNav +GALUtc +GALGstGps +GEONav +BDSNav +BDSCNav2 +QZSNav +NavICLNav +ReceiverSetup +Comment
`Support`	+MeasEpoch +MeasExtra +EndOfMeas +GPSNav +GPSAlm +GPSIon +GPSUtc +GLONav +GLOAlm +GLOTime +GALNav +GALAlm +GALIon +GALUtc +GALGstGps +GEONav +GEOAlm +BDSAlm +BDSNav +BDSIon +BDSUtc +QZSNav +PVTGeodetic +PosCovGeodetic +BaseVectorGeod +DOP +PVTSupport +PVTSupportA +EndOfPVT +ChannelStatus +ReceiverStatus +InputLink +OutputLink +ReceiverSetup +RxComponents +Commands +RxMessage +LBandTrackerStatus +LBAS1DecoderStatus +IPStatus +NTRIPClientStatus +NTRIPServerStatus +WiFiAPStatus +WiFiClientStatus +CosmosStatus +PowerStatus +QualityInd +LBandBeams +DiskStatus +LogStatus +RFStatus +DynDNSStatus +P2PPStatus +SystemInfo

Example

To output the next `MeasEpoch` block, use:

```
COM1>  esoc, COM1, MeasEpoch <CR>
$R: esoc, COM1, MeasEpoch
  SBFOnce, COM1, MeasEpoch
COM1>
```

sso gso	**setSBFOutput** **getSBFOutput**	***Stream*** *Stream*	*Cd*	*Messages*	*Interval*					
		+ Stream1 ... Stream16 + Res1 + Res2 + Res3 + Res4 all	none COM1 COM2 COM3 COM4 USB1 USB2 IP10 ... IP17 NTR1 NTR2 NTR3 IPS1 IPS2 IPS3 IPS4 IPS5 IPR1 IPR2 IPR3 LOG1 ... LOG8	none [SBF List] + Measurements + Meas3 + RawNavBits + GPS + GLO + GAL + GEO + BDS + QZS + NavIC + PVTCart + PVTGeod + PVTExtra + Time + Event + DiffCorr + Status + LBand + UserGroups + Advanced + PostProcess + Rinex + RinexMeas3 + Support	off OnChange msec10 msec20 msec40 msec50 msec100 msec200 msec500 sec1 sec2 sec5 sec10 sec15 sec30 sec60 min2 min5 min10 min15 min30 min60					

RxControl: Communication > Output Settings > SBF Output > SBF Output

Use this command to output a set of SBF blocks on a given connection at a regular interval.

A *Stream* is defined as a list of messages that should be output with the same interval on one connection descriptor (*Cd* - see 1.1.5). In other words, one *Stream* is associated with one *Cd* and one *Interval*, and contains a list of SBF blocks defined by the *Messages* argument.

The list of supported SBF blocks [SBF List] can be found in appendix A.

Predefined groups of SBF blocks (such as `Measurements`) can be addressed in the *Messages* argument. These groups are defined in the table below.

Messages	**Description**
`Measurements`	+MeasEpoch +MeasExtra +EndOfMeas
`Meas3`	+Meas3Ranges +Meas3CN0HiRes +Meas3Doppler +Meas3PP +Meas3MP
`RawNavBits`	+GPSRawCA +GPSRawL2C +GPSRawL5 +GPSRawL1C +GLO-RawCA +GALRawFNAV +GALRawINAV +GALRawCNAV +GEO-RawL1 +GEORawL5 +BDSRaw +BDSRawB1C +BDSRawB2a +BD-SRawB2b +NAVICRaw +QZSRawL1CA +QZSRawL2C +QZSRawL5 +QZSRawL6 +QZSRawL1C +QZSRawL1S +QZSRawL5S
`GPS`	+GPSNav +GPSCNav +GPSAlm +GPSIon +GPSUtc
`GLO`	+GLONav +GLOAlm +GLOTime
`GAL`	+GALNav +GALAlm +GALIon +GALUtc +GALGstGps +GALSAR-RLM

Messages (Continued)	Description
`GEO`	+GEOMT00 +GEOPRNMask +GEOFastCorr +GEOIntegrity +GEOFastCorrDegr +GEONav +GEODegrFactors +GEONetworkTime +GEOAlm +GEOIGPMask +GEOLongTermCorr +GEOIonoDelay +GEOServiceLevel +GEOClockEphCovMatrix
`BDS`	+BDSAlm +BDSNav +BDSCNav2 +BDSIon +BDSUtc
`QZS`	+QZSNav +QZSAlm
`NavIC`	+NavICLNav
`PVTCart`	+PVTCartesian +PosCovCartesian +VelCovCartesian +BaseVectorCart
`PVTGeod`	+PVTGeodetic +PosCovGeodetic +VelCovGeodetic +BaseVectorGeod +PosLocal
`PVTExtra`	+DOP +PVTSatCartesian +PVTResiduals +RAIMStatistics +GEOCorrections +PVTSupport +PVTSupportA +EndOfPVT
`Time`	+ReceiverTime +xPPSOffset
`Event`	+ExtEvent +ExtEventPVTCartesian +ExtEventPVTGeodetic +ExtEventBaseVectGeod
`DiffCorr`	+DiffCorrIn +BaseStation +RTCMDatum
`Status`	+SatVisibility +ChannelStatus +ReceiverStatus +InputLink +OutputLink +IPStatus +NTRIPClientStatus +NTRIPServerStatus +WiFiAPStatus +WiFiClientStatus +CosmosStatus +PowerStatus +QualityInd +DiskStatus +LogStatus +RFStatus +DynDNSStatus +P2PPStatus +GALAuthStatus
`LBand`	+LBandTrackerStatus +LBAS1DecoderStatus +LBAS1Messages +LBandBeams
`UserGroups`	+Group1 +Group2 +Group3 +Group4
`Advanced`	+SystemInfo
`PostProcess`	+MeasEpoch +MeasExtra +GEORawL1 +GPSNav +GPSIon +GPSUtc +GLONav +GLOTime +GALNav +GALIon +GALUtc +GALGstGps +GEONav +BDSNav +BDSIon +BDSUtc +QZSNav +DiffCorrIn +ReceiverSetup +Commands +ExtEvent
`Rinex`	+MeasEpoch +GPSNav +GPSCNav +GPSIon +GPSUtc +GLONav +GALNav +GALUtc +GALGstGps +GEONav +BDSNav +BDSCNav2 +QZSNav +NavICLNav +PVTGeodetic +ReceiverSetup +Comment
`RinexMeas3`	+Meas3Ranges +GPSCNav +GPSNav +GPSIon +GPSUtc +GLONav +GALNav +GALUtc +GALGstGps +GEONav +BDSNav +BDSCNav2 +QZSNav +NavICLNav +ReceiverSetup +Comment

Messages (Continued)	Description
`Support`	+MeasEpoch +MeasExtra +EndOfMeas +GPSRawCA +GPSRawL2C +GPSRawL5 +GPSRawL1C +GLORawCA +GALRawFNAV +GALRawINAV +GALRawCNAV +GEORawL1 +GEORawL5 +BDSRaw +BDSRawB1C +BDSRawB2a +BDSRawB2b +NAVICRaw +QZSRawL1CA +QZSRawL2C +QZSRawL5 +QZSRawL6 +QZSRawL1C +QZSRawL1S +QZSRawL5S +GPSNav +GPSAlm +GPSIon +GPSUtc +GLONav +GLOAlm +GLOTime +GALNav +GALAlm +GALIon +GALUtc +GALGstGps +GALAuthStatus +GEONav +GEOAlm +BDSAlm +BDSNav +BDSIon +BDSUtc +QZSNav +QZSAlm +PVTGeodetic +PosCovGeodetic +BaseVectorGeod +DOP +PVTSupport +PVTSupportA +EndOfPVT +ExtEvent +DiffCorrIn +BaseStation +ChannelStatus +ReceiverStatus +InputLink +OutputLink +ReceiverSetup +RxComponents +Commands +RxMessage +LBandTrackerStatus +LBAS1DecoderStatus +IPStatus +NTRIPClientStatus +NTRIPServerStatus +WiFiAPStatus +WiFiClientStatus +CosmosStatus +PowerStatus +QualityInd +LBandBeams +DiskStatus +LogStatus +RFStatus +DynDNSStatus +P2PPStatus +SystemInfo

The *Interval* argument defines the rate at which the SBF blocks specified in the *Messages* argument are output. If set to `off`, the SBF blocks are disabled. If set to `OnChange`, the SBF blocks are output at their natural renewal rate (see section 4.1.8). If a specific interval is specified (e.g. `sec1` corresponds to an interval of 1 second), the SBF blocks are decimated from their renewal rate to the specified interval. Some blocks can only be output at their renewal rate (e.g. the `GPSNav` block). For these blocks, the receiver ignores the value of the *Interval* argument and always assumes `OnChange`. The list of those blocks can be found in appendix A (see the "Flex Rate" column).

Please make sure that the connection specified by *Cd* is configured to allow SBF output (this is the default for all connections). See the **`setDataInOut`** command.

`Res1` to `Res4` are reserved values of *Stream*. These streams are not saved in the configuration files and, as a consequence, they will always be reset at boot time. For most users, it is not recommended to use these streams.

Example

To output the `MeasEpoch` block at 10Hz and the `PVTGeodetic` block at 1Hz on COM1, use the following sequence:

```
COM1>  sso, Stream1, COM1, MeasEpoch, msec100 <CR>
$R: sso, Stream1, COM1, MeasEpoch, msec100
  SBFOutput, Stream1, COM1, MeasEpoch, msec100
COM1>  sso, Stream2, COM1, PVTGeodetic, sec1 <CR>
$R: sso, Stream2, COM1, PVTGeodetic, sec1
  SBFOutput, Stream2, COM1, PVTGeodetic, sec1
COM1>
```

3.2.17 BINEX Configuration

sbfm gbfm	**setBINEXFormatting** **getBINEXFormatting**	*ObsSignals*	*ExtraObsTypes*								
		+ GPSL1CA + GPSL1PY + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL1P + GLOL2P + GLOL2CA + GLOL3 + GALL1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL1C + QZSL1S + QZSL5S + NAVICL5 + GPS + GLONASS + GALILEO + SBAS + BEIDOU + QZSS + NAVIC all	none + Doppler								

RxControl: Communication > Output Settings > BINEX Output > Record Options

Use these commands to configure the contents of the observation records.

The *ObsSignals* argument specifies the signal types to be encoded in BINEX 0x7f-05 records. `GPS`, `GLONASS`, etc. are aliases to enable/disable all signals from a constellation at once.

For an observable to be actually encoded in BINEX, the corresponding signal type must be enabled with this command and the signal must be enabled for tracking (see the **`setSignalTracking`** command). By default, BINEX 0x7f records contain observables for all tracked signals.

The *ExtraObsTypes* argument is used to include or exclude Doppler observables in record 0x7f-05.

Note that this command is not connection-specific. For example, it is not possible to send observation records without Doppler through COM1, and at the same time to send observation records with Doppler through COM2.

Example

```
COM1>  sbfm, GPSL1CA+GPSL2PY, Doppler<CR>
$R: sbfm, GPSL1CA+GPSL2PY, Doppler
  BINEXFormatting, GPSL1CA+GPSL2PY, Doppler
COM1>
```

ebio gbio	**exeBINEXOnce** **getBINEXOnce**	***Cd***	***Messages***								
		COM1 COM2 COM3 COM4 USB1 USB2 IP10 ... IP17 NTR1 NTR2 NTR3 IPS1 IPS2 IPS3 IPS4 IPS5 IPR1 IPR2 IPR3 LOG1 ... LOG8	+ Rec00 + Rec0101 + Rec0102 + Rec0103 + Rec0105 + Rec0106 + Rec0107 + Rec0114 + Rec05Geod + Rec7D00 + Rec7F05 + Rec01Nav								

RxControl: Communication > Output Settings > BINEX Output Once

Use this command to output a set of BINEX records on a given connection. This command differs from the related **`setBINEXOutput`** command in that it instructs the receiver to output the specified BINEX records only once, instead of at regular intervals.

The *Cd* argument defines the connection descriptor (see 1.1.5) on which the records should be output and the *Messages* argument defines the list of records that should be output.

Refer to appendix B for a short description of the available BINEX records. `Rec01Nav` is an alias for the set of all decoded ephemeris records.

Make sure that the connection specified by *Cd* is configured to allow BINEX output (this is the default for all connections). See also the **`setDataInOut`** command.

Example

To output all the current decoded navigation records once, use:

```
COM1>  ebio, COM1, Rec01Nav <CR>
$R: ebio, COM1, Rec01Nav
  BINEXOnce, COM1, Rec0101+Rec0102+Rec0103+Rec0114+Rec0105+Rec0106
COM1>
```

sbo gbo	**setBINEXOutput** **getBINEXOutput**	***Stream*** *Stream*	*Cd*	*Messages*	*Interval*					
		+ Stream1 ... Stream16 all	none COM1 COM2 COM3 COM4 USB1 USB2 IP10 ... IP17 NTR1 NTR2 NTR3 IPS1 IPS2 IPS3 IPS4 IPS5 IPR1 IPR2 IPR3 LOG1 ... LOG8	none + Rec00 + Rec0101 + Rec0102 + Rec0103 + Rec0105 + Rec0106 + Rec0107 + Rec0114 + Rec0141 ... Rec0147 + Rec05Geod + Rec7D00 + Rec7E01 + Rec7F05 + Rec01Nav + Rec01Raw	off OnChange msec100 msec200 msec500 sec1 sec2 sec5 sec10 sec15 sec30 sec60 min2 min5 min10 min15 min30 min60					

RxControl: Communication > Output Settings > BINEX Output > BINEX Output Intervals

Use this command to output a set of BINEX records on a given connection at a regular interval.

A *Stream* is defined as a list of records that should be output with the same interval on one connection descriptor (*Cd* - see 1.1.5). In other words, one *Stream* is associated with one *Cd* and one *Interval*, and contains a list of BINEX records defined by the *Messages* argument.

Refer to appendix B for a short description of the available BINEX records. `Rec01Nav` is an alias for "`Rec0101+Rec0102+Rec0103+Rec0114+Rec0105+Rec0106+Rec0107`", and `Rec01Raw` is an alias for "`Rec0141+Rec0142+...+Rec0147`".

The *Interval* argument defines the rate at which the BINEX records specified in the *Messages* argument are output. If set to `off`, the BINEX records are disabled. If set to `OnChange`, the BINEX records are output at their OnChange rate defined in appendix B. If another interval is specified (e.g. `sec1` corresponding to an interval of 1 second), the BINEX records are decimated from their OnChange rate to the specified interval. Many records can only be output at their OnChange rate. For these records, the receiver ignores the value of the *Interval* argument and always assumes `OnChange`. See the "Flex Rate" column in appendix B to know if a record can be output at a flexible rate.

Please make sure that the connection specified by *Cd* is configured to allow BINEX output (this is the default for all connections). See the **`setDataInOut`** command.

Example

To output all the decoded ephemeris records at their OnChange rate together with the GNSS observables at 1 Hz on COM1, use the following command:

```
COM1> sbo, Stream1, COM1, Rec01Nav + Rec7F05, sec1 <CR>
$R: sbo, Stream1, COM1, Rec01Nav + Rec7F05, sec1
  BINEXOutput, Stream1, COM1, Rec0101+Rec0102+Rec0103+Rec0114+
     Rec0105+Rec0106+Rec7F05, sec1
COM1>
```

3.2.18 RTCM v2.x Settings

sr2c gr2c	**setRTCMv2Compatibility** **getRTCMv2Compatibility**	*PRCType*	*GLOToD*	*RTKVersion*						
		Standard GroupDelay	Tk Tb	v2.1 v2.2orLater						

RxControl: Communication > Input Settings > Differential Corrections > RTCMv2

Use these commands to define/inquire the compatibility of the RTCM 2.x input correction stream. This command applies to rover receivers only and should be used in case the available base station correction stream is not fully compatible with the latest version of the RTCM 2.x standard.

The *PRCType* argument is used to handle a difference in the interpretation of DGPS corrections between the version 2.0 of the RTCM standard and later versions. If the base station is sending RTCM Message Type 1 based on version 2.0, the value `GroupDelay` must be selected to have a correct usage of incoming corrections.

The *GLOToD* argument specifies how to interpret the time-of-day field in the differential GLONASS correction message (MT31). Select `Tb` to be compatible with RTCM version up to 2.2, and select `Tk` to be compatible with RTCM 2.3 and later.

The *RTKVersion* argument specifies if the base station encodes RTK correction messages (MT18 to MT21) according to version 2.1 of the RTCM standard, or according to version 2.2 or above.

Example

To make to rover receiver compatible with a base station sending RTCM 2.2 corrections, use:

```
COM1>  sr2c, , Tb <CR>
$R: sr2c, , Tb
  RTCMv2Compatibility, Standard, Tb, v2.2orLater
COM1>
```

sr2h gr2h	**setRTCMv2EphemerisHoldoff** **getRTCMv2EphemerisHoldoff**	*TimeGPS*	*TimeGLO*							
		0 . . . 300 s	0 . . . 300 s							

RxControl: Communication > Output Settings > Differential Corrections > RTCMv2

This command specifies the delay in applying new navigation data sets when generating DGPS correction messages such as MT1.

The delay can be set independently for GPS and GLONASS satellites with the *TimeGPS* and *TimeGLO* arguments.

Upon reception of a new navigation data set from GPS(GLONASS) satellites (IOD change), the DGPS corrections continue to refer to the previous data set for *TimeGPS*(*TimeGLO*) seconds.

Example

```
COM1>  sr2h, 60, 60<CR>
$R: sr2h, 60, 60
  RTCMv2EphemerisHoldoff, 60, 60
COM1>
```

sr2f gr2f	**setRTCMv2Formatting** **getRTCMv2Formatting**	*ReferenceID*	*GLOToD*							
		0 . . . 1023	Tk Tb							

RxControl: Communication > Output Settings > Differential Corrections > RTCMv2

Use these commands to define/inquire the reference station ID assigned to the receiver when operating in base station mode. The reference station ID is transmitted in the first word of each outgoing RTCM v2.x message.

The argument *GLOToD* specifies how to encode the time-of-day field in the differential GLONASS correction message (MT31). Select `Tb` to be compatible with RTCM version up to 2.2, and select `Tk` to be compatible with RTCM 2.3 and later.

Examples

```
COM1>  sr2f, 345 <CR>
$R: sr2f, 345
  RTCMv2Formatting, 345, Tk
COM1>

COM1>  gr2f <CR>
$R: gr2f
  RTCMv2Formatting, 345, Tk
COM1>
```

sr2i gr2i	**setRTCMv2Interval** **getRTCMv2Interval**	***Message*** *Message*	*ZCount*							
		+RTCM1 +RTCM3 +RTCM9 +RTCM16 +RTCM17 +RTCM22 +RTCM23|24 +RTCM31 +RTCM32 all	1 . . . 2 . . . 1000							

RxControl: Communication > Output Settings > Differential Corrections > RTCMv2

Use these commands to define/inquire at which interval the RTCM v2.x messages specified in the *Message* argument should be generated. The related **setRTCMv2IntervalObs** command must be used to specify the interval of some RTK-related messages such as messages 18 and 19.

The interval for every message is given in the *ZCount* argument, in units of 0.6 seconds. For example, to generate a message every 6 seconds, *ZCount* should be set to 10.

For the ephemerides message (RTCM17), the ephemerides are sent out one satellite at a time, at a rate specified by this command. For instance, if *ZCount* is set to 1 and there are 12 ephemerides to send out, it takes 0.6*12=7.2 seconds to send the whole ephemerides set.

The intervals specified with this command are not connection-specific: all the connections which output a given RTCM v2.x message will output it with the same interval.

Note that this command only defines the interval of RTCM messages. To make the receiver actually output these messages, use the **setRTCMv2Output** and **setDataInOut** commands.

Refer to appendix D for an overview of the supported RTCM v2.x messages.

Examples

```
COM1>  sr2i, RTCM22, 15 <CR>
$R: sr2i, RTCM22, 15
  RTCMv2Interval, RTCM22, 15
COM1>
```

```
COM1>  gr2i <CR>
$R: gr2i
  RTCMv2Interval, RTCM1, 2
  RTCMv2Interval, RTCM3, 2
  RTCMv2Interval, RTCM16, 2
  RTCMv2Interval, RTCM22, 15
  RTCMv2Interval, RTCM23|24, 2
COM1>
```

sr2b gr2b	**setRTCMv2IntervalObs** **getRTCMv2IntervalObs**	***Message*** *Message*	*Interval*							
		+ RTCM18|19 + RTCM20|21 all	1 . . . 600 s							

RxControl: Communication > Output Settings > Differential Corrections > RTCMv2

Use these commands to define/inquire at which interval the RTCM v2.x messages specified in the *Message* argument should be generated. The related **`setRTCMv2Interval`** command must be used to specify the interval of other supported RCTCM v2.x messages.

The intervals specified with this command are not connection-specific: all the connections which output a given RTCM v2.x message will output it with the same interval.

Note that this command only defines the interval of RTCM messages. To make the receiver actually output these messages, use the **`setRTCMv2Output`** and **`setDataInOut`** commands.

Examples

```
COM1>  sr2b, RTCM20|21, 2 <CR>
$R: sr2b, RTCM20|21, 2
  RTCMv2IntervalObs, RTCM20|21, 2
COM1>

COM1>  gr2b <CR>
$R: gr2b
  RTCMv2IntervalObs, RTCM18|19, 1
  RTCMv2IntervalObs, RTCM20|21, 2
COM1>
```

sr2m gr2m	**setRTCMv2Message16** **getRTCMv2Message16**	*Message (90)*								
		Unknown								

RxControl: Communication > Output Settings > Differential Corrections > RTCMv2

Use these commands to define/inquire the string that will be transmitted in the RTCM v2.x message 16. The argument *Message* can contain up to 90 characters.

Note that this command only defines the content of message 16. To make the receiver actually output this message, use the `setRTCMv2Output` and `setDataInOut` commands.

Example

To send the string "Hello" in message 16 over COM2 at the default interval, use the following sequence:

```
COM1>  sr2m, Hello <CR>
$R: sr2m, Hello
  RTCMv2Message16, "Hello"
COM1>  sr2o, COM2, RTCM16 <CR>
$R: sr2o, COM2, RTCM16
  RTCMv2Output, COM2, RTCM16
COM1>  sdio, COM2, , RTCMv2 <CR>
$R: sdio, COM2, , RTCMv2
  DataInOut, COM2, auto, RTCMv2
COM1>
```

sr2o gr2o	**setRTCMv2Output** **getRTCMv2Output**	***Cd*** *Cd*	***Messages***							
		+COM1 +COM2 +COM3 +COM4 +USB1 +USB2 +IP10 ... IP17 +NTR1 +NTR2 +NTR3 +IPS1 +IPS2 +IPS3 +IPS4 +IPS5 +IPR1 +IPR2 +IPR3 all	none +RTCM1 +RTCM3 +RTCM9 +RTCM16 +RTCM18|19 +RTCM20|21 +RTCM22 +RTCM23|24 +RTCM31 +RTCM32 +RTCM17 +DGPS +RTK all							

RxControl: Communication > Output Settings > Differential Corrections > RTCMv2

Use these commands to define/inquire which RTCM v2.x messages are enabled for output on a given connection descriptor (*Cd* - see 1.1.5). The *Messages* argument specifies the RTCM message types to be enabled. Some pairs of messages are always enabled together, such as messages 18 and 19. `DGPS` is an alias for "`RTCM1+RTCM3+RTCM31`" and `RTK` is an alias for "`RTCM3+RTCM18|19+RTCM22+RTCM31`".

Refer to appendix D for an overview of the supported RTCM v2.x messages.

Please make sure that the connection specified by *Cd* is configured to allow RTCMv2 output, which can be done with the **`setDataInOut`** command. The interval at which each message is output is to be specified with the **`setRTCMv2Interval`** or the **`setRTCMv2IntervalObs`** command.

Example

To enable RTCM v2.x messages 3, 18, 19 and 22 on COM2, use the following sequence:

```
COM1>  sr2o, COM2, RTCM3+RTCM18|19+RTCM22 <CR>
$R: sr2o, COM2, RTCM3+RTCM18|19+RTCM22
  RTCMv2Output, COM2, RTCM3+RTCM18|19+RTCM22
COM1>  sdio, COM2, , RTCMv2 <CR>
$R: sdio, COM2, , RTCMv2
  DataInOut, COM2, auto, RTCMv2
COM1>
```

sr2u gr2u	**setRTCMv2Usage** **getRTCMv2Usage**	*MsgUsage*								
		none +RTCM1 +RTCM3 +RTCM9 +RTCM15 +RTCM18|19 +RTCM20|21 +RTCM22 +RTCM23|24 +RTCM31 +RTCM32 +RTCM34 +RTCM17 +RTCM59 all								

RxControl: Communication > Input Settings > Differential Corrections > RTCMv2

Use this command to restrict the list of incoming RTCM v2.x messages that the receiver is allowed to use in its differential PVT computation.

A short description of the supported RTCM v2.x messages can be found in appendix D.

Example

To only accept RTCM1 and RTCM3 corrections from the base station 1011, use the following sequence:

```
COM1>  sr2u, RTCM1+RTCM3 <CR>
$R: sr2u, RTCM1+RTCM3
  RTCMv2Usage, RTCM1+RTCM3
COM1>  sdcu, , , manual, 1011 <CR>
$R: sdcu, , , manual, 1011
  DiffCorrUsage, LowLatency, 3600.0, manual, 1011 ...
COM1>
```

3.2.19 RTCM v3.x Settings

sr3t gr3t	**setRTCMv3CRSTransfo** **getRTCMv3CRSTransfo**	*Mode*	*TargetName (32)*								
		auto manual									

RxControl: Communication > Input Settings > Differential Corrections > RTCMv3

Use this command to specify how to apply the coordinate reference system (CRS) transformation parameters contained in RTCM v3.x message types 1021 to 1023.

In `auto` mode (the default), the receiver decodes and applies the coordinate transformation parameters from message types 1021-1023. If your RTK provider sends transformation parameters for more than one target CRS, the receiver selects the first transformation parameters it receives.

In `manual` mode, you can force the receiver to only apply the transformation to the target CRS specified with the second argument. The *TargetName* argument must exactly match the name used by the RTK provider. The available target datum names can be found in the `RTCMDatum` SBF block.

Example

To force using the target CRS identified as "4258" by the RTK network, use:

```
COM1>  sr3t, manual, "4258"<CR>
$R: sr3t, manual, "4258"
  RTCMv3CRSTransfo, manual, "4258"
COM1>
```

sr3d gr3d	**setRTCMv3Delay** **getRTCMv3Delay**	*Delay*								
		0.0 . . . 600.0 s								

RxControl: Communication > Output Settings > Differential Corrections > RTCMv3

Use this command to instruct the receiver to generate and output RTCM v3.x messages with a certain delay.

It is possible to impose a global delay to all RTCM v3.x messages by setting the *Delay* to a non-zero value. This can be used in situations where multiple base stations must be configured to transmit their corrections in a time-multiplexed way. For example, base station A would compute and transmit its corrections at every 10-second epoch (in the GPS time scale), and base station B would compute and transmit its corrections 5 seconds after the 10-second epochs. In that case, receiver B would be configured with the *Delay* argument set to 5.

See also the **`setRTCMv3Interval`** command to configure the message interval.

Example

To generate the RTCM1001 message with an interval of 10 seconds and a time shift of 2 seconds, use:

```
COM1>  sr3i, RTCM1001|2, 10 <CR>
$R: sr3i, RTCM1001|2, 10
  RTCMv3Interval, RTCM1001|2, 10
COM1>  sr3d, 2 <CR>
$R: sr3d, 2
  RTCMv3Delay, 2
COM1>
```

sr3f gr3f	**setRTCMv3Formatting** **getRTCMv3Formatting**	*ReferenceID*	*MSMSignals*	*GLOL2*	*RxType (32)*					
		0 . . . 4095	+ GPSL1CA + GPSL1PY + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL1P + GLOL2P + GLOL2CA + GLOL3 + GALL1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL1C + NAVICL5 all	L2CA L2P	default					

RxControl: Communication > Output Settings > Differential Corrections > RTCMv3

Use these commands to configure the RTCM v3.x message contents when operating in base station mode.

The *ReferenceID* argument specifies the reference station ID transmitted in the header of each outgoing RTCM v3.x message.

The *MSMSignals* argument specifies the signal types to be encoded in MSM messages. For an observable to be actually encoded in MSM, the corresponding signal type must be enabled with this command, the signal must be enabled for tracking (see the `setSignalTracking` command), and a suitable MSM message must be enabled with the `setRTCMv3Output` command.

The *GLOL2* argument applies to message types 1011 and 1012 (GLONASS L1 and L2 observables). It specifies which of the L2P or the L2CA observables must be encoded in RTCM1011 and RTCM1012.

The *RxType* argument can be used to change the receiver type that is transmitted in message 1033, i.e. to change the way the receiver identifies itself to the RTCM network. Setting *RxType* to "default" reverts to the default receiver type.

Example

```
COM1>  sr3f, 345 <CR>
$R: sr3f, 345
  RTCMv3Formatting, 345, GPSL1CA+GPSL2PY+GLOL1CA+GLOL2CA+GALL1BC+
     GALE5a+BDSB1I+BDSB2I+QZSL1CA+QZSL2C, L2CA, default
```

```
COM1>
```

sr3i	**setRTCMv3Interval**	***Message***	*Interval*							
gr3i	**getRTCMv3Interval**	*Message*								
		+ RTCM1001|2 + RTCM1003|4 + RTCM1005|6 + RTCM1007|8 + RTCM1009|10 + RTCM1011|12 + RTCM1013 + RTCM1019 + RTCM1020 + RTCM1029 + RTCM1033 + RTCM1041 + RTCM1042 + RTCM1044 + RTCM1045 + RTCM1046 + RTCM1230 + MSM1 ... MSM7 all	0.1 . . . 1.0 . . . 600.0 s							

RxControl: Communication > Output Settings > Differential Corrections > RTCMv3

Use these commands to define/inquire at which interval RTCM v3.x messages should be generated.

The intervals specified with this command are not connection-specific: all the connections which output a given RTCM v3.x message will output it with the same interval.

Using `MSMi` for the *Message* argument sets the interval of all Multiple Signal Messages of type `i`. Refer to appendix D for an overview of the supported RTCM v3.x messages.

For the ephemerides messages (e.g. RTCM1019), the ephemerides are sent out one satellite at a time, at a rate specified by this command. For instance, if *Interval* is set to 1 and there are 12 GPS ephemerides to send out, it takes 12 seconds to send the whole GPS ephemerides set.

By default, RTCM v3.x messages are generated at integer multiples of the specified interval in the GPS time scale. The command **`setRTCMv3Delay`** can be used to introduce a time offset.

Note that this command only defines the interval of RTCM messages. To make the receiver actually output these messages, use the **`setRTCMv3Output`** and **`setDataInOut`** commands.

Example

```
COM1>  sr3i, RTCM1001|2, 2 <CR>
$R: sr3i, RTCM1001|2, 2
  RTCMv3Interval, RTCM1001|2, 2
COM1>
```

sr3m gr3m	**setRTCMv3Message1029** **getRTCMv3Message1029**	*Message (120)*								
		Unknown								

RxControl: Communication > Output Settings > Differential Corrections > RTCMv3

Use these commands to define/inquire the string that will be transmitted in the RTCM v3.x message 1029. The argument *Message* can contain up to 120 characters.

Note that this command only defines the content of message 1029. To make the receiver actually output this message, use the **`setRTCMv3Output`** and **`setDataInOut`** commands.

Example

To send the string "Hello" in message 1029 over COM2 at the default interval, use the following sequence:

```
COM1>  sr3m, Hello <CR>
$R: sr3m, Hello
  RTCMv3Message1029, "Hello"
COM1>  sr3o, COM2, RTCM1029 <CR>
$R: sr3o, COM2, RTCM1029
  RTCMv3Output, COM2, RTCM1029
COM1>  sdio, COM2, , RTCMv3 <CR>
$R: sdio, COM2, , RTCMv3
  DataInOut, COM2, auto, RTCMv3
COM1>
```

sr3o gr3o	**setRTCMv3Output** **getRTCMv3Output**	***Cd*** *Cd*	*Messages*								
		+COM1 +COM2 +COM3 +COM4 +USB1 +USB2 +IP10 ... IP17 +NTR1 +NTR2 +NTR3 +IPS1 +IPS2 +IPS3 +IPS4 +IPS5 +IPR1 +IPR2 +IPR3 all	none +RTCM1001 +RTCM1002 +RTCM1003 +RTCM1004 +RTCM1005 +RTCM1006 +RTCM1007 +RTCM1008 +RTCM1009 +RTCM1010 +RTCM1011 +RTCM1012 +RTCM1013 +RTCM1019 +RTCM1020 +RTCM1029 +RTCM1033 +RTCM1041 +RTCM1042 +RTCM1044 +RTCM1045 +RTCM1046 +RTCM1071 ... RTCM1077 +RTCM1081 ... RTCM1087 +RTCM1091 ... RTCM1097 +RTCM1101 ... RTCM1107 +RTCM1111 ... RTCM1117 +RTCM1121 ... RTCM1127 +RTCM1131 ... RTCM1137 +RTCM1230 +MSM1 +MSM2 +MSM3 +MSM4 +MSM5 +MSM6 +MSM7 all								

RxControl: Communication > Output Settings > Differential Corrections > RTCMv3

Use these commands to define/inquire which RTCM v3.x messages are enabled for output on a given connection descriptor (*Cd* - see 1.1.5). The *Messages* argument specifies the RTCM message types to be enabled.

A short description of the supported RTCM v3.x message types can be found in appendix D. `MSMi` enables the Multiple Signal Message - Type i from all constellations. Make sure to disable the legacy observation messages (MT1001-1004 and MT1009-1012) when enabling MSM messages as it is not advised to mix them.

Please make sure that the connection specified by *Cd* is configured to allow RTCMv3 output, which can be done with the **`setDataInOut`** command. The interval at which each message is output is to be specified with the **`setRTCMv3Interval`** command.

Example

To enable RTCM v3.x messages 1001, 1002, 1005 and 1006 on COM2, use the following sequence:

```
COM1>  sr3o, COM2, RTCM1001+RTCM1002+RTCM1005+RTCM1006 <CR>
$R: sr3o, COM2, RTCM1001+RTCM1002+RTCM1005+RTCM1006
  RTCMv3Output, COM2, RTCM1001+RTCM1002+RTCM1005+RTCM1006
COM1>  sdio, COM2, , RTCMv3 <CR>
$R: sdio, COM2, , RTCMv3
  DataInOut, COM2, auto, RTCMv3
COM1>
```

sr3u gr3u	**setRTCMv3Usage** **getRTCMv3Usage**	*MsgUsage*							
		none + RTCM1001 ... RTCM1013 + RTCM1015 + RTCM1016 + RTCM1017 + RTCM1019 ... RTCM1027 + RTCM1029 + RTCM1033 + RTCM1037 + RTCM1038 + RTCM1039 + RTCM1041 + RTCM1042 + RTCM1044 + RTCM1045 + RTCM1046 + RTCM1071 ... RTCM1077 + RTCM1081 ... RTCM1087 + RTCM1091 ... RTCM1097 + RTCM1111 ... RTCM1117 + RTCM1121 ... RTCM1127 + RTCM1230 + MSM1 + MSM2 + MSM3 + MSM4 + MSM5 + MSM6 + MSM7 all							

RxControl: Communication > Input Settings > Differential Corrections > RTCMv3

Use this command to restrict the list of incoming RTCM v3.x messages that the receiver is allowed to use in its differential PVT computation.

A short description of the supported RTCM v3.x messages can be found in appendix D. `MSMi` is an alias to enable the Multiple Signal Message - Type i from all constellations at once.

Example

To only accept RTCM1001 and RTCM1002 corrections from the base station 1011, use the following sequence:

```
COM1> sr3u, RTCM1001+RTCM1002 <CR>
$R: sr3u, RTCM1001+RTCM1002
  RTCMv3Usage, RTCM1001+RTCM1002
COM1> sdcu, , , manual, 1011 <CR>
$R: sdcu, , , manual, 1011
  DiffCorrUsage, LowLatency, 3600.0, manual, 1011 ...
COM1>
```

3.2.20 CMR v2.0 Settings

sc2f gc2f	**setCMRv2Formatting** **getCMRv2Formatting**	*ReferenceID*								
		0 . . . 31								

RxControl: Communication > Output Settings > Differential Corrections > CMRv2

Use these commands to define/inquire the reference station ID assigned to the receiver when operating in base station mode. The reference station ID is transmitted in the header of each outgoing CMR v2.0 message.

Examples

```
COM1>  sc2f, 12 <CR>
$R: sc2f, 12
  CMRv2Formatting, 12
COM1>

COM1>  gc2f <CR>
$R: gc2f
  CMRv2Formatting, 12
COM1>
```

sc2i gc2i	**setCMRv2Interval** **getCMRv2Interval**	***Message*** *Message*	*Interval*							
		+CMR0 +CMR1 +CMR2 +CMR3 all	0.1 . . . <u>1.0</u> . . . 600.0 s							

RxControl: Communication > Output Settings > Differential Corrections > CMRv2

Use these commands to define/inquire at which interval CMR v2.0 messages should be generated.

The intervals specified with this command are not connection-specific: all the connections which output a given CMR v2.0 message will output it with the same interval.

Note that this command only defines the interval of CMR messages. To make the receiver actually output these messages, use the **`setCMRv2Output`** and **`setDataInOut`** commands.

Refer to appendix D for an overview of the supported CMR v2.0 messages.

Examples

```
COM1>  sc2i, CMR0, 2 <CR>
$R: sc2i, CMR0, 2
  CMRv2Interval, CMR0, 2
COM1>

COM1>  gc2i <CR>
$R: gc2i CMRv2Interval, CMR0, 2
  CMRv2Interval, CMR1, 1 CMRv2Interval, CMR2, 1
COM1>
```

sc2m gc2m	**setCMRv2Message2** **getCMRv2Message2**	*ShortID (8)*	*LongID (50)*	*COGO (16)*							
		Unknown	Unknown	Unknown							

RxControl: Communication > Output Settings > Differential Corrections > CMRv2

Use these commands to define/inquire the strings that will be transmitted in the CMR v2.0 message 2.

The argument *ShortID* is the short station ID. It can contain up to 8 characters in compliance with the CMR standard. If less than 8 characters are defined, the string will be right justified and padded with spaces.

The argument *LongID* is the long station ID. It can contain up to 50 characters in compliance with the CMR standard. If less than 50 characters are defined, the string will be right justified and padded with spaces.

The argument *COGO* is the COGO code. It can contain up to 16 characters in compliance with the CMR standard. If less than 16 characters are defined, the string will be right justified and padded with spaces.

Note that this command only defines the contents of message 2. To make the receiver actually output this message, use the **`setCMRv2Output`** and **`setDataInOut`** commands.

Example

To send the string "Hello" as short station ID and send CMR2 messages through COM2, use the following sequence:

```
COM1>  sc2m, Hello <CR>
$R: sc2m, Hello
  CMRv2Message2, "Hello", "Unknown", "Unknown"
COM1>  sc2o, COM2, CMR2 <CR>
$R: sc2o, COM2, CMR2
  CMRv2Output, COM2, CMR2
COM1>  sdio, COM2, , CMRv2 <CR>
$R: sdio, COM2, , CMRv2
  DataInOut, COM2, auto, CMRv2
COM1>
```

sc2o	**setCMRv2Output**	***Cd***	*Messages*							
gc2o	**getCMRv2Output**	*Cd*								
		+COM1 +COM2 +COM3 +COM4 +USB1 +USB2 +IP10 ... IP17 +NTR1 +NTR2 +NTR3 +IPS1 +IPS2 +IPS3 +IPS4 +IPS5 +IPR1 +IPR2 +IPR3 all	none +CMR0 +CMR1 +CMR2 +CMR3 all							

RxControl: Communication > Output Settings > Differential Corrections > CMRv2

Use these commands to define/inquire which CMR v2.0 messages are enabled for output on a given connection descriptor (*Cd* - see 1.1.5). The *Messages* argument specifies the CMR message types to be enabled. Refer to appendix D for an overview of the supported CMR v2.0 messages.

Please make sure that the connection specified by *Cd* is configured to allow CMRv2 output, which can be done with the `setDataInOut` command. The interval at which each message is output is to be specified with the `setCMRv2Interval` command.

Example

To enable CMR v2.0 message 0 on COM2, use the following sequence:

```
COM1> sc2o, COM2, CMR0 <CR>
$R: sc2o, COM2, CMR0
  CMRv2Output, COM2, CMR0
COM1> sdio, COM2, , CMRv2 <CR>
$R: sdio, COM2, , CMRv2
  DataInOut, COM2, auto, CMRv2
COM1>
```

sc2u gc2u	**setCMRv2Usage** **getCMRv2Usage**	*MsgUsage*								
		none +CMR0 +CMR1 +CMR2 +CMR3 +CMR0p +CMR0w all								

RxControl: Communication > Input Settings > Differential Corrections > CMRv2

Use this command to restrict the list of incoming CMR v2.0 messages that the receiver is allowed to use in its differential PVT computation. `CMR0p` and `CMR0w` refer to the CMR+ and CMR-W variants respectively.

A short description of the supported CMR v2.0 messages can be found in appendix D.

Example

To only accept CMR0 from the base station 12, use the following sequence:

```
COM1>  sc2u, CMR0 <CR>
$R: sc2u, CMR0
  CMRv2Usage, CMR0
COM1>  sdcu, , , manual, 12 <CR>
$R: sdcu, , , manual, 12
  DiffCorrUsage, LowLatency, 3600.0, manual, 12 ...
COM1>
```

3.2.21 Internal Disk Logging

sblp gblp	**setBINEXLoggingParameters** **getBINEXLoggingParameters**	*Cd* *Cd*	*Duration*	*Compression*						
		+ LOG1 ... LOG8 all	min15 hour1 hour6 hour24	off on						

RxControl: Logging > Internal Logging Settings > BINEX Logging and Upload

Use these commands to define the BINEX logging parameters.

The first argument defines the file duration: the receiver automatically starts a new file every 15 minutes, every hour, every 6 hours or every 24 hours according to the *Duration* argument.

The second argument enables or disables compression. Compression takes place when the file is complete, i.e. at the end of each *Duration* interval. After compression, the original (non-compressed) file is deleted.

The BINEX file name is `ssssdddfmm.yy.bnx`, where `ssssdddfmm.yy` follows the RINEX 2.11 definition. The 4-character station identifier is the first four letters of the station code as set by the **`setMarkerParameters`** command. If the station code is empty, the first four letters of the marker name are used. After compression, the `.gz` extension is appended to the file name.

If desired, it is also possible to add a log session ID prefix to all BINEX file names. This is enabled with the **`setGlobalFileNamingOptions`** command.

The BINEX files are put in daily directories, the directory name being of the form `yyddd` with `yy` the 2-digit year and `ddd` the day of year.

Example

```
COM1>  sblp, LOG1, hour1, on<CR>
$R: sblp, LOG1, hour1, on
  BINEXLoggingParameters, LOG1, hour1, on
COM1>
```

scgl gcgl	**setCGGTTSLogging** **getCGGTTSLogging**	***Cd*** *Cd*	*FileTypes*							
		+LOG1 +LOG2 ... LOG8 all	none +GPS +GLONASS +GALILEO +BEIDOU all							

RxControl: Logging > *Internal CGGTTS Logging* > *CGGTTS File Logging*

Use this command to enable/disable logging of CGGTTS files in one or more logging sessions.

The receiver can generate CGGTTS files for the GPS, GLONASS, Galileo and BeiDou constellations. These files can be logged in a single session, or in different sessions if preferred.

The CGGTTS files always cover a full day, and they are put in daily directories, the directory name being of the form `yyddd` with `yy` the 2-digit year and `ddd` the day of year.

The file name conforms to the CGGTTS-v2E convention: `XFLLmodd.ddd`, with `X` the constellation code, `F` the frequency type (always set to "Z"), `LL` the laboratory identifier set with **`setTimeLabParameters`**, `mo` the receiver identifier set with **`setTimeLabParameters`**, and `dd.ddd` the modified Julian date.

It is recommended to configure all CGGTTS parameters before enabling logging (see the **`setTimeLabParameters`**, **`setCGGTTSPosition`**, **`setCGGTTSTrackParameters`**, **`setINTDLY`**, **`setCABDLY`** and **`setREFDLY`** commands). If the CGGTTS parameters are changed while logging is ongoing, the CGGTTS file header information may not correspond to the actual content of the file.

Note that it can take up to 30 minutes between enabling CGGTTS logging and actually seeing the CGGTTS file on the disk. This is because CGGTTS data are generated at a very low rate (typically every 16 minutes).

Example

```
COM1>  scgl, LOG1, GPS+GALILEO<CR>
$R: scgl, LOG1, GPS+GALILEO
  CGGTTSLogging, LOG1, GPS+GALILEO
COM1>
```

lde	lstDiskEvent	*Disk*	*SessionName (60*							
		DSK1 DSK2 all								

Use this command to retrieve information about the disk identified by the *Disk* argument. `DSK1` is the internal disk and `DSK2` is the external disk (USB drive). The reply to this command lists the events that have occurred for a given log session in the context of the "preserve on event" feature. See also the **`setPreserveOnEvent`** command.

When only using the *Disk* argument and omitting the *SessionName* argument, the different log session names are listed.

When filling in both the *Disk* and the *SessionName* arguments, a list of events for the specified log session is generated.

The events are grouped per minute. The listed `<etime>` means that at least one event of type `<Event type>` happened between 30s before and 30s after `<etime>`. `etime` is a Unix timestamp. The fields `<before>` and `<after>` correspond to the applicable value of the *TimeBefore* and *TimeAfter* arguments of the **`setPreserveOnEvent`** command.

For the future events the Planned tag is diplayed.

Example

```
COM1>  lde, DSK1, LOG1_a_name <CR>
$R; lde, DSK1, LOG1_a_name
---->
$-- BLOCK 1 / 0
<?xml version="1.0" encoding="ISO-8859-1" ?>
<DiskEvent version="0.1">
  <Disk name="DSK1" total="15618400256" free="10437935104" >
    <Dir name="LOG1_a_name">
      <Dir name="19260" mtime="1568764782" />
        <Event type="EventA" etime="1568727400" before="1440" after
           ="1440" />
        <Event type="EventB" etime="1568732640" before="1440" after
           ="1440" />
      </Dir>
      <Dir name="19261" mtime="1568793582" />
        <Event type="EventA" etime="1568791150" before="1440" after
           ="1440" />
        <Event type="EventB" etime="1568796180" before="1440" after
           ="1440" />
        <Event type="EventA" etime="1568796190" before="1440" after
           ="1440" />
      </Dir>
    </Dir>
    <Planned>
      <Event type="Command" etime="1586529330" before="1440" after=
         "1440" />
      <Event type="Command" etime="1586615730" before="1440" after=
         "1440" />
      <Event type="Command" etime="1588343730" before="1440" after=
         "1440" />
    </Planned>
  </Disk>
</DiskEvent>
COM1>
```

sdfa gdfa	**setDiskFullAction** **getDiskFullAction**	***Disk*** *Disk*	*Action*							
		+ DSK1 + DSK2 all	DeleteOldest StopLogging							

RxControl: Logging > Internal Logging Settings > Global Logging Options

Use these commands to define/inquire what the receiver should do when the disk identified by *Disk* is full. `DSK1` is the internal disk and `DSK2` is the external disk (USB drive).

The currently supported actions are as follows:

Action	**Description**
`DeleteOldest`	The receiver deletes old files from the disk that is becoming full. The files in unused or disabled log sessions (see the **`setLogSession`** command) are first to be deleted. Then the receiver scans the enabled low-priority sessions. The oldest file in these sessions is identified and deleted. If no file could be deleted, the receiver scans the medium-priority sessions, and finally it will scan the high-priority sessions. See the **`setLogSession`** command to define the log session priority.
`StopLogging`	All logging activity stops on the specified disk.

Examples

```
COM1>  sdfa, DSK1, StopLogging <CR>
$R: sdfa, DSK1, StopLogging
  DiskFullAction, DSK1, StopLogging
COM1>

COM1>  gdfa <CR>
$R: gdfa
  DiskFullAction, DSK1, StopLogging
COM1>
```

ldi	**lstDiskInfo**	***Disk***	***Directory (60)***							
		DSK1 DSK2 all								

Use this command to retrieve information about the disk identified by the *Disk* argument. `DSK1` is the internal disk and `DSK2` is the external disk (USB drive). The reply to this command contains the disk size and free space in bytes and the list of all recorded files and directories.

The content of directories is not shown by default. To list the content of a directory, use the second argument to specify the directory name.

Example

```
COM1>  ldi, DSK1 <CR>
$R; ldi, dsk1
---->
$-- BLOCK 1 / 0
<?xml version="1.0" encoding="ISO-8859-1" ?>
<DiskInfo version="0.1"gt;
  <Disk name="DSK1" total="2030927872" free="2030764032" >
    <File name="log.sbf" size="16384" />
    <File name="leuv2050.07_" size="35196" />
  </Disk>
</DiskInfo>
COM1>
```

sfn gfn	**setFileNaming** **getFileNaming**	***Cd*** *Cd*	*NamingType*	*FileName (20)*	*Compression*					
		+LOG1 ... LOG8 all	FileName Incremental IGS15M IGS1H IGS6H IGS24H	log	off on					

RxControl: Logging > Internal Logging Settings > SBF Logging and Upload

Use these commands to define/inquire the file naming convention for the internal SBF files, and to enable compression.

If *NamingType* is `FileName`, the file name is given by the third argument *FileName*, followed by the extension `.sbf`.

If *NamingType* is `Incremental`, the receiver appends a 3-digit counter to the file name provided in the *FileName* argument. The counter increments each time logging is stopped and restarted. When the counter reaches 1000, it restarts at zero. If the resulting file name (*FileName*+counter) already exists on the disk, the existing file is overwritten.

The set of allowed characters for the *FileName* argument is:

`_0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`

If *NamingType* is `IGS15M`, `IGS1H`, `IGS6H` or `IGS24H`, the receiver automatically creates a new file every 15 minutes, every hour, every 6 hours or every 24 hours respectively, and the file name adheres to the IGS/RINEX2.11 naming convention.

The 4-character station identifier is the first four letters of the station code as set by the **`setMarkerParameters`** command. If the station code is empty, the first four letters of the marker name are used.

If desired, it is also possible to add a log session ID prefix to all file names logged in IGS naming mode. This is enabled with the **`setGlobalFileNamingOptions`** command.

In IGS naming mode, the files are put in daily directories, the directory name being of the form `yyddd` with `yy` the 2-digit year and `ddd` the day of year. If *NamingType* is `FileName` or `Incremental`, the file is put in the root directory of the log session.

If the *Compression* argument is set to `on`, the SBF files are compressed. Compression is only applied to files logged in one of the IGS naming modes, and takes place when the files are complete. After compression, the original (non-compressed) files are deleted.

If the naming convention is changed while logging is ongoing, the current file is closed and the logging continues in a new file with the name as specified.

Example

To have a fixed file name "`mytest.sbf`" in the first log session, use:

```
COM1>  sfn, LOG1, FileName, mytest <CR>
$R: sfn, LOG1, FileName, mytest
  FileNaming, LOG1, FileName, "mytest", off
COM1>
```

sfno gfno	**setGlobalFileNamingOptions** **getGlobalFileNamingOptions**	*BusyTag*	*SessionTag*							
		off on	off on							

RxControl: Logging > Internal Logging Settings > Global Logging Options

By default, files names follow the convention described with the logging commands (**`setFileNaming`**, **`setBINEXLoggingParameters`**, **`setNMEALogging`**, **`setRTCMMSMLogging`**, **`setRINEXLogging`**).

By setting the *BusyTag* argument to `on`, a "`.A`" suffix is added to all files that are currently written to for easy identification. The suffix is removed when the file is closed.

By setting the *SessionTag* argument to `on`, the log session ID is prefixed to all file names, with the exception of the manually-specified SBF file names. This makes that all file names logged by the receiver are unique.

Example

```
COM1>  sfno, on, on<CR>
$R: sfno, on, on
  GlobalFileNamingOptions, on, on
COM1>
```

sls	**setLogSession**	***Cd***	*State*	*Disk*	*Name (30)*	*AutoDelete*	*Priority*	*Type*		
gls	**getLogSession**	*Cd*								
		+LOG1 ... LOG8	Unused	DSK1		Never	Low	Continuous		
		all	Disabled	DSK2		After1Day	Medium	Scheduled		
			Enabled			After2Days	High	PreserveOnEvent		
						After3Days				
						After4Days				
						After7Days				
						After14Days				
						After21Days				
						After30Days				
						After60Days				
						After90Days				
						After180Days				
						After1Year				
						After2Years				

RxControl: Logging > Internal Logging Settings > Log Sessions

This command defines the parameters of the log sessions. For each session, the user can specify where to log (DSK1: internal disk or DSK2: USB drive), the session name, when to auto-delete the files, and the session priority.

The state of each log session is set by the *State* argument. If *State* is `Enabled`, the log session is enabled for logging. If *State* is `Unused` or `Disabled`, the log session stops logging data.

Log files of session *i* are logged in a folder called "LOG*i*_*Name*" where *Name* is the session name provided in this command. If a folder named "LOG*i*_..." already exists on the disk, the existing folder is renamed to the new name without loosing its contents. If there is no "LOG*i*_..." folder yet, it is created.

The log files are automatically deleted after the interval specified in the *AutoDelete* argument (unless they are flagged as "to be preserved", see the *Type* argument below). If the *AutoDelete* argument is set to `Never`, files are never deleted automatically (except when the disk is full, see the **`setDiskFullAction`** command). It is the responsibility of the user to timely download and delete the files, or to timely stop the logging to avoid filling up the disk.

The *Priority* argument defines the session priority when the disk is full and the receiver starts to free-up disk space by deleting old files (see the **`setDiskFullAction`** command for details). Sessions with a low priority will be the first to be deleted upon a disk full condition.

Logging sessions can be of different types:

Type	Description
`Continuous`	The log session is logging continuously and deleting old files as per the *AutoDelete* settings.
`Scheduled`	Logging is enabled/disabled according to the schedule set with the **`setLogSessionSchedule`** command. Auto-deletion is the same as in `Continuous` mode.

Type (Continued)	Description
`PreserveOnEvent`	Logging is continuous, but auto-deletion will not delete files flagged as "to be preserved". The criterium to set the "to be preserved" flag must be defined with the **`setPreserveOnEvent`** command. Only files logged in one of the IGS naming modes can be preserved. When the disk is full and the receiver tries to delete old files to free-up space (see the **`setDiskFullAction`** command), preserved files will be the last to be deleted (starting from the session with lowest priority).

Example

```
COM1>  sls, LOG1, Enabled, DSK1, MySession, After1Year, High,
   Continuous<CR>
$R: sls, LOG1, Enabled, DSK1, MySession, After1Year, High,
   Continuous
  LogSession, LOG1, Enabled, DSK1, MySession, After1Year, High,
     Continuous
COM1>
```

slss glss	**setLogSessionSchedule** **getLogSessionSchedule**	***Cd*** *Cd*	*StartTime (30)*	*Duration*	*RepetitionIntrvl*					
		+ LOG1 ... LOG8 all	2000-01-01 00:00:00	0 . . . 2678400 s	0 . . . 2678400 s					

RxControl: Logging > *Internal Logging Settings* > *Log Sessions*

This command defines the logging schedule for log sessions configured in `Scheduled` mode with the **`setLogSession`** command.

The *StartTime* argument defines the epoch when the receiver should start logging the first time. It also serves as reference epoch for the *RepetitionIntrvl* argument. It refers to the GPS time scale. The format of the *StartTime* argument is "YYYY-MM-DD hh:mm:ss".

The *Duration* argument defines the period for which the receiver should stay logging. If this argument is set to 0 (the default value), the receiver will remain logging indefinitely.

The *RepetitionIntrvl* can be used to repeat the logging pattern at regular interval. *RepetitionIntrvl* should be at least 5 seconds longer than *Duration*. If *RepetitionIntrvl* is set to a value smaller than *Duration*, the repetition functionality is disabled.

Be aware that the receiver must know the time to automatically schedule the logging: if no antenna is connected to the receiver or if not enough satellites could be tracked since boot, the receiver will not log any data.

Example

If you want to log for one hour every day from 00:00:00 till 01:00:00 (GPS time), use:

```
COM1> sls, LOG1, Enabled, DSK1, "Example", After2Days, Medium,
   Scheduled <CR>
$R: sls, LOG1, Enabled, DSK1, "Example", After2Days, Medium,
   Scheduled
  LogSession, LOG1, Enabled, DSK1, "Example", After2Days, Medium,
     Scheduled
COM1> slss, , 3600, 86400 <CR>
$R: slss, , 3600, 86400
  LogSessionSchedule, "2000-01-01 00:00:00", 3600, 86400
COM1>
```

emd gmd	**exeManageDisk** **getManageDisk**	*Disk*	*Action*							
		DSK1 DSK2	Unmount Mount Format							

RxControl: Logging > Disk(s) > Disk Management

Use this command to manage the disk identified by the *Disk* argument. `DSK1` is the internal disk and `DSK2` is the external disk (USB drive).

Specify the action `Format` to format the disk (all data will be lost).

The `Mount` and `Unmount` actions mount and unmount the disk respectively. Unmounting an internal disk makes it available as a mass-storage device when the USB cable is connected to a PC, i.e. it makes the disk appear as a drive on most file browsers. When the disk is mounted, it cannot be accessed as a mass-storage device. Internal logging is only possible when the disk is mounted.

Prior to formatting or unmounting the disk, make sure to stop all disk activities. If the specified action could not be performed, an error message is returned.

Example

To format the first disk (`DSK1`), use:

```
COM1>  emd, DSK1, Format <CR>
               $R: emd, DSK1, Format
               ManageDisk, DSK1, Format
               COM1>
```

snlp gnlp	**setNMEALogging** **getNMEALogging**	***Cd*** *Cd*	*NamingType*	*FileName (20)*	*Compression*					
		+LOG1 ... LOG8 all	FileName Incremental IGS15M IGS1H IGS6H IGS24H	log	off on					

RxControl: Logging > Internal Logging Settings > NMEA Logging and Upload

Use these commands to define/inquire the NMEA logging parameters.

If *NamingType* is `FileName`, the file name is given by the third argument *FileName*, followed by the extension `.nma`.

If *NamingType* is `Incremental`, the receiver appends a 3-digit counter to the file name provided in the *FileName* argument. The counter increments each time logging is stopped and restarted. When the counter reaches 1000, it restarts at zero. If the resulting file name (*FileName*+counter) already exists on the disk, the existing file is overwritten.

The set of allowed characters for the *FileName* argument is:

`_0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`

If *NamingType* is `IGS15M`, `IGS1H`, `IGS6H` or `IGS24H`, the receiver automatically creates a new file every 15 minutes, every hour, every 6 hours or every 24 hours respectively, and the file name adheres to the IGS/RINEX2.11 naming convention. The 4-character station identifier is the first four letters of the station code as set by the **`setMarkerParameters`** command. If the station code is empty, the first four letters of the marker name are used.

If desired, it is also possible to add a log session ID prefix to all file names logged in IGS naming mode. This is enabled with the **`setGlobalFileNamingOptions`** command.

In IGS naming mode, the files are put in daily directories, the directory name being of the form `yyddd` with `yy` the 2-digit year and `ddd` the day of year. If *NamingType* is `FileName` or `Incremental`, the file is put in the root directory of the log session.

If the *Compression* argument is set to `on`, the NMEA files are compressed. Compression is only applied to files logged in one of the IGS naming modes, and takes place when the files are complete. After compression, the original (non-compressed) files are deleted.

If the naming convention is changed while logging is ongoing, the current file is closed and the logging continues in a new file with the name as specified.

Example

To have a fixed file name "`mytest.nma`" in the first log session, use:

```
COM1>  nlp, LOG1, FileName, mytest <CR>
 $R: nlp, LOG1, FileName, mytest
   FileNaming, LOG1, FileName, "mytest", off
 COM1>
```

epl gpl	**exePreserveLogging** **getPreserveLogging**	***EventTime (30)***								
		2000-01-01 00:00:00								

RxControl: Logging > Preserve Log Files

This command prevents the automatic deletion of a specified event's logs, pointed by *EventTime*. Preserved files will not be deleted by the auto-delete functionality and they will also be the last to be deleted when the receiver attempts to free-up disk space when the disk is full. The command can add future events also and these can be viewed in the Planned tag using **`lstDiskEvent`** command. Refer to the **`lstDiskEvent`**, **`setLogSession`** and **`PreserveOnEvent`** commands for further details.

The *EventTime* argument defines the time of the event that the receiver should preserve. The format of the *EventTime* argument is "YYYY-MM-DD hh:mm:ss". Note: the Preserve-On-Event logic expects events to be marked at half-minute time, so *EventTime* argument is adjusted accordingly. When adding an already registered event this is updated using the current sessions params.

Example

```
COM1>  epl, 2019-01-01 00:00:00<CR>
$R: epl, 2019-01-01 00:00:00
  PreserveLogging, 2019-01-01 00:00:00
COM1>
```

spoe	**setPreserveOnEvent**	***Cd***	*Event*	*TimeBefore*	*TimeAfter*					
gpoe	**getPreserveOnEvent**	*Cd*								
		+LOG1 ... LOG8 all	none +EventA +EventB +Command	0 . . . 15 . . . 1440 min	0 . . . 15 . . . 1440 min					

RxControl: Logging > Internal Logging Settings > Log Sessions

This command defines the criterium to use to preserve a file from automatic deletion. It is applicable to log sessions configured in `PreserveOnEvent` mode with the **`setLogSession`** command, and only to files logged in one of the IGS file naming modes.

Preserved files will not be deleted by the auto-delete functionality, and they will also be the last to be deleted when the receiver attempts to free-up disk space when the disk is full. Refer to the **`setLogSession`** command for further details.

The *Event* argument specifies the events that trigger files to be preserved. `EventA` and `EventB` refer to the two external event inputs of the receiver. `Command` refers to the on demand event log preserving command.

TimeBefore tells the receiver to preserve all the files that contain data for the last *TimeBefore* minutes before the event occurred.

TimeAfter tells the receiver to preserve all the files that contain data until *TimeAfter* minutes past the occurence of the event.

Note that only files that are logged under the IGS file naming can be preserved. Preserved files are marked as "`preserve=on`" when issuing the **`lstDiskInfo`** command. A list of events that occurred during logging can be seen with the **`lstDiskEvent`** command.

Example

If, for the LOG1 session, you want to preserve NMEA files that contain data at least 15 minutes before and 15 minutes after an EventA or EventB occurred, use:

```
COM1> sls, LOG1, Enabled, DSK1, "Example", After2Days, Medium,
   PreserveOnEvent <CR>
$R: sls, LOG1, Enabled, DSK1, "Example", After2Days, Medium,
   PreserveOnEvent
  LogSession, LOG1, Enabled, DSK1, "Example", After2Days, Medium,
     PreserveOnEvent
COM1> snlp, LOG1, IGS1H, "log", off <CR>
$R: snlp, LOG1, IGS1H, "log", off
  NMEALogging, LOG1, IGS1H, "log", off
COM1> spoe, LOG1, EventA+EventB <CR>
$R: spoe, LOG1, EventA+EventB
  PreserveOnEvent, LOG1, EventA+EventB, 15, 15
COM1>
```

lrf	**lstRecordedFile**	***Disk***	***FileName (60)***							
		DSK1 DSK2								

Use this command to retrieve the contents of one of the log files on the disk identified with the *Disk* argument. `DSK1` is the internal disk and `DSK2` is the external disk (USB drive).

The reply to this command consists in a succession of blocks starting with the "$−− BLOCK" header, and terminating with the pseudo-prompt "−−−−>" (see section Section 3.1.3, "Command Replies" for details). The decoding program must remove these headers and pseudo-prompts to recover the original file contents.

The download speed is highly influenced by the processor load. To speed up the download, it is recommended to stop the signal tracking, which can be done by typing the following command before starting the download: **`setSatelliteTracking, none`**.

The file download can be interrupted by sending ten uppercase "S" characters (simply by holding the "shift-S" key pressed) to the connection through which the download is taking place.

Examples

To output the contents of the internal log file named `log.sbf` on the first disk (DSK1), use:

```
COM1>  lrf, DSK1, log.sbf <CR>
$R; lrf, DSK1, log.sbf
... Here comes the content of log.sbf ...
COM1>
```

If the file `log.sbf` does not exist, an error is returned:

```
COM1>  lrf, DSK1, log.sbf <CR>
$R? lstRecordedFile: Argument ’FileName’ could not be handled!
COM1>
```

erf grf	**exeRemoveFile** **getRemoveFile**	***Disk***	***FileName (200)***							
		DSK1 DSK2	none all							

RxControl: Logging > Remove Internal File

Use this command to remove one file or an entire directory from the disk identified by the *Disk* argument. `DSK1` is the internal disk and `DSK2` is the external disk (USB drive).

If *FileName* is the name of a file, only that single file is removed from the disk. Files in a directory can be specified using dirname/filename.

If *FileName* is the name of a directory, the entire directory is deleted, except the file currently written to, if any.

If the reserved string `all` is used for the *FileName* argument, all files are removed from the selected disk, except the file currently written to, if any.

If there is no file nor directory named *FileName* on the disk or if the file is currently written to, an error message is returned.

Examples

To remove the file "`ATRX2980.03_`" from directory "`03298`", use:

```
COM1>  erf, DSK1, 03298/ATRX2980.03_ <CR>
$R: erf, DSK1, 03298/ATRX2980.03_
  RemoveFile, DSK1, "03298/ATRX2980.03_"
COM1>
```

To remove all files from DSK1, use:

```
COM1>  erf, DSK1, all <CR>
$R: erf, DSK1, all
  RemoveFile, DSK1, all
COM1>
```

srxl grxl	**setRINEXLogging** **getRINEXLogging**	***Cd*** *Cd*	*FileDuration*	*ObsInterval*	*SignalTypes*	*ExtraObsTypes*	*RINEXVersion*	*MixedNav*	*FileTypes*	*Archiving*
		+ LOG1 ... LOG8 all	none hour1 hour6 hour24 minute15	sec1 sec2 sec5 sec10 sec15 sec30 sec60	none + GPSL1CA + GPSL1PY + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL1P + GLOL2P + GLOL2CA + GLOL3 + GALL1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL1C + QZSL1S + QZSL5S + QZSL1CB + NAVICL5 all	none + Dx + Sx + Channel all	v211 v304ShortName v304 v305 v400	off on	+ Obs + Nav + Met all	none + Hatanaka + GroupInTar + Compress

RxControl: Logging > Internal RINEX Logging > RINEX Logging Options

Use this command to configure RINEX file logging in the log session identified with the *Cd* argument.

The argument *FileDuration* specifies whether a new RINEX file should be started every 15 minutes, every hour, 6 hours or every day. When *FileDuration* is set to `none`, RINEX logging is disabled and all following arguments are ignored.

ObsInterval specifies the interval of the observation records.

SignalTypes sets the list of signals to encode in the RINEX observation files. The more signals are selected, the bigger the RINEX files.

By default, the RINEX files contain the code and carrier phase observables. The *ExtraObsTypes* argument allows to also include the Doppler (obs code Dx), the C/N_0 (obs code Sx), or the channel number (obs code X).

The argument *RinexVersion* selects which RINEX version to use. `v304ShortName` generates a v3.04 file, but still using the DOS file naming convention of v2.xx.

The argument *MixedNav* specifies whether the navigation data is stored in separate files for each constellation (*MixedNav* set to `off`), or in a single mixed file (*MixedNav* set to `on`). This argument is ignored if *RINEXVersion* is `v211`. Note that QZSS and BeiDou navigation data are only available in the mixed navigation file.

By default, all types of RINEX files will be logged (observation, navigation and meteo) if the relevant data is available. This can be changed with the *FileTypes* argument.

The *Archiving* argument defines the type of post-processing (archiving and/or compression) to apply at the end of each *FileDuration* interval. `Hatanaka` enables Hatanaka compression of the observation file, `GroupInTar` groups all RINEX files (possibly after Hatanaka compression) into a `.tar` archive, and `Compress` enables gzip compression. The different options can be enabled individually, or combined with the "+" sign. The order of execution is `Hatanaka`, `GroupInTar` and, finally, `Compress`. After post-processing, the original files are deleted.

In the RINEX file names, the 4-character station identifier is the first four letters of the station code as set by the **`setMarkerParameters`** command. If the station code is empty, the first four letters of the marker name are used instead. The monument index, receiver index and country code needed to generate long file names (used by the receiver for RINEX v3.04 or above) are also set with the **`setMarkerParameters`** command.

If desired, it is also possible to add a log session ID prefix to all RINEX file names. This is enabled with the **`setGlobalFileNamingOptions`** command.

RINEX files are put in daily directories, the directory name being of the form `yyddd` with `yy` the 2-digit year and `ddd` the day of year.

If a RINEX file is currently being logged when issuing this command, the new settings will only be applied when the next RINEX file will be started. This occurs at a rate specified by *FileDuration*. To force the new settings to be immediately applied, RINEX logging must be temporarily stopped (*FileDuration* set to `none`) and then re-enabled. Changing the RINEX settings (e.g. changing the list of signals to be stored in RINEX) results in the past data to be overwritten in the RINEX file.

Example

To create daily RINEX files with the observation file containing only GPS L1CA data at a 30-s interval, and to enable Hatanaka and gzip compression, use:

```
COM1>  srxl, LOG1, hour24, sec30, GPSL1CA,,,,, Hatanaka+Compress <CR>
$R: srxl, LOG1, hour24, sec30, GPSL1CA,,,,, Hatanaka+Compress
  RINEXLogging, LOG1, hour24, sec30, GPSL1CA, none, v304, on, Obs+
     Nav+Met, Hatanaka+Compress
COM1>
```

smsl gmsl	**setRTCMMSMLogging** **getRTCMMSMLogging**	***Cd*** *Cd*	*FileDuration*	*Compression*	*Messages*	*ObsInterval*				
		+ LOG1 ... LOG8	none	off	none	msec100				
		all	hour1	on	+ RTCM1006	msec200				
			hour6		+ RTCM1008	msec500				
			hour24		+ RTCM1013	sec1				
			minute15		+ RTCM1019	sec2				
					+ RTCM1020	sec5				
					+ RTCM1033	sec10				
					+ RTCM1041	sec15				
					+ RTCM1042	sec30				
					+ RTCM1044	sec60				
					+ RTCM1045					
					+ RTCM1046					
					+ RTCM1074					
					+ RTCM1075					
					+ RTCM1076					
					+ RTCM1077					
					+ RTCM1084					
					+ RTCM1085					
					+ RTCM1086					
					+ RTCM1087					
					+ RTCM1094					
					+ RTCM1095					
					+ RTCM1096					
					+ RTCM1097					
					+ RTCM1104					
					+ RTCM1105					
					+ RTCM1106					
					+ RTCM1107					
					+ RTCM1114					
					+ RTCM1115					
					+ RTCM1116					
					+ RTCM1117					
					+ RTCM1124					
					+ RTCM1125					
					+ RTCM1126					
					+ RTCM1127					
					+ RTCM1134					
					+ RTCM1135					
					+ RTCM1136					
					+ RTCM1137					
					+ RTCM1230					
					+ MSM4					
					+ MSM5					
					+ MSM6					
					+ MSM7					
					+ Nav					
					+ Station					

RxControl: Logging > Internal Logging Settings > RTCM-MSM Logging and Upload

Use this command to configure the logging of RTCM-MSM messages in the log session identified with the *Cd* argument.

The *FileDuration* argument specifies whether a new RTCM-MSM file should be started every 15 minutes, every hour, 6 hours or every day. When *FileDuration* is set to `none`, RTCM logging is disabled and all following arguments are ignored.

If compression is enabled with the *Compression* argument, the RTCM files are compressed. Compression takes place at file completion and the original non-compressed file is deleted.

The *Messages* argument specifies the RTCM message types to be logged. A short description of the supported RTCM v3.x message types can be found in appendix D. `MSMi` enables the Multiple Signal Message - Type i from all constellations. `Nav` enables all navigation messages (ephemerides) and `Station` enables all station-related messages, as summarized in the table below.

Messages	Description
`MSM4`	+RTCM1074 +RTCM1084 +RTCM1094 +RTCM1104 +RTCM1114 +RTCM1124 +RTCM1134
`MSM5`	+RTCM1075 +RTCM1085 +RTCM1095 +RTCM1105 +RTCM1115 +RTCM1125 +RTCM1135
`MSM6`	+RTCM1076 +RTCM1086 +RTCM1096 +RTCM1106 +RTCM1116 +RTCM1126 +RTCM1136
`MSM7`	+RTCM1077 +RTCM1087 +RTCM1097 +RTCM1107 +RTCM1117 +RTCM1127 +RTCM1137
`Nav`	+RTCM1019 +RTCM1020 +RTCM1041 +RTCM1042 +RTCM1044 +RTCM1045 +RTCM1046
`Station`	+RTCM1006 +RTCM1008 +RTCM1013 +RTCM1033 +RTCM1230

The *ObsInterval* argument specifies the interval of the observation messages (RTCM1074 to RTCM1137). The ephemerides-related messages (eg. RTCM1019) are always logged on-change and the station messages are always logged once at the beginning of all files.

The RTCM-MSM file name follows the RINEX v3.02 convention, with the content characters being omitted and the format characters being set to `msm` (e.g. `SEPT00BEL_R_20173260000_06H_60S.msm`). The 4-character station identifier is the first four letters of the station code as set by the **`setMarkerParameters`** command. If the station code is empty, the first four letters of the marker name are used instead. The other elements of the file names (monument index, receiver index and country code) are also set with the **`setMarkerParameters`** command.

If desired, it is also possible to add a log session ID prefix to all RTCM-MSM file names. This is enabled with the **`setGlobalFileNamingOptions`** command.

RTCM-MSM files are put in daily directories, the directory name being of the form `yyddd` with `yy` the 2-digit year and `ddd` the day of year.

Example

To create daily RTCM-MSM files with the observation file containing all MSM7 messages at a 10-s interval, use:

```
COM1>  sr3l, LOG1, hour24, off, MSM7+Nav+Station, sec10 <CR>
$R: sr3l, LOG1, hour24, off, MSM7+Nav+Station, sec10
  RTCMMSMLogging, LOG1, Hour24, off, MSM7+Nav+Station, sec10
COM1>
```

3.2.22 FTP Push of Log Files

sbfp gbfp	**setBINEXFTP** **getBINEXFTP**	***Cd*** *Cd*	*Enable*	*Server (40)*	*Path (64)*	*User (20)*	*Password (40)*	*FTPPort*	*RetryIntrvl*	
		+LOG1 ... LOG8	off			anonymous		1 . . . 21 . . . 65535	off	
		all	on						min15	
									min30	
									hour1	
									hour6	
									day1	

RxControl: Logging > Internal Logging Settings > BINEX Logging and Upload

Use this command to specify where to FTP BINEX files logged in the log session identified by the *Cd* argument (BINEX FTP push).

The FTP push configuration is done for each session independently. The *Enable* arguments enables or disables FTP push for a given session. The next arguments specify the FTP server hostname or IP address, the path to the remote directory where to put the BINEX files, the login and password to use, and the FTP port. Note that the receiver encrypts the password so that it cannot be read back with the command **getBINEXFTP**.

The files are FTPed when they are complete, as prescribed by the *Duration* settings in the **setBINEXLoggingParameters** command.

If BINEX file compression is enabled with the **setBINEXLoggingParameters** command, the compressed files are FTPed.

The files are put in the remote directory specified in the *Path* argument. Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, `%j` with the day of the year (starting with `001`) and `%b` with the 3-letter month indication (e.g. Apr for April). To put a literal "`%`" in the path, use `%%`. After expansion, *Path* must not be longer than 80 characters.

If the directory specified in the *Path* argument does not exist on the remote server, it is created.

If the transfer or the directory creation fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the FTP retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **setLogSession**). Pending retries are also cancelled when the *Enable* argument is set to `off`, or when the *Server* is changed.

Example

```
COM1>  sbfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30<CR>
$R: sbfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30
  BINEXFTP, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
     min30
COM1>
```

scfp gcfp	**setCGGTTSFTP** **getCGGTTSFTP**	***Cd*** *Cd*	*Enable*	*Server (40)*	*Path (64)*	*User (20)*	*Password (40)*	*FTPPort*	*RetryIntrvl*	
		+LOG1 ... LOG8 all	off on			anonymous		1 . . . 21 . . . 65535	off min15 min30 hour1 hour6 day1	

RxControl: Logging > *Internal CGGTTS Logging* > *CGGTTS FTP Push Options*

Use this command to specify where to FTP CGGTTS files logged in the log session identified by the *Cd* argument (CGGTTS FTP push).

The FTP push configuration is done for each session independently. The *Enable* arguments enables or disables FTP push for a given session. The next arguments specify the FTP server hostname or IP address, the path to the remote directory where to put the CGGTTS files, the login and password to use, and the FTP port. Note that the receiver encrypts the password so that it cannot be read back with the command **`getCGGTTSFTP`**.

The CGGTTS files from a given day are FTPed within the first hour of the next day.

The files are put in the remote directory specified in the *Path* argument. Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, and `%j` with the day of the year (starting with `001`). To put a literal "`%`" in the path, use `%%`. After expansion, *Path* must not be longer than 80 characters.

If the directory specified in the *Path* argument does not exist on the remote server, it is created.

If the transfer or the directory creation fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the FTP retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Enable* argument is set to `off`, or when the *Server* is changed.

Example

```
COM1>  scfp, LOG1, on, tai.bipm.org, /data/UTC/xxxx/links/cggtts/,
   myname, mypwd, 21, min30<CR>
$R: scfp, LOG1, on, tai.bipm.org, /data/UTC/xxxx/links/cggtts/,
   myname, mypwd, 21, min30
  CGGTTSFTP, LOG1, on, tai.bipm.org, /data/UTC/xxxx/links/cggtts/,
     myname, mypwd, 21, min30
COM1>
```

efpt gfpt	**exeFTPPushTest** **getFTPPushTest**	***Server (40)***	***Path (64)***	***User (20)***	***Password (40)***	***FTPPort***				
				anonymous		1 . . . 21 . . . 65535				

RxControl: Logging > Test FTP Push to Server

Use this command to test write access to a FTP server.

The arguments specify the FTP server hostname or IP address, the path to the remote directory where write access will be tested, and the login and password to use. The *Path* may contain variable fields as explained in the **`setSBFFTP`** command.

Upon receiving this command, the receiver tries to connect to the specified FTP server and to write a short file in the directory specified with the *Path* argument. The file is then deleted from the FTP server.

The process can take several minutes depending on the network latency. At the end, a report message is posted in the activity log. This message can be retrieved with the **`lstInternalFile, RxMessages`**, and is also available in the `RxMessage` SBF block.

Example

```
COM1>  efpt, myftp.com, mydata/%Y%m%d, myname, mypwd, 21<CR>
$R: efpt, myftp.com, mydata/%Y%m%d, myname, mypwd, 21
  FTPPushTest, myftp.com, mydata/%Y%m%d, myname, mypwd, 21
COM1>
```

snfp gnfp	**setNMEAFTP** **getNMEAFTP**	***Cd*** *Cd*	*Enable*	*Server (40)*	*Path (64)*	*User (20)*	*Password (40)*	*FTPPort*	*RetryIntrvl*	
		+LOG1 ... LOG8 all	off on			anonymous		1 . . . 21 . . . 65535	off min15 min30 hour1 hour6 day1	

RxControl: Logging > Internal Logging Settings > NMEA Logging and Upload

Use this command to specify where to FTP NMEA files logged in the log session identified by the *Cd* argument (NMEA FTP push).

The FTP push configuration is done for each session independently. The *Enable* arguments enables or disables FTP push for a given session. The next arguments specify the FTP server hostname or IP address, the path to the remote directory where to put the NMEA files, the login and password to use, and the FTP port. Note that the receiver encrypts the password so that it cannot be read back with the command **getNMEAFTP**.

FTP push is only available in IGS file naming mode (see the *NamingType* argument of the **setNMEALogging** command). Each time an NMEA file is ready in a given log session, it is FTPed to the specified server. For example, in `IGS1H` file naming mode, files are FTPed every hour.

If NMEA file compression is enabled with the **setNMEALogging** command, the compressed files are FTPed.

The files are put in the remote directory specified in the *Path* argument. Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, `%j` with the day of the year (starting with `001`) and `%b` with the 3-letter month indication (e.g. Apr for April). To put a literal "`%`" in the path, use `%%`. After expansion, *Path* must not be longer than 80 characters.

If the directory specified in the *Path* argument does not exist on the remote server, it is created.

If the transfer or the directory creation fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the FTP retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **setLogSession**). Pending retries are also cancelled when the *Enable* argument is set to `off`, or when the *Server* is changed.

Example

```
COM1> snfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30<CR>
$R: snfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30
  NMEAFTP, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
     min30
COM1>
```

srfp grfp	**setRINEXFTP** **getRINEXFTP**	***Cd*** *Cd*	*Enable*	*Server (40)*	*Path (64)*	*User (20)*	*Password (40)*	*FTPPort*	*RetryIntrvl*	
		+LOG1 ... LOG8 all	off on			anonymous		1 . . . 21 . . . 65535	off min15 min30 hour1 hour6 day1	

RxControl: Logging > Internal RINEX Logging > RINEX FTP Push Options

Use this command to specify where to FTP RINEX files logged in the log session identified by the *Cd* argument (RINEX FTP push).

The FTP push configuration is done for each session independently. The *Enable* arguments enables or disables FTP push for a given session. The next arguments specify the FTP server hostname or IP address, the path to the remote directory where to put the RINEX files, the login and password to use, and the FTP port. Note that the receiver encrypts the password so that it cannot be read back with the command **`getRINEXFTP`**.

The RINEX files are FTPed when they are complete, as prescribed by the *FileDuration* settings in the **`setRINEXLogging`** command.

If RINEX file compression is enabled with the **`setRinexLogging`** command, the compressed files are FTPed.

The files are put in the remote directory specified in the *Path* argument. Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, and `%j` with the day of the year (starting with `001`) and `%b` with the 3-letter month indication (e.g. Apr for April). To put a literal "`%`" in the path, use `%%`. After expansion, *Path* must not be longer than 80 characters.

If the directory specified in the *Path* argument does not exist on the remote server, it is created.

If the transfer or the directory creation fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the FTP retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Enable* argument is set to `off`, or when the *Server* is changed.

Example

```
COM1> srfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30<CR>
$R: srfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30
  RINEXFTP, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
    min30
COM1>
```

smfp gmfp	**setRTCMMSMFTP** **getRTCMMSMFTP**	***Cd*** *Cd*	*Enable*	*Server (40)*	*Path (64)*	*User (20)*	*Password (40)*	*FTPPort*	*RetryIntrvl*	
		+LOG1 ... LOG8 all	off on			anonymous		1 . . . 21 . . . 65535	off min15 min30 hour1 hour6 day1	

RxControl: Logging > Internal Logging Settings > RTCM-MSM Logging and Upload

Use this command to specify where to FTP RTCM-MSM files logged in the log session identified by the *Cd* argument.

The FTP push configuration is done for each session independently. The *Enable* arguments enables or disables FTP push for a given session. The next arguments specify the FTP server hostname or IP address, the path to the remote directory where to put the RTCM-MSM files, the login and password to use, and the FTP port. Note that the receiver encrypts the password so that it cannot be read back with the command **`getRTCMMSMFTP`**.

The files are FTPed when they are complete, as prescribed by the *FileDuration* settings in the **`setRTCMMSMLogging`** command.

If file compression is enabled with the **`setRTCMMSMLogging`** command, the compressed files are FTPed.

The files are put in the remote directory specified in the *Path* argument. Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, `%j` with the day of the year (starting with `001`) and `%b` with the 3-letter month indication (e.g. Apr for April). To put a literal "`%`" in the path, use `%%`. After expansion, *Path* must not be longer than 80 characters.

If the directory specified in the *Path* argument does not exist on the remote server, it is created.

If the transfer or the directory creation fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the FTP retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Enable* argument is set to `off`, or when the *Server* is changed.

Example

```
COM1>  smfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30<CR>
$R: smfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30
  RTCMMSMFTP, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd,
     21, min30
COM1>
```

ssfp gsfp	**setSBFFTP** **getSBFFTP**	***Cd*** *Cd*	*Enable*	*Server (40)*	*Path (64)*	*User (20)*	*Password (40)*	*FTPPort*	*RetryIntrvl*	
		+LOG1 ... LOG8	off			anonymous		1 . . . 21 . . . 65535	off	
		all	on						min15	
									min30	
									hour1	
									hour6	
									day1	

RxControl: Logging > Internal Logging Settings > SBF Logging and Upload

Use this command to specify where to FTP SBF files logged in the log session identified by the *Cd* argument (SBF FTP push).

The FTP push configuration is done for each session independently. The *Enable* arguments enables or disables FTP push for a given session. The next arguments specify the FTP server hostname or IP address, the path to the remote directory where to put the SBF files, the login and password to use, and the FTP port. Note that the receiver encrypts the password so that it cannot be read back with the command **`getSBFFTP`**.

FTP push is only available in IGS file naming mode (see the **`setFileNaming`** command). Each time an SBF file is ready in a given log session, it is FTPed to the specified server. For example, in `IGS1H` file naming mode, files are FTPed every hour.

If SBF file compression is enabled with the **`setFileNaming`** command, the compressed files are FTPed.

The files are put in the remote directory specified in the *Path* argument. Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, `%j` with the day of the year (starting with `001`) and `%b` with the 3-letter month indication (e.g. Apr for April). To put a literal "`%`" in the path, use `%%`. After expansion, *Path* must not be longer than 80 characters.

If the directory specified in the *Path* argument does not exist on the remote server, it is created.

If the transfer or the directory creation fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the FTP retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Enable* argument is set to `off`, or when the *Server* is changed.

Example

```
COM1> ssfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30<CR>
$R: ssfp, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
   min30
  SBFFTP, LOG1, on, myftp.com, mydata/%Y%m%d, myname, mypwd, 21,
     min30
COM1>
```

3.2.23 CloudIt Configuration

lal	lstAuthorizationLinkCloudIt	*Server*								
		+Cloud1 +Cloud2 ... Cloud8 all								

CloudIt is a beta feature and the command description below is subject to change.

Use this command to get the authorization link of a specific CloudIt server defined by the *Server* argument. The selected server should be already configured using the **`setCloudItConfig`**. You need to visit the given link through a web browser.

If it is your first time to login, you should get a login web page. After entering your credentials and logging in, you should be directed to a "consent screen" where you give your consent to the receiver to access your storage server (you may not get this "consent screen" if your server administrator has configured the authorization server to skip this step).

The next page should display a code that you will need to copy back to the receiver using the **`exeAuthorizeCloudIt`** command.

The full workflow of CloudIt is described in section 1.24.

Example

To ouput the authorization link of `Cloud1`, use:

```
COM1>  lstAuthorizationLinkCloudIt, Cloud1 <CR>
$R; lstAuthorizationLinkCloudIt, cloud1
---->
$-- BLOCK 1 / 1
http://myauthurl.com/auth?client_id=myclientid&scope=readwrite&
   response_type=code
COM1>
```

sbci gbci	**setBINEXCloudIt** **getBINEXCloudIt**	***Cd*** *Cd*	*Server*	*Path (64)*	*Retry*					
		+ LOG1 ... LOG8 all	off Cloud1 ... Cloud8		off min15 min30 hour1 hour6 day1					

RxControl: Logging > Internal Logging Settings > BINEX Logging and Upload

CloudIt is a beta feature and the command description below is subject to change.

Use this command to specify where to upload BINEX files logged in the log session identified by the *Cd* argument (BINEX CloudIt). CloudIt configuration is done for each session independently. The *Server* argument enables CloudIt for a given log session and specifies to which server the BINEX files will be uploaded to. The selected *Server* must have been already configured using **setCloudItConfig** command before enabling CloudIt for a given session. If a non-configured *Server* is selected, the upload attempt will fail. To disable CloudIt for a given session, the *Server* argument should be set to `off`.

The next argument specifies the path to the remote directory where to put the BINEX files in the remote server.

The RINEX files are uploaded when they are complete, as prescribed by the *FileDuration* settings in the **setBINEXLoggingParameters** command.

If BINEX file compression is enabled with the **setBINEXLoggingParameters** command, the compressed files are uploaded. The files are put in the remote directory specified in the *Path* argument.

Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, and `%j` with the day of the year (starting with 001). To put a literal "`%`" in the path, use `%%`. After expansion, Path must not be longer than 80 characters.

The *Path* argument is sent as a parameter to the *Server* according to the API description, and it is up to the server to manage the folder creation.

If the transfer fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the CloudIt retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **setLogSession**). Pending retries are also cancelled when the *Server* is changed or set to `off`.

Example

```
COM1>  sbci, LOG1, Cloud1, newfolder, min30<CR>
$R: sbci, LOG1, Cloud1, newfolder, min30
  BINEXCloudIt, LOG1, Cloud1, newfolder, min30
COM1>
```

scic gcic	**setCloudItConfig** **getCloudItConfig**	***Server*** *Server*	*Name (32)*	*ClientID (128)*	*ClientSecret (128)*	*AuthURL (255)*	*TokenURL (255)*	*Endpoint (255)*	*Scope (64)*	*RedirectURI (255)*
		+ Cloud1 ... Cloud8 all								

RxControl: Logging > CloudIt > CloudIt Servers

CloudIt is a beta feature and the command description below is subject to change.

As described in the section 1.24, CloudIt uses the OAuth2 protocol to authorize a user to upload logged files to a resource server.

Use this command to configure a CloudIt server identified by the *Server* argument. The *Server* argument defines where the logged files will be uploaded.

The *Name* argument is optional and helps the user to identify the server during the configuration process.

All the next arguments are required. The *ClientID* and *ClientSecret* arguments identify the application to the authentication server as described in section 1.24. Note that the receiver encrypts the *ClientSecret* so that it cannot be read back with the command **`getCloudItConfig`**.

The *AuthURL* is the Authorization Endpoint. The *TokenURL* is the token endpoint that the receiver will use to obtain tokens. The *Endpoint* argument is the upload endpoint in the resource server where the files will be stored. The *RedirectURI* (optional) is the URI where the user will be redirected to after a successfull authentication to get the authorization code. All the endpoints should be valid http(s):// URLs.

The *Scope* defines the access scope of the receiver to the resource server. For example, if SBF files will be uploaded, the receiver should have the access to write files to the remote directory defined by the *Path* argument in **`setSBFCloudIt`**. If the receiver does not have the right to write to the remote directory, the upload attempts will fail.

When all the arguments are all set, the next step is authentication using the link given by the **`lstAuthorizationLinkCloudIt`** command.

The full workflow of CloudIt is described in section 1.24.

Example

```
COM1>  scic, Cloud1, MyServerName, myclientid, mysecret,
    http://www.mysauthserver.com/oauth2/auth/,
    http://www.mysauthserver.com/oauth2/token/,
    http://www.resourceserver.com/api/upload/, readwrite,
    http://www.myserver.com/api/<CR>
$R: scic, Cloud1, MyServerName, myclientid, mysecret, http://www.
    mysauthserver.com/oauth2/auth/, http://www.mysauthserver.com/
    oauth2/token/, http://www.resourceserver.com/api/upload/,
    readwrite, http://www.myserver.com/api/
   CloudItConfig, Cloud1, MyServerName, myclientid, mysecret, http
      ://www.mysauthserver.com/oauth2/auth/, http://www.
      mysauthserver.com/oauth2/token/, http://www.resourceserver.com
      /api/upload/, readwrite, http://www.myserver.com/api/
```

```
COM1>
```

edci gdci	**exeDisconnectCloudIt** **getDisconnectCloudIt**	***Server***								
		Cloud1 ... Cloud8								

RxControl: Logging > CloudIt > Disconnect from CloudIt Server

⚠ CloudIt is a beta feature and the command description below is subject to change.

Use this command to disconnect the receiver from the resource server defined by the *Server* argument.

When the receiver is disconnected, it cannot upload files to the server anymore. If this command is used while a file is being uploaded, the upload will continue until it succeeds or fails. In case it succeeds, no additional files are uploaded. In case it fails, the next upload attempts for the same file will fail and the file will not be uploaded.

Example

```
COM1>  edci, Cloud1<CR>
$R: edci, Cloud1
  DisconnectCloudIt, Cloud1
COM1>
```

erac	**exeAuthorizeCloudIt**	***Server***	***AuthCode (1024)***							
		Cloud1 Cloud2 ... Cloud8								

⚠ CloudIt is a beta feature and the command description below is subject to change.

Use this command to authorize the receiver to access the resource server by setting the authorization code *Authcode* to the specified server defined by the *Server* argument.

The authorization code *Authcode* lifetime is usually short and depends on the admin configuration of the authorization server. If the lifetime of the authorization code is exceeded before executing **`exeAuthorizeCloudIt`**, the authorization will fail.

If an *Authcode* has been already used for a successful authorization, it cannot be used anymore. If the server needs to be reauthorized, the server will need a new code. This new code is generated by revisiting the link provided by the **`lstAuthorizationLinkCloudIt`**.

The process can take several minutes depending on the network latency. At the end, a report message is posted in the activity log. The result of the authorization attempt can be found in the receiver messages with the **`lstInternalFile, RxMessages`** command, and is also available in the `RxMessage` SBF block.

Example

```
COM1>  erac, Cloud1, L2544MLKJLKJSDsd5444QSD<CR>
$R: erac, Cloud1, L2544MLKJLKJSDsd5444QSD
  exeAuthorizeCloudIt, Cloud1, L2544MLKJLKJSDsd5444QSD
COM1>
```

snci	**setNMEACloudIt**	***Cd***	*Server*	*Path (64)*	*Retry*					
gnci	**getNMEACloudIt**	*Cd*								
		+LOG1 ... LOG8 all	off Cloud1 ... Cloud8		off min15 min30 hour1 hour6 day1					

RxControl: Logging > *Internal Logging Settings* > *NMEA Logging and Upload*

CloudIt is a beta feature and the command description below is subject to change.

Use this command to specify where to upload NMEA files logged in the log session identified by the *Cd* argument (NMEA CloudIt). CloudIt configuration is done for each session independently. The *Server* argument enables CloudIt for a given log session and specifies to which server the NMEA files will be uploaded to. The selected *Server* must have been already configured using **`setCloudItConfig`** command before enabling CloudIt for a given session. If a non-configured *Server* is selected, the upload attempt will fail. To disable CloudIt for a given session, the *Server* argument should be set to `off`.

The next argument specifies the path to the remote directory where to put the NMEA files in the remote server.

CloudIt is only available in IGS file naming mode (see the setFileNaming command). Each time an NMEA file is ready in a given log session, it is uploaded with CloudIt to the specified server. For example, in `IGS1H` file naming mode, files are uploaded every hour.

If NMEA file compression is enabled with the **`setFileNaming`** command, the compressed files are uploaded. The files are put in the remote directory specified in the *Path* argument.

Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, and `%j` with the day of the year (starting with 001). To put a literal "`%`" in the path, use `%%`. After expansion, Path must not be longer than 80 characters.

The *Path* argument is sent as a parameter to the *Server* according to the API description, and it is up to the server to manage the folder creation. If the transfer fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the CloudIt retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Server* is changed or set to `off`.

Example

```
COM1> snci, LOG1, Cloud1, newfolder, min30<CR>
$R: snci, LOG1, Cloud1, newfolder, min30
  NMEACloudIt, LOG1, Cloud1, newfolder, min30
COM1>
```

srci grci	**setRINEXCloudIt** **getRINEXCloudIt**	***Cd*** *Cd*	*Server*	*Path (64)*	*Retry*					
		+LOG1 ... LOG8 all	off Cloud1 ... Cloud8		off min15 min30 hour1 hour6 day1					

RxControl: Logging > Internal RINEX Logging > RINEX CloudIt Options

⚠ CloudIt is a beta feature and the command description below is subject to change.

Use this command to specify where to upload RINEX files logged in the log session identified by the *Cd* argument (RINEX CloudIt). CloudIt configuration is done for each session independently. The *Server* argument enables CloudIt for a given log session and specifies to which server the RINEX files will be uploaded to. The selected *Server* must have been already configured using **`setCloudItConfig`** command before enabling CloudIt for a given session. If a non-configured *Server* is selected, the upload attempt will fail. To disable CloudIt for a given session, the *Server* argument should be set to `off`. The next argument specifies the path to the remote directory where to put the RINEX files in the remote server.

The RINEX files are uploaded when they are complete, as prescribed by the *FileDuration* settings in the **`setRinexLogging`** command.

If RINEX file compression is enabled with the **`setRinexLogging`** command, the compressed files are uploaded. The files are put in the remote directory specified in the *Path* argument.

Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, and `%j` with the day of the year (starting with 001). To put a literal "`%`" in the path, use `%%`. After expansion, Path must not be longer than 80 characters.

The *Path* argument is sent as a parameter to the *Server* according to the API description, and it is up to the server to manage the folder creation.

If the transfer fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the CloudIt retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Server* is changed or set to `off`.

Example

```
COM1>  srci, LOG1, Cloud1, newfolder, min30<CR>
$R: srci, LOG1, Cloud1, newfolder, min30
  RINEXCloudIt, LOG1, Cloud1, newfolder, min30
COM1>
```

srmi grmi	**setRTCMMSMCloudIt** **getRTCMMSMCloudIt**	***Cd*** *Cd*	*Server*	*Path (64)*	*Retry*					
		+LOG1 ... LOG8	off		off					
		all	Cloud1 ... Cloud8		min15					
					min30					
					hour1					
					hour6					
					day1					

RxControl: Logging > Internal Logging Settings > RTCM-MSM Logging and Upload

⚠ CloudIt is a beta feature and the command description below is subject to change.

Use this command to specify where to upload RTCM-MSM files logged in the log session identified by the *Cd* argument (RTCM-MSM CloudIt). CloudIt configuration is done for each session independently. The *Server* argument enables CloudIt for a given log session and specifies to which server the RTCM-MSM files will be uploaded. The selected *Server* must have been already configured using **`setCloudItConfig`** command before enabling CloudIt for a given session. If a non-configured *Server* is selected, the upload attempt will fail. To disable CloudIt for a given session, the *Server* argument should be set to `off`.

The next argument specifies the path to the remote directory where to put the RTCM-MSM files in the remote server.

The files are uploaded when they are complete, as prescribed by the *FileDuration* settings in the **`setRTCMMSMLogging`** command.

If RTCM-MSM file compression is enabled with the **`setRTCMMSMLogging`** command, the compressed files are uploaded. The files are put in the remote directory specified in the *Path* argument.

Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, and `%j` with the day of the year (starting with 001). To put a literal "`%`" in the path, use `%%`. After expansion, Path must not be longer than 80 characters.

The *Path* argument is sent as a parameter to the *Server* according to the API description, and it is up to the server to manage the folder creation.

If the transfer fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the CloudIt retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state (see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Server* is changed or set to `off`.

Example

```
COM1> srmi, LOG1, Cloud1, newfolder, min30<CR>
$R: srmi, LOG1, Cloud1, newfolder, min30
  RTCMMSMCloudIt, LOG1, Cloud1, newfolder, min30
COM1>
```

ssci gsci	**setSBFCloudIt** **getSBFCloudIt**	***Cd*** *Cd*	*Server*	*Path (64)*	*Retry*					
		+LOG1 ... LOG8 all	off Cloud1 ... Cloud8		off min15 min30 hour1 hour6 day1					

RxControl: Logging > Internal Logging Settings > SBF Logging and Upload

CloudIt is a beta feature and the command description below is subject to change.

Use this command to specify where to upload SBF files logged in the log session identified by the *Cd* argument (SBF CloudIt). CloudIt configuration is done for each session independently. The *Server* argument enables CloudIt for a given log session and specifies to which server the SBF files will be uploaded. The selected *Server* must have been already configured using **`setCloudItConfig`** command before enabling CloudIt for a given session. If a non-configured *Server* is selected, the upload attempt will fail. To disable CloudIt for a given session, the *Server* argument should be set to `off`.

The next arguments specify the path to the remote directory where to put the SBF files in the remote server.

CloudIt is only available in IGS file naming mode (see the setFileNaming command). Each time an SBF file is ready in a given log session, it is uploaded with CloudIt to the specified server. For example, in `IGS1H` file naming mode, files are uploaded every hour.

If SBF file compression is enabled with the **`setFileNaming`** command, the compressed files are uploaded. The files are put in the remote directory specified in the *Path* argument.

Special character sequences can be used to encode the file date in the path: `%y` is replaced with the 2-digit year, `%Y` with the 4-digit year, `%m` with the month, `%d` with the day of the month, and `%j` with the day of the year (starting with 001). To put a literal "`%`" in the path, use `%%`. After expansion, Path must not be longer than 80 characters.

The *Path* argument is sent as a parameter to the *Server* as described in section 1.24.2.2, and it is up to the server to manage the folder creation.

If the transfer fails, the receiver will periodically retry at a rate specified with the *RetryIntrvl* argument. The receiver will periodically retry the last 80 failed transfers for each log session. Setting the *RetryIntrvl* argument to `off` disables the CloudIt retries. Periodic retries continue for log sessions in disabled state, but are cancelled for log sessions in unused state(see the *State* argument of **`setLogSession`**). Pending retries are also cancelled when the *Server* is changed or set to `off`.

Example

```
COM1> ssci, LOG1, Cloud1, newfolder, min30<CR>
$R: ssci, LOG1, Cloud1, newfolder, min30
  SBFCloudIt, LOG1, Cloud1, newfolder, min30
COM1>
```

3.2.24 MSS/L-Band Configuration

llbb	**lstLBandBeams**										

Use this command to retrieve the list of user-defined and auto-defined L-Band beams.

The list contains user-defined beams (`User1`, `User2`,...) defined with the **`setLBandBeams`** command and service-specific beams which are automatically updated by the L-Band service provider. Only the enabled user-defined beams are shown.

For each beam, the list contains the beam carrier frequency in Hz, the baud rate, the beam name and the region code. For service-specific beams, the satellite longitude in degrees (from -180 to 180, positive east of Greenwich) and the grant status are also provided. The last entry shows the SVID to which the beam is mapped.

This command is very similar to the command **`getLBandBeams`**, the only difference being that the latter only reports the list of user-defined beams.

Example

```
COM1>  llbb <CR>
$R; llbb
---->
$-- BLOCK 1 / 1
LBandBeams, User1, 1535165000, baud1200, "User", "E", Enabled, L13
...
COM1>
```

slbb glbb	**setLBandBeams** **getLBandBeams**	***Beam*** *Beam*	*Frequency*	*Rate*	*Name (8)*	*Region (8)*	*Usage*			
		+ User1 + User2 all	1525000000 . . . 1559000000 Hz	baud600 baud1200 baud2400 baud4800	Unknown	Unknown	Disabled Enabled			

RxControl: L-band > Generic L-Band Settings > Satellite Beam Configuration

This command can be used to define/inquire the parameters of user-defined L-Band beams. A beam is characterized by its frequency and baud rate (the *Frequency* and *Rate* arguments). Optionally, a beam name and region ID can also be associated to each beam, for information only. A beam can be enabled or disabled, as set by the *Usage* argument. Only enabled beams can be locked to.

Example

```
COM1>  slbb, User1, 1537460000, baud1200, 25East, E, Enabled <CR>
$R: slbb, User1, 1537460000, baud1200, 25East, E, Enabled
  LBandBeams, User1, 1537460000, baud1200, "25East", "E", Enabled
COM1>
```

slcs glcs	**setLBandCustomServiceID** **getLBandCustomServiceID**	*ServiceID (4)*	*ScramblingVector*	*NDAUsage*						
		0000	0000	off on						

RxControl: L-band > Generic L-Band Settings > Satellite Beam Configuration

This command can be used to define the Service ID, scrambling vector and Null-Data-Algorithm (NDA) usage of the L-Band service provider. The *ServiceID* and *ScramblingVector* are 4-digit hexadecimal numbers.

This command should only be used for test and maintenance purposes.

Example

```
COM1>  slcs, A5A5, 0101, on<CR>
$R: slcs, A5A5, 0101, on
  LBandCustomServiceID, A5A5, 0101, on
COM1>
```

slnd glnd	**setLBandNTRIPDelivery** **getLBandNTRIPDelivery**	*Cd*								
		none NTR1 NTR2 NTR3 DC1|SID08 DC2|SID08								

RxControl: L-band > Generic L-Band Settings > NTRIP Delivery

Use this command to enable reception of L-Band corrections over the NTRIP connection identified with the *Cd* argument.

The selected NTRIP connection (`NTRi`) must be configured in client mode with the **`setNtripSettings`** command.

The `DC1|...` and `DC2|...` connections should normally not be used. They are reserved for applications where the NTRIP client is running outside the receiver and the L-Band corrections are injected into one of the internal daisy-chain pipes (see the **`setDataInOut`** command for details on the daisy-chain pipes). In that case, the NTRIP mountpoint (SID08) must also be specified.

Example

```
COM1>  slnd, NTR1<CR>
$R: slnd, NTR1
  LBandNTRIPDelivery, NTR1
COM1>
```

slsm glsm	**setLBandSelectMode** **getLBandSelectMode**	*Mode*	*Service*	*Beam1*	*Beam2*					
		auto	LBAS1	User1	User1					
		off	Inmarsat	User2	User2					
		manual		LBAS1|1	LBAS1|1					
				LBAS1|2	LBAS1|2					
				LBAS1|3	LBAS1|3					
				LBAS1|4	LBAS1|4					
				LBAS1|5	LBAS1|5					
				LBAS1|6	LBAS1|6					
				LBAS1|7	LBAS1|7					
				LBAS1|8	LBAS1|8					
				LBAS1|9	LBAS1|9					
				LBAS1|10	LBAS1|10					
				LBAS1|11	LBAS1|11					
				LBAS1|12	LBAS1|12					
				LBAS1|13	LBAS1|13					
				LBAS1|14	LBAS1|14					
				LBAS1|15	LBAS1|15					
				LBAS1|16	LBAS1|16					

RxControl: L-band > Generic L-Band Settings > Satellite Beam Configuration

This command can be used to define/inquire the main operation mode of the L-Band demodulator.

The following modes are available through the *Mode* argument:

Mode	**Description**
`auto`	The demodulator will try to lock to a visible beam, preferring beams to which access has been granted. The list of beams and their status can be retrieved by the command **`lstLBandBeams`**.
`off`	The demodulator will be disabled and will not attempt to lock to any beam.
`manual`	The demodulator will attempt to lock to the beams identified in the *Beami* arguments and ignore all other beams. The parameters of the beams (frequency and baud rate) can be retrieved by the command **`lstLBandBeams`**. Make sure that the beams identified in the *Beami* arguments are enabled (see command **`setLBandBeams`**).

The second argument *Service* specifies which service the demodulator has to lock to.

Example

```
COM1>  slsm, manual, Inmarsat, User1, User2<CR>
$R: slsm, manual, Inmarsat, User1, User2
  LBandSelectMode, manual, Inmarsat, User1, User2
COM1>
```

llrs	lstLBAS1RefStations									

Use this LBAS1-specific command to inquire the list of the reference stations of which corrections are available in the LBAS1 L-Band beam currently locked to. Usage of the corrections from a given reference station can be granted or not, as indicated in the list.

Example

```
COM1>  llrs <CR>
$R; llrs
---->
$-- BLOCK 1 / 1
LBAS1ReferenceStation, 0001, granted
LBAS1ReferenceStation, 0002, granted
LBAS1ReferenceStation, 0003, granted
LBAS1ReferenceStation, 0004, granted
LBAS1ReferenceStation, 0005, granted
LBAS1ReferenceStation, 0007, granted
LBAS1ReferenceStation, 0009, granted
LBAS1ReferenceStation, 0015, granted
LBAS1ReferenceStation, 0125, denied
LBAS1ReferenceStation, 0556, denied
...
COM1>
```

slrs glrs	**setLBAS1RefStations** **getLBAS1RefStations**	***Stream*** *Stream*	*StationID (255)*							
		+ RTCMV all	all							

RxControl: L-band > LBAS1-Specific Settings > Reference Stations

This LBAS1-specific command defines the set of reference stations from which differential corrections have to be demodulated from the L-Band beam and included in the decoded correction stream. Only one decoded stream is defined, the RTCMV stream. That stream is fed into the PVT algorithm and serves as source of differential corrections in DGPS-rover and in PPP positioning modes.

The argument *StationID* is a list of 4-digit reference station IDs, separated by the "+" or "-" sign. Use the keyword "none" to empty the selected decoded correction stream, and "all" to include all reference stations in the stream.

Examples

To restrain the decoded differential correction stream to reference stations 1, 2, 3 and 15, use:

```
COM1>  slrs, RTCMV, 0001+0002+0003+0015 <CR>
$R: slrs, RTCMV, 0001+0002+0003+0015
  LBAS1RefStations, RTCMV, "0001+0002+0003+0015"
COM1>
```

To add reference station 4 to the set, use:

```
COM1>  slrs, RTCMV, +0004 <CR>
$R: slrs, RTCMV, +0004
  LBAS1RefStations, RTCMV, "0001+0002+0003+0004+0015"
COM1>
```

To select all reference stations except the one with ID 0015, use:

```
COM1>  slrs, RTCMV, all-0015 <CR>
$R: slrs, RTCMV, all-0015
  LBAS1RefStations, RTCMV, "all-0015"
COM1>
```

To empty the RTCMV stream, use:

```
COM1>  slrs, RTCMV, none <CR>
$R: slrs, RTCMV, none
  LBAS1RefStations, RTCMV, "none"
COM1>
```

3.2.25 Cosmos Configuration

scoc gcoc	**setCosmosConfig** **getCosmosConfig**	*Enable*	*CustomerID (24)*							
		off on								

RxControl: Communication > Cosmos

⚠ Cosmos support is a beta feature and the command description below is subject to change.

Cosmos is a tool that makes it possible to visualize the health and status information of multiple GNSS receivers in a single dashboard, as well as to functionally manage these receivers.

To include the receiver in the Cosmos tool, among others the Cosmos service must be installed on the receiver, and this service must be correctly configured.

Use this command to configure the Cosmos service on the receiver.

The *Enable* argument determines whether or not the Cosmos service is executed on the receiver. The *CustomerID* argument is used to identify the customer in whose Cosmos dashboard the receiver must be included. This value is unique per customer and must be obtained from Septentrio.

Besides the configuration in this command, Cosmos certificate and key files must be uploaded to the receiver. These files must also be obtained from Septentrio and can be uploaded using the web interface (using the "Admin > About > Cosmos" page).

Please refer to the Cosmos documentation and/or to Septentrio sales/support for more information on deploying Cosmos.

Example

```
COM1>  scoc, on, customerx<CR>
$R: scoc, on, customerx
  CosmosConfig, on, customerx
COM1>
```

Chapter 4

SBF Reference

4.1 SBF Outline

SBF is the binary output format of Septentrio receivers. In this format, the data are arranged in binary blocks referred to as SBF blocks.

Each SBF block consists of a sequence of numeric or alphanumeric fields of different types and sizes. The total block size is always a multiple of 4 bytes.

The fields of an SBF block may have one of the following types:

Type	Description
u1	Unsigned integer on 1 byte (8 bits)
u2	Unsigned integer on 2 bytes (16 bits)
u4	Unsigned integer on 4 bytes (32 bits)
u8	Unsigned integer on 8 bytes (64 bits)
i1	Signed integer on 1 byte (8 bits)
i2	Signed integer on 2 bytes (16 bits)
i4	Signed integer on 4 bytes (32 bits)
i8	Signed integer on 8 bytes (64 bits)
f4	IEEE float on 4 bytes (32 bits)
f8	IEEE float on 8 bytes (64 bits)
c1[X]	String of X ASCII characters, right padded with bytes set to 0 if needed.

Each multi-byte binary type is transmitted as little-endian, meaning that the least significant byte is the first one to be transmitted by the receiver. Signed integers are coded as two's complement.

Every SBF block begins with an 8-byte block header, which is followed by the block body.

4.1.1 SBF Block Header Format

Every SBF block starts with an 8-byte header having the following contents:

Parameter	Type	Description
`Sync`	c1[2]	The Sync field is a 2-byte array always set to 0x24, 0x40. The first byte of every SBF block has hexadecimal value 24 (decimal 36, ASCII '$'). The second byte of every SBF block has hexadecimal value 40 (decimal 64, ASCII '@'). These two bytes identify the beginning of any SBF block and can be used for synchronization.
`CRC`	u2	The CRC field is the 16-bit CRC of all the bytes in an SBF block from and including the ID field to the last byte of the block. The generator polynomial for this CRC is the so-called CRC-CCITT polynomial: $x^{16}+x^{12}+x^5+x^0$. The CRC is computed in the forward direction using a seed of 0, no reverse and no final XOR.
`ID`	u2	The ID field is a 2-byte block ID, which uniquely identifies the block type and its contents. It is a bit field with the following definition: bits 0-12: block number; bits 13-15: block revision number, starting from 0 at the initial block definition, and incrementing each time backwards-compatible changes are performed to the block (see section 4.1.6).
`Length`	u2	The Length field is a 2-byte unsigned integer containing the size of the SBF block. It is the total number of bytes in the SBF block including the header. It is always a multiple of 4.

4.1.2 SBF Block Names and Numbers

The structure and contents of an SBF block are unambiguously identified by the block ID. For easier readability, a block name is also defined for each block. When invoking the **`setSBFOutput`** command to enable a given block, the block name should be specified.

The list of SBF blocks available on your receiver can be found in Appendix A.

4.1.3 SBF Block Time Stamp (TOW and WNc)

Each SBF header is directly followed by a time stamp, which consists of two fields: `TOW` and `WNc`:

Parameter	Type	Units & Scale	Do-Not-Use	Description
		Factor	Value	
`TOW`	u4	0.001 s	4294967295	Time-Of-Week : Time-tag, expressed in whole milliseconds from the beginning of the current GPS week.
`WNc`	u2	1 week	65535	The GPS week number associated with the `TOW`. `WNc` is a continuous week count (hence the "c"). It is not affected by GPS week rollovers, which occur every 1024 weeks. By definition of the Galileo system time, `WNc` is also the Galileo week number plus 1024.

In the SBF time stamps, the definition of the week always follows the GPS convention even if the block contains data for another constellation. This means that WNc 0, TOW 0 corresponds to Jan 06,1980 at 00:00:00 UTC.

If the time-of-week or the week number is unknown, which is typically the case for a few seconds after start-up, the corresponding field is set to its Do-Not-Use value (see section 4.1.7). It does not mean that the SBF block is unusable, but simply that the receiver could not time-tag it. It is typical that the `TOW` field becomes valid before the WNc field.

The interpretation to give to the time stamp is block-dependent. Three types of time stamps are possible:

- *Receiver time stamp*: this type of time stamp is used for the SBF blocks containing synchronous data, i.e. data generated at a given epoch in the receiver time scale. Examples of such blocks are the measurement and PVT blocks (`MeasEpoch` and `PVTCartesian`). The time stamp is always a multiple of the output interval as specified by the **`setSBFOutput`** command (see also section 4.1.8). As soon as the receiver time is aligned with the GNSS time, the receiver time stamp is guaranteed to never decrease in successive SBF blocks.
- *SIS time stamp*: it is used for asynchronous blocks containing navigation message data from the signal-in-space. The time stamp corresponds to the time of transmission of the end of the last navigation bit used to build the SBF block. It always follows the GPS convention, as explained above.
- *External time stamp*: this type of time stamp is used for SBF blocks triggered by external asynchronous events, such as the `ExtEvent` block.

For the blocks with a SIS or an external time stamp, there is no strict relation between the time stamp of the SBF blocks and their order of transmission. For example, the SBF stream may contain a `GPSNav` block with ephemeris parameters received one hour in the past (i.e. the time stamp is one hour in the past) followed by another block with a current receiver time stamp.

4.1.4 Sub-blocks

Some blocks contain sub-blocks. For example, the `SatVisibility` block contains `N` `SatInfo` sub-blocks, each sub-block containing data for one particular satellite. Unless the size of the sub-blocks is mentioned explicitly in the block description, SBF blocks that contain sub-blocks also contain a `SBLength` field, which indicates the size of the sub-blocks in bytes.

4.1.5 Padding Bytes

Padding bytes are foreseen at the end of most SBF blocks and sub-blocks, so that their total size is equal to `Length` or `SBLength` respectively. The padding bytes are just placeholders and should not be looked at by the decoding software. Their value is not defined.

4.1.6 SBF Revision Number

Each SBF block has an associated revision number. The revision number is incremented each time a backwards-compatible change is implemented.

As described in section 4.1.1, the block number is to be found in bits 0 to 12 of the `ID` field, and the revision is in bits 13 to 15 of that field.

A backwards-compatible change consists of adding one or more fields in the padding bytes, or in the fields marked as "reserved" in the block description. Such change should be unnoticed by properly written decoding software that ignore the contents of padding and reserved fields (see also section 4.1.12). Each time such change happens, the revision number is incremented. The revision at which a given field has been introduced is documented in the block description in chapter 4.2, unless that revision is 0 (see the `ReceiverSetup` block as an example). It is guaranteed that if a given field exists in revision N, it will also exist in all revisions after N: no fields are withdrawn from SBF.

4.1.7 Do-Not-Use Value

It might happen that one or more pieces of data in an SBF block are not known at block creation time. For example, when there are insufficient satellite measurements to compute a position solution, the position components found in the `X`, `Y` and `Z` fields of the `PVTCartesian` block will not be available. To indicate that a given data item is not available or is currently not provided by the receiver, the corresponding field is set to a ’Do-Not-Use’ value that is never reached in normal operation.

When applicable, the Do-Not-Use value is mentioned in the block description. The Do-Not-Use value refers to the raw contents of the field, without applying the scale factor. A field set to its Do-Not-Use value should always be discarded by the decoding software.

4.1.8 Output Rate

In general, the default output rate for each SBF block is the renewal rate of the information. For instance, the `GPSNav` block is output each time a new ephemeris data set is received from a given GPS satellite. The default output rates of GNSS measurement blocks, PVT blocks and integrated INS/GNSS blocks depend on your permission set. These three rates can be checked by the command **`getReceiverCapabilities`**.

The default output rate is specified for each block in chapter 4.2. To instruct the receiver to output a given block at its default rate, the "`OnChange`" rate has to be specified in the **`setSBFOutput`** command.

Some blocks can only be output at their default rate (e.g. the `GPSNav` block). Others can be decimated to a user-selectable rate. A subset of blocks can also be output "once" using the **`exeSBFOnce`** command. This can be handy to get a one-shot overview of a particular receiver state. Whether a given block supports a user-selectable rate ("Flex Rate") and whether it belongs to the "output once" set is indicated in the SBF block list in Appendix A.

Attempting to force another rate than the default one for those blocks that do not support "Flex Rate" has no effect: those blocks are always output at their default rate.

4.1.9 Satellite ID and GLONASS Frequency Number

Satellites are identified by the `SVID` (or `PRN`) and `FreqNr` fields, defined as in the table below.

This table is only valid for the currently-supported constellations and signal types (see 4.1.10). To ensure compatibility with future SBF upgrades, decoding software must ignore SBF blocks and sub-blocks of which the satellite ID field or the signal number field is undefined in this document.

Field	Type	Do-Not-Use Value	Description	RINEX satellite code
`SVID or PRN`	u1	0	Satellite ID: The following ranges are defined:	
			1-37: PRN number of a GPS satellite	*Gnn* (*nn* = SVID)
			38-61: Slot number of a GLONASS satellite with an offset of 37 (R01 to R24)	*Rnn* (*nn* = SVID-37)
			62: GLONASS satellite of which the slot number is not known	NA
			63-68: Slot number of a GLONASS satellite with an offset of 38 (R25 to R30)	*Rnn* (*nn* = SVID-38)
			71-106: PRN number of a GALILEO satellite with an offset of 70	*Enn* (*nn* = SVID-70)
			107-119: L-Band (MSS) satellite. Corresponding satellite name can be found in the `LBandBeams` block.	NA
			120-140: PRN number of an SBAS satellite (S120 to S140)	*Snn* (*nn* = SVID-100)
			141-180: PRN number of a BeiDou satellite with an offset of 140	*Cnn* (*nn* = SVID-140)
			181-187: PRN number of a QZSS satellite with an offset of 180	*Jnn* (*nn* = SVID-180)
			191-197: PRN number of a NavIC/IRNSS satellite with an offset of 190 (I01 to I07)	*Inn* (*nn* = SVID-190)
			198-215: PRN number of an SBAS satellite with an offset of 57 (S141 to S158)	*Snn* (*nn* = SVID-157)
			216-222: PRN number of a NavIC/IRNSS satellite with an offset of 208 (I08 to I14)	*Inn* (*nn* = SVID-208)
			223-245: PRN number of a BeiDou satellite with an offset of 182 (C41 to C63)	*Cnn* (*nn* = SVID-182)
`FreqNr`	u1	0	GLONASS frequency number, with an offset of 8. It ranges from 1 (corresponding to an actual frequency number of -7) to 21 (corresponding to an actual frequency number of 13). For non-GLONASS satellites, `FreqNr` is reserved and must be ignored by the decoding software.	

4.1.10 Signal Type

Some sub-blocks contain a signal type field, which identifies the type of signal and modulation the sub-blocks applies to. The signal numbering is defined as follows:

Signal number	Signal Type	Constellation	Carrier frequency (MHz)	RINEX obs code
0	L1CA	GPS	1575.42	1C
1	L1P	GPS	1575.42	1W
2	L2P	GPS	1227.60	2W
3	L2C	GPS	1227.60	2L
4	L5	GPS	1176.45	5Q
5	L1C	GPS	1575.42	1L
6	L1CA	QZSS	1575.42	1C
7	L2C	QZSS	1227.60	2L
8	L1CA	GLONASS	1602.00+(`FreqNr`-8)*9/16, with `FreqNr` as defined in section 4.1.9.	1C
9	L1P	GLONASS	1602.00+(`FreqNr`-8)*9/16	1P
10	L2P	GLONASS	1246.00+(`FreqNr`-8)*7/16	2P
11	L2CA	GLONASS	1246.00+(`FreqNr`-8)*7/16	2C
12	L3	GLONASS	1202.025	3Q
13	B1C	BeiDou	1575.42	1P
14	B2a	BeiDou	1176.45	5P
15	L5	NavIC/IRNSS	1176.45	5A
16	Reserved			
17	E1	Galileo	1575.42	1C
18	Reserved			
19	E6	Galileo	1278.75	6C or 6B (*)
20	E5a	Galileo	1176.45	5Q
21	E5b	Galileo	1207.14	7Q
22	E5 AltBOC	Galileo	1191.795	8Q
23	LBand	MSS	L-band beam specific	NA
24	L1CA	SBAS	1575.42	1C
25	L5	SBAS	1176.45	5I
26	L5	QZSS	1176.45	5Q
27	L6	QZSS	1278.75	
28	B1I	BeiDou	1561.098	2I
29	B2I	BeiDou	1207.14	7I
30	B3I	BeiDou	1268.52	6I
31	Reserved			
32	L1C	QZSS	1575.42	1L
33	L1S	QZSS	1575.42	1Z
34	B2b	BeiDou	1207.14	7D
35-37	Reserved			
38 **(Tentative!)**	L1CB	QZSS	1575.42	1E
39	L5S	QZSS	1176.45	5P

(*) See the `E6B Used` bit in the `MeasEpoch` SBF block.

4.1.11 Channel Numbering

Some blocks contain a reference to the receiver channel number. Channel numbering starts at one. The maximum value for the channel number depends on the receiver type.

4.1.12 Decoding of SBF Blocks

In order to decode an SBF block, one has to identify the block boundaries in the data stream coming from the receiver. This involves searching for the initial "$@" characters that mark the beginning of each SBF block. Since the "$@" sequence can occur in the middle of an SBF block as well, additional checking is needed to make sure that a given "$@" is indeed the beginning of a block. The following procedure is recommended to decode SBF data stream.

1. Wait until the "$@" character sequence appears in the data stream from the receiver. When it is found, go to point 2.
2. Read the next two bytes. It should be the block CRC. Store this value for future reference.
3. Read the next two bytes and store them in a buffer. It should be the block ID.
4. Read the next two bytes and append them to the buffer. It should be the `Length` field of the SBF block. It should be a multiple of 4. If not, go back to point 1.
5. Read the next (`Length-8`) bytes and append them to the buffer. Compute the CRC of the buffer. The computed CRC should be equal to the CRC stored at point 2. If not, go back to point 1, else a valid SBF block has been detected and can be interpreted by the reading software.
6. If the block number (bits 0 to 12 of the `ID` field decoded at point 3) is of interest to your application, decode the SBF block.
7. Go back to point 1 and search for the new occurrence of the "$@" sequence after the end of the last byte of the block that was just identified.

To ensure compatibility with future upgrades of SBF, it is recommended that the decoding software observes the following rules:

- Only bits 0 to 12 of the `ID` field must be used to identify a block. Bits 13 to 15 represent the revision number.
- The lengths of SBF blocks and sub-blocks should not be considered constant and hard-coded in the decoding software. Instead, the decoding software must use the `Length` and `SBLength` fields encoded in the SBF block.
- Padding bytes should be ignored.
- Reserved fields and reserved bits in bit-fields should be ignored.
- SBF blocks or sub-blocks of which the satellite ID field or the signal number field is undefined in this document should be ignored (see section 4.1.9).

4.2 SBF Block Definitions

4.2.1 Measurement Blocks

GNSS observables are available in the following SBF blocks:

- the legacy `MeasEpoch` block, possibly complemented by `MeasExtra`.
- the `Meas3Ranges` block, possibly complemented by `Meas3Doppler` and `Meas3CN0HiRes`.

The `MeasEpoch` block contains pseudorange, carrier phase, C/N0 and Doppler observables. The `Meas3Ranges` block contains pseudoranges, carrier phases and C/N0, while Doppler is available in the companion `Meas3Doppler` block. The observable resolution is shown in the table below.

	`MeasEpoch`	`Meas3Ranges`
Pseudorange	1mm	1mm
Carrier phase	0.001cycles	0.001cycles
C/N0	0.25dB-Hz 0.03125dB-Hz with `MeasExtra`	1dB-Hz 0.0625dB-Hz with `Meas3CN0HiRes`
Doppler	0.0001Hz	No Doppler in `Meas3Ranges` 1mm/s with `Meas3Doppler`

The main advantage of the `Meas3` blocks is their reduced size compared to the `MeasEpoch` blocks. As an illustration, the following table shows the disk space required to log the different measurement-related blocks over one day at a 1-s interval. In this example, measurements from all GPS L1/L2/L5, GLONASS L1/L2, Galileo E1/E6/E5a/E5b and BeiDou B1/B2/B3 signals have been logged (constellation status as of beginning of 2017).

SBF Block	Disk space (1 day, 1 Hz)
`MeasEpoch`	104MB
`MeasExtra`	110MB
`Meas3Ranges`	28MB
`Meas3Doppler`	10MB
`Meas3CN0HiRes`	5MB

`MeasEpoch`	Number:	4027
	`"OnChange"` interval:	internal measurement rate (receiver-type dependent)

This block contains all the GNSS measurements (observables) taken at the time given by the `TOW` and `WNc` fields.

For each tracked signal, the following measurement set is available:

- the pseudorange
- the carrier phase
- the Doppler
- the C/N0
- the lock-time.

To decrease the block size, all the measurements from a given satellite are referenced to one master measurement set. For instance, the L2 pseudorange (C2) is not much different from the L1 pseudorange (C1), such that the difference between C2 and C1 is encoded, instead of the absolute value of C2.

This is done by using a two-level sub-block structure. All the measurements from a given satellite are stored in a `MeasEpochChannelType1` sub-block. The first part of this sub-block contains the master measurements, encoded as absolute values. The second part contains slave measurements, for which only the delta values are encoded in smaller `MeasEpochChannelType2` sub-blocks.

Every `MeasEpochChannelType1` sub-block contains a field "`N2`", which gives the number of nested `MeasEpochChannelType2` sub-blocks. If there is only one signal tracked for a given satellite, there are no slave measurements and `N2` is set to 0.

Decoding is done as follows:

1. Decode the master measurements and the `N2` value from the `MeasEpochChannelType1` sub-block.
2. If `N2` is not 0, decode the N2 nested `MeasEpochChannelType2` sub-blocks.
3. Go back to 1 till the `N1` `MeasEpochChannelType1` sub-blocks have been decoded.

⚠ Note that measurements in this block are scrambled if the "Measurement Availability" permission is not granted on your receiver. See also bit 7 of the `CommonFlags` field.

Parameter	Type	Units	Do-Not-Use	Description
Sync1	c1			Block Header, see 4.1.1
Sync2	c1			
CRC	u2			
ID	u2			
Length	u2	1 byte		
TOW	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
WNc	u2	1 week	65535	
N1	u1			Number of `MeasEpochChannelType1` sub-blocks in this `MeasEpoch` block.
SB1Length	u1	1 byte		Length of a `MeasEpochChannelType1` sub-block, excluding the nested `MeasEpochChannelType2` sub-blocks
SB2Length	u1	1 byte		Length of a `MeasEpochChannelType2` sub-block
CommonFlags	u1			Bit field containing flags common to all measurements. Bit 0: Multipath mitigation: if this bit is set, multipath mitigation is enabled. (see the **setMultipathMitigation** command). Bit 1: Smoothing of code: if this bit is set, at least one of the code measurements are smoothed values (see **setSmoothingInterval** command). Bit 2: Reserved Bit 3: Clock steering: this bit is set if clock steering is active (see **setClockSyncThreshold** command). Bit 4: Not applicable. Bit 5: High dynamics: this bit is set when the receiver is in high-dynamics mode (either on request of the user using the **setReceiverDynamics, high** command, or based on the receiver's built-in high-dynamics detection algorithms). Bit 6: E6B used: this bit is set if the Galileo E6 measurements in this block are obtained using the E6B signal instead of the default E6C signal. This happens when E6C encryption is active on at least one of the Galileo satellites. Bit 7: Scrambling: bit set when the measurements are scrambled. Scrambling is applied when the "Measurement Availability" permission is not granted (see the **lif,Permissions** command).
CumClkJumps	u1	0.001 s		Cumulative millisecond clock jumps since start-up, with an ambiguity of k*256 ms. For example, if two clock jumps of -1 ms have occurred since startup, this field contains the value 254.
Reserved	u1			Reserved for future use, to be ignored by decoding software
Type1	...	...		*A succession of N1* `MeasEpochChannelType1` *sub-blocks, see definition below*
Padding	u1[..]			Padding bytes, see 4.1.5

Rev 1

`MeasEpochChannelType1` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`RxChannel`	u1			Receiver channel on which this satellite is currently tracked (see 4.1.11).
`Type`	u1			Bit field indicating the signal type and antenna ID: Bits 0-4: `SigIdxLo`: if not 31, this is the signal number (see 4.1.10), otherwise the signal number can be found in the `ObsInfo` field below. Bits 5-7: Antenna ID: 0 for main, 1 for *Aux1* and 2 for *Aux2*
`SVID`	u1			Satellite ID, see 4.1.9
`Misc`	u1	4294967.296 m	0 (1)	Bit field containing the MSB of the pseudorange. Bits 0-3: `CodeMSB`: MSB of the pseudorange (this is an unsigned value). Bits 4-7: Reserved
`CodeLSB`	u4	0.001 m	0 (1)	LSB of the pseudorange. The pseudorange expressed in meters is computed as follows: PR_{type1}[m] = (`CodeMSB`*4294967296+`CodeLSB`)*0.001 where `CodeMSB` is part of the `Misc` field.
`Doppler`	i4	0.0001 Hz	−2147483648	Carrier Doppler (positive for approaching satellites). To compute the Doppler in Hz, use: D_{type1}[Hz] = `Doppler`*0.0001
`CarrierLSB`	u2	0.001 cycles	0 (2)	LSB of the carrier phase relative to the pseudorange
`CarrierMSB`	i1	65.536 cycles	−128 (2)	MSB of the carrier phase relative to the pseudorange. The full carrier phase can be computed by: L[cycles] = PR_{type1}[m]/λ +(`CarrierMSB`*65536+`CarrierLSB`)*0.001 where λ is the carrier wavelength corresponding to the frequency of the signal type in the `Type` field above: λ=299792458/f_L m, with f_L the carrier frequency as listed in section 4.1.10.
`CN0`	u1	0.25 dB-Hz	255	The C/N0 in dB-Hz is computed as follows, depending on the signal type in the `Type` field: C/N_0[dB-Hz] = `CN0`*0.25 if the signal number is 1 or 2 C/N_0[dB-Hz] = `CN0`*0.25+10 otherwise Users requiring a higher C/N0 resolution can use the `MeasExtra` SBF block. The `Misc` field of that block allows to extend the resolution to 0.03125dB-Hz.
`LockTime`	u2	1 s	65535	Duration of continuous carrier phase. The lock-time is reset at the initial lock of the phase-locked-loop, and whenever a loss of lock condition occurs. If the lock-time is longer than 65534s, it is clipped to 65534s. If the carrier phase measurement is not available, this field is set to its Do-Not-Use value.

`ObsInfo`	u1			Bit field: Bit 0: if set, the pseudorange measurement is smoothed Bit 1: Reserved Bit 2: this bit is set when the carrier phase (L) has a half-cycle ambiguity Bits 3-7: The interpretation of these bits depends on the value of `SigIdxLo` from the `Type` field. If `SigIdxLo` equals 31, these bits contain the signal number with an offset of 32 (see 4.1.10). For example, a value of 1 corresponds to signal number 33 (QZSS L1S). If `SigIdxLo` is 8, 9, 10 or 11, these bits contain the GLONASS frequency number with an offset of 8. For example, a value of 1 corresponds to frequency number -7. Otherwise, these bits are reserved.
`N2`	u1			Number of `MeasEpochChannelType2` sub-blocks contained in this `MeasEpochChannelType1` sub-block.
`Padding`	u1[..]			Padding bytes, see 4.1.5
Type2	...	...		*A succession of N2* `MeasEpochChannelType2` *sub-blocks, see definition below*

`MeasEpochChannelType2` sub-block definition:

Parameter	**Type**	**Units**	**Do-Not-Use**	**Description**
`Type`	u1			Bit field indicating the signal type and antenna ID: Bits 0-4: `SigIdxLo`: if not 31, this is the signal number (see 4.1.10), otherwise the signal number can be found in the `ObsInfo` field below. Bits 5-7: Antenna ID: 0 for main, 1 for *Aux1* and 2 for *Aux2*
`LockTime`	u1	1 s	255	See corresponding field in the `MeasEpochChannelType1` sub-block above, except that the value is clipped to 254 instead of 65534.
`CN0`	u1	0.25 dB-Hz	255	See corresponding field in the `MeasEpochChannelType1` sub-block above.
`OffsetsMSB`	u1	 65.536 m 6.5536 Hz	 −4 (3) −16 (4)	Bit field containing the MSB of the code and of the Doppler offsets with respect to the `MeasEpochChannelType1` sub-block. Bits 0-2: `CodeOffsetMSB`: MSB of the code offset. Bits 3-7: `DopplerOffsetMSB`: MSB of the Doppler offset. `CodeOffsetMSB` and `DopplerOffsetMSB` are coded as two's complement. Refer to the `CodeOffsetLSB` and `DopplerOffsetLSB` fields to see how to use this field.
`CarrierMSB`	i1	65.536 cycles	−128 (5)	MSB of the carrier phase relative to the pseudorange.
`ObsInfo`	u1			Bit field: Bit 0: if set, the pseudorange measurement is smoothed Bit 1: Reserved Bit 2: this bit is set when the carrier phase (L) has a half-cycle ambiguity Bits 3-7: If `SigIdxLo` from the `Type` field of this sub-block equals 31, these bits contain the signal number with an offset of 32 (see 4.1.10), e.g. 1 corresponds to signal number 33 (QZSS L1S). Otherwise they are reserved and must be ignored by the decoding software.

(1) The pseudorange is invalid if both `CodeMSB` is 0 and `CodeLSB` is 0.

(2) The carrier phase is invalid if both `CarrierMSB` is -128 and `CarrierLSB` is 0.

`CodeOffsetLSB`	u2	0.001 m	0 [3]	LSB of the code offset with respect to pseudorange in the `MeasEpochChannelType1` sub-block. To compute the pseudorange, use: PR_{type2} [m] = PR_{type1}[m] + (`CodeOffsetMSB`*65536+`CodeOffsetLSB`)*0.001
`CarrierLSB`	u2	0.001 cycles	0 [5]	LSB of the carrier phase relative to the pseudorange. The full carrier phase can be computed by: L[cycles]= PR_{type2}[m]/λ +(`CarrierMSB`*65536+`CarrierLSB`)*0.001 where λ is the carrier wavelength corresponding to the signal type in the `Type` field.
`DopplerOffsetLSB`	u2	0.0001 Hz	0 [4]	LSB of the Doppler offset relative to the Doppler in the `MeasEpochChannelType1` sub-block. To compute the Doppler, use: D_{type2}[Hz] = D_{type1}[Hz]*α +(`DopplerOffsetMSB`*65536+`DopplerOffsetLSB`) *1e-4, where α is the ratio of the carrier frequency corresponding to the observable type in this `MeasEpochChannelType2` sub-block, and that of the master observable type in the parent `MeasEpochChannelType1` sub-block (see section 4.1.10 for a list of all carrier frequencies).
`Padding`	u1[..]			Padding bytes, see 4.1.5

---

[3] The pseudorange is invalid if both `CodeOffsetMSB` is -4 and `CodeOffsetLSB` is 0.

[4] The Doppler is invalid if both `DopplerOffsetMSB` is -16 and `DopplerOffsetLSB` is 0.

[5] The carrier phase is invalid if both `CarrierMSB` is -128 and `CarrierLSB` is 0.

`MeasExtra`	Number:	4000
	`"OnChange"` interval:	internal measurement rate (receiver-type dependent)

This block contains extra information associated with the measurements contained in the `MeasEpoch` block, such as the internal corrections parameters applied during the measurement pre-processing, and the noise variances.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of sub-blocks in this `MeasExtra` block.
`SBLength`	u1	1 byte		Length of a sub-block
`DopplerVarFactor`	f4	1 Hz^2 / $cycle^2$		Factor to be used to compute the Doppler variance from the carrier phase variance. More specifically, the Doppler variance in *mHz*^{2} can be computed by: $\sigma^2_{\mathrm{Doppler}}$[$mHz^2$] = `CarrierVariance` * `DopplerVarFactor`, Where `CarrierVariance` can be found for each measurement type in the `MeasExtraChannelSub` sub-blocks.
ChannelSub	...	...		*A succession of N `MeasExtraChannelSub` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`MeasExtraChannelSub` sub-block definition:

	Parameter	Type	Units	Do-Not-Use	Description
	`RxChannel`	u1			Receiver channel on which this satellite is currently tracked (see 4.1.11).
	`Type`	u1			Bit field indicating the signal type and antenna ID: Bits 0-4: `SigIdxLo`: if not 31, this is the signal number (see 4.1.10), otherwise the signal number can be found in the `Misc` field below. A value of 31 can only happen on block revision 3 or above. Bits 5-7: Antenna ID: 0 for main, 1 for *Aux1* and 2 for *Aux2*
	`MPCorrection`	i2	0.001 m		Multipath correction applied to the pseudorange. This number has to be added to the pseudorange to recover the raw pseudorange as it would be if multipath mitigation was not used.
	`SmoothingCorr`	i2	0.001 m		Smoothing correction applied to the pseudorange. This number has to be added to the pseudorange to recover the raw pseudorange as it would be if smoothing was disabled.
	`CodeVar`	u2	0.0001 m^2	65535	Estimated code tracking noise variance. If the variance is larger than 65534 cm^2, it is clipped to 65534 cm^2.
	`CarrierVar`	u2	1 $mcycle^2$	65535	Estimated carrier tracking noise variance. This value can be multiplied by `DopplerVarFactor` to compute the Doppler measurement variance. If the variance is larger than 65534 $mcycles^2$, it is clipped to 65534 $mcycles^2$.
	`LockTime`	u2	1 s	65535	Duration of continuous carrier phase. The lock-time is reset at the initial lock after a signal (re)acquisition. If the lock-time is longer than 65534s, it is clipped to 65534s. If the carrier phase measurement is not available, this field is set to its Do-Not-Use value.
Rev 1	`CumLossCont`	u1			Carrier phase cumulative loss-of-continuity counter (modulo 256) for the signal type, antenna and satellite this sub-block refers to. This counter starts at zero at receiver start-up, and is incremented at each initial lock after signal (re)acquisition, or when a cycle slip is detected.
	`CarMPCorr`	i1	1.953125 mcycle		Multipath correction applied to the carrier phase, in units of 1/512 cycles. This number has to be added to the carrier phase to recover the raw phase as it would be if multipath mitigation was not used.
Rev 2	`Info`	u1			Bit field: Bits 0-3: Reserved. Bits 4-7: Reserved.
Rev 3	`Misc`	u1	0.03125 dB-Hz		Bit field: Bits 0-2: `CN0HighRes`: high-resolution extension of the C/N0 (unsigned value from 0 to 7). The C/N0 value in the `MeasEpoch` SBF block has a resolution of 0.25dB-Hz. `CN0HighRes` can be used to extend the resolution to 0.03125dB-Hz. The high-resolution C/N0, in dB-Hz, is computed as follows: $C/N_{0,\mathrm{HighRes}} = C/N_{0,\mathrm{MeasEpoch}} + \mathtt{CN0HighRes} * 0.03125$. where $C/N_{0,\mathrm{MeasEpoch}}$ is the C/N0 value coming from the `MeasEpoch` SBF block. Bits 3-7: If `SigIdxLo` from the `Type` field equals 31, these bits contain the signal number with an offset of 32 (see 4.1.10). Otherwise they are reserved.
	`Padding`	u1[..]			Padding bytes, see 4.1.5

`Meas3Ranges`	Number:	4109
	`"OnChange"` interval:	internal measurement rate (receiver-type dependent)

This block contains all the code, carrier phase and C/N0 observables at a given measurement epoch. The resolution is 0.001m, 0.001cycles and 1dB-Hz for the code, carrier and C/N0 measurements respectively.

Applications requiring Doppler measurements can log the `Meas3Doppler` SBF block in addition to the `Meas3Ranges` block. Applications requiring extended C/N0 resolution (1/16dB-Hz) can log the `Meas3CN0HiRes` SBF block in addition to the `Meas3Ranges` block.

The advantage of this block compared to the `MeasEpoch` SBF block is its reduced size while offering the full resolution for the code and carrier measurements. One of the techniques used to reduce the size is to only encode full measurements (reference epochs) every N epochs. Between these reference epochs, `Meas3Ranges` contains delta epochs where the difference between the current measurements and the ones at the applicable reference epoch is encoded. The decoder must have received and stored the applicable reference epoch to be able to decode delta epochs. When streaming SBF over an unreliable communication link, if the reference epoch is lost, subsequent `Meas3Ranges` blocks cannot be decoded until the next reference epoch is received. The interval at which reference epochs are encoded can be controlled with the **`setMeas3MaxRefInterval`** command. A longer interval generally reduces the average block size, at the expense of a longer data gap in case a reference epoch is lost.

See also page 301 for additional information.

⚠ The format of this block and of the other Meas3 blocks is complex and is not provided here. Details can be obtained from Septentrio Support. The RxTools installation contains the complete source code of a decoder in C language, together with `sbf2asc`, a small application showing how to use it. All C files can be found under the `sbf2asc` folder in the RxTools installation. The main measurement decoding function is `sbfread_MeasCollectAndDecode()` in the `sbfread_meas.c` file. Users interested in decoding the Meas3 blocks are strongly advised to use the provided source code instead of writing their own decoder.

The detailed definition of this block is not available in this document.

`Meas3CN0HiRes`	Number:	4110
	`"OnChange"` interval:	internal measurement rate (receiver-type dependent)

The `Meas3CN0HiRes` block is an extension of the `Meas3Ranges` block containing the fractional part of the C/N0 values. The resolution of the C/N0 value in the `Meas3Ranges` SBF block is 1dB-Hz. Applications requiring a finer C/N0 resolution (0.0625dB-Hz) must log the `Meas3CN0HiRes` block together with the `Meas3Ranges` block.

The detailed definition of this block is not available in this document.

`Meas3Doppler`	Number:	4111
	`"OnChange"` interval:	internal measurement rate (receiver-type dependent)

The `Meas3Doppler` block is an extension of the `Meas3Ranges` block containing the range-rate (Doppler) values. Applications requiring range-rate or Doppler observables must log the `Meas3Doppler` block together with the `Meas3Ranges` block.

The detailed definition of this block is not available in this document.

`Meas3PP`	Number:	4112
	`"OnChange"`	interval: internal measurement rate (receiver-type dependent)

The `Meas3PP` block is an extension of the `Meas3Ranges` block containing various Septentrio-proprietary flags and values needed for accurate post-processing or re-processing of the PVT from the measurements in the `Meas3Ranges` SBF block. This block must be logged together with `Meas3Ranges`.

The detailed definition of this block is not available in this document.

`Meas3MP`	Number:	4113
	`"OnChange"`	interval: internal measurement rate (receiver-type dependent)

The `Meas3MP` block is an extension of the `Meas3Ranges` block containing the multipath correction applied by the receiver. It can be used for research purposes to undo the receiver multipath mitigation and revert to unmitigated data. This block must be logged together with `Meas3Ranges`.

The detailed definition of this block is not available in this document.

`EndOfMeas`	Number:	5922
	`"OnChange"` interval:	internal measurement rate (receiver-type dependent)

This block marks the end of the transmission of all measurement-related blocks belonging to a given epoch.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.2 Navigation Page Blocks

`GPSRawCA`	Number:	4017
	`"OnChange"`	interval: 6s

This block contains the 300 bits of a GPS C/A subframe. It is generated each time a new subframe is received, i.e. every 6 seconds.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Not applicable
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NAVBits` contains the 300 bits of a GPS C/A subframe. Encoding: For easier parsing, the bits are stored as a succession of 10 32-bit words. Since the actual words in the subframe are 30-bit long, two unused bits are inserted in each 32-bit word. More specifically, each 32-bit word has the following format: Bits 0-5: 6 parity bits (referred to as D_{25} to D_{30} in the GPS ICD), XOR-ed with the last transmitted bit of the previous word (D^*_{30}). Bits 6-29: source data bits (referred to as d_n in the GPS ICD). The first received bit is the MSB. Bits 30-31: Reserved
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GPSRawL2C`	Number: 4018 `"OnChange"` interval: 12s

This block contains the 300 bits of a GPS L2C CNAV subframe (the so-called $D_c(t)$ data stream).

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the subframe
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NAVBits` contains the 300 bits of a GPS CNAV subframe. Encoding: `NAVBits` contains all the bits of the frame, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[9]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GPSRawL5`	Number:	4019
	`"OnChange"`	interval: 6s

This block contains the 300 bits of a GPS L5 CNAV subframe (the so-called $D_c(t)$ data stream).

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the subframe
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NAVBits` contains the 300 bits of a GPS CNAV subframe. Encoding: `NAVBits` contains all the bits of the frame, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[9]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GPSRawL1C`	Number:	4221
	`"OnChange"`	interval: 18s

This block contains the 1800 symbols of a GPS L1C navigation frame (itself containing three subframes).

The symbols are deinterleaved. The receiver attempts to correct bit errors using the LDPC parity bits, but unrecoverable errors are still possible at low C/N0. It is therefore always needed to check the CRC status before using the navigation bits. A separate CRC status is provided for subframe 2 and 3.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCSF2`	u1			Status of the CRC check of subframe 2: 0: failed 1: passed
`CRCSF3`	u1			Status of the CRC check of subframe 3: 0: failed 1: passed
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[57]			`NAVBits` contains the 1800 deinterleaved symbols of a GPS L1C navigation frame. Encoding: `NAVBits` contains all the symbols of the frame. The first received symbol (i.e. the first symbol of subframe 1) is stored as the MSB of `NAVBits[0]`. The 24 unused bits in NAVBits[56] must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GLORawCA`	Number:	4026
	`"OnChange"` interval:	2s

This block contains the 85 bits of a GLONASS L1CA or L2CA navigation string.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Not applicable
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`FreqNr`	u1			Frequency number, with an offset of 8. See 4.1.9
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[3]			`NAVBits` contains the first 85 bits of a GLONASS C/A string (i.e. all bits of the string with the exception of the time mark). Encoding: The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[2]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

GALRawFNAV	Number: 4022 "OnChange" interval: 10s

This block contains the 244 bits of a Galileo F/NAV navigation page, after deinterleaving and Viterbi decoding.

Parameter	Type	Units	Do-Not-Use	Description
Sync1	c1			Block Header, see 4.1.1
Sync2	c1			
CRC	u2			
ID	u2			
Length	u2	1 byte		
TOW	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
WNc	u2	1 week	65535	
SVID	u1			Satellite ID, see 4.1.9
CRCPassed	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
ViterbiCnt	u1			Viterbi decoder error count over the page
Source	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-6: Reserved Bit 7: Reserved
FreqNr	u1			Not applicable
RxChannel	u1			Receiver channel (see 4.1.11).
NAVBits	u4[8]			`NavBits` contains the 244 bits of a Galileo F/NAV page. Encoding: `NAVBits` contains all the bits of the frame, with the exception of the synchronization field. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[7]` must be ignored by the decoding software.
Padding	u1[..]			Padding bytes, see 4.1.5

`GALRawINAV`	Number: 4023 `"OnChange"` interval: 2s

This block contains the 234 bits of a Galileo I/NAV navigation page, after deinterleaving and Viterbi decoding.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the page
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bit 5: Set when the nav page is the concatenation of a sub-page received from E5b, and a sub-page received from L1BC. In that case, bits 0-4 are set to L1BC. Bit 6: Reserved Bit 7: Reserved
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[8]			`NAVBits` contains the 234 bits of an I/NAV navigation page (in nominal or alert mode). Note that the I/NAV page is transmitted as two sub-pages (the so-called even and odd pages) of duration 1 second each (120 bits each). In this block, the even and odd pages are concatenated, even page first and odd page last. The 6 tails bits at the end of the even page are removed (hence a total of 234 bits). If the even and odd pages have been received from two different carriers (E5b and L1), bit 5 of the `Source` field is set. Encoding: `NAVBits` contains all the bits of the frame, with the exception of the synchronization field. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[7]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GALRawCNAV`	Number:	4024
	`"OnChange"` interval: 1s	

This block contains the 492 bits of a Galileo C/NAV navigation page, after deinterleaving and Viterbi decoding.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the page
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-6: Reserved Bit 7: Reserved
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[16]			`NAVBits` contains the 492 bits of a Galileo C/NAV page. Encoding: `NAVBits` contains all the bits of the frame, with the exception of the synchronization field. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[15]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEORawL1`	Number: 4020 `"OnChange"` interval: 1s

This block contains the 250 bits of a SBAS L1 navigation frame, after Viterbi decoding.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the navigation frame
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[8]			`NAVBits` contains the 250 bits of a SBAS navigation frame. Encoding: `NAVBits` contains all the bits of the frame, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[7]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEORawL5`	Number:	4021
	`"OnChange"`	interval: 1s

This block contains the 250 bits of a SBAS L5 navigation frame, after Viterbi decoding.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the navigation frame
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[8]			`NAVBits` contains the 250 bits of a SBAS navigation frame. Encoding: `NAVBits` contains all the bits of the frame, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[7]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSRaw`	Number:	4047
	`"OnChange"`	interval: 6 seconds (non GEOs), 0.6 s (GEOs)

This block contains the 300 bits of a BeiDou navigation page, as received from the B1I, B2I or B3I signal.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Not applicable
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NAVBits` contains the 300 deinterleaved bits of a BeiDou navigation subframe. Encoding: `NAVBits` contains all the bits of the subframe, including the preamble and the parity bits. The first received bit is stored as the MSB of `NAVBits[0]`. The 20 unused bits in `NAVBits[9]` must be ignored by the decoding software. The bits are deinterleaved.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSRawB1C`	Number:	4218
	`"OnChange"`	interval: 18s

This block contains the 1800 symbols of a BeiDou B-CNAV1 navigation frame (itself containing three subframes), as received from the B1C signal.

The symbols are deinterleaved. The receiver attempts to correct bit errors using the LDPC parity bits, but unrecoverable errors are still possible at low C/N0. It is therefore always needed to check the CRC status before using the navigation bits. A separate CRC check is provided for subframe 2 and 3.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCSF2`	u1			Status of the CRC check of subframe 2: 0: failed 1: passed
`CRCSF3`	u1			Status of the CRC check of subframe 3: 0: failed 1: passed
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[57]			`NAVBits` contains the 1800 deinterleaved symbols of a BeiDou B1C (B-CNAV1) navigation frame. Encoding: `NAVBits` contains all the symbols of the frame. The first received symbol (i.e. the first symbol of subframe 1) is stored as the MSB of `NAVBits[0]`. The 24 unused bits in NAVBits[56] must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSRawB2a`	Number: 4219 `"OnChange"` interval: 3s

This block contains the 576 symbols of a BeiDou B-CNAV2 navigation frame, as received from the B2a signal.

The receiver attempts to correct bit errors using the LDPC parity bits, but unrecoverable errors are still possible at low C/N0. It is therefore always needed to check the CRC status before using the navigation bits.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Not applicable
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[18]			`NAVBits` contains the 576 symbols of a BeiDou B2a (B-CNAV2) navigation frame. Encoding: `NAVBits` contains all the symbols of the frame, excluding the preamble. The first received symbol (i.e. the MSB of the PRN field) is stored as the MSB of `NAVBits[0]`.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSRawB2b`	Number:	4242
	`"OnChange"`	interval: 1s

This block contains the raw symbols of a BeiDou B-CNAV3 or PPP-B2b_I navigation frame, as received from the B2b signal.

The receiver attempts to correct bit errors using the LDPC parity bits, but unrecoverable errors are still possible at low C/N0. It is therefore always needed to check the CRC status before using the navigation bits.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC check on the 486 bits of the message: 0: CRC check failed 1: CRC check passed
`Reserved1`	u1			Reserved for future use, to be ignored by decoding software.
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved2`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[31]			`NAVBits` contains the 984 symbols of a BeiDou B2b navigation frame. Encoding: `NAVBits` contains all the symbols of the frame, excluding the preamble. The first received symbol (i.e. the MSB of the PRN field) is stored as the MSB of `NAVBits[0]`. The 8 unused bits in NAVBits[30] must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSRawL1CA`	Number:	4066
	`"OnChange"`	interval: 6s

This block contains the 300 bits of a QZSS L1C/A or L1C/B subframe.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`Reserved`	u1			Reserved
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved2`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NAVBits` contains the 300 bits of a QZSS C/A subframe. Encoding: Same as `GPSRawCA` block.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSRawL2C`	Number:	4067
	`"OnChange"`	interval: 12s

This block contains the 300 bits of a QZSS L2C CNAV subframe.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the subframe
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NAVBits` contains the 300 bits of a QZSS CNAV subframe. Encoding: `NAVBits` contains all the bits of the frame, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[9]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSRawL5`	Number:	4068
	`"OnChange"`	interval: 6s

This block contains the 300 bits of a QZSS L5 CNAV subframe.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the subframe
`Source`	u1			Bit field: Bits 0-4: Signal type from which the bits have been received, as defined in 4.1.10 Bits 5-7: Reserved
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NAVBits` contains the 300 bits of a QZSS CNAV subframe. Encoding: `NAVBits` contains all the bits of the frame, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[9]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSRawL6`	Number:	4069
	`"OnChange"`	interval: 1s

This block contains the 2000 bits of a QZSS L6 message.

The receiver attempts to correct bit errors using the Reed-Solomon parity symbols. The block contains the corrected bits and there is no need to have a Reed-Solomon decoder at the user side. The `Parity` field indicates whether the error recovery was successful or not.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`Parity`	u1			Status of the Reed-Solomon decoding: 0: Failed: unrecoverable errors found. There is at least one wrong bit in `NavBits`. 1: Passed: all bit errors could be recovered, or the message was received without bit error.
`RSCnt`	u1			Number of symbol errors that were successfully corrected by the Reed-Solomon decoder.
`Source`	u1			Source of the message: 0: Unknown 1: QZSS L6D 2: QZSS L6E
`Reserved`	u1			Reserved
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[63]			`NAVBits` contains the 2000 bits of a QZSS L6 message. Encoding: `NAVBits` contains all the bits of the message after Reed-Solomon decoding, including the preamble and the Reed-Solomon parity symbols themselves. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[63]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSRawL1C`	Number: 4227 `"OnChange"` interval: 18s

This block contains the 1800 symbols of a QZSS L1C navigation frame (itself containing three subframes).

The symbols are deinterleaved. The receiver attempts to correct bit errors using the LDPC parity bits, but unrecoverable errors are still possible at low C/N0. It is therefore always needed to check the CRC status before using the navigation bits. A separate CRC status is provided for subframe 2 and 3.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCSF2`	u1			Status of the CRC check of subframe 2: 0: failed 1: passed
`CRCSF3`	u1			Status of the CRC check of subframe 3: 0: failed 1: passed
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[57]			`NAVBits` contains the 1800 deinterleaved symbols of a QZSS L1C navigation frame. Encoding: `NAVBits` contains all the symbols of the frame. The first received symbol (i.e. the first symbol of subframe 1) is stored as the MSB of `NAVBits[0]`. The 24 unused bits in NAVBits[56] must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSRawL1S`	Number: 4228 `"OnChange"` interval: 1s

This block contains the 250 bits of a QZSS L1S navigation message, after Viterbi decoding.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the navigation message
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[8]			`NAVBits` contains the 250 bits of a QZSS L1S navigation message. Encoding: `NAVBits` contains all the bits of the message, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[7]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSRawL5S`	Number: 4246 `"OnChange"` interval: 1s

This block contains the 250 bits of a QZSS L5S navigation message, after Viterbi decoding.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the navigation message
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`FreqNr`	u1			Not applicable
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[8]			`NAVBits` contains the 250 bits of a QZSS L5S navigation message. Encoding: `NAVBits` contains all the bits of the message, including the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[7]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`NAVICRaw`	Number:	4093
	`"OnChange"`	interval: 12s

This block contains the 292 bits of a NavIC/IRNSS subframe.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			Satellite ID, see 4.1.9
`CRCPassed`	u1			Status of the CRC or parity check: 0: CRC or parity check failed 1: CRC or parity check passed
`ViterbiCnt`	u1			Viterbi decoder error count over the subframe
`Source`	u1			Signal type from which the bits have been received, as defined in 4.1.10
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`RxChannel`	u1			Receiver channel (see 4.1.11).
`NAVBits`	u4[10]			`NavBits` contains the 292 bits of a NavIC/IRNSS subframe. Encoding: `NAVBits` contains all the bits of the frame, with the exception of the preamble. The first received bit is stored as the MSB of `NAVBits[0]`. The unused bits in `NAVBits[9]` must be ignored by the decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.3 GPS Decoded Message Blocks

`GPSNav`	Number:	5891
	`"OnChange"` interval:	block generated each time a new navigation data set is received from a GPS satellite

The `GPSNav` block contains the decoded navigation data for one GPS satellite. These data are conveyed in subframes 1 to 3 of the satellite navigation message. Refer to GPS ICD for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the GPS satellite of which the ephemeris is given in this block (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`WN`	u2	1 week	65535	Week number (10 bits from subframe 1, word 3)
`CAorPonL2`	u1			Code(s) on L2 channel (2 bits from subframe 1, word 3)
`URA`	u1			User Range accuracy index (4 bits from subframe 1 word 3)
`health`	u1			6-bit health from subframe 1, word 3 (6 bits from subframe 1, word 3)
`L2DataFlag`	u1			Data flag for L2 P-code (1 bit from subframe 1, word 4)
`IODC`	u2			Issue of data, clock (10 bits from subframe 1)
`IODE2`	u1			Issue of data, ephemeris (8 bits from subframe 2)
`IODE3`	u1			Issue of data, ephemeris (8 bits from subframe 3)
`FitIntFlg`	u1			Curve Fit Interval, (1 bit from subframe 2, word 10)
`Reserved2`	u1			unused, to be ignored by decoding software
`T_gd`	f4	1 s		Estimated group delay differential
`t_oc`	u4	1 s		clock data reference time
`a_f2`	f4	1 s / s^2		SV clock aging
`a_f1`	f4	1 s / s		SV clock drift
`a_f0`	f4	1 s		SV clock bias
`C_rs`	f4	1 m		Amplitude of the sine harmonic correction term to the orbit radius
`DEL_N`	f4	1 semi-circle / s		Mean motion difference from computed value
`M_0`	f8	1 semi-circle		Mean anomaly at reference time
`C_uc`	f4	1 rad		Amplitude of the cosine harmonic correction term to the argument of latitude
`e`	f8			Eccentricity
`C_us`	f4	1 rad		Amplitude of the sine harmonic correction term to the argument of latitude
`SQRT_A`	f8	1 m$^{1/2}$		Square root of the semi-major axis
`t_oe`	u4	1 s		Reference time ephemeris
`C_ic`	f4	1 rad		Amplitude of the cosine harmonic correction term to the angle of inclination
`OMEGA_0`	f8	1 semi-circle		Longitude of ascending node of orbit plane at weekly epoch
`C_is`	f4	1 rad		Amplitude of the sine harmonic correction term to the angle of inclination

`i_0`	f8	1 semi-circle		Inclination angle at reference time
`C_rc`	f4	1 m		Amplitude of the cosine harmonic correction term to the orbit radius
`omega`	f8	1 semi-circle		Argument of perigee
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`IDOT`	f4	1 semi-circle / s		Rate of inclination angle
`WNt_oc`	u2	1 week		WN associated with t_oc, modulo 1024
`WNt_oe`	u2	1 week		WN associated with t_oe, modulo 1024
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GPSAlm`	Number:	5892
	`"OnChange"` interval:	block generated each time a new almanac data set is received from a GPS satellite

The `GPSAlm` block contains the decoded almanac data for one GPS satellite. These data are conveyed in subframes 4 and 5 of the satellite navigation message. Refer to GPS ICD for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the GPS satellite of which the almanac is given in this block (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`e`	f4			Eccentricity
`t_oa`	u4	1 s		almanac reference time of week
`delta_i`	f4	1 semi-circle		Inclination angle at reference time, relative to i_0 = 0.3 semi-circles
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`SQRT_A`	f4	1 $m^{1/2}$		Square root of the semi-major axis
`OMEGA_0`	f4	1 semi-circle		Longitude of ascending node of orbit plane at weekly epoch
`omega`	f4	1 semi-circle		Argument of perigee
`M_0`	f4	1 semi-circle		Mean anomaly at reference time
`a_f1`	f4	1 s / s		SV clock drift
`a_f0`	f4	1 s		SV clock bias
`WN_a`	u1	1 week		Almanac reference week, to which t_oa is referenced
`config`	u1			Anti-spoofing and satellite configuration (4 bits from subframe 4, page 25)
`health8`	u1			health on 8 bits from the almanac page
`health6`	u1			health summary on 6 bits (from subframe 4, page 25 and subframe 5 page 25)
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GPSIon`	Number:	5893
	`"OnChange"` interval:	block generated each time subframe 4, page 18, is received from a GPS satellite

The `GPSIon` block contains the decoded ionosphere data (the Klobuchar coefficients). These data are conveyed in subframes 4, page 18 of the satellite navigation message. Refer to GPS ICD for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the GPS satellite from which the coefficients have been received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`alpha_0`	f4	1 s		vertical delay coefficient 0
`alpha_1`	f4	1 s / semi-circle		vertical delay coefficient 1
`alpha_2`	f4	1 s / semi-circle^2		vertical delay coefficient 2
`alpha_3`	f4	1 s / semi-circle^3		vertical delay coefficient 3
`beta_0`	f4	1 s		model period coefficient 0
`beta_1`	f4	1 s / semi-circle		model period coefficient 1
`beta_2`	f4	1 s / semi-circle^2		model period coefficient 2
`beta_3`	f4	1 s / semi-circle^3		model period coefficient 3
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GPSUtc`	Number:	5894
	`"OnChange"` interval:	block generated each time subframe 4, page 18, is received from a GPS satellite

The `GPSUtc` block contains the decoded UTC data. These data are conveyed in subframes 4, page 18 of the satellite navigation message. Refer to GPS ICD for further details.

Parameter	**Type**	**Units**	**Do-Not-Use**	**Description**
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the GPS satellite from which these UTC parameters have been received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`A_1`	f4	1 s / s		first order term of polynomial
`A_0`	f8	1 s		constant term of polynomial
`t_ot`	u4	1 s		reference time for UTC data
`WN_t`	u1	1 week		UTC reference week number, to which `t_ot` is referenced
`DEL_t_LS`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is not in the past
`WN_LSF`	u1	1 week		Effectivity time of leap second (week)
`DN`	u1	1 day		Effectivity time of leap second (day, from 1 to 7)
`DEL_t_LSF`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is in the past
`Padding`	u1[..]			Padding bytes, see 4.1.5

GPSCNav	Number:	4042
	"OnChange" interval:	block generated at each change of ephemeris, clock or group delay parameters

The GPSCNav block contains the decoded CNAV navigation data for one GPS satellite. The ephemeris and health parameters are extracted from message types (MT) 10 and 11. The clock correction and group delay correction parameters are extracted from MT30.

Parameter	Type	Units	Do-Not-Use	Description
Sync1	c1			Block Header, see 4.1.1
Sync2	c1			
CRC	u2			
ID	u2			
Length	u2	1 byte		
TOW	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
WNc	u2	1 week	65535	
PRN	u1			ID of the GPS satellite of which the parameters are given in this block (see 4.1.9)
Flags	u1			Bit field: Bit 0: Alert: Set to 1 if at least one of the message types of which the contents is included in this block had its alert bit (bit 38) set to 1. Set to 0 otherwise. Bit 1: Integrity Status Flag (bit 272 of MT10). Bit 2: L2C Phasing (bit 273 of MT10). Bits 3-5: Reserved Bit 6: L2C used: bit set if at least part of the navigation data has been decoded from L2C. Bit 7: L5 used: bit set if at least part of the navigation data has been decoded from L5.
WN	u2	1 week		Week number (WN_n, 13 bits from MT10)
Health	u1			L1, L2 and L5 signal health (3 bits from MT10). LSB is L5 health.
URA_ED	i1			Elevation-Dependant accuracy index (URA_{ED}, 5 bits from MT10)
t_op	u4	1 s		Data predict time of week ($300 * t_{op}$, 11 bits from MT10/30)
t_oe	u4	1 s		Ephemeris data reference time of week ($300 * t_{oe}$, 11 bits from MT10/11)
A	f8	1 m		Semi-major axis at reference time (ΔA, 26 bits from MT10, plus A_{ref})
A_DOT	f8	1 m / s		Change rate in semi-major axis ($\dot{A}$, 25 bits from MT10)
DELTA_N	f4	1 semi-circle / s		Mean motion difference from computed value at reference time (Δn_0, 17 bits from MT10)
DELTA_N_DOT	f4	1 semi-circle / s^2		Rate of mean motion difference from computed value ($\Delta \dot{n}_0$, 23 bits from MT10)
M_0	f8	1 semi-circle		Mean anomaly at reference time (M_{0-n}, 33 bits from MT10)
e	f8			Eccentricity (e_n, 33 bits from MT10)
omega	f8	1 semi-circle		Argument of perigee (ω_n, 33 bits from MT10)
OMEGA_0	f8	1 semi-circle		Reference right ascension angle (Ω_{0-n}, 33 bits from MT11)
OMEGADOT	f8	1 semi-circle / s		Rate of right ascension ($\Delta\dot{\Omega}$, 17 bits from MT11, plus $\dot{\Omega}_{ref}$)
i_0	f8	1 semi-circle		Inclination angle at reference time (i_{0-n}, 33 bits from MT11)
IDOT	f4	1 semi-circle / s		Rate of inclination angle ($\dot{i}_{0-n}$, 15 bits from MT11)
C_is	f4	1 rad		Amplitude of the sine harmonic correction term to the angle of inclination (C_{is-n}, 16 bits from MT11)
C_ic	f4	1 rad		Amplitude of the cosine harmonic correction term to the angle of inclination (C_{ic-n}, 16 bits from MT11)

C_rs	f4	1 m		Amplitude of the sine harmonic correction term to the orbit radius (C_{rs-n}, 24 bits from MT11)
C_rc	f4	1 m		Amplitude of the cosine harmonic correction term to the orbit radius (C_{rc-n}, 24 bits from MT11)
C_us	f4	1 rad		Amplitude of the sine harmonic correction term to the argument of latitude (C_{us-n}, 21 bits from MT11)
C_uc	f4	1 rad		Amplitude of the cosine harmonic correction term to the argument of latitude (C_{uc-n}, 21 bits from MT11)
t_oc	u4	1 s		Clock data reference time ($300 * t_{oc}$, 11 bits from MT30)
URA_NED0	i1			Non-Elevation-Dependant accuracy index (URA_{NED0}, 5 bits from MT30)
URA_NED1	u1			Non-Elevation-Dependant accuracy change index (URA_{NED1}, 3 bits from MT30)
URA_NED2	u1			Non-Elevation-Dependant accuracy change rate index (URA_{NED2}, 3 bits from MT30)
WN_op	u1	1 week		Week number associated with t_op, modulo 256 (WN_{op}, 8 bits from MT30)
a_f2	f4	1 s / s^2		Clock drift rate correction coefficient (a_{f2-n}, 10 bits from MT30)
a_f1	f4	1 s / s		Clock drift correction coefficient (a_{f1-n}, 20 bits from MT30)
a_f0	f8	1 s		Clock bias correction coefficient (a_{f0-n}, 26 bits from MT30)
T_gd	f4	1 s	$-2 \cdot 10^{10}$ (6)	Group delay differential (T_{GD}, 13 bits from MT30)
ISC_L1CA	f4	1 s	$-2 \cdot 10^{10}$ (6)	Group delay differential L1C/A ($ISC_{L1C/A}$, 13 bits from MT30)
ISC_L2C	f4	1 s	$-2 \cdot 10^{10}$ (6)	Group delay differential L2C (ISC_{L2C}, 13 bits from MT30)
ISC_L5I5	f4	1 s	$-2 \cdot 10^{10}$ (6)	Group delay differential L5I (ISC_{L5I5}, 13 bits from MT30)
ISC_L5Q5	f4	1 s	$-2 \cdot 10^{10}$ (6)	Group delay differential L5Q (ISC_{L5Q5}, 13 bits from MT30)
Padding	u1[..]			Padding bytes, see 4.1.5

---

(6) The Do-Not-Use value is used when the bit string is "1000000000000".

4.2.4 GLONASS Decoded Message Blocks

`GLONav`	Number:	4004
	`"OnChange"` interval:	block generated each time a new navigation data set is received from a GLONASS satellite

The `GLONav` block contains the decoded ephemeris data for one GLONASS satellite.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			ID of the GLONASS satellite for which ephemeris is provided in this block (see 4.1.9).
`FreqNr`	u1			Frequency number of the GLONASS satellite for which ephemeris is provided in this block (see 4.1.9).
`X`	f8	1000 m		x-component of satellite position in PZ-90
`Y`	f8	1000 m		y-component of satellite position in PZ-90
`Z`	f8	1000 m		z-component of satellite position in PZ-90
`Dx`	f4	1000 m / s		x-component of satellite velocity in PZ-90
`Dy`	f4	1000 m / s		y-component of satellite velocity in PZ-90
`Dz`	f4	1000 m / s		z-component of satellite velocity in PZ-90
`Ddx`	f4	1000 m / s^2		x-component of satellite acceleration in PZ-90
`Ddy`	f4	1000 m / s^2		y-component of satellite acceleration in PZ-90
`Ddz`	f4	1000 m / s^2		z-component of satellite acceleration in PZ-90
`gamma`	f4	1 Hz / Hz		$\gamma_{\mathrm{n}}(\mathrm{t_b})$:relative deviation of predicted carrier frequency
`tau`	f4	1 s		$\tau_{\mathrm{n}}(\mathrm{t_b})$: time correction to GLONASS time
`dtau`	f4	1 s		$\Delta\tau_{\mathrm{n}}$: time difference between L2 and L1 sub-band
`t_oe`	u4	1 s		reference time-of-week in GPS time frame
`WN_toe`	u2	1 week		reference week number in GPS time frame (modulo 1024)
`P1`	u1	1 minute		time interval between adjacent values of $\mathrm{t_b}$
`P2`	u1			1-bit odd/even flag of $\mathrm{t_b}$
`E`	u1	1 day		age of data
`B`	u1			3-bit health flag, satellite unhealthy if MSB set
`tb`	u2	1 minute		time of day (center of validity interval)
`M`	u1			2-bit GLONASS-M satellite identifier (01, otherwise 00)
`P`	u1			2-bit mode of computation of time parameters
`l`	u1			1-bit health flag, 0=healthy, 1=unhealthy
`P4`	u1			1-bit 'updated' flag of ephemeris data
`N_T`	u2	1 day		current day number within 4-year interval
`F_T`	u2	0.01 m		predicted user range accuracy at time $\mathrm{t_b}$
`C`	u1		255	1-bit health Cn flag from the almanac. This field is set to the last 'Cn' flag received from the almanac. If no 'Cn' flag has been received yet, it is set to its do-not-use value.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GLOAlm`	Number:	4005
	`"OnChange"` interval:	block generated each time a new almanac data set is received from a GLONASS satellite

The `GLOAlm` block contains the decoded navigation data for one GLONASS satellite.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			ID of the GLONASS satellite for which almanac is provided in this block (see 4.1.9).
`FreqNr`	u1			Frequency number of the GLONASS satellite for which almanac is provided in this block (see 4.1.9). This number corresponds to the H_n^A parameter in the GLONASS ICD.
`epsilon`	f4			ϵ_n^A: orbit eccentricity
`t_oa`	u4	1 s		Reference time-of-week in GPS time frame
`Delta_i`	f4	1 semi-circle		Δ i_n^A: correction to inclination
`lambda`	f4	1 semi-circle		λ_n^A: Longitude of first ascending node
`t_ln`	f4	1 s		$t^A_{\lambda\ n}$: time of first ascending node passage
`omega`	f4	1 semi-circle		ω_n^A: argument of perigee
`Delta_T`	f4	1 s / orbit-period		Δ T_n^A: correction to mean Draconian period
`dDelta_T`	f4	1 s / orbit-period^2		dΔ T_n^A: rate of change correction to mean Draconian period
`tau`	f4	1 s		τ_n^A: coarse correction to satellite time
`WN_a`	u1	1 week		Reference week in GPS time frame (modulo 256)
`C`	u1			$C_n{}^A$: 1-bit general health flag (1 indicates healthy)
`N`	u2	1 day		N^A: calendar day number within 4 year period
`M`	u1			$M_n{}^A$: 2-bit GLONASS-M satellite identifier
`N_4`	u1			N_4: 4 year interval number, starting from 1996
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GLOTime`	Number:	4036
	`"OnChange"` interval:	block generated at the end of each GLONASS superframe, i.e. every 2.5 minutes.

The `GLOTime` block contains the decoded non-immediate data related to the difference between GLONASS and GPS, UTC and UT1 time scales.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			ID of the GLONASS satellite from which the data in this block has been decoded (see 4.1.9).
`FreqNr`	u1			Frequency number of the GLONASS satellite from which the data in this block has been decoded (see 4.1.9).
`N_4`	u1			4 year interval number, starting from 1996
`KP`	u1			notification of leap second
`N`	u2	1 day		calendar day number within 4 year period
`tau_GPS`	f4	$1 \cdot 10^9$ ns		difference with respect to GPS time
`tau_c`	f8	$1 \cdot 10^9$ ns		GLONASS time scale correction to UTC(SU)
`B1`	f4	1 s		difference between UT1 and UTC(SU)
`B2`	f4	1 s / msd		daily change of B1
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.5 Galileo Decoded Message Blocks

`GALNav`	Number:	4002
	`"OnChange"`	interval: output each time a new navigation data batch is decoded.

The `GalNav` block contains the following decoded navigation data for one Galileo satellite:

- orbital elements and clock corrections
- health, Signal-In-Space Accuracy (SISA) indexes and Broadcast Group Delays (BGDs) for each carrier or carrier combinations.

The interpretation of the clock correction parameters (`t_oc`, `a_f0`, `a_f1`, `a_f2`) depends on the value of the `Source` field:

Source	Message type	Applicable Clock Model
2	I/NAV	(L1,E5b)
16	F/NAV	(L1,E5a)

If the receiver is decoding both the I/NAV and the F/NAV data stream, it will output a `GalNav` block for the I/NAV stream, containing the (L1, E5b) clock model, and a different `GalNav` block for the F/NAV stream, containing the (L1, E5a) clock model.

Depending on the message type being decoded, some health, SISA or BGD values may not be available (in that case they are set to their respective Do-Not-Use values). The following health, SISA and BGD values are guaranteed to be available for a given value of the `Source` field:

Source	Health, SISA and BGD availability
2 (I/NAV)	At least L1-B$_{\mathrm{DVS}}$, L1-B$_{\mathrm{HS}}$, E5b$_{\mathrm{DVS}}$,E5b$_{\mathrm{HS}}$, SISA_L1E5b and BGD_L1E5b are available
16 (F/NAV)	At least E5a$_{\mathrm{DVS}}$,E5a$_{\mathrm{HS}}$, SISA_L1E5a and BGD_L1E5a are available

The `IODNav` field identifies the issue of data. All orbital elements, clock parameters and SISA values in the block are guaranteed to refer to the same data batch identified by `IODNav`. The fields `Health_OSSOL`, `BGD_L1E5a`, `BGD_L1E5b` and `CNAVenc` are not covered by the issue of data, and the block simply contains the latest received value.

Please refer to the Galileo Signal-In-Space ICD for the interpretation and usage of the parameters contained in this SBF block.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			SVID of the Galileo satellite (see 4.1.9)
`Source`	u1			See table above: this field indicates how to interpret the clock correction parameters.

SQRT_A	f8	1 m$^{1/2}$		Square root of the semi-major axis
M_0	f8	1 semi-circle		Mean anomaly at reference time
e	f8			Eccentricity
i_0	f8	1 semi-circle		Inclination angle at reference time
omega	f8	1 semi-circle		Argument of perigee
OMEGA_0	f8	1 semi-circle		Longitude of ascending node of orbit plane at weekly epoch
OMEGADOT	f4	1 semi-circle / s		Rate of right ascension
IDOT	f4	1 semi-circle / s		Rate of inclination angle
DEL_N	f4	1 semi-circle / s		Mean motion difference from computed value
C_uc	f4	1 rad		Amplitude of the cosine harmonic correction term to the argument of latitude
C_us	f4	1 rad		Amplitude of the sine harmonic correction term to the argument of latitude
C_rc	f4	1 m		Amplitude of the cosine harmonic correction term to the orbit radius
C_rs	f4	1 m		Amplitude of the sine harmonic correction term to the orbit radius
C_ic	f4	1 rad		Amplitude of the sine harmonic correction term to the angle of inclination
C_is	f4	1 rad		Amplitude of the cosine harmonic correction term to the angle of inclination
t_oe	u4	1 s		Reference time, ephemeris
t_oc	u4	1 s		Reference time, clock. The Source field indicates which clock model t_oc refers to.
a_f2	f4	1 s / s^2		SV clock aging. The Source field indicates which clock model a_f2 refers to.
a_f1	f4	1 s / s		SV clock drift. The Source field indicates which clock model a_f1 refers to.
a_f0	f8	1 s		SV clock bias. The Source field indicates which clock model a_f0 refers to.
WNt_oe	u2	1 week		WN associated with t_oe, in GPS time frame, modulo 4096
WNt_oc	u2	1 week		WN associated with t_oc, in GPS time frame, modulo 4096
IODnav	u2			Issue of data, navigation (10 bits)
Health_OSSOL	u2			Bit field indicating the last received Health Status (HS) and Data Validity Status (DVS) of the E5a, E5b and L1-B signals: Bit 0: If set, bits 1 to 3 are valid, otherwise they must be ignored. Bit 1: 1-bit L1-B$_{\mathrm{DVS}}$ Bits 2-3: 2-bit L1-B$_{\mathrm{HS}}$ Bit 4: If set, bits 5 to 7 are valid, otherwise they must be ignored. Bit 5: 1-bit E5b$_{\mathrm{DVS}}$ Bits 6-7: 2-bit E5b$_{\mathrm{HS}}$ Bit 8: If set, bits 9 to 11 are valid, otherwise they must be ignored. Bit 9: 1-bit E5a$_{\mathrm{DVS}}$ Bits 10-11: 2-bit E5a$_{\mathrm{HS}}$ Bits 12-15: Reserved
Health_PRS	u1			Reserved
SISA_L1E5a	u1		255	Signal-In-Space Accuracy Index (L1, E5a)
SISA_L1E5b	u1		255	Signal-In-Space Accuracy Index (L1, E5b)
SISA_L1AE6A	u1		255	Reserved
BGD_L1E5a	f4	1 s	$-2 \cdot 10^{10}$	Last received broadcast group delay (L1, E5a)
BGD_L1E5b	f4	1 s	$-2 \cdot 10^{10}$	Last received broadcast group delay (L1, E5b)
BGD_L1AE6A	f4	1 s	$-2 \cdot 10^{10}$	Reserved

`CNAVenc`	u1		255	2-bit C/NAV encryption status: Bit 0: Bit set if E6B is unencrypted Bit 1: Bit set if E6C is unencrypted Bits 2-7: Reserved
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GALAlm`	Number:	4003
	`"OnChange"` interval:	output each time a new almanac set is received for a satellite.

The `GalAlm` block contains the decoded almanac data for one Galileo satellite.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			SVID of the Galileo satellite from which these almanac parameters have been received (see 4.1.9)
`Source`	u1			See corresponding field in the `GalNav` block. `Source` can take the value 18 to indicate that the almanac data contained in this block has been merged from INAV and FNAV pages.
`e`	f4			Eccentricity
`t_oa`	u4	1 s		almanac reference time of week
`delta_i`	f4	1 semi-circle		Inclination angle at reference time, relative to nominal
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`SQRT_A`	f4	1 $m^{1/2}$		Square root of the semi-major axis, relative to nominal
`OMEGA_0`	f4	1 semi-circle		Longitude of ascending node of orbit plane at weekly epoch
`omega`	f4	1 semi-circle		Argument of perigee
`M_0`	f4	1 semi-circle		Mean anomaly at reference time
`a_f1`	f4	1 s / s		SV clock drift
`a_f0`	f4	1 s		SV clock bias
`WN_a`	u1	1 week		2-bit almanac reference week
`SVID_A`	u1			SVID of the Galileo satellite of which the almanac parameters are provided in this block (see 4.1.9 for the SVID numbering convention).
`health`	u2			Bit field indicating the health status (HS) of the E5a, E5b, L1-B, L1-A and E6-A signals: Bit 0: If set, bits 1 and 2 are valid, otherwise they must be ignored. Bits 1-2: 2-bit L1-B$_{\mathrm{HS}}$ Bit 3: If set, bits 4 and 5 are valid, otherwise they must be ignored. Bits 4-5: 2-bit E5b$_{\mathrm{HS}}$ Bit 6: If set, bits 7 and 8 are valid, otherwise they must be ignored. Bits 7-8: 2-bit E5a$_{\mathrm{HS}}$ Bit 9: Not applicable Bits 10-11: Not applicable Bit 12: Not applicable Bits 13-14: Not applicable Bit 15: Reserved
`IODa`	u1			4-bit Issue of Data for the almanac.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GALIon`	Number:	4030
	`"OnChange"` interval:	output each time the ionospheric parameters are received from a Galileo satellite.

The `GalIon` block contains the decoded ionosphere model parameters of the Galileo system.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			SVID of the Galileo satellite from which these parameters have been received (see 4.1.9)
`Source`	u1			Message type from which the data has been decoded: 2: I/NAV 16: F/NAV
`a_i0`	f4	$1 \cdot 10^{-22}$ W / (m^2 Hz)		Effective ionization level, a_{i0}
`a_i1`	f4	$1 \cdot 10^{-22}$ W / (m^2 Hz) / deg		Effective ionization level, a_{i1}
`a_i2`	f4	$1 \cdot 10^{-22}$ W / (m^2 Hz) / deg^2		Effective ionization level, a_{i2}
`StormFlags`	u1			Bit field containing the five ionospheric storm flags: Bit 0: SF5 Bit 1: SF4 Bit 2: SF3 Bit 3: SF2 Bit 4: SF1 Bits 5-7: Reserved
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GALUtc`	Number:	4031
	`"OnChange"` interval:	output each time the UTC offset parameters are received from a Galileo satellite.

The `GalUtc` block contains the decoded UTC parameter information.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			SVID of the Galileo satellite from which these parameters have been received (see 4.1.9)
`Source`	u1			Message type from which the data has been decoded: 2: I/NAV 16: F/NAV
`A_1`	f4	1 s / s	$-2 \cdot 10^{10}$	first order term of polynomial
`A_0`	f8	1 s	$-2 \cdot 10^{10}$	constant term of polynomial
`t_ot`	u4	1 s		reference time of week for UTC data
`WN_ot`	u1	1 week		UTC reference week number, to which `t_ot` is referenced
`DEL_t_LS`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is not in the past
`WN_LSF`	u1	1 week		Effectivity time of leap second (week)
`DN`	u1	1 day		Effectivity time of leap second (day, from 1 to 7)
`DEL_t_LSF`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is in the past
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GALGstGps`	Number:	4032
	`"OnChange"` interval:	output each time valid GST-GPS offset parameters are received from a Galileo satellite.

This block contains the decoded GPS to Galileo System Time offset parameters. This block is only output if these parameters are valid in the navigation page (i.e. if they are not set to "all ones").

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			SVID of the Galileo satellite from which these parameters have been received (see 4.1.9)
`Source`	u1			Message type from which the data has been decoded: 2: I/NAV 16: F/NAV
`A_1G`	f4	$1 \cdot 10^9$ ns / s		Rate of change of the offset
`A_0G`	f4	$1 \cdot 10^9$ ns		Constant term of the offset
`t_oG`	u4	1 s		Reference time of week
`WN_oG`	u1	1 week		6-bit reference week number.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GALSARRLM`	Number:	4034
	`"OnChange"` interval:	generated each time a SAR RLM message is decoded.

This block contains a decoded Galileo search-and-rescue (SAR) return link message (RLM).

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SVID`	u1			SVID of the Galileo satellite from which this RLM has been received.
`Source`	u1			Message type from which the data has been decoded: 2: I/NAV 16: F/NAV
`RLMLength`	u1			Length of the RLM message in bits. `RLMLength` can be either 80 for a short message or 160 for a long message.
`Reserved`	u1[3]			Reserved for future use, to be ignored by decoding software
`RLMBits`	u4[*N*]			Bits in the RLM message, with the first bit being the MSB of `RLMBits[0]`. *N* is 3 for a short message (i.e. if `RLMLength` is 80), and 5 for a long message (i.e. if `RLMLength` is 160). The 16 unused bits of a short message are set to 0. These bits correspond to the 16 LSBs of `RLMBits[2]`.
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.6 BeiDou Decoded Message Blocks

`BDSNav`	Number:	4081
	`"OnChange"` interval:	block generated each time a new navigation data set is received from a BeiDou satellite

The `BDSNav` block contains the decoded navigation data for one BeiDou satellite, as received from the D1 or D2 nav message.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the BeiDou satellite of which the ephemeris is given in this block (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`WN`	u2	1 week		BeiDou week number as received from the navigation message (from 0 to 8191)
`URA`	u1			User range accuracy index (4-bit value)
`SatH1`	u1			1-bit autonomous health
`IODC`	u1			Age of data, clock (5 bits)
`IODE`	u1			Age of data, ephemeris (5 bits)
`Reserved2`	u2			unused, to be ignored by decoding software
`T_GD1`	f4	1 s		B1I equipment group delay differential
`T_GD2`	f4	1 s	$-2\cdot10^{10}$	B2I equipment group delay differential (set to the Do-Not-Use value when unknown)
`t_oc`	u4	1 s		clock data reference time, in BeiDou system time (lagging GPS time by 14 seconds).
`a_f2`	f4	1 s / s^2		SV clock aging
`a_f1`	f4	1 s / s		SV clock drift
`a_f0`	f4	1 s		SV clock bias
`C_rs`	f4	1 m		Amplitude of the sine harmonic correction term to the orbit radius
`DEL_N`	f4	1 semi-circle / s		Mean motion difference from computed value
`M_0`	f8	1 semi-circle		Mean anomaly at reference time
`C_uc`	f4	1 rad		Amplitude of the cosine harmonic correction term to the argument of latitude
`e`	f8			Eccentricity
`C_us`	f4	1 rad		Amplitude of the sine harmonic correction term to the argument of latitude
`SQRT_A`	f8	1 m$^{1/2}$		Square root of the semi-major axis
`t_oe`	u4	1 s		Reference time ephemeris, in BeiDou system time (lagging GPS time by 14 seconds).
`C_ic`	f4	1 rad		Amplitude of the cosine harmonic correction term to the angle of inclination
`OMEGA_0`	f8	1 semi-circle		Longitude of ascending node of orbit plane at weekly epoch
`C_is`	f4	1 rad		Amplitude of the sine harmonic correction term to the angle of inclination
`i_0`	f8	1 semi-circle		Inclination angle at reference time

`C_rc`	f4	1 m		Amplitude of the cosine harmonic correction term to the orbit radius
`omega`	f8	1 semi-circle		Argument of perigee
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`IDOT`	f4	1 semi-circle / s		Rate of inclination angle
`WNt_oc`	u2	1 week		BeiDou week number associated with t_oc, modulo 8192. Note that this value relates to the BeiDou system time.
`WNt_oe`	u2	1 week		BeiDou week number associated with t_oe, modulo 8192. Note that this values relates to the BeiDou system time.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSCNav2`	Number:	4252
	`"OnChange"`	interval: block generated when a new data set is received

The `BDSCNav2` block contains the B-CNAV2 navigation data decoded from the B2a signal of a BeiDou satellite.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRNidx`	u1			PRN number of the satellite in the BeiDou constellation (1 for C01, 2 for C02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 4.1.9.
`Flags`	u1			Bit field: Bits 0-1: Satellite type (1: GEO, 2: IGSO, 3: MEO). Bits 2-7: Reserved
`t_oe`	u4	1 s		Ephemeris reference time of week
`A`	f8	1 m		Semi-major axis at reference time (ΔA plus A_{ref})
`A_DOT`	f8	1 m / s		Change rate in semi-major axis ($\dot{A}$)
`DELTA_n0`	f4	1 semi-circle / s		Mean motion difference from computed value at reference time
`DELTA_n0_DOT`	f4	1 semi-circle / s^2		Rate of mean motion difference from computed value
`M_0`	f8	1 semi-circle		Mean anomaly
`e`	f8			Eccentricity
`omega`	f8	1 semi-circle		Argument of perigee
`OMEGA_0`	f8	1 semi-circle		Longitude of ascending node
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`i_0`	f8	1 semi-circle		Inclination angle
`IDOT`	f4	1 semi-circle / s		Rate of inclination angle
`C_is`	f4	1 rad		Amplitude of the sine harmonic correction term to the angle of inclination
`C_ic`	f4	1 rad		Amplitude of the cosine harmonic correction term to the angle of inclination
`C_rs`	f4	1 m		Amplitude of the sine harmonic correction term to the orbit radius
`C_rc`	f4	1 m		Amplitude of the cosine harmonic correction term to the orbit radius
`C_us`	f4	1 rad		Amplitude of the sine harmonic correction term to the argument of latitude
`C_uc`	f4	1 rad		Amplitude of the cosine harmonic correction term to the argument of latitude
`t_oc`	u4	1 s		Clock correction parameters reference time
`a_2`	f4	1 s / s^2		Clock drift rate
`a_1`	f4	1 s / s		Clock drift
`a_0`	f8	1 s		Clock bias
`t_op`	u4	1 s		Time of week for data prediction
`SISAI_ocb`	u1			Satellite orbit radius and fixed satellite clock bias accuracy index

`SISAI_oc12`	u1			Bit field: Bits 0-2: SISAI_oc2: Satellite clock drift accuracy index Bits 3-5: SISAI_oc1: Satellite clock bias accuracy index Bits 6-7: Reserved
`SISAI_oe`	u1			Satellite orbit along-track and cross-track accuracy index
`SISMAI`	u1			Signal in space monitoring accuracy index
`HealthIF`	u1			Health and integrity flags from the last message type used in this SBF block: Bit 0: Accuracy integrity flag ($AIF_{(B1C)}$) Bit 1: Signal integrity flag ($SIF_{(B1C)}$) Bit 2: Data integrity flag ($DIF_{(B1C)}$) Bit 3: Accuracy integrity flag ($AIF_{(B2a)}$) Bit 4: Signal integrity flag ($SIF_{(B2a)}$) Bit 5: Data integrity flag ($DIF_{(B2a)}$) Bits 6-7: Satellite health status (0 if healthy)
`IODE`	u1			Issue Of Data Ephemeris
`IODC`	u2			Issue Of Data Clock
`ISC_B2ad`	f4	1 s	$-2\cdot10^{10}$	Group delay differential between the B2a data and pilot components
`T_GDB2ap`	f4	1 s	$-2\cdot10^{10}$	Group delay differential of the B2a pilot component
`T_GDB1Cp`	f4	1 s	$-2\cdot10^{10}$	Group delay differential of the B1C pilot component
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSAlm`	Number:	4119
	`"OnChange"` interval:	block generated each time a new almanac data set is received from a BeiDou satellite

The `BDSAlm` block contains the decoded almanac data for one BeiDou satellite.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the BeiDou satellite of which the almanac is given in this block (see 4.1.9)
`WN_a`	u1	1 week		Almanac week number
`t_oa`	u4	1 s		Almanac reference time
`SQRT_A`	f4	1 $m^{1/2}$		Square root of the semi-major axis
`e`	f4			Eccentricity
`omega`	f4	1 semi-circle		Argument of perigee
`M_0`	f4	1 semi-circle		Mean anomaly at reference time
`OMEGA_0`	f4	1 semi-circle		Longitude of ascending node of orbital plane computed according to reference time
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`delta_i`	f4	1 semi-circle		Correction of orbit reference inclination at reference time
`a_f0`	f4	1 s		Satellite clock bias
`a_f1`	f4	1 s / s		Satellite clock drift
`Health`	u2			Satellite health information (9 bits)
`Reserved`	u1[2]			Reserved for future use, to be ignored by decoding software
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSIon`	Number:	4120
	`"OnChange"` interval:	output each time the ionospheric parameters are received from a BeiDou satellite

The `BDSIon` block contains the BeiDou ionosphere data (the Klobuchar coefficients), as received from the D1 or D2 nav message.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the BeiDou satellite from which the coefficients have been received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`alpha_0`	f4	1 s		vertical delay coefficient 0
`alpha_1`	f4	1 s / semi-circle		vertical delay coefficient 1
`alpha_2`	f4	1 s / semi-circle^2		vertical delay coefficient 2
`alpha_3`	f4	1 s / semi-circle^3		vertical delay coefficient 3
`beta_0`	f4	1 s		model period coefficient 0
`beta_1`	f4	1 s / semi-circle		model period coefficient 1
`beta_2`	f4	1 s / semi-circle^2		model period coefficient 2
`beta_3`	f4	1 s / semi-circle^3		model period coefficient 3
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BDSUtc`	Number:	4121
	`"OnChange"` interval:	output each time the UTC offset parameters are received from a BeiDou satellite

The `BDSUtc` block contains the BeiDou UTC data, as received from the D1 or D2 nav message.

Note that BDT (BeiDou time) started on January 1st, 2006 (GPS week 1356). Therefore the delta time between BDT and UTC due to leap seconds is 14 less than the value in `GPSUtc`.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the BeiDou satellite from which the coefficients have been received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`A_1`	f4	1 s / s		first order term of polynomial
`A_0`	f8	1 s		constant term of polynomial
`DEL_t_LS`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is not in the past
`WN_LSF`	u1	1 week		Effectivity time of leap second (week)
`DN`	u1	1 day		Effectivity time of leap second (day, from 0 to 6)
`DEL_t_LSF`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is in the past
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.7 QZSS Decoded Message Blocks

`QZSNav`	Number:	4095
	`"OnChange"` interval:	block generated each time a new navigation data set is received from a QZSS satellite

The `QZSNav` block contains the decoded navigation data for one QZSS satellite. The data is decoded from the navigation message transmitted by the L1C/A or L1C/B signal. Refer to the QZSS ICD for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the QZSS satellite of which the ephemeris is given in this block (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`WN`	u2	1 week	65535	Week number (10 bits from subframe 1, word 3)
`CAorPonL2`	u1			Code(s) on L2 channel (2 bits from subframe 1, word 3). Always 2 for QZSS satellites.
`URA`	u1			User Range accuracy index (4 bits from subframe 1 word 3)
`health`	u1			6-bit health from subframe 1, word 3 (6 bits from subframe 1, word 3)
`L2DataFlag`	u1			Data flag for L2 P-code (1 bit from subframe 1, word 4). Always 1 for QZSS satellites.
`IODC`	u2			Issue of data, clock (10 bits from subframe 1)
`IODE2`	u1			Issue of data, ephemeris (8 bits from subframe 2)
`IODE3`	u1			Issue of data, ephemeris (8 bits from subframe 3)
`FitIntFlg`	u1			Curve Fit Interval, (1 bit from subframe 2, word 10)
`Reserved2`	u1			unused, to be ignored by decoding software
`T_gd`	f4	1 s	$-2 \cdot 10^{10}$	Estimated group delay differential
`t_oc`	u4	1 s		clock data reference time
`a_f2`	f4	1 s / s^2		SV clock aging
`a_f1`	f4	1 s / s		SV clock drift
`a_f0`	f4	1 s		SV clock bias
`C_rs`	f4	1 m		Amplitude of the sine harmonic correction term to the orbit radius
`DEL_N`	f4	1 semi-circle / s		Mean motion difference from computed value
`M_0`	f8	1 semi-circle		Mean anomaly at reference time
`C_uc`	f4	1 rad		Amplitude of the cosine harmonic correction term to the argument of latitude
`e`	f8			Eccentricity
`C_us`	f4	1 rad		Amplitude of the sine harmonic correction term to the argument of latitude
`SQRT_A`	f8	1 m$^{1/2}$		Square root of the semi-major axis
`t_oe`	u4	1 s		Reference time ephemeris
`C_ic`	f4	1 rad		Amplitude of the cosine harmonic correction term to the angle of inclination
`OMEGA_0`	f8	1 semi-circle		Longitude of ascending node of orbit plane at weekly epoch

`C_is`	f4	1 rad		Amplitude of the sine harmonic correction term to the angle of inclination
`i_0`	f8	1 semi-circle		Inclination angle at reference time
`C_rc`	f4	1 m		Amplitude of the cosine harmonic correction term to the orbit radius
`omega`	f8	1 semi-circle		Argument of perigee
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`IDOT`	f4	1 semi-circle / s		Rate of inclination angle
`WNt_oc`	u2	1 week		WN associated with t_oc, modulo 1024
`WNt_oe`	u2	1 week		WN associated with t_oe, modulo 1024
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QZSAlm`	Number:	4116
	`"OnChange"` interval:	block generated each time a new almanac data set is received from a QZSS satellite

The `QZSAlm` block contains the decoded almanac data for one QZSS satellite. These data are conveyed in subframes 4 and 5 of the satellite navigation message. Refer to QZSS ICD for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the QZSS satellite of which the almanac is given in this block (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`e`	f4			Difference from reference eccentricity
`t_oa`	u4	1 s		almanac reference time of week
`delta_i`	f4	1 semi-circle		Difference from reference angle of inclination
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`SQRT_A`	f4	1 $m^{1/2}$		Square root of the semi-major axis
`OMEGA_0`	f4	1 semi-circle		Longitude of ascending node of orbit plane at weekly epoch
`omega`	f4	1 semi-circle		Argument of perigee
`M_0`	f4	1 semi-circle		Mean anomaly at reference time
`a_f1`	f4	1 s / s		SV clock drift
`a_f0`	f4	1 s		SV clock bias
`WN_a`	u1	1 week		Almanac reference week, to which t_oa is referenced
`Reserved2`	u1			Reserved for future use, to be ignored by decoding software
`health8`	u1			health on 8 bits from the almanac page
`health6`	u1			health summary on 6 bits (from subframe 4, page 25 and subframe 5 page 25)
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.8 NavIC/IRNSS Decoded Message Blocks

`NavICLNav`	Number:	4254
	`"OnChange"` interval:	block generated each time a new navigation data set is received from a NavIC satellite

The `NavICLNav` block contains the decoded LNAV navigation data for one NavIC/IRNSS satellite.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRNidx`	u1			PRN number of the satellite in the NavIC/IRNSS constellation (1 for I01, 2 for I02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 4.1.9.
`IODEC`	u1			Issue of data, ephemeris and clock
`t_oe`	u4	1 s		Ephemeris reference time of week
`SQRT_A`	f8	1 $m^{1/2}$		Square root of the semi-major axis
`DELTA_N`	f4	1 semi-circle / s		Mean motion difference from computed value at reference time
`M_0`	f8	1 semi-circle		Mean anomaly
`e`	f8			Eccentricity
`omega`	f8	1 semi-circle		Argument of perigee
`OMEGA_0`	f8	1 semi-circle		Longitude of ascending node
`OMEGADOT`	f4	1 semi-circle / s		Rate of right ascension
`i_0`	f8	1 semi-circle		Inclination angle
`IDOT`	f4	1 semi-circle / s		Rate of inclination angle
`C_is`	f4	1 rad		Amplitude of the sine harmonic correction term to the angle of inclination
`C_ic`	f4	1 rad		Amplitude of the cosine harmonic correction term to the angle of inclination
`C_rs`	f4	1 m		Amplitude of the sine harmonic correction term to the orbit radius
`C_rc`	f4	1 m		Amplitude of the cosine harmonic correction term to the orbit radius
`C_us`	f4	1 rad		Amplitude of the sine harmonic correction term to the argument of latitude
`C_uc`	f4	1 rad		Amplitude of the cosine harmonic correction term to the argument of latitude
`t_oc`	u4	1 s		Clock correction parameters reference time
`a_f0`	f4	1 s		Clock bias
`a_f1`	f4	1 s / s		Clock drift
`a_f2`	f4	1 s / s^2		Clock drift rate
`T_GD`	f4	1 s		Total group delay between L5-SPS and S-SPS signals
`Flags`	u1			Health, Alert and AutoNav flags: Bit 0: S-SPS health flag (0 if healthy) Bit 1: L5-SPS health flag (0 if healthy) Bit 2: Current value of the Alert flag Bit 3: Current value of the AutoNav flag Bits 4-7: Reserved
`URA`	u1			User Range accuracy index
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.9 SBAS L1 Decoded Message Blocks

`GEOMT00`	Number:		5925
	`"OnChange"`	interval:	block generated each time an empty MT00 is received from an SBAS satellite on the L1 signal

This block is sent to indicate that an empty SBAS message type 0 has been received.

Depending on the SBAS operational mode, message type 0 can contain the contents of message type 2. Upon reception of a message type 0, the receiver checks whether the message is empty (it contains only 0's) or whether it contains the message type 2 contents. In the former case, a `GEOMT00` block will be generated. In the latter case, a `GEOFastCorr` block will be generated. Refer to section A.4.4.1 of the DO 229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEOPRNMask`	Number:	5926
	`"OnChange"` interval:	block generated each time MT01 is received from an SBAS satellite

This block contains the decoded PRN mask transmitted in SBAS message type 1. Refer to section A.4.4.2 of the DO 229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`IODP`	u1			Issue of data - PRN.
`NbrPRNs`	u1			Number of PRNs designated in the mask.
`PRNMask`	u1[NbrPRNs]			List of the PRNs in the PRN mask. `PRNMask[0]` is the first PRN designated in the PRN mask (from 1 to 210), `PRNMask[1]` is the 2^{nd} PRN designated in the PRN mask, etc...
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEOFastCorr`	Number:	5927
	`"OnChange"` interval:	block generated each time MT02, MT03, MT04, MT05, MT24 and possibly MT00 is received from an SBAS satellite

This block contains the decoded fast corrections transmitted in the SBAS message types 2, 3, 4, 5, 24 and possibly 0 if the type 0 message contains the type 2 contents. Refer to section A.4.4.3 and A.4.4.8 of the DO 229 standard for further details.

<table>
<tr><th>Parameter</th><th>Type</th><th>Units</th><th>Do-Not-Use</th><th>Description</th></tr>
<tr><td><code>Sync1</code></td><td>c1</td><td></td><td></td><td>Block Header, see 4.1.1</td></tr>
<tr><td><code>Sync2</code></td><td>c1</td><td></td><td></td><td></td></tr>
<tr><td><code>CRC</code></td><td>u2</td><td></td><td></td><td></td></tr>
<tr><td><code>ID</code></td><td>u2</td><td></td><td></td><td></td></tr>
<tr><td><code>Length</code></td><td>u2</td><td>1 byte</td><td></td><td></td></tr>
<tr><td><code>TOW</code></td><td>u4</td><td>0.001 s</td><td>4294967295</td><td>SIS time stamp, see 4.1.3</td></tr>
<tr><td><code>WNc</code></td><td>u2</td><td>1 week</td><td>65535</td><td></td></tr>
<tr><td><code>PRN</code></td><td>u1</td><td></td><td></td><td>ID of the SBAS satellite from which the message has been received (see 4.1.9)</td></tr>
<tr><td><code>MT</code></td><td>u1</td><td></td><td></td><td>Message type from which these fast corrections come, either 0, 2, 3, 4, 5 or 24.</td></tr>
<tr><td><code>IODP</code></td><td>u1</td><td></td><td></td><td>Issue of data - PRN.</td></tr>
<tr><td><code>IODF</code></td><td>u1</td><td></td><td></td><td>Issue of data - fast corrections.</td></tr>
<tr><td><code>N</code></td><td>u1</td><td></td><td></td><td>Number of fast correction sets in this message. This is the number of <code>FastCorr</code> sub-blocks. <code>N</code> depends on the message type as follows.
<table>
<tr><th>Message type</th><th>N</th></tr>
<tr><td>MT00, MT02, MT03, MT04</td><td>13</td></tr>
<tr><td>MT05</td><td>12</td></tr>
<tr><td>MT24</td><td>6</td></tr>
</table></td></tr>
<tr><td><code>SBLength</code></td><td>u1</td><td></td><td></td><td>Length of the <code>FastCorr</code> sub-blocks in bytes</td></tr>
<tr><td><i>FastCorr</i></td><td>...</td><td>...</td><td></td><td><i>A succession of N <code>FastCorr</code> sub-blocks, see definition below</i></td></tr>
<tr><td><code>Padding</code></td><td>u1[..]</td><td></td><td></td><td>Padding bytes, see 4.1.5</td></tr>
</table>

`FastCorr` sub-block definition:

Parameter	Type	Units	Description
`PRNMaskNo`	u1		Sequence number in the PRN mask. This field may be set to zero. In that case, all following fields in this sub-block must be discarded.
`UDREI`	u1		User Differential Range Error Indicator for the PRN at index `PRNMaskNo`.
`Reserved`	u1[2]		Reserved for future use, to be ignored by decoding software
`PRC`	f4	1 m	Pseudorange correction for the PRN at index `PRNMaskNo`.
`Padding`	u1[..]		Padding bytes, see 4.1.5

`GEOIntegrity`	Number:	5928
	`"OnChange"` interval:	block generated each time MT06 is received from an SBAS satellite

This block contains the decoded integrity information transmitted in SBAS message type 6. Refer to section A.4.4.4 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`IODF`	u1[4]			Issue of data - fast corrections for MT02, MT03, MT04 and MT05.
`UDREI`	u1[51]			User Differential Range Error Indicator for each of the 51 slots in the PRN mask.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEOFastCorrDegr`	Number:	5929
	`"OnChange"` interval:	block generated each time MT07 is received from an SBAS satellite

This block contains the decoded fast correction degradation factors transmitted in SBAS message type 7. Refer to section A.4.4.5 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`IODP`	u1			Issue of data - PRN.
`t_lat`	u1	1 s		System latency.
`ai`	u1[51]			Degradation factor indicator (from 0 to 15) for each of the 51 slots in the PRN mask.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEONav`	Number:	5896
	`"OnChange"` interval:	block generated each time MT09 is received from an SBAS satellite

This block contains the decoded navigation data transmitted in SBAS message type 9. Refer to section A.4.4.11 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite of which the navigation data is provided here (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`IODN`	u2			Issue of data - navigation (DO 229-B) Spare (DO 229-C)
`URA`	u2			Accuracy exponent
`t0`	u4	1 s		Time of applicability (time-of-day)
`Xg`	f8	1 m		X position at time-of-day `t0`
`Yg`	f8	1 m		Y position at time-of-day `t0`
`Zg`	f8	1 m		Z position at time-of-day `t0`
`Xgd`	f8	1 m / s		X velocity at time-of-day `t0`
`Ygd`	f8	1 m / s		Y velocity at time-of-day `t0`
`Zgd`	f8	1 m / s		Z velocity at time-of-day `t0`
`Xgdd`	f8	1 m / s^2		X acceleration at time-of-day `t0`
`Ygdd`	f8	1 m / s^2		Y acceleration at time-of-day `t0`
`Zgdd`	f8	1 m / s^2		Z acceleration at time-of-day `t0`
`aGf0`	f4	1 s		Time offset with respect to SBAS network time
`aGf1`	f4	1 s / s		Time drift with respect to SBAS network time
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEODegrFactors`	Number:	5930
	`"OnChange"` interval:	block generated each time MT10 is received from an SBAS satellite

This block contains the decoded degradation factors transmitted in SBAS message type 10. Refer to section A.4.5 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`Brrc`	f8	1 m		A parameter associated with the relative estimation noise and round-off error.
`Cltc_lsb`	f8	1 m		Maximum round-off error due to the LSB resolution of the orbit and clock information.
`Cltc_v1`	f8	1 m / s		Velocity error bound on the maximum range rate difference of missed messages due to clock and orbit rate differences.
`Iltc_v1`	u4	1 s		Update interval for long term corrections when the velocity code is 1.
`Cltc_v0`	f8	1 m		Bound on the update delta between successive long term corrections.
`Iltc_v0`	u4	1 s		Minimum update interval for long term messages when the velocity code is 0.
`Cgeo_lsb`	f8	1 m		Maximum round-off error due to the LSB resolution of the orbit and clock information.
`Cgeo_v`	f8	1 m / s		Velocity error bound on the maximum range rate difference of missed messages due to clock and orbit rate differences.
`Igeo`	u4	1 s		Update interval for GEO navigation messages.
`Cer`	f4	1 m		A degradation parameter.
`Ciono_step`	f8	1 m		Bound on the difference between successive ionospheric grid delay values.
`Iiono`	u4	1 s		Minimum update interval for ionospheric correction messages.
`Ciono_ramp`	f8	1 m / s		Rate of change of the ionospheric corrections.
`RSSudre`	u1			Root-sum-square flag (UDRE)
`RSSiono`	u1			Root-sum-square flag (IONO)
`Reserved2`	u1[2]			Reserved for future use, to be ignored by decoding software
`Ccovariance`	f8			A parameter used to compensate for the errors introduced by quantization (introduced in DO 229-C). To be multiplied by the `SF` parameter from the `GEOClockEphCovMatrix` block.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEONetworkTime`	Number:	5918
	`"OnChange"` interval:	block generated each time MT12 is received from an SBAS satellite

This block contains the decoded network time offset parameters transmitted in SBAS message type 12. Refer to section A.4.4.15 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which this Network Time data was received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`A_1`	f4	1 s / s		first order term of polynomial
`A_0`	f8	1 s		constant term of polynomial
`t_ot`	u4	1 s		reference time for UTC data (time of week)
`WN_t`	u1	1 week		UTC reference week number, to which `t_ot` is referenced
`DEL_t_LS`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is not in the past
`WN_LSF`	u1	1 week		Effectivity time of leap second (week)
`DN`	u1	1 day		Effectivity time of leap second (day)
`DEL_t_LSF`	i1	1 s		Delta time due to leap seconds whenever the effectivity time is in the past
`UTC_std`	u1			UTC Standard Identifier
`GPS_WN`	u2	1 week		GPS week number (modulo 1024)
`GPS_TOW`	u4	1 s		GPS time-of-week
`GlonassID`	u1			Glonass Indicator
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEOAlm`	Number:	5897
	`"OnChange"` interval:	block generated each time MT17 is received from an SBAS satellite

This block contains the decoded almanac data for one SBAS satellite, as transmitted in SBAS message type 17. A different `GEOAlm` block is generated for each of the up to three almanac data sets in MT17. Refer to section A.4.4.12 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite of which the almanac is provided here (see 4.1.9)
`Reserved0`	u1			Reserved for future use, to be ignored by decoding software
`DataID`	u1			Data ID
`Reserved1`	u1			Reserved for future use, to be ignored by decoding software
`Health`	u2			Health bits
`t_oa`	u4	1 s		Time of applicability with the day ambiguity resolved. This is the time in GPS seconds from Jan 6th, 1980.
`Xg`	f8	1 m		X position at `t_oa`
`Yg`	f8	1 m		Y position at `t_oa`
`Zg`	f8	1 m		Z position at `t_oa`
`Xgd`	f8	1 m / s		X velocity at `t_oa`
`Ygd`	f8	1 m / s		Y velocity at `t_oa`
`Zgd`	f8	1 m / s		Z velocity at `t_oa`
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEOIGPMask`	Number:	5931
	`"OnChange"` interval:	block generated each time MT18 is received from an SBAS satellite

This block contains the decoded ionospheric grid point mask transmitted in SBAS message type 18. Refer to section A.4.4.9 of the DO-229 standard for further details.

Parameter	**Type**	**Units**	**Do-Not-Use**	**Description**
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`NbrBands`	u1			Number of bands being broadcast.
`BandNbr`	u1			Band number.
`IODI`	u1			Issue of data - ionosphere.
`NbrIGPs`	u1			Number of ionospheric grid points (IGP) designated in the mask.
`IGPMask`	u1[NbrIGPs]			List of the IGPs in the IGP mask. `IGPMask[0]` is the first IGP designated in the IGP mask (from 1 to 201), `IGPMask[1]` is the 2nd IGP designated in the IGP mask, etc...
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GEOLongTermCorr`	Number:	5932
	`"OnChange"` interval:	block generated each time MT24 or MT25 is received from an SBAS satellite

This block contains the decoded long term corrections transmitted in SBAS message types 24 and 25. Refer to section A.4.4.7 and A.4.4.8 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`N`	u1			Number of long-term corrections in this message. This is the number of `LTCorr` sub-blocks. `N` can be 0, 1, 2, 3 or 4.
`SBLength`	u1	1 byte		Length of the `LTCorr` sub-blocks in bytes
`Reserved`	u1[3]			Reserved for future use, to be ignored by decoding software
LTCorr	...	...		*A succession of N `LTCorr` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`LTCorr` sub-block definition:

Parameter	Type	Units	Description
`VelocityCode`	u1		Velocity code (0 or 1)
`PRNMaskNo`	u1		Sequence in the PRN mask, from 1 to 51. Note that if the PRN mask No. from the original message is 0, the corresponding long term corrections are ignored, and hence not included in the `GEOLongTermCorr` block.
`IODP`	u1		Issue of data - PRN.
`IODE`	u1		Issue of data - ephemeris.
`dx`	f4	1 m	Satellite position offset (x).
`dy`	f4	1 m	Satellite position offset (y).
`dz`	f4	1 m	Satellite position offset (z).
`dxRate`	f4	1 m / s	Satellite velocity offset (x), or 0.0 if `VelocityCode` is 0.
`dyRate`	f4	1 m / s	Satellite velocity offset (y), or 0.0 if `VelocityCode` is 0.
`dzRate`	f4	1 m / s	Satellite velocity offset (z), or 0.0 if `VelocityCode` is 0.
`da_f0`	f4	1 s	Satellite clock offset.
`da_f1`	f4	1 s / s	Satellite drift correction, or 0.0 if `VelocityCode` is 0.
`t_oe`	u4	1 s	Time-of-day of applicability, or 0 if `VelocityCode` is 0.
`Padding`	u1[..]		Padding bytes, see 4.1.5

GEOIonoDelay	Number:	5933
	"OnChange" interval:	block generated each time MT26 is received from an SBAS satellite

This block contains the decoded ionospheric delays transmitted in SBAS message type 26. Refer to section A.4.4.10 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which the message has been received (see 4.1.9)
`BandNbr`	u1			Band number
`IODI`	u1			Issue of data - ionosphere.
`N`	u1			Number of ionospheric delay corrections in this message. This is the number of `IDC` sub-blocks. `N` is always 15.
`SBLength`	u1	1 byte		Length of the `IDC` sub-blocks in bytes.
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
IDC	...	...		*A succession of N `IDC` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`IDC` sub-block definition:

Parameter	Type	Units	Description
`IGPMaskNo`	u1		Sequence number in the IGP mask (see `GEOIGPMask` block), from 1 to 201.
`GIVEI`	u1		Grid Ionospheric Vertical Error Indicator, from 0 to 15
`Reserved`	u1[2]		Reserved for future use, to be ignored by decoding software
`VerticalDelay`	f4	1 m	IGP vertical delay estimate.
`Padding`	u1[..]		Padding bytes, see 4.1.5

`GEOServiceLevel`	Number: 5917 "`OnChange`" interval: block generated each time MT27 is received from an SBAS satellite

This block contains a decoded service level message for a geostationary SBAS satellite as sent in message type 27. Refer to section A.4.4.13 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			ID of the SBAS satellite from which this service level message was received (see 4.1.9)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`IODS`	u1			Issue of Data Service level, ranging from 0 to 7
`nrMessages`	u1			Number of service messages (MT27), from 1 to 8
`MessageNR`	u1			Service message number, from 1 to 8
`PriorityCode`	u1			Priority Code, from 0 to 3
`dUDREI_In`	u1			δUDRE Indicator for users inside the service region, from 0 to 15
`dUDREI_Out`	u1			δUDRE Indicator for users outside the service region, from 0 to 15
`N`	u1			Number of Regions in this message. This is the number of `ServiceRegion` sub-blocks. Ranging from 0 to 7
`SBLength`	u1	1 byte		Length of the `ServiceRegion` sub-blocks in bytes
Regions	...	...		*A succession of N `ServiceRegion` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`ServiceRegion` sub-block definition:

Parameter	Type	Units	Description
`Latitude1`	i1	1 degree	Coordinate 1 latitude, from -90 to +90
`Latitude2`	i1	1 degree	Coordinate 2 latitude, from -90 to +90
`Longitude1`	i2	1 degree	Coordinate 1 longitude, from -180 to +180
`Longitude2`	i2	1 degree	Coordinate 2 longitude, from -180 to +180
`RegionShape`	u1		Region Shape: 0=triangular, 1=square
`Padding`	u1[..]		Padding bytes, see 4.1.5

`GEOClockEphCovMatrix`	Number:	5934
	`"OnChange"` interval:	block generated each time MT28 is received from an SBAS satellite

This block contains the decoded clock-ephemeris covariance Cholesky factor matrix transmitted in SBAS message type 28. Refer to section A.4.4.16 of the DO-229 standard for further details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	SIS time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PRN`	u1			Satellite ID, see 4.1.9
`IODP`	u1			Issue of data - PRN.
`N`	u1			Number of covariance matrices in this message. This is the number of `CovMatrix` sub-blocks. `N` can be 1 or 2.
`SBLength`	u1	1 byte		Length of the `CovMatrix` sub-blocks in bytes
`Reserved`	u1[2]			Reserved for future use, to be ignored by decoding software
CovMatrix	...	...		*A succession of N* `CovMatrix` *sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`CovMatrix` sub-block definition:

Parameter	Type	Units	Description
`PRNMaskNo`	u1		Sequence number in the PRN mask, from 1 to 51. Note that if the PRN mask No. from the original message is 0, the corresponding matrix is ignored, and hence not included in the `GEOClockEphCovMatrix` block.
`Reserved`	u1[2]		Reserved for future use, to be ignored by decoding software
`ScaleExp`	u1		Scale exponent; scale factor (= $2^{(\text{scale exponent - 5})}$)
`E11`	u2		$E_{1,1}$
`E22`	u2		$E_{2,2}$
`E33`	u2		$E_{3,3}$
`E44`	u2		$E_{4,4}$
`E12`	i2		$E_{1,2}$
`E13`	i2		$E_{1,3}$
`E14`	i2		$E_{1,4}$
`E23`	i2		$E_{2,3}$
`E24`	i2		$E_{2,4}$
`E34`	i2		$E_{3,4}$
`Padding`	u1[..]		Padding bytes, see 4.1.5

4.2.10 GNSS Position, Velocity and Time Blocks

`PVTCartesian`	Number:	4006
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the GNSS-based position, velocity and time (PVT) solution at the time specified in the `TOW` and `WNc` fields. The time of applicability is specified in the receiver time frame.

The computed position (x, y, z) and velocity (v_x, v_y, v_z) are reported in a Cartesian coordinate system using the datum indicated in the `Datum` field. The position is that of the marker. The ARP-to-marker offset is set through the command **`setAntennaOffset`**.

The PVT solution is also available in ellipsoidal form in the `PVTGeodetic` block.

The variance-covariance information associated with the reported PVT solution can be found in the `PosCovCartesian` and `VelCovCartesian` blocks.

If no PVT solution is available, the `Error` field indicates the cause of the unavailability and all fields after the `Error` field are set to their respective Do-Not-Use values.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **setPVTMode, Static, , auto** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`X`	f8	1 m	$-2 \cdot 10^{10}$	X coordinate in coordinate frame specified by `Datum`
`Y`	f8	1 m	$-2 \cdot 10^{10}$	Y coordinate in coordinate frame specified by `Datum`
`Z`	f8	1 m	$-2 \cdot 10^{10}$	Z coordinate in coordinate frame specified by `Datum`
`Undulation`	f4	1 m	$-2 \cdot 10^{10}$	Geoid undulation. See the **setGeoidUndulation** command.
`Vx`	f4	1 m / s	$-2 \cdot 10^{10}$	Velocity in the X direction
`Vy`	f4	1 m / s	$-2 \cdot 10^{10}$	Velocity in the Y direction
`Vz`	f4	1 m / s	$-2 \cdot 10^{10}$	Velocity in the Z direction
`COG`	f4	1 degree	$-2 \cdot 10^{10}$	Course over ground: this is defined as the angle of the vehicle with respect to the local level North, ranging from 0 to 360, and increasing towards east. Set to the Do-Not-Use value when the speed is lower than 0.1m/s.
`RxClkBias`	f8	1 ms	$-2 \cdot 10^{10}$	Receiver clock bias relative to the GNSS system time reported in the `TimeSystem` field. Positive when the receiver time is ahead of the system time. To transfer the receiver time to the system time, use: $t_{GPS/GST} = t_{rx}$ - `RxClkBias`
`RxClkDrift`	f4	1 ppm	$-2 \cdot 10^{10}$	Receiver clock drift relative to the GNSS system time (relative frequency error). Positive when the receiver clock runs faster than the system time.

`TimeSystem`	u1		255	Time system of which the offset is provided in this sub-block: 0: GPS time 1: Galileo time 3: GLONASS time 4: BeiDou time 5: QZSS time
`Datum`	u1		255	This field defines in which datum the coordinates are expressed: 0: WGS84/ITRS 19: Datum equal to that used by the DGNSS/RTK base station 30: ETRS89 (ETRF2000 realization) 31: NAD83(2011), North American Datum (2011) 32: NAD83(PA11), North American Datum, Pacific plate (2011) 33: NAD83(MA11), North American Datum, Marianas plate (2011) 34: GDA94(2010), Geocentric Datum of Australia (2010) 35: GDA2020, Geocentric Datum of Australia 2020 36: JGD2011, Japanese Geodetic Datum 2011 250: First user-defined datum 251: Second user-defined datum
`NrSV`	u1		255	Total number of satellites used in the PVT computation.
`WACorrInfo`	u1		0	Bit field providing information about which wide area corrections have been applied: Bit 0: set if orbit and satellite clock correction information is used Bit 1: set if range correction information is used Bit 2: set if ionospheric information is used Bit 3: set if orbit accuracy information is used (UERE/SISA) Bit 4: set if DO229 Precision Approach mode is active Bits 5-7: Reserved
`ReferenceID`	u2		65535	This field indicates the reference ID of the differential information used. In case of DGPS or RTK operation, this field is to be interpreted as the base station identifier. In SBAS operation, this field is to be interpreted as the PRN of the geostationary satellite used (from 120 to 158). If multiple base stations or multiple geostationary satellites are used the value is set to 65534.
`MeanCorrAge`	u2	0.01 s	65535	In case of DGPS or RTK, this field is the mean age of the differential corrections. In case of SBAS operation, this field is the mean age of the 'fast corrections' provided by the SBAS satellites. In case of PPP, this is the age of the last received clock or orbit correction message.
`SignalInfo`	u4		0	Bit field indicating the type of GNSS signals having been used in the PVT computations. If a bit *i* is set, the signal type having index *i* has been used. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.
`AlertFlag`	u1		0	Bit field indicating integrity related information: Bits 0-1: RAIM integrity flag: 0: RAIM not active (integrity not monitored) 1: RAIM integrity test successful 2: RAIM integrity test failed 3: Reserved Bit 2: set if integrity has failed as per Galileo HPCA (HMI Probability Computation Algorithm) Bit 3: set if Galileo ionospheric storm flag is active Bit 4: Reserved Bits 5-7: Reserved
`NrBases`	u1		0	Number of base stations used in the PVT computation.

Rev 1	`PPPInfo`	u2	1 s	0	Bit field containing PPP-related information: Bits 0-11: Age of the last seed, in seconds. The age is clipped to 4091s. This field must be ignored when the seed type is 0 (see bits 13-15 below). Bit 12: Reserved Bits 13-15: Type of last seed: 0: Not seeded or not in PPP positioning mode 1: Manual seed 2: Seeded from DGPS 3: Seeded from RTKFixed
Rev 2	`Latency`	u2	0.0001 s	65535	Time elapsed between the time of applicability of the position fix and the generation of this SBF block by the receiver. This time includes the receiver processing time, but not the communication latency.
	`HAccuracy`	u2	0.01 m	65535	2DRMS horizontal accuracy: twice the root-mean-square of the horizontal distance error. The horizontal distance between the true position and the computed position is expected to be lower than `HAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m
	`VAccuracy`	u2	0.01 m	65535	2-sigma vertical accuracy. The vertical distance between the true position and the computed position is expected to be lower than `VAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m.
	`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: In DGNSS or RTK mode, set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating whether the marker position reported in this block is also the ARP position (i.e. whether the ARP-to-marker offset provided with the **`setAntennaOffset`** command is zero or not) 0: Unknown 1: The ARP-to-marker offset is zero 2: The ARP-to-marker offset is not zero
	`Padding`	u1[..]			Padding bytes, see 4.1.5

`PVTGeodetic`	Number: 4007 `"OnChange"` interval: default PVT output rate (see 4.1.8)

This block contains the GNSS-based position, velocity and time (PVT) solution at the time specified in the `TOW` and `WNc` fields. The time of applicability is specified in the receiver time frame.

The computed position (ϕ,λ,h) and velocity (v_n, v_e, v_u) are reported in an ellipsoidal co-ordinate system using the datum indicated in the `Datum` field. The velocity vector is expressed relative to the local-level Cartesian coordinate frame with north-, east-, up-unit vectors. The position is that of the marker. The ARP-to-marker offset is set through the command **`setAntennaOffset`**.

The PVT solution is also available in Cartesian form in the `PVTCartesian` block.

The variance-covariance information associated with the reported PVT solution can be found in the `PosCovGeodetic` and `VelCovGeodetic` blocks.

If no PVT solution is available, the `Error` field indicates the cause of the unavailability and all fields after the `Error` field are set to their respective Do-Not-Use values.

Parameter	Type	Units	Do-Not-Use	Description
Sync1	c1			Block Header, see 4.1.1
Sync2	c1			
CRC	u2			
ID	u2			
Length	u2	1 byte		
TOW	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
WNc	u2	1 week	65535	
Mode	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **setPVTMode, Static, , auto** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
Error	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
Latitude	f8	1 rad	$-2\cdot10^{10}$	Latitude, from $-\pi/2$ to $+\pi/2$, positive North of Equator
Longitude	f8	1 rad	$-2\cdot10^{10}$	Longitude, from $-\pi$ to $+\pi$, positive East of Greenwich
Height	f8	1 m	$-2\cdot10^{10}$	Ellipsoidal height (with respect to the ellipsoid specified by `Datum`)
Undulation	f4	1 m	$-2\cdot10^{10}$	Geoid undulation. See the **setGeoidUndulation** command.
Vn	f4	1 m / s	$-2\cdot10^{10}$	Velocity in the North direction
Ve	f4	1 m / s	$-2\cdot10^{10}$	Velocity in the East direction
Vu	f4	1 m / s	$-2\cdot10^{10}$	Velocity in the 'Up' direction
COG	f4	1 degree	$-2\cdot10^{10}$	Course over ground: this is defined as the angle of the vehicle with respect to the local level North, ranging from 0 to 360, and increasing towards east. Set to the Do-Not-Use value when the speed is lower than 0.1m/s.
RxClkBias	f8	1 ms	$-2\cdot10^{10}$	Receiver clock bias relative to the GNSS system time reported in the `TimeSystem` field. Positive when the receiver time is ahead of the system time. To transfer the receiver time to the system time, use: $t_{GPS/GST} = t_{rx}$ - `RxClkBias`
RxClkDrift	f4	1 ppm	$-2\cdot10^{10}$	Receiver clock drift relative to the GNSS system time (relative frequency error). Positive when the receiver clock runs faster than the system time.

`TimeSystem`	u1		255	Time system of which the offset is provided in this sub-block: 0: GPS time 1: Galileo time 3: GLONASS time 4: BeiDou time 5: QZSS time
`Datum`	u1		255	This field defines in which datum the coordinates are expressed: 0: WGS84/ITRS 19: Datum equal to that used by the DGNSS/RTK base station 30: ETRS89 (ETRF2000 realization) 31: NAD83(2011), North American Datum (2011) 32: NAD83(PA11), North American Datum, Pacific plate (2011) 33: NAD83(MA11), North American Datum, Marianas plate (2011) 34: GDA94(2010), Geocentric Datum of Australia (2010) 35: GDA2020, Geocentric Datum of Australia 2020 36: JGD2011, Japanese Geodetic Datum 2011 250: First user-defined datum 251: Second user-defined datum
`NrSV`	u1		255	Total number of satellites used in the PVT computation.
`WACorrInfo`	u1		0	Bit field providing information about which wide area corrections have been applied: Bit 0: set if orbit and satellite clock correction information is used Bit 1: set if range correction information is used Bit 2: set if ionospheric information is used Bit 3: set if orbit accuracy information is used (UERE/SISA) Bit 4: set if DO229 Precision Approach mode is active Bits 5-7: Reserved
`ReferenceID`	u2		65535	This field indicates the reference ID of the differential information used. In case of DGPS or RTK operation, this field is to be interpreted as the base station identifier. In SBAS operation, this field is to be interpreted as the PRN of the geostationary satellite used (from 120 to 158). If multiple base stations or multiple geostationary satellites are used the value is set to 65534.
`MeanCorrAge`	u2	0.01 s	65535	In case of DGPS or RTK, this field is the mean age of the differential corrections. In case of SBAS operation, this field is the mean age of the 'fast corrections' provided by the SBAS satellites. In case of PPP, this is the age of the last received clock or orbit correction message.
`SignalInfo`	u4		0	Bit field indicating the type of GNSS signals having been used in the PVT computations. If a bit *i* is set, the signal type having index *i* has been used. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.
`AlertFlag`	u1		0	Bit field indicating integrity related information: Bits 0-1: RAIM integrity flag: 0: RAIM not active (integrity not monitored) 1: RAIM integrity test successful 2: RAIM integrity test failed 3: Reserved Bit 2: set if integrity has failed as per Galileo HPCA (HMI Probability Computation Algorithm) Bit 3: set if Galileo ionospheric storm flag is active Bit 4: Reserved Bits 5-7: Reserved
`NrBases`	u1		0	Number of base stations used in the PVT computation.

Rev 1

Rev 2

`PPPInfo`	u2	1 s	0	Bit field containing PPP-related information: Bits 0-11: Age of the last seed, in seconds. The age is clipped to 4091s. This field must be ignored when the seed type is 0 (see bits 13-15 below). Bit 12: Reserved Bits 13-15: Type of last seed: 0: Not seeded or not in PPP positioning mode 1: Manual seed 2: Seeded from DGPS 3: Seeded from RTKFixed
`Latency`	u2	0.0001 s	65535	Time elapsed between the time of applicability of the position fix and the generation of this SBF block by the receiver. This time includes the receiver processing time, but not the communication latency.
`HAccuracy`	u2	0.01 m	65535	2DRMS horizontal accuracy: twice the root-mean-square of the horizontal distance error. The horizontal distance between the true position and the computed position is expected to be lower than `HAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m
`VAccuracy`	u2	0.01 m	65535	2-sigma vertical accuracy. The vertical distance between the true position and the computed position is expected to be lower than `VAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m.
`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: In DGNSS or RTK mode, set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating whether the marker position reported in this block is also the ARP position (i.e. whether the ARP-to-marker offset provided with the **`setAntennaOffset`** command is zero or not) 0: Unknown 1: The ARP-to-marker offset is zero 2: The ARP-to-marker offset is not zero
`Padding`	u1[..]			Padding bytes, see 4.1.5

`PosCovCartesian`	Number:	5905
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the elements of the symmetric variance-covariance matrix of the position expressed relative to the Cartesian axes of the coordinate system datum requested by the user:

$$\begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} & \sigma_{xb} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} & \sigma_{yb} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 & \sigma_{zb} \\ \sigma_{bx} & \sigma_{by} & \sigma_{bz} & \sigma_b^2 \end{pmatrix}$$

This variance-covariance matrix contains an indication of the accuracy of the estimated parameters (see diagonal elements) and the correlation between these estimates (see off-diagonal elements). Note that the variances and covariances are estimated: they are not necessarily indicative of the actual scatter of the position estimates at a given site.

The position variance results from the propagation of all pseudorange variances using the observation geometry. The receiver implements a stochastic error model for individual measurements, based on parameters such as the C/N$_0$, the satellite elevation, the pseudorange type, the URA of the broadcast ephemeris and the ionospheric model.

If the ellipsoidal height is not estimated (2D-mode), all components of the variance-covariance matrix are undefined and set to their Do-Not-Use value.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).

`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`Cov_xx`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of the x estimate
`Cov_yy`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of the y estimate
`Cov_zz`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of the z estimate
`Cov_bb`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of the clock bias estimate
`Cov_xy`	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the x and y estimates
`Cov_xz`	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the x and z estimates
`Cov_xb`	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the x and clock bias estimates
`Cov_yz`	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the y and z estimates
`Cov_yb`	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the y and clock bias estimates
`Cov_zb`	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the z and clock bias estimates
`Padding`	u1[..]			Padding bytes, see 4.1.5

`PosCovGeodetic`	Number: 5906 `"OnChange"` interval: default PVT output rate (see 4.1.8)

This block contains the elements of the symmetric variance-covariance matrix of the position expressed in the geodetic coordinates in the datum requested by the user:

$$\begin{pmatrix} \sigma^2_\phi & \sigma_{\phi\lambda} & \sigma_{\phi h} & \sigma_{\phi b} \\ \sigma_{\lambda\phi} & \sigma^2_\lambda & \sigma_{\lambda h} & \sigma_{\lambda b} \\ \sigma_{h\phi} & \sigma_{h\lambda} & \sigma^2_h & \sigma_{hb} \\ \sigma_{b\phi} & \sigma_{b\lambda} & \sigma_{bh} & \sigma^2_b \end{pmatrix}$$

Please refer to the `PosCovCartesian` block description for a general explanation of the contents.

Note that the units of measure for all the variances and covariances, for height as well as for latitude and longitude, are m^2 for ease of interpretation.

If the ellipsoidal height is not estimated (2D-mode), all height related components of the variance-covariance matrix are undefined and set to their Do-Not-Use value.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **setPVTMode, Static, , auto** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`Cov_latlat`	f4	1 m^2	$-2\cdot10^{10}$	Variance of the latitude estimate
`Cov_lonlon`	f4	1 m^2	$-2\cdot10^{10}$	Variance of the longitude estimate
`Cov_hgthgt`	f4	1 m^2	$-2\cdot10^{10}$	Variance of the height estimate
`Cov_bb`	f4	1 m^2	$-2\cdot10^{10}$	Variance of the clock-bias estimate
`Cov_latlon`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the latitude and longitude estimates
`Cov_lathgt`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the latitude and height estimates
`Cov_latb`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the latitude and clock-bias estimates
`Cov_lonhgt`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the longitude and height estimates
`Cov_lonb`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the longitude and clock-bias estimates
`Cov_hb`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the height and clock-bias estimates
`Padding`	u1[..]			Padding bytes, see 4.1.5

`VelCovCartesian`	Number: 5907 `"OnChange"` interval: default PVT output rate (see 4.1.8)

This block contains the elements of the symmetric variance-covariance matrix of the velocity expressed in the Cartesian coordinates of the coordinate system datum requested by the user:

$$\begin{pmatrix} \sigma^2_{v_x} & \sigma_{v_x v_y} & \sigma_{v_x v_z} & \sigma_{v_x d} \\ \sigma_{v_y v_x} & \sigma^2_{v_y} & \sigma_{v_y v_z} & \sigma_{v_y d} \\ \sigma_{v_z v_x} & \sigma_{v_z v_y} & \sigma^2_{v_z} & \sigma_{v_z d} \\ \sigma_{d v_x} & \sigma_{d v_y} & \sigma_{d v_z} & \sigma^2_{d} \end{pmatrix}$$

Please refer to the `PosCovCartesian` block description for a general explanation of the contents.

If the up-velocity is not estimated (2D-mode), all components of the variance-covariance matrix are undefined and set to their Do-Not-Use value.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions

`Cov_VxVx`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the x-velocity estimate
`Cov_VyVy`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the y-velocity estimate
`Cov_VzVz`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the z-velocity estimate
`Cov_DtDt`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the clock drift estimate
`Cov_VxVy`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the x- and y-velocity estimates
`Cov_VxVz`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the x- and z-velocity estimates
`Cov_VxDt`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the x-velocity and the clock drift estimates
`Cov_VyVz`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the y- and z-velocity estimates
`Cov_VyDt`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the y-velocity and the clock drift estimates
`Cov_VzDt`	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the z-velocity and the clock drift estimates
`Padding`	u1[..]			Padding bytes, see 4.1.5

`VelCovGeodetic`	Number:	5908
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the elements of the symmetric variance-covariance matrix of the velocity expressed in the geodetic coordinates in the datum requested by the user:

$$\begin{pmatrix} \sigma^2_{v_N} & \sigma_{v_N v_E} & \sigma_{v_N v_U} & \sigma_{v_N d} \\ \sigma_{v_E v_N} & \sigma^2_{v_E} & \sigma_{v_E v_U} & \sigma_{v_E d} \\ \sigma_{v_U v_N} & \sigma_{v_U v_E} & \sigma^2_{v_U} & \sigma_{v_U d} \\ \sigma_{d v_N} & \sigma_{d v_E} & \sigma_{d v_U} & \sigma^2_{d} \end{pmatrix}$$

Please refer to the `PosCovCartesian` block description for a general explanation of the contents.

If the up-velocity is not estimated (2D-mode), all up-velocity related components of the variance-covariance matrix are undefined and set to their Do-Not-Use value.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions

Cov_VnVn	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the north-velocity estimate
Cov_VeVe	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the east-velocity estimate
Cov_VuVu	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the up-velocity estimate
Cov_DtDt	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Variance of the clock drift estimate
Cov_VnVe	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the north- and east-velocity estimates
Cov_VnVu	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the north- and up-velocity estimates
Cov_VnDt	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the north-velocity and clock drift estimates
Cov_VeVu	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the east- and up-velocity estimates
Cov_VeDt	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the east-velocity and clock drift estimates
Cov_VuDt	f4	1 m^2 / s^2	$-2 \cdot 10^{10}$	Covariance between the up-velocity and clock drift estimates
Padding	u1[..]			Padding bytes, see 4.1.5

`DOP`	Number:	4001
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains both Dilution of Precision (DOP) values and SBAS protection levels. The DOP values result from a trace of the unit position variance-covariance matrices:

$$\text{Position Dilution of Precision:} \quad PDOP = \sqrt{\mathbf{Q}_{xx} + \mathbf{Q}_{yy} + \mathbf{Q}_{zz}}$$

$$\text{Time Dilution of Precision:} \quad TDOP = \sqrt{\mathbf{Q}_{bb}}$$

$$\text{Horizontal Dilution of Precision:} \quad HDOP = \sqrt{\mathbf{Q}_{\lambda\lambda} + \mathbf{Q}_{\phi\phi}}$$

$$\text{Vertical Dilution of Precision:} \quad VDOP = \sqrt{\mathbf{Q}_{hh}}$$

In these equations, the matrix **Q** is the inverse of the unweighted normal matrix used for the computation of the position. The normal matrix equals the product of the geometry matrix A with its transpose ($A^t A$). The term "unweighted" implies that the DOP factor only addresses the effect of the geometric factors on the quality of the position.

The DOP values can be used to interpret the current constellation geometry. This is an important parameter for the quality of the position fix: the DOP parameter is the propagation factor of the pseudorange variance. For example, if an error of 5 m is present in the pseudorange, it will propagate into the horizontal plane with a factor expressed by the HDOP. Hence a low DOP value indicates that the satellites used for the position fix result in a low multiplication of the systematic ranging errors. A value of six (6) for the PDOP is generally considered as the maximum value allowed for an acceptable position computation.

The horizontal and vertical protection levels (HPL and VPL) indicate the integrity of the computed horizontal and vertical position components as per the DO 229 specification. In SBAS-aided PVT mode (see the `Mode` field of the `PVTCartesian` SBF block), HPL and VPL are based upon the error estimates provided by SBAS. Otherwise they are based upon internal position-mode dependent error estimates.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`NrSV`	u1		0	Total number of satellites used in the DOP computation, or 0 if the DOP information is not available (in that case, the `xDOP` fields are all set to 0)
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`PDOP`	u2	0.01	0	If 0, PDOP not available, otherwise divide by 100 to obtain PDOP.
`TDOP`	u2	0.01	0	If 0, TDOP not available, otherwise divide by 100 to obtain TDOP.
`HDOP`	u2	0.01	0	If 0, HDOP not available, otherwise divide by 100 to obtain HDOP.
`VDOP`	u2	0.01	0	If 0, VDOP not available, otherwise divide by 100 to obtain VDOP.
`HPL`	f4	1 m	$-2 \cdot 10^{10}$	Horizontal Protection Level (see the DO 229 standard).
`VPL`	f4	1 m	$-2 \cdot 10^{10}$	Vertical Protection Level (see the DO 229 standard).
`Padding`	u1[..]			Padding bytes, see 4.1.5

`PosCart`	Number:	4044
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the absolute and relative (relative to the nearest base station) position at the time specified in the `TOW` and `WNc` fields. The time of applicability is specified in the receiver time frame.

The absolute position (`X`, `Y`, `Z`) is reported in a Cartesian coordinate system using the datum indicated in the `Datum` field. The position is that of the marker. The ARP-to-marker offset is set through the command **`setAntennaOffset`**.

For highest accuracy, the receiver tries to compute the baseline (`Base2RoverX`, `Base2RoverY`, `Base2RoverZ`) from rover ARP to base ARP. See the description of the `BaseVectorCart` block for details.

Accurate ARP-to-ARP baseline is guaranteed only if both bits 0 and 1 of the `Misc` field are set. Otherwise, centimeter-level offsets may arise because the receiver cannot make the distinction between phase center and ARP positions. See section 2.5 for a discussion on the phase center and ARP positions.

This block also contains the variance-covariance information and DOP factors associated with the position.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **setPVTMode, Static, , auto** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`X`	f8	1 m	$-2\cdot10^{10}$	X coordinate in coordinate frame specified by `Datum`
`Y`	f8	1 m	$-2\cdot10^{10}$	Y coordinate in coordinate frame specified by `Datum`
`Z`	f8	1 m	$-2\cdot10^{10}$	Z coordinate in coordinate frame specified by `Datum`
`Base2RoverX`	f8	1 m	$-2\cdot10^{10}$	X baseline component (from base to rover)
`Base2RoverY`	f8	1 m	$-2\cdot10^{10}$	Y baseline component (from base to rover)
`Base2RoverZ`	f8	1 m	$-2\cdot10^{10}$	Z baseline component (from base to rover)
`Cov_xx`	f4	1 m^2	$-2\cdot10^{10}$	Variance of the x estimate
`Cov_yy`	f4	1 m^2	$-2\cdot10^{10}$	Variance of the y estimate
`Cov_zz`	f4	1 m^2	$-2\cdot10^{10}$	Variance of the z estimate
`Cov_xy`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the x and y estimates
`Cov_xz`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the x and z estimates
`Cov_yz`	f4	1 m^2	$-2\cdot10^{10}$	Covariance between the y and z estimates
`PDOP`	u2	0.01	0	If 0, PDOP not available, otherwise divide by 100 to obtain PDOP.
`HDOP`	u2	0.01	0	If 0, HDOP not available, otherwise divide by 100 to obtain HDOP.
`VDOP`	u2	0.01	0	If 0, VDOP not available, otherwise divide by 100 to obtain VDOP.

`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: In DGNSS or RTK mode, set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating whether the marker position reported in this block is also the ARP position (i.e. whether the ARP-to-marker offset provided with the **`setAntennaOffset`** command is zero or not) 0: Unknown 1: The ARP-to-marker offset is zero 2: The ARP-to-marker offset is not zero
`Reserved`	u1			Reserved for future use.
`AlertFlag`	u1		0	Bit field indicating integrity related information: Bits 0-1: RAIM integrity flag: 0: RAIM not active (integrity not monitored) 1: RAIM integrity test successful 2: RAIM integrity test failed 3: Reserved Bit 2: set if integrity has failed as per Galileo HPCA (HMI Probability Computation Algorithm) Bit 3: set if Galileo ionospheric storm flag is active Bit 4: Reserved Bits 5-7: Reserved
`Datum`	u1		255	This field defines in which datum the coordinates are expressed: 0: WGS84/ITRS 19: Datum equal to that used by the DGNSS/RTK base station 30: ETRS89 (ETRF2000 realization) 31: NAD83(2011), North American Datum (2011) 32: NAD83(PA11), North American Datum, Pacific plate (2011) 33: NAD83(MA11), North American Datum, Marianas plate (2011) 34: GDA94(2010), Geocentric Datum of Australia (2010) 35: GDA2020, Geocentric Datum of Australia 2020 36: JGD2011, Japanese Geodetic Datum 2011 250: First user-defined datum 251: Second user-defined datum
`NrSV`	u1		255	Total number of satellites used in the PVT computation.
`WACorrInfo`	u1		0	Bit field providing information about which wide area corrections have been applied: Bit 0: set if orbit and satellite clock correction information is used Bit 1: set if range correction information is used Bit 2: set if ionospheric information is used Bit 3: set if orbit accuracy information is used (UERE/SISA) Bit 4: set if DO229 Precision Approach mode is active Bits 5-7: Reserved
`ReferenceId`	u2		65535	This field indicates the reference ID of the differential information used. In case of DGPS or RTK operation, this field is to be interpreted as the base station identifier. In SBAS operation, this field is to be interpreted as the PRN of the geostationary satellite used (from 120 to 158). If multiple base stations or multiple geostationary satellites are used the value is set to 65534.
`MeanCorrAge`	u2	0.01 s	65535	In case of DGPS or RTK, this field is the mean age of the differential corrections. In case of SBAS operation, this field is the mean age of the 'fast corrections' provided by the SBAS satellites. In case of PPP, this is the age of the last received clock or orbit correction message.

`SignalInfo`	u4		0	Bit field indicating the type of GNSS signals having been used in the PVT computations. If a bit *i* is set, the signal type having index *i* has been used. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`PosLocal`	Number:	4052
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the position at the time specified in the `TOW` and `WNc` fields. The time of applicability is specified in the receiver time frame.

The position (Lat, Lon, Alt) relates to the local datum identified with the `Datum` field. The coordinate transformation to the local datum is done using parameters transmitted by the RTK service provider in RTCM message types MT1021 to MT1023.

The position is that of the marker. The ARP-to-marker offset is set through the command **`setAntennaOffset`**.

If no position is available, the `Error` field indicates the cause of the unavailability and all fields after the `Error` field are set to their respective Do-Not-Use values.

To be able to output a position in the `PosLocal` block, the receiver needs to have received the relevant RTCM transformation messages (at least either MT1021 or MT1022 is required). If they have not been received yet, the local position is not available and the `Error` field is set to value 17. See also section 2.4.6.

The corresponding `RTCMDatum` block provides information on the local datum name and transformation quality indicators. The corresponding `RTCMDatum` block is the one of which the `Datum` field matches the `Datum` field in the `PosLocal` block.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **setPVTMode, Static, , auto** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions 17: Datum transformation parameters unknown
`Lat`	f8	1 rad	$-2\cdot10^{10}$	Latitude, from $-\pi/2$ to $+\pi/2$, positive North of Equator
`Lon`	f8	1 rad	$-2\cdot10^{10}$	Longitude, from $-\pi$ to $+\pi$, positive East of Greenwich
`Alt`	f8	1 m	$-2\cdot10^{10}$	Height. See the `HeightType` field of the corresponding `RTCMDatum` block for the interpretation of the height.
`Datum`	u1			Reference frame to which the position relate. If the value is in the 20 to 24 range, the corresponding datum parameters can be found in the `RTCMDatum` block having a matching `Datum` field.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`PosProjected`	Number: 4094 `"OnChange"` interval: default PVT output rate (see 4.1.8)

This block contains the projected coordinates at the time specified in the `TOW` and `WNc` fields. The time of applicability is specified in the receiver time frame.

The coordinates (Northing, Easting, Alt) relate to the local datum identified with the `Datum` field. The coordinate transformation and projection is done using parameters transmitted by the RTK service provider in RTCM message types MT1021 to MT1027.

The position is that of the marker. The ARP-to-marker offset is set through the command **`setAntennaOffset`**.

If no position is available, the `Error` field indicates the cause of the unavailability and all fields after the `Error` field are set to their respective Do-Not-Use values.

To be able to output a position in the `PosProjected` block, the receiver needs to have received at least one RTCM message in the MT1025 to MT1027 range. If none of these messages is sent out by the service provider, or if they have not been received yet, the projected position is not available and the `Error` field is set to value 17. See also section 2.4.6.

The corresponding `RTCMDatum` block provides information on the local datum name and transformation/projection quality indicators. The corresponding `RTCMDatum` block is the one of which the `Datum` field matches the `Datum` field in the `PosProjected` block.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions 17: Datum transformation parameters unknown
`Northing`	f8	1 m	$-2 \cdot 10^{10}$	Northing coordinate in the plane grid representation.
`Easting`	f8	1 m	$-2 \cdot 10^{10}$	Easting coordinate in the plane grid representation.
`Alt`	f8	1 m	$-2 \cdot 10^{10}$	Height. If the `Datum` field is in the 20 to 24 range, see the `HeightType` field of the corresponding `RTCMDatum` block for the interpretation of the height.
`Datum`	u1			Reference frame to which the position relate. If the value is in the 20 to 24 range, the corresponding datum parameters can be found in the `RTCMDatum` block having a matching `Datum` field.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`PVTSatCartesian`	Number:	4008
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the position and velocity of all the satellites used in the PVT solution, together with slant ionospheric and tropospheric delays. Coordinates are referred to the time of signal transmission computed by the receiver and are corrected for the Sagnac effect.

The reference frame the coordinates are related to is the one specified in the respective ICDs (WGS84 for GPS satellites, GTRF for Galileo satellites, PZ90 for GLONASS satellites, etc).

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of satellites for which satellite position is provided in this SBF block, i.e. number of `SatPos` sub-blocks. If `N` is 0, there are no satellite positions available for this epoch.
`SBLength`	u1	1 byte		Length of one sub-block
SatPos	...	...		*A succession of N `SatPos` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`SatPos` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`SVID`	u1			Satellite ID, see 4.1.9
`FreqNr`	u1		0	For GLONASS FDMA signals, this is the frequency number, with an offset of 8. It ranges from 1 (corresponding to an actual frequency number of -7) to 21 (corresponding to an actual frequency number of 13). Otherwise, `FreqNr` is reserved and must be ignored by the decoding software.
`IODE`	u2			IODE of the data set used to compute the values in this sub-block. 1mm
`x`	f8	1 m	$-2 \cdot 10^{10}$	X coordinate
`y`	f8	1 m	$-2 \cdot 10^{10}$	Y coordinate
`z`	f8	1 m	$-2 \cdot 10^{10}$	Z coordinate
`Vx`	f4	1 m / s	$-2 \cdot 10^{10}$	Satellite velocity in the X direction
`Vy`	f4	1 m / s	$-2 \cdot 10^{10}$	Satellite velocity in the Y direction
`Vz`	f4	1 m / s	$-2 \cdot 10^{10}$	Satellite velocity in the Z direction
`IonoMSB`	i2	1 dm	-32768 (7)	Total slant ionospheric delay at the L1 carrier frequency (1575.42MHz), with a decimeter resolution.
`TropoMSB`	i2	1 dm	-32768 (8)	Total slant tropospheric delay, with a decimeter resolution.
`IonoLSB`	u1	1.0/256.0 dm	0 (7)	Sub-decimeter part of the slant ionospheric delay. The high-resolution ionospheric delay, expressed in meters, can be computed as: IonoDelay[m] = 0.1*(`IonoMSB` + `IonoLSB`/256)
`TropoLSB`	u1	1.0/256.0 dm	0 (8)	Sub-decimeter part of the slant tropospheric delay. The high-resolution tropospheric delay, expressed in meters, can be computed as: TropoDelay[m] = 0.1*(`TropoMSB` + `TropoLSB`/256)
`IonoModel`	u1			Model used to compute the ionospheric delay: 0: Not applicable 1: Klobuchar 2: DO229 3: NeQuick 4: Measured (from dual frequency measurements) 5: Estimated 6: KlobucharBeiDou
`Padding`	u1[..]			Padding bytes, see 4.1.5

Rev 1

---

(7) The ionospheric delay should not be used when `IonoMSB` is -32768.

(8) The tropospheric delay should not be used when `TropoMSB` is -32768.

`PVTResiduals`	Number:	4009
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the residuals of all measurements used in PVT solution computed at the epoch specified in the `TOW` and `WNc` fields. The PVT solution itself can be found in the `PVTCartesian` or `PVTGeodetic` blocks.

For each measurement from each satellite and each modulation used in the PVT solution, detailed PVT residual information is output for each observable type (code phase, carrier phase and Doppler):

- a-posteriori measurement residual (e_i)
- absolute value of the w-test statistic (w_i)
- Minimal detectable bias (MDB).

In case of multi-base differential operation, a set of residuals is provided for all base stations.

This block uses a two-level sub-block structure analogous to that of the `MeasEpoch` block. It contains one `SatSignalInfo` sub-block for each satellite/signal type pair used in the PVT or attitude computation. Each `SatSignalInfo` sub-block contains a number of `ResidualInfo` sub-blocks, each of them containing the residuals of a given observable type.

The standard deviation of the residual (σ_e) for satellite i and the "a priori" measurement standard deviation (σ_y) can be computed from e_i, w_i and MDB by using the following formulas (see also section 2.7):

$$\sigma_{e_i} = \frac{|e_i|}{w_i} \text{ and } \sigma_{y_i} = \sqrt{\frac{MDB}{\sqrt{\lambda_0}} . \sigma_{e_i}}$$

where λ_0 is the non-centrality parameter and:

$$\sqrt{\lambda_0} = \sqrt{2}[\text{erfinv}(1 - P_{fa}) + \text{erfinv}(1 - 2P_{md})]$$

with P_{fa} and P_{md} being the probability of false alarm and of missed detection respectively, as set by the **`setRAIMLevels`** command, and the "`erfinv`" function being the inverse error function. The output of `erfinv(x)` is the value `y` that satisfies the following equality:

$$x = \frac{2}{\sqrt{\pi}} \int_0^y e^{-t^2} dt$$

This block can be used to monitor the quality of the measurements. Under normal circumstances, the residuals lie within -2 and +2 times the a-priori variance of the corresponding measurements.

Parameter	**Type**	**Units**	**Do-Not-Use**	**Description**
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of satellite/signal pairs for which residual blocks are provided in this SBF block, i.e. number of `SatSignalInfo` sub-blocks. If `N` is 0, there are no satellite residuals available for this epoch.
`SB1Length`	u1	1 byte		Length of a `SatSignalInfo` sub-block, excluding the nested `ResidualInfoCode`, `ResidualInfoPhase` and `ResidualInfoDoppler` sub-blocks
`SB2Length`	u1	1 byte		Length of a `ResidualInfoCode`, `ResidualInfoPhase` and `ResidualInfoDoppler` sub-block
`Reserved`	u1[3]			Reserved for future use, to be ignored by decoding software
Residuals	...	...		*A succession of N `SatSignalInfo` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`SatSignalInfo` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`SVID`	u1			Satellite ID, see 4.1.9
`FreqNr`	u1		0	For GLONASS FDMA signals, this is the frequency number, with an offset of 8. It ranges from 1 (corresponding to an actual frequency number of -7) to 21 (corresponding to an actual frequency number of 13). Otherwise, `FreqNr` is reserved and must be ignored by the decoding software.
`Type`	u1			Bit field indicating the signal type and antenna ID: Bits 0-4: Signal number, see 4.1.10. Bits 5-7: Antenna ID: 0 for the main antenna
`RefSVID`	u1		255, 62	Satellite ID of the reference satellite used for double differencing, see 4.1.9. Set to 255 if not in double difference mode, or set to 62 if in double difference mode, and GLONASS slot number unknown.
`RefFreqNr`	u1		255, 0	GLONASS frequency number for the reference satellite, see 4.1.9. Set to 255 if not in double difference mode, or set to 0 if in double difference mode, but non-GLONASS satellite.
`MeasInfo`	u1			Bit field: Bits 0-1: Type of residual this sub-block refers to: 0: zero-difference residual (standalone) 1: single-difference residual (SBAS, DGPS) 2: double-difference residual. If the antenna ID is 0 (see the `Type` field above), this sub-block contains an RTK residual, else it contains an attitude residual. Bit 2: Set if a `ResidualInfoCode` sub-block containing pseudorange residuals follows. Bit 3: Set if a `ResidualInfoPhase` sub-block containing carrier-phase residuals follows. Bit 4: Set if a `ResidualInfoDoppler` sub-block containing Doppler residuals follows. Bits 5-6: Reserved Bit 7: Set if ambiguity is fixed for the signal type identified by the `Type` field. The number of `ResidualInfo` sub-blocks to follow is equal to the number of bits set to 1 between bit 2 and bit 4. The order of these `ResidualInfo` sub-blocks is fixed: the code-phase residuals come first (if any), then the carrier phase residuals (if any), and the Doppler residuals as last.
`IODE`	u2			Issue of Data Ephemeris used for the satellite and signal type identified by `SVID` and `Type`. For GLONASS satellites, this is the t_b value in minutes.
`CorrAge`	u2	0.01 s	65535	Age of corrections, either from SBAS, DGPS, RTK etc, truncated to 655.34 seconds.
Rev 1 `ReferenceID`	u2		65535	ID of the base station the residuals apply to. Set to 65535 in case of standalone operation.
`Padding`	u1[..]			Padding bytes, see 4.1.5
If the `Pseudorange residuals` field is 1 then this sub block is available:				
`ResidualInfoCode`	...	...		A `ResidualInfoCode` sub-block, see definition below
If the `Carrier-phase residuals` field is 1 then this sub block is available:				
`ResidualInfoPhase`	...	...		A `ResidualInfoPhase` sub-block, see definition below
If the `Doppler residuals` field is 1 then this sub block is available:				
`ResidualInfoDoppler`	...	...		A `ResidualInfoDoppler` sub-block, see definition below

`ResidualInfoCode` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`Residual`	f4	1 m	$-2 \cdot 10^{10}$	Code Residual with respect to PVT solution reported in `PVTCartesian` and/or `PVTGeodetic` block.
`W`	u2	0.001	65535	Absolute value of the *w*-test statistic based on probability of false alarm set by user
`MDB`	u2	0.1 m	65535	Minimal detectable bias based on probability of missed detection set by user
`Padding`	u1[..]			Padding bytes, see 4.1.5

`ResidualInfoPhase` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`Residual`	f4	1 cycle	$-2 \cdot 10^{10}$	Phase Residual with respect to PVT solution reported in `PVTCartesian` and/or `PVTGeodetic` block. Double-difference carrier phase residuals include the double difference ambiguity as long as the ambiguity is not fixed (i.e. as long as bit 7 of `MeasInfo` is not set). When the ambiguity is fixed, e_i does not contain the ambiguity anymore.
`W`	u2	0.001	65535	Absolute value of the *w*-test statistic based on probability of false alarm set by user
`MDB`	u2	0.01 cycles	65535	Minimal detectable bias based on probability of missed detection set by user
`Padding`	u1[..]			Padding bytes, see 4.1.5

`ResidualInfoDoppler` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`Residual`	f4	1 m / s	$-2 \cdot 10^{10}$	Doppler Residual with respect to PVT solution reported in `PVTCartesian` and/or `PVTGeodetic` block.
`W`	u2	0.001	65535	Absolute value of the *w*-test statistic based on probability of false alarm set by user
`MDB`	u2	0.01 m / s	65535	Minimal detectable bias based on probability of missed detection set by user
`Padding`	u1[..]			Padding bytes, see 4.1.5

`RAIMStatistics`	Number:	4011
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the integrity statistics that are computed by the receiver RAIM algorithm.

The output of the RAIM algorithm contains integrity information, which can be used in user applications. First, the RAIM algorithm generates its own integrity flag based on the probability of false-alarm, which can be used by a user as a receiver-level indication of positional integrity. If the internal integrity test is successful, a user has an opportunity to introduce a more stringent application-specific integrity criterion by using External Reliability Levels (XERL). The positional solution is deemed as passed an application-level integrity test if the XERLs are within user-defined (and application-dependent) alarm limits. This comparison (and the definition of alarm limits as well) takes place in a user application and is outside of the receiver scope. Please also refer to section 2.7.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`IntegrityFlag`	u1			RAIM integrity flag: 0: Integrity test successful 1: Integrity test failed 2: Integrity not available
`Reserved1`	u1			Reserved for future use, to be ignored by decoding software
`HERL-position`	f4	1 m	$-2\cdot10^{10}$	Horizontal external reliability level of the position
`VERL-position`	f4	1 m	$-2\cdot10^{10}$	Vertical external reliability level of the position
`HERL-velocity`	f4	1 m / s	$-2\cdot10^{10}$	Horizontal external reliability level of the velocity
`VERL-velocity`	f4	1 m / s	$-2\cdot10^{10}$	Vertical external reliability level of the velocity
`OverallModel`	u2	1/50000	65535 (9)	Overall model test statistic for the estimated PVT parameters divided by the test threshold
`Padding`	u1[..]			Padding bytes, see 4.1.5

---

(9) This field is clipped to 65534, i.e. if the actual value is larger than 65534, it is set to 65534.

`GEOCorrections`	Number:	5935
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains the SBAS corrections that the receiver has applied to the pseudoranges used in the PVT computation computed at the epoch specified in the `TOW` and `WNc` fields. The PVT solution itself can be found in the `PVTCartesian` or `PVTGeodetic` blocks.

The corrections are based on the messages received from an SBAS satellite. They compensate for the following errors:

- Satellite orbit
- Satellite clock
- Ionospheric delay.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of satellites for which corrections are provided in this SBF block, i.e. number of `SatCorr` sub-blocks. If `N` is 0, there are no corrections available for this epoch.
`SBLength`	u1	1 byte		Length of one sub-block in bytes
SatCorr	...	...		*A succession of N `SatCorr` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`SatCorr` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`SVID`	u1			Satellite ID, see 4.1.9
`IODE`	u1			Issue of Data Ephemeris related to the orbit and clock corrections
`Reserved`	u1[2]			Reserved for future use, to be ignored by decoding software
`PRC`	f4	1 m		Applied pseudorange correction based on the fast correction data received in MT02-MT05 or MT24
`CorrAgeFC`	f4	1 s	$-2\cdot10^{10}$	Age of applied fast correction
`DeltaX`	f4	1 m		X-component of applied orbit correction based on the long term correction data received in MT24 or MT25
`DeltaY`	f4	1 m		Y-component of applied orbit correction based on the long term correction data received in MT24 or MT25
`DeltaZ`	f4	1 m		Z-component of applied orbit correction based on the long term correction data received in MT24 or MT25
`DeltaClock`	f4	1 s		Satellite clock correction based on the long term correction data received in MT24 or MT25
`CorrAgeLT`	f4	1 s	$-2\cdot10^{10}$	Age of applied long term correction
`IonoPPlat`	f4	1 rad	$-2\cdot10^{10}$	Latitude of ionospheric pierce point
`IonoPPlon`	f4	1 rad	$-2\cdot10^{10}$	Longitude of ionospheric pierce point
`SlantIono`	f4	1 m	$-2\cdot10^{10}$	Slant ionospheric delay at the L1 carrier at the ionosphere pierce point based on the data received in MT18 and MT26
`CorrAgeIono`	f4	1 s	$-2\cdot10^{10}$	Maximum of the ionospheric correction age at each of the grid locations used for the interpolation of the ionospheric delay.

`VarFLT`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of fast and long-term corrections (used for XPL computation)
`VarUIRE`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of ionospheric delay corrections (used for XPL computation)
`VarAir`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of unmodeled receiver errors, such as tracking noise and multipath (used for XPL computation)
`VarTropo`	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of tropospheric delay corrections (used for XPL computation)
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BaseVectorCart`	Number: 4043
	`"OnChange"` interval: default PVT output rate (see 4.1.8)

The `BaseVectorCart` block contains the relative position and orientation of one or more base stations, as seen from the rover (i.e. this receiver). The relative position is expressed in the Cartesian X, Y, Z directions.

For highest accuracy, the receiver tries to compute the baseline from rover antenna reference point (ARP) to base ARP. This requires to compensate for the phase center offset at both the base and the rover antennas. This is possible if two conditions are met:

- the base station must transmit its antenna parameters in RTCM2 message types 23 and 24 or in RTCM3 message types 1005/1006 and 1007/1008. Older RTCM2 messages and CMR do not allow phase center offset compensation.
- the base and rover antenna types must belong to the list returned by the command **`lstAntennaInfo, overview`**. (see the description of the commands **`setAntennaOffset`** and **`lstAntennaInfo`** for details).

Accurate ARP-to-ARP baseline is guaranteed only if both bits 0 and 1 of the `Misc` field are set. Otherwise, centimeter-level offsets may arise because the receiver cannot make the distinction between phase center and ARP positions. See section 2.5 for a discussion on the phase center and ARP positions.

The block supports multi-base operation. It contains as many sub-blocks as available base stations, each sub-block containing the baseline relative to a single base station identified by the `ReferenceID` field.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of baselines for which relative position, velocity and direction are provided in this SBF block, i.e. number of `VectorInfoCart` sub-blocks. If `N` is 0, there are no baseline available for this epoch.
`SBLength`	u1	1 byte		Length of one sub-block
VectorInfoCart	...	...		*A succession of N `VectorInfoCart` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`VectorInfoCart` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`nrSV`	u1			Number of satellites for which corrections are available from the base station identified by the `ReferenceID` field.
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: Set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover (i.e. the baseline starts from the antenna ARP), unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Reserved
`DeltaX`	f8	1 m	$-2 \cdot 10^{10}$	X baseline component (from rover to base)
`DeltaY`	f8	1 m	$-2 \cdot 10^{10}$	Y baseline component (from rover to base)
`DeltaZ`	f8	1 m	$-2 \cdot 10^{10}$	Z baseline component (from rover to base)
`DeltaVx`	f4	1 m / s	$-2 \cdot 10^{10}$	X velocity of base with respect to rover
`DeltaVy`	f4	1 m / s	$-2 \cdot 10^{10}$	Y velocity of base with respect to rover
`DeltaVz`	f4	1 m / s	$-2 \cdot 10^{10}$	Z velocity of base with respect to rover
`Azimuth`	u2	0.01 degrees	65535	Azimuth of the base station (from 0 to 360°, increasing towards east)
`Elevation`	i2	0.01 degrees	−32768	Elevation of the base station (from -90° to 90°)
`ReferenceID`	u2			Base station ID
`CorrAge`	u2	0.01 s	65535	Age of the oldest differential correction used for this baseline computation.

`SignalInfo`	u4		0	Bit field indicating the GNSS signals for which differential corrections are available from the base station identified by `ReferenceID`. If bit *i* is set, corrections for the signal type having index *i* are available. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.
`Padding`	u1[..]			Padding bytes, see 4.1.5

BaseVectorGeod	Number: 4028 "OnChange" interval: default PVT output rate (see 4.1.8)

The BaseVectorGeod block contains the relative position and orientation of one or more base stations, as seen from the rover (i.e. this receiver). The relative position is expressed in the East-North-Up directions.

For highest accuracy, the receiver tries to compute the baseline from rover antenna reference point (ARP) to base ARP. See the description of the BaseVectorCart block for details.

Accurate ARP-to-ARP baseline is guaranteed only if both bits 0 and 1 of the Misc field are set. Otherwise, centimeter-level offsets may arise because the receiver cannot make the distinction between phase center and ARP positions. See section 2.5 for a discussion on the phase center and ARP positions.

The block supports multi-base operation. It contains as many sub-blocks as available base stations, each sub-block containing the baseline coordinates relative to a single base station identified by the ReferenceID field.

Parameter	Type	Units	Do-Not-Use	Description
Sync1	c1			Block Header, see 4.1.1
Sync2	c1			
CRC	u2			
ID	u2			
Length	u2	1 byte		
TOW	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
WNc	u2	1 week	65535	
N	u1			Number of baselines for which relative position, velocity and direction are provided in this SBF block, i.e. number of VectorInfoGeod sub-blocks. If N is 0, there are no baseline available for this epoch.
SBLength	u1	1 byte		Length of one sub-block
VectorInfoGeod	...	...		*A succession of N VectorInfoGeod sub-blocks, see definition below*
Padding	u1[..]			Padding bytes, see 4.1.5

VectorInfoGeod sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
NrSV	u1			Number of satellites for which corrections are available from the base station identified by the ReferenceID field.
Error	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions

`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: Set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover (i.e. the baseline starts from the antenna ARP), unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Reserved
`DeltaEast`	f8	1 m	$-2 \cdot 10^{10}$	East baseline component (from rover to base)
`DeltaNorth`	f8	1 m	$-2 \cdot 10^{10}$	North baseline component (from rover to base)
`DeltaUp`	f8	1 m	$-2 \cdot 10^{10}$	Up baseline component (from rover to base)
`DeltaVe`	f4	1 m / s	$-2 \cdot 10^{10}$	East velocity of base with respect to rover
`DeltaVn`	f4	1 m / s	$-2 \cdot 10^{10}$	North velocity of base with respect to rover
`DeltaVu`	f4	1 m / s	$-2 \cdot 10^{10}$	Up velocity of base with respect to rover
`Azimuth`	u2	0.01 degrees	65535	Azimuth of the base station (from 0 to 360°, increasing towards east)
`Elevation`	i2	0.01 degrees	-32768	Elevation of the base station (from -90° to 90°)
`ReferenceID`	u2			Base station ID
`CorrAge`	u2	0.01 s	65535	Age of the oldest differential correction used for this baseline computation.
`SignalInfo`	u4		0	Bit field indicating the GNSS signals for which differential corrections are available from the base station identified by `ReferenceID`. If bit *i* is set, corrections for the signal type having index *i* are available. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`PVTSupport`	Number:	4076
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains various internal parameters that can be used for maintenance and support.

The detailed definition of this block is not available in this document.

`PVTSupportA`	Number:	4079
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block contains various internal parameters that can be used for maintenance and support.

The detailed definition of this block is not available in this document.

`EndOfPVT`	Number:	5921
	`"OnChange"`	interval: default PVT output rate (see 4.1.8)

This block marks the end of transmission of all PVT related blocks belonging to the same epoch.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.11 Receiver Time Blocks

`ReceiverTime`	Number: 5914 `"OnChange"` interval: 1s

The `ReceiverTime` block provides the current time with a 1-second resolution in the receiver time scale and UTC.

The level of synchronization of the receiver time with the satellite system time is provided in the `SyncLevel` field.

UTC time is provided if the UTC parameters have been received from at least one GNSS satellite. If the UTC time is not available, the corresponding fields are set to their Do-Not-Use value.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`UTCYear`	i1	1 year	−128	Current year in the UTC time scale (2 digits). From 0 to 99, or -128 if not available
`UTCMonth`	i1	1 month	−128	Current month in the UTC time scale. From 1 to 12, or -128 if not available
`UTCDay`	i1	1 day	−128	Current day in the UTC time scale. From 1 to 31, or -128 if not available
`UTCHour`	i1	1 hour	−128	Current hour in the UTC time scale. From 0 to 23, or -128 if not available
`UTCMin`	i1	1 minute	−128	Current minute in the UTC time scale. From 0 to 59, or -128 if not available
`UTCSec`	i1	1 s	−128	Current second in the UTC time scale. From 0 to 59, or -128 if not available
`DeltaLS`	i1	1 s	−128	Integer second difference between UTC time and GPS system time. Positive if GPS time is ahead of UTC. Set to -128 if not available.
`SyncLevel`	u1			Bit field indicating the synchronization level of the receiver time. If bits 0 to 2 are set, full synchronization is achieved: Bit 0: WNSET: if this bit is set, the receiver week number is set. Bit 1: TOWSET: if this bit is set, the receiver time-of-week is set to within 20ms. Bit 2: FINETIME: if this bit is set, the receiver time-of-week is within the limit specified by the **`setClockSyncThreshold`** command. Bit 3: Reserved Bit 4: Reserved Bits 5-7: Reserved
`Padding`	u1[..]			Padding bytes, see 4.1.5

`xPPSOffset`	Number:	5911
	`"OnChange"`	interval: PPS rate

The `xPPSOffset` block contains the offset between the true xPPS pulse and the actual pulse output by the receiver. It is output right after each xPPS pulse.

On receivers with more than one independent PPS outputs, this block always refers to the first PPS output.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SyncAge`	u1	1 s		Age of the last synchronization to system time. The xPPS pulse is regularly resynchronized with system time. This field indicates the number of seconds elapsed since the last resynchronization. `SyncAge` is constrained to the 0-255s range. If the age is higher than 255s, `SyncAge` is set to 255. If the PPS is synchronized with the internal receiver time (`Timescale` = 3), `SyncAge` is always set to 0.
`TimeScale`	u1			Time scale to which the xPPS pulse is referenced, as set with the **`setPPSParameters`** command: 1: GPS time 2: UTC 3: Receiver time 4: GLONASS time 5: Galileo time 6: BeiDou time
`Offset`	f4	$1 \cdot 10^{-9}$ s		Offset of the xPPS output by the receiver with respect to its true position. `Offset` is negative when the xPPS pulse is in advance with respect to its true position. See also section 1.27 for an explanation of the xPPS generation principle, and for a description of the xPPS offset.
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.12 External Event Blocks

These blocks report the state of the receiver applicable at the instant of a level transition on one of its “Event” pins. The receiver time is reported in the `ExtEvent` SBF block, and the receiver position is reported in the `ExtEventPVTCartesian` and the `ExtEventPVTGeodetic` blocks.

If enabled, upon detection of an event, these three blocks are output in the following order, with no other SBF blocks in between them:

1. `ExtEvent`;
2. `ExtEventPVTCartesian`;
3. `ExtEventPVTGeodetic`.

All blocks referring to the same event contain the same time stamp in the `TOW` and `WNc` fields.

`ExtEvent`	Number:	5924
	`"OnChange"`	interval: each time an event is detected

The `ExtEvent` block contains the time tag of a voltage transition on one of the "Event" input pins.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	External time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Source`	u1			Input pin where this external event has been detected. The following values are defined: 1: EventA 2: EventB
`Polarity`	u1			0: rising edge event 1: falling edge event
`Offset`	f4	1 s		Event time offset with respect to `TOW`, including the potential delay specified with the **`setEventParameters`** command. The time of week of the external event is given by: $t_{\mathrm{ext,rx}}$ [s] = `TOW`/1000 + `Offset` $t_{\mathrm{ext,rx}}$ refers to the receiver system time scale. Use the `RxClkBias` field to convert this time to the GNSS time scale.
`RxClkBias`	f8	1 s	$-2 \cdot 10^{10}$	Receiver clock bias at the time of event. The clock bias is relative to the time system of the last PVT computation (see the `TimeSystem` field of the `PVTCartesian` or `PVTGeodetic` blocks). To get the time of week of the external event in GNSS time, use: $t_{\mathrm{ext,GNSS}}$ [s] = `TOW`/1000 + `Offset` - `RxClkBias`. The accuracy of the clock bias is dependent on the age of the last PVT solution. When the receiver has been unable to compute a PVT during the last 10 minutes, this field is set to its Do-Not-Use value.
`PVTAge`	u2	1 s		Age of the last PVT solution. If the PVT age is larger than 10 minutes (600s), this value is clipped to 600.
`Padding`	u1[..]			Padding bytes, see 4.1.5

Rev 1

`ExtEventPVTCartesian`	Number:	4037
	`"OnChange"` interval:	each time an external event is detected

This block contains the position, velocity and time (PVT) solution applicable at the time of an external event, in a Cartesian coordinate system.

This block has the same structure and description as the `PVTCartesian` block, except that the `TOW` and `WNc` fields refer to the time at which the electrical transition on the event pin has been detected (with a millisecond resolution), and that the position is computed at the event time, taking into account a possible user-defined delay set by the **`setEventParameters`** command.

A user needing the sub-millisecond part of the event time must refer to the `Offset` field of the corresponding `ExtEvent` block. The corresponding `ExtEvent` block is the last of the `ExtEvent` blocks having been output by the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	External time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`X`	f8	1 m	$-2\cdot 10^{10}$	X coordinate in coordinate frame specified by `Datum`

`Y`	f8	1 m	$-2 \cdot 10^{10}$	Y coordinate in coordinate frame specified by `Datum`
`Z`	f8	1 m	$-2 \cdot 10^{10}$	Z coordinate in coordinate frame specified by `Datum`
`Undulation`	f4	1 m	$-2 \cdot 10^{10}$	Geoid undulation. See the **`setGeoidUndulation`** command.
`Vx`	f4	1 m / s	$-2 \cdot 10^{10}$	Not applicable
`Vy`	f4	1 m / s	$-2 \cdot 10^{10}$	Not applicable
`Vz`	f4	1 m / s	$-2 \cdot 10^{10}$	Not applicable
`COG`	f4	1 degree	$-2 \cdot 10^{10}$	Course over ground: this is defined as the angle of the vehicle with respect to the local level North, ranging from 0 to 360, and increasing towards east. Set to the Do-Not-Use value when the speed is lower than 0.1m/s.
`RxClkBias`	f8	1 ms	$-2 \cdot 10^{10}$	Receiver clock bias relative to the GNSS system time reported in the `TimeSystem` field. Positive when the receiver time is ahead of the system time. To transfer the receiver time to the system time, use: $t_{GPS/GST} = t_{rx}$ - `RxClkBias`
`RxClkDrift`	f4	1 ppm	$-2 \cdot 10^{10}$	Receiver clock drift relative to the GNSS system time (relative frequency error). Positive when the receiver clock runs faster than the system time.
`TimeSystem`	u1		255	Time system of which the offset is provided in this sub-block: 0: GPS time 1: Galileo time 3: GLONASS time 4: BeiDou time 5: QZSS time
`Datum`	u1		255	This field defines in which datum the coordinates are expressed: 0: WGS84/ITRS 19: Datum equal to that used by the DGNSS/RTK base station 30: ETRS89 (ETRF2000 realization) 31: NAD83(2011), North American Datum (2011) 32: NAD83(PA11), North American Datum, Pacific plate (2011) 33: NAD83(MA11), North American Datum, Marianas plate (2011) 34: GDA94(2010), Geocentric Datum of Australia (2010) 35: GDA2020, Geocentric Datum of Australia 2020 36: JGD2011, Japanese Geodetic Datum 2011 250: First user-defined datum 251: Second user-defined datum
`NrSV`	u1		255	Total number of satellites used in the PVT computation.
`WACorrInfo`	u1		0	Bit field providing information about which wide area corrections have been applied: Bit 0: set if orbit and satellite clock correction information is used Bit 1: set if range correction information is used Bit 2: set if ionospheric information is used Bit 3: set if orbit accuracy information is used (UERE/SISA) Bit 4: set if DO229 Precision Approach mode is active Bits 5-7: Reserved
`ReferenceID`	u2		65535	This field indicates the reference ID of the differential information used. In case of DGPS or RTK operation, this field is to be interpreted as the base station identifier. In SBAS operation, this field is to be interpreted as the PRN of the geostationary satellite used (from 120 to 158). If multiple base stations or multiple geostationary satellites are used the value is set to 65534.
`MeanCorrAge`	u2	0.01 s	65535	In case of DGPS or RTK, this field is the mean age of the differential corrections. In case of SBAS operation, this field is the mean age of the 'fast corrections' provided by the SBAS satellites. In case of PPP, this is the age of the last received clock or orbit correction message.
`SignalInfo`	u4		0	Bit field indicating the type of GNSS signals having been used in the PVT computations. If a bit *i* is set, the signal type having index *i* has been used. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.

	`AlertFlag`	u1		0	Bit field indicating integrity related information: Bits 0-1: RAIM integrity flag: 0: RAIM not active (integrity not monitored) 1: RAIM integrity test successful 2: RAIM integrity test failed 3: Reserved Bit 2: set if integrity has failed as per Galileo HPCA (HMI Probability Computation Algorithm) Bit 3: set if Galileo ionospheric storm flag is active Bit 4: Reserved Bits 5-7: Reserved
Rev 1	`NrBases`	u1		0	Number of base stations used in the PVT computation.
	`PPPInfo`	u2	1 s	0	Bit field containing PPP-related information: Bits 0-11: Age of the last seed, in seconds. The age is clipped to 4091s. This field must be ignored when the seed type is 0 (see bits 13-15 below). Bit 12: Reserved Bits 13-15: Type of last seed: 0: Not seeded or not in PPP positioning mode 1: Manual seed 2: Seeded from DGPS 3: Seeded from RTKFixed
Rev 2	`Latency`	u2	0.0001 s	65535	Time elapsed between the time of applicability of the position fix and the generation of this SBF block by the receiver. This time includes the receiver processing time, but not the communication latency.
	`HAccuracy`	u2	0.01 m	65535	2DRMS horizontal accuracy: twice the root-mean-square of the horizontal distance error. The horizontal distance between the true position and the computed position is expected to be lower than `HAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m
	`VAccuracy`	u2	0.01 m	65535	2-sigma vertical accuracy. The vertical distance between the true position and the computed position is expected to be lower than `VAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m.
	`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: In DGNSS or RTK mode, set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating whether the marker position reported in this block is also the ARP position (i.e. whether the ARP-to-marker offset provided with the **`setAntennaOffset`** command is zero or not) 0: Unknown 1: The ARP-to-marker offset is zero 2: The ARP-to-marker offset is not zero
	`Padding`	u1[..]			Padding bytes, see 4.1.5

`ExtEventPVTGeodetic`	Number:	4038
	`"OnChange"` interval:	each time an external event is detected

This block contains the position, velocity and time (PVT) solution applicable at the time of an external event, in an ellipsoidal coordinate system.

This block has the same structure and description as the `PVTGeodetic` block, except that the `TOW` and `WNc` fields refer to the time at which the electrical transition on the event pin has been detected (with a millisecond resolution), and that the position is computed at the event time, taking into account a possible user-defined delay set by the **`setEventParameters`** command.

A user needing the sub-millisecond part of the event time must refer to the `Offset` field of the corresponding `ExtEvent` block. The corresponding `ExtEvent` block is the last of the `ExtEvent` blocks having been output by the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	External time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`Latitude`	f8	1 rad	$-2\cdot10^{10}$	Latitude, from $-\pi/2$ to $+\pi/2$, positive North of Equator

`Longitude`	f8	1 rad	$-2\cdot10^{10}$	Longitude, from $-\pi$ to $+\pi$, positive East of Greenwich
`Height`	f8	1 m	$-2\cdot10^{10}$	Ellipsoidal height (with respect to the ellipsoid specified by `Datum`)
`Undulation`	f4	1 m	$-2\cdot10^{10}$	Geoid undulation. See the **`setGeoidUndulation`** command.
`Vn`	f4	1 m / s	$-2\cdot10^{10}$	Not applicable
`Ve`	f4	1 m / s	$-2\cdot10^{10}$	Not applicable
`Vu`	f4	1 m / s	$-2\cdot10^{10}$	Not applicable
`COG`	f4	1 degree	$-2\cdot10^{10}$	Course over ground: this is defined as the angle of the vehicle with respect to the local level North, ranging from 0 to 360, and increasing towards east. Set to the Do-Not-Use value when the speed is lower than 0.1m/s.
`RxClkBias`	f8	1 ms	$-2\cdot10^{10}$	Receiver clock bias relative to the GNSS system time reported in the `TimeSystem` field. Positive when the receiver time is ahead of the system time. To transfer the receiver time to the system time, use: $t_{GPS/GST} = t_{rx}$ - `RxClkBias`
`RxClkDrift`	f4	1 ppm	$-2\cdot10^{10}$	Receiver clock drift relative to the GNSS system time (relative frequency error). Positive when the receiver clock runs faster than the system time.
`TimeSystem`	u1		255	Time system of which the offset is provided in this sub-block: 0: GPS time 1: Galileo time 3: GLONASS time 4: BeiDou time 5: QZSS time
`Datum`	u1		255	This field defines in which datum the coordinates are expressed: 0: WGS84/ITRS 19: Datum equal to that used by the DGNSS/RTK base station 30: ETRS89 (ETRF2000 realization) 31: NAD83(2011), North American Datum (2011) 32: NAD83(PA11), North American Datum, Pacific plate (2011) 33: NAD83(MA11), North American Datum, Marianas plate (2011) 34: GDA94(2010), Geocentric Datum of Australia (2010) 35: GDA2020, Geocentric Datum of Australia 2020 36: JGD2011, Japanese Geodetic Datum 2011 250: First user-defined datum 251: Second user-defined datum
`NrSV`	u1		255	Total number of satellites used in the PVT computation.
`WACorrInfo`	u1		0	Bit field providing information about which wide area corrections have been applied: Bit 0: set if orbit and satellite clock correction information is used Bit 1: set if range correction information is used Bit 2: set if ionospheric information is used Bit 3: set if orbit accuracy information is used (UERE/SISA) Bit 4: set if DO229 Precision Approach mode is active Bits 5-7: Reserved
`ReferenceID`	u2		65535	This field indicates the reference ID of the differential information used. In case of DGPS or RTK operation, this field is to be interpreted as the base station identifier. In SBAS operation, this field is to be interpreted as the PRN of the geostationary satellite used (from 120 to 158). If multiple base stations or multiple geostationary satellites are used the value is set to 65534.
`MeanCorrAge`	u2	0.01 s	65535	In case of DGPS or RTK, this field is the mean age of the differential corrections. In case of SBAS operation, this field is the mean age of the 'fast corrections' provided by the SBAS satellites. In case of PPP, this is the age of the last received clock or orbit correction message.
`SignalInfo`	u4		0	Bit field indicating the type of GNSS signals having been used in the PVT computations. If a bit *i* is set, the signal type having index *i* has been used. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.

	`AlertFlag`	u1		0	Bit field indicating integrity related information: Bits 0-1: RAIM integrity flag: 0: RAIM not active (integrity not monitored) 1: RAIM integrity test successful 2: RAIM integrity test failed 3: Reserved Bit 2: set if integrity has failed as per Galileo HPCA (HMI Probability Computation Algorithm) Bit 3: set if Galileo ionospheric storm flag is active Bit 4: Reserved Bits 5-7: Reserved
Rev 1	`NrBases`	u1		0	Number of base stations used in the PVT computation.
	`PPPInfo`	u2	1 s	0	Bit field containing PPP-related information: Bits 0-11: Age of the last seed, in seconds. The age is clipped to 4091s. This field must be ignored when the seed type is 0 (see bits 13-15 below). Bit 12: Reserved Bits 13-15: Type of last seed: 0: Not seeded or not in PPP positioning mode 1: Manual seed 2: Seeded from DGPS 3: Seeded from RTKFixed
Rev 2	`Latency`	u2	0.0001 s	65535	Time elapsed between the time of applicability of the position fix and the generation of this SBF block by the receiver. This time includes the receiver processing time, but not the communication latency.
	`HAccuracy`	u2	0.01 m	65535	2DRMS horizontal accuracy: twice the root-mean-square of the horizontal distance error. The horizontal distance between the true position and the computed position is expected to be lower than `HAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m
	`VAccuracy`	u2	0.01 m	65535	2-sigma vertical accuracy. The vertical distance between the true position and the computed position is expected to be lower than `VAccuracy` with a probability of at least 95%. The value is clipped to 65534 =655.34m.
	`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: In DGNSS or RTK mode, set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating whether the marker position reported in this block is also the ARP position (i.e. whether the ARP-to-marker offset provided with the **`setAntennaOffset`** command is zero or not) 0: Unknown 1: The ARP-to-marker offset is zero 2: The ARP-to-marker offset is not zero
	`Padding`	u1[..]			Padding bytes, see 4.1.5

`ExtEventBaseVectGeod`	Number:	4217
	`"OnChange"` interval:	each time an external event is detected

This block contains the relative position and orientation of one or more base stations at the time of an external event. The relative position is expressed in the East-North-Up directions.

This block has the same structure and description as the `BaseVectorGeod` block, except that the `TOW` and `WNc` fields refer to the time at which the electrical transition on the event pin has been detected (with a millisecond resolution), and that the position is computed at the event time, taking into account a possible user-defined delay set by the **`setEventParameters`** command.

A user needing the sub-millisecond part of the event time must refer to the `Offset` field of the corresponding `ExtEvent` block. The corresponding `ExtEvent` block is the last of the `ExtEvent` blocks having been output by the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	External time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of baselines for which relative position, velocity and direction are provided in this SBF block, i.e. number of `ExtEventVectorInfoGeod` sub-blocks. If `N` is 0, there are no baseline available for this epoch.
`SBLength`	u1	1 byte		Length of one sub-block
ExtEventVectorInfoGeod	...	...		*A succession of N `ExtEventVectorInfoGeod` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`ExtEventVectorInfoGeod` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`NrSV`	u1			Number of satellites for which corrections are available from the base station identified by the `ReferenceID` field.
`Error`	u1			PVT error code. The following values are defined: 0: No Error 1: Not enough measurements 2: Not enough ephemerides available 3: DOP too large (larger than 15) 4: Sum of squared residuals too large 5: No convergence 6: Not enough measurements after outlier rejection 7: Position output prohibited due to export laws 8: Not enough differential corrections available 9: Base station coordinates unavailable 10: Ambiguities not fixed and user requested to only output RTK-fixed positions
`Mode`	u1			Bit field indicating the GNSS PVT mode, as follows: Bits 0-3: type of PVT solution: 0: No GNSS PVT available (the `Error` field indicates the cause of the absence of the PVT solution) 1: Stand-Alone PVT 2: Differential PVT 3: Fixed location 4: RTK with fixed ambiguities 5: RTK with float ambiguities 6: SBAS aided PVT 7: moving-base RTK with fixed ambiguities 8: moving-base RTK with float ambiguities 10: Precise Point Positioning (PPP) 12: Reserved Bits 4-5: Reserved Bit 6: Set if the user has entered the command **`setPVTMode, Static, , auto`** and the receiver is still in the process of determining its fixed position. Bit 7: 2D/3D flag: set in 2D mode (height assumed constant and not computed).
`Misc`	u1			Bit field containing miscellaneous flags: Bit 0: Set if the baseline points to the base station ARP. Unset if it points to the antenna phase center, or if unknown. Bit 1: Set if the phase center offset is compensated for at the rover (i.e. the baseline starts from the antenna ARP), unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Reserved
`DeltaEast`	f8	1 m	$-2\cdot10^{10}$	East baseline component (from rover to base)
`DeltaNorth`	f8	1 m	$-2\cdot10^{10}$	North baseline component (from rover to base)
`DeltaUp`	f8	1 m	$-2\cdot10^{10}$	Up baseline component (from rover to base)
`DeltaVe`	f4	1 m / s	$-2\cdot10^{10}$	East velocity of base with respect to rover
`DeltaVn`	f4	1 m / s	$-2\cdot10^{10}$	North velocity of base with respect to rover
`DeltaVu`	f4	1 m / s	$-2\cdot10^{10}$	Up velocity of base with respect to rover
`Azimuth`	u2	0.01 degrees	65535	Azimuth of the base station (from 0 to 360°, increasing towards east)
`Elevation`	i2	0.01 degrees	−32768	Elevation of the base station (from -90° to 90°)
`ReferenceID`	u2			Base station ID
`CorrAge`	u2	0.01 s	65535	Age of the oldest differential correction used for this baseline computation.

`SignalInfo`	u4		0	Bit field indicating the GNSS signals for which differential corrections are available from the base station identified by `ReferenceID`. If bit *i* is set, corrections for the signal type having index *i* are available. The signal numbers are listed in section 4.1.10. Bit 0 (GPS-C/A) is the LSB of `SignalInfo`.
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.13 Differential Correction Blocks

`DiffCorrIn`	Number:	5919
	`"OnChange"`	interval: each time a RTCM or CMR message is received

The `DiffCorrIn` block contains incoming RTCM or CMR messages. The length of the block depends on the message type and contents.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Mode`	u1			0: RTCMv2 1: CMRv2 2: RTCMv3 3: RTCMV (a proprietary variant of RTCM2) 4: SPARTN 5: Reserved
`Source`	u1		255	Indicates the receiver connection from which the message has been received: 0: COM1 1: COM2 2: COM3 3: COM4 4: USB1 5: USB2 6: IP connection 7: SBF file 8: L-Band (message decoded by the built-in L-band demodulator) 9: NTRIP 10: OTG1 11: OTG2 12: Bluetooth 15: UHF modem 16: IPR connection 17: Direct call port 18: IPS connection
If the `Mode` field is 0 then this field is available:				
`RTCM2Words`	u4[*N*]			30-bit words of the RTCM2 message. The Data Word Length (number of 32 bit words) is variable and depends on the RTCM2 message contents. It can be computed by the following piece of C code: `N = 2 + ((RTCM2Words[1]»9) & 0x1f);` *N* can range from 2 to 33. The first two words are the RTCM2 message header and they are always present. Each of the words is organized as follows: Bits 0-5: 6 parity bits. They are provided for the sake of completeness. Parity doesn't need to be checked, since the `DiffCorrIn` block only contains valid words. Bits 6-29: 24 information-containing bits of the word. The first received bit is the MSB. Bits 30-31: bit 0 and 1 of the preceding word
If the `Mode` field is 1 then this field is available:				
`CMRMessage`	u1[*N*]			*N* depends on the CMR message type.
If the `Mode` field is 2 then this field is available:				

RTCM3Message	u1[*N*]			*N* depends on the RTCM 3 message type.
If the Mode field is 3 then this field is available:				
RTCMVMessage	u1[*N*]			*N* depends on the RTCMV message type.
Padding	u1[..]			Padding bytes, see 4.1.5

`BaseStation`	Number:	5949
	`"OnChange"` interval:	block generated each time a differential correction message related to the base station coordinates is received

The `BaseStation` block contains the ECEF coordinates of the base station the receiver is currently connected to. This block helps users accessing the base station coordinates via SBF instead of having to decode the specific differential correction message (see the `DiffCorrIn` SBF block above).

The interpretation to give to the X, Y, Z ECEF coordinates is dependent on the value of the `Source` field:

Value of Source	Interpretation of X, Y, Z
0, 4 or 10	Coordinate of the L1 phase center
2 or 8	Antenna reference point
9	Proprietary

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`BaseStationID`	u2			The base station ID
`BaseType`	u1			Base station type: 0: Fixed 1: Moving (reserved for future use) 255: Unknown
`Source`	u1			Source of the base station coordinates: 0: RTCM 2.x (Msg 3) 2: RTCM 2.x (Msg 24) 4: CMR 2.x (Msg 1) 8: RTCM 3.x (Msg 1005 or 1006) 9: RTCMV (Msg 3) 10: CMR+ (Type 2)
`Datum`	u1		255	Not applicable
`Reserved`	u1			Reserved for future use, to be ignored by decoding software
`X`	f8	1 m		Antenna X coordinate expressed in the datum specified by the `Datum` field
`Y`	f8	1 m		Antenna Y coordinate
`Z`	f8	1 m		Antenna Z coordinate
`Padding`	u1[..]			Padding bytes, see 4.1.5

`RTCMDatum`	Number:	4049
	`"OnChange"` interval:	block generated each time a set of transformation parameters is received

This block reports the source and target datum names as transmitted in RTCM 3.x message types 1021 or 1022. It also reports the corresponding height and quality indicators.

If a service provider only sends out message types 1021 or 1022, this block is transmitted immediately after reception of MT1021 or MT1022. If message types 1023 or 1024 are also sent out, this block is transmitted after the reception of these messages and the `QualityInd` field is set accordingly.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SourceCRS`	c1[32]			Name of the source Coordinate Reference System, right-padded with zeros.
`TargetCRS`	c1[32]			Name of the target Coordinate Reference System, right-padded with zeros.
`Datum`	u1			See the `Datum` field in the `PosLocal` and `PosProjected` SBF blocks. Datum is set to 255 if this `SourceCRS/TargetCRS` pair is currently not used by the receiver.
`HeightType`	u1			Height Indicator field from MT1021 and MT1022. This field indicates how to interpret the height reported in the `PosLocal` and the `PosProjected` SBF blocks: 0: Geometrical height 1: Physical height (height definition in target CRS) 2: Physical height (height definition in source CRS)
`QualityInd`	u1			Bit field indicating the maximum approximation error after applying the transformation: Bits 0-3: horizontal quality indicator: 0: Unknown quality 1: Quality better than 21 mm (from MT1021/1022) 2: Quality 21 to 50 mm (from MT1021/1022) 3: Quality 51 to 200 mm (from MT1021/1022) 4: Quality 201 to 500 mm (from MT1021/1022) 5: Quality 501 to 2000 mm (from MT1021/1022) 6: Quality 2001 to 5000 mm (from MT1021/1022) 7: Quality worse than 5001 mm (from MT1021/1022) 9: Quality 0 to 10 mm (from MT1023/1024) 10: Quality 11 to 20 mm (from MT1023/1024) 11: Quality 21 to 50 mm (from MT1023/1024) 12: Quality 51 to 100 mm (from MT1023/1024) 13: Quality 101 to 200 mm (from MT1023/1024) 14: Quality 201 to 500 mm (from MT1023/1024) 15: Quality worse than 501 mm (from MT1023/1024) Bits 4-7: vertical quality indicator, same definition as bits 0-3.
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.14 L-Band Demodulator Blocks

`LBandTrackerStatus`	Number:	4201
	`"OnChange"`	interval: 1s

The `LBandTrackerStatus` block provides general information on the tracking status of the L-band signals.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of L-band trackers for which data is provided in this SBF block, i.e. number of `TrackData` sub-blocks.
`SBLength`	u1	1 byte		Length of one sub-block
TrackData	...	...		*A succession of N `TrackData` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`TrackData` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`Frequency`	u4	1 Hz	0	Nominal frequency of the beam for which data is provided in this sub-block.
`Baudrate`	u2	1 baud	0	Baudrate of the beam
`ServiceID`	u2			Service ID of the beam. Set to 0 for the LBAS1 beam. Set to 1 for the LBAS2 beam when received through an NTRIP connection. This field must be ignored if the `Status` field is set to anything else than 3 (Locked).
`FreqOffset`	f4	1 Hz	$-2 \cdot 10^{10}$	Frequency offset of the demodulator, if available
`CN0`	u2	0.01 dB-Hz	0	Current C/N_0 value
`AvgPower`	i2	0.01 dB	−32768	Not applicable.
`AGCGain`	i1	1 dB	−128	L-band AGC gain, in dB.
`Mode`	u1			Current operation mode: 0: normal
`Status`	u1			Current status: 0: Idle 1: Search 2: FrameSearch 3: Locked
Rev 2 `SVID`	u1			Satellite ID, see 4.1.9
Rev 1 `LockTime`	u2	1 s		Lock time to the L-band signal, clipped to 65535 seconds.
Rev 3 `Source`	u1			L-band tracking module: 0: Unknown 1: Internal 2: LBR board 3: NTRIP. L-band data received over NTRIP. In that case, the other fields in this sub-block are not applicable and set to their Do-Not-Use value.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`LBAS1DecoderStatus`	Number:	4202
	`"OnChange"` interval:	Block generated each time a status update is received from the LBAS1 decoder

The `LBAS1DecoderStatus` block provides general information on the LBAS1 L-band decoder.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Reserved`	u1[2]			Reserved for future use. to be ignored by decoding software
`Status`	u1			Status of the Decoder: 0: No Signal (more than 30 seconds no valid data) 1: Search 2: Locked 3: Locked w Error
`Access`	u1			Access status: 0: Access disabled 1: Access enabled
`GeoGatingMode`	u1			GeoGating mode 0: N/A 1: Non-maritime usage 2: Maritime and non-maritime usage 3: Coastal and non-maritime usage 4: 60 km nearshore and non-maritime usage
`GeoGatingStatus`	u1			GeoGating status.
`Event`	u4			Bit field indicating whether an event occurred previously. If this field is not equal to zero, at least one event has occurred. Bit 0: Beamtable Update Bit 1: Station List Update Bit 2: Access Changed Bit 3: Message Received Bit 4: Subscription Error Bits 5-31: Reserved
`LeaseTime`	u4	1 s	4294967295	Allocated lease time
`LeaseRemaining`	u4	1 s	4294967295	Remaining lease time
`LocalAreaLat`	i4	1.0/3600.0 degrees	−2147483648	Local area center latitude, positive North.
`LocalAreaLon`	i4	1.0/3600.0 degrees	−2147483648	Local area center longitude, positive East of Greenwich.
`LocalAreaRadius`	u2	1000 m	65535	Local area radius.
`LocalAreaStatus`	u1			Local area status: 0: Local area disabled 1: Waiting for position 16: Range check 129: User is in range 130: User is out of range 255: Position is too old
`Reserved1`	u1			Reserved for future use, to be ignored by decoding software.
`SubscrEndYear`	i1	1 year	−128	Subscription end date, year (2 digits). From 0 to 99.
`SubscrEndMonth`	i1	1 month	−128	Subscription end date, month. From 1 to 12.

Rev 1

Rev 1	`SubscrEndDay`	i1	1 day	−128	Subscription end date, day. From 1 to 31.
	`SubscrEndHour`	i1	1 hour	−128	Subscription end date, hour (UTC). From 0 to 23.
	`PAC`	c1[20]			Product activation code, right padded with zeros.
Rev 2	`VelocityLimit`	u1	1 m / s		Speed Gating : Velocity Limit.
	`SpeedGatingStatus`	u1			Speed Gating status: 0: Speed Gating disabled 1: Waiting for velocity 129: User is in speed limit 130: Speed limit exceeded 255: Velocity error
	`Padding`	u1[..]			Padding bytes, see 4.1.5

`LBAS1Messages`	Number:	4203
	`"OnChange"` interval:	Block generated each time an over-the-air message is received by the LBAS1 decoder

The `LBAS1Messages` block contains the over-the-air message decoded from LBAS1.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`MessageLength`	u2	1 byte		Length of the message in this block. Maximum message length is 512 bytes
`Message`	c1[MessageLength]			Over-The-Air message
`Padding`	u1[..]			Padding bytes, see 4.1.5

`LBandBeams`	Number:	4204
	`"OnChange"` interval:	Block generated each time beam status data is decoded

This block contains the name, longitude and beam frequency of the L-band geostationary satellites known by the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of L-band beams for which data is provided in this SBF block, i.e. number of `BeamInfo` sub-blocks.
`SBLength`	u1	1 byte		Length of one sub-block
BeamInfo	...	...		*A succession of N `BeamInfo` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BeamInfo` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`SVID`	u1			SVID associated to the satellite for which information is provided in this sub-block. SVID ranges from 107 to 119. See also section 4.1.9.
`SatName`	c1[9]			Satellite Name, right padded with zeros
`SatLongitude`	i2	0.01 degrees	−32768	Satellite Longitude (positive east of Greenwich)
`BeamFreq`	u4	1 Hz	0	L-band beam center frequency
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.15 Status Blocks

`ChannelStatus`	Number: 4013
	`"OnChange"` interval: default PVT output rate (see 4.1.8)

This block describes the current satellite allocation and tracking status of the active receiver channels. Active channels are channels to which a satellite has been allocated.

This block uses a two-level sub-block structure analogous to that of the `MeasEpoch` block. For each active channel, a `ChannelSatInfo` sub-block contains all satellite-dependent information such as health, azimuth and elevation. Each of these sub-blocks contains `N2` `ChannelStateInfo` sub-blocks, `N2` being the number of active antennas in a given channel (for single-antenna receivers, `N2` is one). The `ChannelStateInfo` reports information such as the tracking status and PVT usage of a given signal type tracked on a given antenna.

Inactive channels are not contained in the `ChannelStatus` block.

Health, tracking and PVT status fields are available for each satellite. These status fields consist of a sequence of up to 8 two-bit fields. Each 2-bit field contains the status of one of the signals transmitted by the satellite. The position of the 2 bits corresponding to a given signal is dependent on the constellation, but is otherwise fixed. It is indicated in the tables below.

GPS:

Reserved		Reserved		L1C		L5		L2C		P2(Y)		P1(Y)		L1CA	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

GLONASS:

Reserved		Reserved		Reserved		L3		L2CA		L2P		L1P		L1CA	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Galileo:

Reserved		E5-AltBOC		E5b		E5a		E6BC		E6A		L1BC		L1A	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SBAS:

Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		L5		L1	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

BeiDou:

Reserved		Reserved		B2b		B2a		B1C		B3I		B2I		B1I	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

QZSS:

L5S		L1CB		L1S		L1C		L6		L5		L2C		L1CA	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

NavIC/IRNSS:

Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		L5	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of channels for which status are provided in this SBF block, i.e. number of `ChannelSatInfo` sub-blocks. If `N` is 0, there are no active channels available for this epoch.
`SB1Length`	u1	1 byte		Length of a `ChannelSatInfo` sub-block, excluding the nested `ChannelStateInfo` sub-blocks
`SB2Length`	u1	1 byte		Length of a `ChannelStateInfo` sub-block
`Reserved`	u1[3]			Reserved for future use, to be ignored by decoding software
SatInfo	...	...		*A succession of N `ChannelSatInfo` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`ChannelSatInfo` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`SVID`	u1			Satellite ID, see 4.1.9
`FreqNr`	u1		0	For GLONASS FDMA signals, this is the frequency number, with an offset of 8. It ranges from 1 (corresponding to an actual frequency number of -7) to 21 (corresponding to an actual frequency number of 13). Otherwise, `FreqNr` is reserved and must be ignored by the decoding software.
`Reserved1`	u1[2]			Reserved for future use, to be ignored by decoding software
`Azimuth/RiseSet`	u2	1 degree	511 3	bit field: Bits 0-8: Azimuth [0,359]. 0 is North, and Azimuth increases towards East. Bits 9-13: Reserved Bits 14-15: Rise/Set Indicator: 0: Satellite setting 1: Satellite rising 3: Elevation rate unknown
`HealthStatus`	u2			Sequence of 2-bit health status fields, each of them taking one of the following values: 0 : health unknown, or not applicable 1 : healthy 3 : unhealthy The 2-bit health status is a condensed version of the health status as sent by the satellite. For SBAS, the health status is set from the almanac data (MT17).
`Elevation`	i1	1 degree	−128	Elevation [-90,90] relative to local horizontal plane
`N2`	u1			Number of `ChannelStateInfo` blocks following this `ChannelSatInfo` block. There is one `ChannelStateInfo` sub-block per antenna.
`RxChannel`	u1			Channel number, see section 4.1.11.
`Reserved2`	u1			Reserved for future use, to be ignored by decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5
StateInfo	...	...		*A succession of N2 `ChannelStateInfo` sub-blocks, see definition below*

`ChannelStateInfo` sub-block definition:

Parameter	Type	Units	Description
`Antenna`	u1		Antenna number (0 for main antenna)
`Reserved`	u1		Reserved for future use, to be ignored by decoding software
`TrackingStatus`	u2		Sequence of 2-bit tracking status fields, each of them taking one of the following values: 0: idle or not applicable 1: Search 2: Sync 3: Tracking
`PVTStatus`	u2		Sequence of 2-bit PVT status fields, each of them taking one of the following values: 0: not used 1: waiting for ephemeris 2: used 3: rejected
`PVTInfo`	u2		Internal info
`Padding`	u1[..]		Padding bytes, see 4.1.5

`ReceiverStatus`	Number:	4014
	`"OnChange"`	interval: 1s

The `ReceiverStatus` block provides general information on the status of the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`CPULoad`	u1	1 %	255	Load on the receiver's CPU. The load should stay below 80% in normal operation. Higher loads might result in data loss.
`ExtError`	u1			Bit field reporting external errors, i.e. errors detected in external data. Upon detection of an error, the corresponding bit is set for a duration of one second, and then resets. Bit 0: SISERROR: set if a violation of the signal-in-space ICD has been detected for at least one satellite while that satellite is reported as healthy. Use the command "**`lif,SisError`**" for details. Bit 1: DIFFCORRERROR: set when an anomaly has been detected in an incoming differential correction stream, causing the receiver to fail to decode the corrections. Use the command "**`lif,DiffCorrError`**" for details. Bit 2: EXTSENSORERROR: set when a malfunction has been detected on at least one of the external sensors connected to the receiver. Use the command "**`lif, ExtSensorError`**" for details. Bit 3: SETUPERROR: set when a configuration/setup error has been detected. An example of such error is when a remote NTRIP Caster is not reachable. Use the command "**`lif, SetupError`**" for details. Bits 4-7: Reserved
`UpTime`	u4	1 s		Number of seconds elapsed since the start-up of the receiver, or since the last reset.

`RxState`	u4			Bit field indicating the status of key components of the receiver: Bit 0: Reserved Bit 1: ACTIVEANTENNA: this bit is set when an active GNSS antenna is sensed, i.e. when current is drawn from the antenna connector. In dual-antenna receivers, this bit alternatively reports the status of the main and the auxiliary antenna in intervals of 10 seconds. For example, if only the main antenna is drawing current, the bit will be cyclically set for 10s and unset for 10s. Bit 2: EXT_FREQ: this bit is set if an external frequency reference is detected at the 10 MHz input, and cleared if the receiver uses its own internal clock. Bit 3: EXT_TIME: this bit is set if a pulse has been detected on the PPS IN connector. Bit 4: WNSET: see corresponding bit in the `SyncLevel` field of the `ReceiverTime` block. Bit 5: TOWSET: see corresponding bit in the `SyncLevel` field of the `ReceiverTime` block. Bit 6: FINETIME: see corresponding bit in the `SyncLevel` field of the `ReceiverTime` block. Bit 7: INTERNALDISK_ACTIVITY: this bit is set for one second each time data is logged to the internal disk (`DSK1`). If the logging rate is larger than 1 Hz, set continuously. Bit 8: INTERNALDISK_FULL: this bit is set when the internal disk (`DSK1`) is full. A disk is full when it is filled to 95% of its total capacity. Bit 9: INTERNALDISK_MOUNTED: this bit is set when the internal disk (`DSK1`) is mounted. Bit 10: INT_ANT: this bit is set when the GNSS RF signal is taken from the internal antenna input, and cleared when it comes from the external antenna input (only applicable on receiver models featuring an internal antenna input). Bit 11: REFOUT_LOCKED: if set, the 10-MHz frequency provided at the REF OUT connector is locked to GNSS time. Otherwise it is free-running. Bit 12: Reserved Bit 13: EXTERNALDISK_ACTIVITY: this bit is set for one second each time data is logged to the external disk (`DSK2`). If the logging rate is larger than 1 Hz, set continuously. Bit 14: EXTERNALDISK_FULL: this bit is set when the external disk (`DSK2`) is full. A disk is full when it is filled to 95% of its total capacity. Bit 15: EXTERNALDISK_MOUNTED: this bit is set when the external disk (`DSK2`) is mounted. Bit 16: PPS_IN_CAL: this bit is set when PPS IN delay calibration is ongoing. Only applicable to PolaRx5TR receivers. Bit 17: DIFFCORR_IN: this bit is set for one second each time differential corrections are decoded. If the input rate is larger than 1 Hz, set continuously. Bit 18: INTERNET: this bit is set when the receiver has internet access. If not set, there is either no internet access, or the receiver could not reliably determine the status. Bits 19-31: Reserved

Rev 1

`RxError`	u4			Bit field indicating whether an error occurred previously. If this field is not equal to zero, at least one error has been detected. Bit 0: Reserved Bit 1: Reserved Bit 2: Reserved Bit 3: SOFTWARE: set upon detection of a software warning or error. This bit is reset by the command "**`lif, error`**". Bit 4: WATCHDOG: set when the watchdog expired at least once since the last power-on. Bit 5: ANTENNA: set when antenna overcurrent condition is detected. Bit 6: CONGESTION: set when an output data congestion has been detected on at least one of the communication ports of the receiver during the last second. Bit 7: Reserved Bit 8: MISSEDEVENT: set when an external event congestion has been detected during the last second. It indicates that the receiver is receiving too many events on its EVENTx pins. Bit 9: CPUOVERLOAD: set when the CPU load is larger than 90%. Bit 10: INVALIDCONFIG: set if one or more configuration file (e.g. permissions) is invalid or absent. Bit 11: OUTOFGEOFENCE: set if the receiver is currently out of its permitted region of operation (geofencing). Bit 12: Reserved Bit 13: Reserved Bit 14: Reserved Bit 15: Reserved Bit 16: Reserved Bits 17-31: Reserved
`N`	u1			Number of `AGCState` sub-blocks this block contains.
`SBLength`	u1	1 byte		Length of a `AGCState` sub-block.
`CmdCount`	u1		0	Command cyclic counter, incremented each time a command is entered that changes the receiver configuration. After the counter has reached 255, it resets to 1.
`Temperature`	u1	1 °C	0	Receiver temperature with an offset of 100. Remove 100 to get the temperature in degree Celsius.
AGCState	...	...		*A succession of N `AGCState` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`AGCState` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`FrontEndID`	u1			Bit field indicating the frontend code and antenna ID: Bits 0-4: frontend code: 0: GPSL1/E1 1: GLOL1 2: E6 3: GPSL2 4: GLOL2 5: L5/E5a 6: E5b/B2I 7: E5(a+b) 8: Combined GPS/GLONASS/SBAS/Galileo L1 9: Combined GPS/GLONASS L2 10: MSS/L-band 11: B1I 12: B3I 13: S-band Bits 5-7: Antenna ID: 0 for main, 1 for *Aux1* and 2 for *Aux2*

`Gain`	i1	1 dB	−128	AGC gain, in dB. The Do-Not-Use value is used to indicate that the frontend PLL is not locked.
`SampleVar`	u1		0	Normalized variance of the IF samples. The nominal value for this variance is 100.
`BlankingStat`	u1	1 %		Current percentage of samples being blanked by the pulse blanking unit. This field is always 0 for receiver without pulse blanking unit.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`SatVisibility`	Number:	4012
	`"OnChange"`	interval: 1s

This block contains the azimuth and elevation of all the satellites above the horizon for which the ephemeris or almanac is available.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of satellites for which information is provided in this SBF block, i.e. number of `SatInfo` sub-blocks.
`SBLength`	u1	1 byte		Length of one `SatInfo` sub-block
SatInfo	...	...		*A succession of N `SatInfo` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`SatInfo` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`SVID`	u1			Satellite ID, see 4.1.9
`FreqNr`	u1		0	For GLONASS FDMA signals, this is the frequency number, with an offset of 8. It ranges from 1 (corresponding to an actual frequency number of -7) to 21 (corresponding to an actual frequency number of 13). Otherwise, `FreqNr` is reserved and must be ignored by the decoding software.
`Azimuth`	u2	0.01 degrees	65535	Azimuth. 0 is North, and azimuth increases towards East.
`Elevation`	i2	0.01 degrees	−32768	Elevation relative to local horizontal plane.
`RiseSet`	u1			Rise/set indicator: 0: satellite setting 1: satellite rising 255: elevation rate unknown
`SatelliteInfo`	u1			Satellite visibility info based on: 1: almanac 2: ephemeris 255: unknown
`Padding`	u1[..]			Padding bytes, see 4.1.5

`InputLink`	Number: 4090 `"OnChange"` interval: 1s

The `InputLink` block reports statistics of the number of bytes and messages received and accepted on each active connection descriptor.

Per connection descriptor, the receiver maintains two byte counters (`NrBytesReceived` and `NrBytesAccepted`) and two message counters (`NrMsgReceived` and `NrMsgAccepted`), which are reported in the sub-blocks. These counters provide useful information on the quality of the transmission link, and of the bandwidth efficiency.

These counters (as well as the age of the last message) are reset simultaneously on the following events:

- start-up of the receiver
- overflow of one of the counters
- change of input type
- deactivation of a connection descriptor, e.g. on disconnection of USB or IP ports.

There is one sub-block per connection descriptor for which statistics is available.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of connection descriptors for which communication link statistics are included
`SBLength`	u1	1 byte		Length of one `InputStatsSub` sub-block.
InputStats	...	...		*A succession of N `InputStatsSub` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`InputStatsSub` sub-block definition:

<table>
<tr><th>Parameter</th><th>Type</th><th>Units</th><th>Do-Not-Use</th><th>Description</th></tr>
<tr><td>CD</td><td>u1</td><td></td><td></td><td>Identifier of the connection to which this information applies:
<table>
<tr><th>Value of CD</th><th>Connection type</th><th>Example</th></tr>
<tr><td>0-31</td><td>COMx, with <i>x</i>=CD</td><td>1: COM1</td></tr>
<tr><td>32-47</td><td>USBx, with <i>x</i>=CD-32</td><td>33: USB1</td></tr>
<tr><td>48-63</td><td>OTGx, with <i>x</i>=CD-48</td><td>49: OTG1</td></tr>
<tr><td>64-95</td><td>IPx, with <i>x</i>=CD-54</td><td>64:IP10</td></tr>
<tr><td>96-127</td><td>DSKx, with <i>x</i>=CD-96</td><td>97:DSK1</td></tr>
<tr><td>128-159</td><td>NTRx, with <i>x</i>=CD-128 (NTRIP connections)</td><td>129:NTR1</td></tr>
<tr><td>160-191</td><td>IPSx, with <i>x</i>=CD-160 (IP server connections)</td><td>161:IPS1</td></tr>
<tr><td>192</td><td>BT01 (Bluetooth connection)</td><td></td></tr>
<tr><td>193</td><td>BT02 (Bluetooth connection)</td><td></td></tr>
<tr><td>196</td><td>UHF1 (UHF Modem)</td><td></td></tr>
<tr><td>200-205</td><td>IPRx, with <i>x</i>=CD-200 (IP receive connections)</td><td>201:IPR1</td></tr>
<tr><td>210</td><td>DCL1 (cellular data-call connection)</td><td></td></tr>
<tr><td>214</td><td>CAN1 (CAN stream interface)</td><td></td></tr>
<tr><td>215-219</td><td>Reserved</td><td></td></tr>
<tr><td>220</td><td>SPI1 (SPI interface)</td><td></td></tr>
<tr><td>221-255</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td>Type</td><td>u1</td><td></td><td></td><td>Type of data:
0: none
1: DaisyChain (includes "echo" messages)
32: CMD
33: SBF
34: AsciiDisplay (see <b>setDataInOut</b> command)
35: RINEX
36: CGGTTS
40: BINEX
64: NMEA
96: RTCMv2
97: RTCMv3
98: CMRv2
99: RTCMV (a proprietary variant of RTCMv2)
100: SPARTN
101: LBMP
110: raw LBAS1 from e.g. NTRIP
111: raw LBAS2 from e.g. NTRIP
118: raw LBAND data from Beam1
119: raw LBAND data from Beam2
120: raw LBAND data from Beam3
121: raw LBAND data from Beam4
128: Reserved
129: Reserved
130: Reserved
131: SBG (IMU sensor)
132: Reserved
133: Reserved
134: Reserved
135: Reserved
136: Reserved
137: ADIS
160: ASCIIIn</td></tr>
<tr><td>AgeOfLastMessage</td><td>u2</td><td>1 s</td><td>65535</td><td>Age of the last accepted message.
If the age is older than 65534s, it is clipped to 65534s.</td></tr>
<tr><td>NrBytesReceived</td><td>u4</td><td>1 byte</td><td>4294967295</td><td>Total number of bytes received (10)</td></tr>
</table>

`NrBytesAccepted`	u4	1 byte	4294967295	Total number of bytes [10] in messages that passed the check for this type of input (CRC, parity check, ...). The ratio of `NrBytesAccepted` to `NrBytesReceived` gives an indication of the quality of the communication link.
`NrMsgReceived`	u4	1 message		Total number of messages of type `Type` received.
`NrMsgAccepted`	u4	1 message		Total number of messages of type `Type` that were interpreted and used by the receiver. The ratio of `NrMsgAccepted` to `NrMsgReceived` gives an indication of the bandwidth usage efficiency
`Padding`	u1[..]			Padding bytes, see 4.1.5

---

(10) Note that, for RTCM 2.x, one 8-bit byte contains 6 RTCM data bits.

`OutputLink`	Number:	4091
	`"OnChange"`	interval: 1s

The `OutputLink` block reports statistics of the number of bytes sent on each active connection descriptor.

Per connection descriptor, the receiver maintains two byte counters `NrBytesProduced` and `NrBytesSent`, which are reported in the sub-block. They provide an indication of the amount of data output and data lost on a given connection.

These counters are reset simultaneously on the following events:

- start-up of the receiver
- overflow of one of the counters
- deactivation of a connection descriptor, e.g. on disconnection of USB or IP ports
- change of COM port settings.

There is one `OutputStatsSub` sub-block per connection descriptor for which statistics is available. Each `OutputStatsSub` sub-block contains a number of `OutputTypeSub` sub-blocks. These sub-blocks indicate which data type has been output through the connection in question during the last second. If no output happened during the last second, there is no `OutputTypeSub` sub-block.

Parameter	**Type**	**Units**	**Do-Not-Use**	**Description**
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N1`	u1			Number of `OutputStatsSub` sub-blocks in this `OutputLink` block.
`SB1Length`	u1	1 byte		Length of an `OutputStatsSub` sub-block, excluding the nested `OutputTypeSub` sub-block
`SB2Length`	u1	1 byte		Length of an `OutputTypeSub` sub-block
`Reserved`	u1[3]			Reserved for future use
OutputStats	...	...		*A succession of N1 `OutputStatsSub` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`OutputStatsSub` sub-block definition:

<table>
<tr><th>Parameter</th><th>Type</th><th>Units</th><th>Description</th></tr>
<tr><td>`CD`</td><td>u1</td><td></td><td>Identifier of the connection to which this information applies:
<table>
<tr><th>Value of CD</th><th>Connection type</th><th>Example</th></tr>
<tr><td>0-31</td><td>COMx, with *x*=CD</td><td>1: COM1</td></tr>
<tr><td>32-47</td><td>USBx, with *x*=CD-32</td><td>33: USB1</td></tr>
<tr><td>48-63</td><td>OTGx, with *x*=CD-48</td><td>49: OTG1</td></tr>
<tr><td>64-95</td><td>IPx, with *x*=CD-54</td><td>64:IP10</td></tr>
<tr><td>96-127</td><td>DSKx, with *x*=CD-96</td><td>97:DSK1</td></tr>
<tr><td>128-159</td><td>NTRx, with *x*=CD-128 (NTRIP connections)</td><td>129:NTR1</td></tr>
<tr><td>160-191</td><td>IPSx, with *x*=CD-160 (IP server connections)</td><td>161:IPS1</td></tr>
<tr><td>192</td><td>BT01 (Bluetooth connection)</td><td></td></tr>
<tr><td>193</td><td>BT02 (Bluetooth connection)</td><td></td></tr>
<tr><td>196</td><td>UHF1 (UHF Modem)</td><td></td></tr>
<tr><td>200-205</td><td>IPRx, with *x*=CD-200 (IP receive connections)</td><td>201:IPR1</td></tr>
<tr><td>210</td><td>DCL1 (cellular data-call connection)</td><td></td></tr>
<tr><td>214</td><td>CAN1 (CAN stream interface)</td><td></td></tr>
<tr><td>215-219</td><td>Reserved</td><td></td></tr>
<tr><td>220</td><td>SPI1 (SPI interface)</td><td></td></tr>
<tr><td>221-255</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td>`N2`</td><td>u1</td><td></td><td>Number of `OutputTypeSub` sub-blocks included at the end of this `OutputStatsSub` sub-block</td></tr>
<tr><td>`AllowedRate`</td><td>u2</td><td>1 kbyte / s</td><td>Maximum datarate recommended on this connection</td></tr>
<tr><td>`NrBytesProduced`</td><td>u4</td><td>1 byte</td><td>Total number of bytes produced by the receiver. See also the `NrBytesSent` field.</td></tr>
<tr><td>`NrBytesSent`</td><td>u4</td><td>1 byte</td><td>Total number of bytes actually sent (i.e. without congestions or transmission errors).
The ratio of `NrBytesSent` to `NrBytesProduced` gives an indication of the amount of bandwidth overload.
`NrBytesSent` and `NrBytesProduced` are 32-bit counters. If one of them overflows, both counters are reset to zero.</td></tr>
<tr><td>`NrClients`</td><td>u1</td><td></td><td>Number of clients currently connected to this connection. Most connection types can only serve one client at a time, but each IP server (IPS) port can serve up to eight simultaneous clients.
Note that when `NrClients` is more than one, the fields `NrBytesProduced` and `NrBytesSent` are the number of bytes produced and sent to each individual client.</td></tr>
<tr><td>`Reserved`</td><td>u1[3]</td><td></td><td>Reserved for future use</td></tr>
<tr><td>`Padding`</td><td>u1[..]</td><td></td><td>Padding bytes, see 4.1.5</td></tr>
<tr><td>*OutputType*</td><td>...</td><td>...</td><td>*A succession of N2 `OutputTypeSub` sub-blocks, see definition below*</td></tr>
</table>

Rev 1

`OutputTypeSub` sub-block definition:

Parameter	Type	Units	Description
`Type`	u1		Type of data: 0: none 1: DaisyChain (includes "echo" messages) 32: CMD 33: SBF 34: AsciiDisplay (see **`setDataInOut`** command) 35: RINEX 36: CGGTTS 40: BINEX 64: NMEA 96: RTCMv2 97: RTCMv3 98: CMRv2 99: RTCMV (a proprietary variant of RTCMv2) 118: raw LBAND data from Beam1 119: raw LBAND data from Beam2 120: raw LBAND data from Beam3 121: raw LBAND data from Beam4
`Percentage`	u1	1 %	Percentage of the produced bytes that belong to this type (during the last second)
`Padding`	u1[..]		Padding bytes, see 4.1.5

`NTRIPClientStatus`	Number: 4053 "OnChange" interval: 1s

This block reports the current status of the NTRIP client connections.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of NTRIP client connections for which status is provided in this block, i.e. number of `NTRIPClientConnection` sub-blocks.
`SBLength`	u1	1 byte		Length of one `NTRIPClientConnection` sub-block
NTRIPClientConnection	...	...		*A succession of N `NTRIPClientConnection` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`NTRIPClientConnection` sub-block definition:

Parameter	Type	Units	Description
`CDIndex`	u1		Index of the NTRIP connection (1 for NTR1, 2 for NTR2, etc) for which status is provided in this sub-block.
`Status`	u1		NTRIP client status: 0: Connection disabled 1: Initializing 2: Running, differential corrections are being received and the link statistics is available in the `InputLink` block. 3: Error detected, the error code is provided in the next field. 4: Retrying, client encountered an error, we are trying to reconnect. The error code is provided in the next field. 5: Disabled since the settings are a duplicate of another active NTRIP connection.
`ErrorCode`	u1		NTRIP error code: 0: No error 1: Initialization error (e.g. source table retrieval failure) 2: Authentication error 3: Connection error 4: Mountpoint does not exist 5: Mountpoint unavailable 6: Waiting for GGA 7: GGA sending disabled when required by mountpoint 8: Resolving host failed 9: Out of region 10: TLS setup error 11: TLS handshake error 12: TLS fingerprint error 13: TLS time not known 254: Unknown error
`Info`	u1		Bit field indicating miscellaneous info about the Connection status: Bit 0: TLS was used to make secure NTRIP connection if this bit is set Bits 1-7: Reserved
`Padding`	u1[..]		Padding bytes, see 4.1.5

`NTRIPServerStatus`	Number: 4122 `"OnChange"` interval: 1s

This block reports the current status of the NTRIP server connections.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of NTRIP server connections for which status is provided in this block, i.e. number of `NTRIPServerConnection` sub-blocks.
`SBLength`	u1	1 byte		Length of one `NTRIPServerConnection` sub-block
NTRIPServerConnection	...	...		*A succession of N `NTRIPServerConnection` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`NTRIPServerConnection` sub-block definition:

Parameter	Type	Units	Description
`CDIndex`	u1		Index of the NTRIP connection (1 for NTR1, 2 for NTR2, etc) for which status is provided in this sub-block.
`Status`	u1		NTRIP server status: 0: Connection disabled 1: Initializing 2: Running, differential corrections are being sent and the link statistics is available in the `OutputLink` block. 3: Error detected, the error code is provided in the next field. 4: Error detected. Currently trying to reconnect. The error code is provided in the next field. 5: Disabled since the settings are a duplicate of another active NTRIP connection.
`ErrorCode`	u1		NTRIP error code: 0: No error 1: Initialization error 2: Authentication error 3: Connection error 4: Mountpoint does not exist 5: Configuration conflict error 6: Resolving host failed 7: TLS setup error 8: TLS handshake error 9: TLS fingerprint error 10: TLS time not known 254: Unknown error
`Info`	u1		Bit field indicating miscellaneous info about the Connection status: Bit 0: TLS was used to make secure NTRIP connection if this bit is set Bits 1-7: Reserved
`Padding`	u1[..]		Padding bytes, see 4.1.5

`IPStatus`	Number:	4058
	`"OnChange"`	interval: output each time one or more IP parameters change

This block contains information on the receiver's Ethernet interface (hostname, IP address, gateway, netmask and MAC address).

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`MACAddress`	u1[6]			MAC address. The first byte corresponds to the MSB of the address.
`IPAddress`	u1[16]		All elements set to 0	IP address. For future upgradability, this field can contain a 128-bit IPv6 address. In the current firmware version, the first 12 bytes are always set to 0, and the last 4 bytes contain the IPv4 IP address, or are set to zero if the IP address is not known or not applicable.
`Gateway`	u1[16]		All elements set to 0	Gateway address. For future upgradability, this field can contain a 128-bit IPv6 address. In the current firmware version, the first 12 bytes are always set to 0, and the last 4 bytes contain the IPv4 IP address, or are set to zero if the gateway address is not known or not applicable.
`Netmask`	u1		255	Number of bits used to identify the network (CIDR notation).
`Reserved`	u1[3]			Reserved for future use, to be ignored by decoding software.
`HostName`	c1[32]			Receiver hostname on the Ethernet interface, or empty if not known.
`Padding`	u1[..]			Padding bytes, see 4.1.5

Rev 1

`WiFiAPStatus`	Number:	4054
	`"OnChange"`	interval: 1s

This block contains the IP address of the receiver when configured in WiFi access point. It also contains the list of all connected clients.

The current WiFi mode is reported in the `Mode` argument. When the receiver is configured in WiFi client mode or when WiFi is disabled, many fields are not applicable and are set to their Do-Not-Use value.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of WiFi clients currently connected to the receiver.
`SBLength`	u1	1 byte		Length of one `WiFiClient` sub-block
`APIPAddress`	u1[16]		All elements set to 0	IP address of the WiFi access point. For future upgradability, this field can contain a 128-bit IPv6 address. In the current firmware version, the first 12 bytes are always set to 0, and the last 4 bytes contain the IPv4 IP address, or are set to zero if the IP address is not known or not applicable.
`Mode`	u1			WiFi mode: 0: WiFi disabled 1: WiFi enabled in access point mode 2: WiFi enabled in client mode
`Hotspot`	u1			WiFi hotspot: 0: Hotspot disabled 1: Hotspot enabled and no Internet access 2: Hotspot enabled and Internet access 255: Hotspot not supported
`Reserved`	u1[2]			Reserved for future use, to be ignored by decoding software
WiFiClient	...	...		*A succession of N `WiFiClient` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`WiFiClient` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`ClientHostName`	c1[32]			Hostname of a WiFi client connected to the receiver, or empty if not known.
`ClientMACAddress`	u1[6]			MAC address of a WiFi client connected to the receiver. The first byte corresponds to the MSB of the address.
`ClientIPAddress`	u1[16]		All elements set to 0	IP address of a WiFi client connected to the receiver. For future upgradability, this field can contain a 128-bit IPv6 address. In the current firmware version, the first 12 bytes are always set to 0, and the last 4 bytes contain the IPv4 IP address, or are set to zero if the IP address is not known or not applicable.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`WiFiClientStatus`	Number: 4096 `"OnChange"` interval: 1s

This block contains WiFi status information of the receiver when configured in WiFi client mode.

When the receiver is not configured in WiFi client mode, many fields are not applicable and are set to their Do-Not-Use value.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`SSID_AP`	c1[32]			SSID of the access point the receiver is currently connected to. Empty when not connected.
`IPAddress`	u1[16]		All elements set to 0	IP address of the receiver as WiFi client. For future upgradability, this field can contain a 128-bit IPv6 address. In the current firmware version, the first 12 bytes are always set to 0, and the last 4 bytes contain the IPv4 IP address. All bytes are set to zero if the IP address is not applicable or not known yet (e.g. when the receiver is currently obtaining its IP address from the access point).
`APFrequency`	u1			The frequency of the WIFI access point the receiver is connected to. Possible WiFi AP Frequencies: 0: Unknown frequency of access point 1: 2.4GHz 2: 5GHz
`SigLevel`	i1	1 dBm	−128	WiFi signal power level
`Status`	u1			Bit field: Bits 0-3: WiFi client connection status: 0: Not connected, see the `ErrorCode` field for reason 1: Connecting 2: Connected Bits 4-7: Reserved
`ErrorCode`	u1			WiFi client error code: 0: No error 1: WiFi disabled or not in client mode 2: No reachable WiFi access point found 3: No known access point in reach (use the **`exeAddWiFiAccessPoint`** command to add an access point to the list of known access points) 4: Known access points were found but the receiver could not connect to them 5: Failed to connect to access point, wrong credentials
`Padding`	u1[..]			Padding bytes, see 4.1.5

`DynDNSStatus`	Number: 4105 `"OnChange"` interval: 1s

This block contains dynamic DNS (DynDNS) status information.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Status`	u1			DynDNS status: 0: DynDNS disabled 1: Updating IP address 2: IP address updated at the DynDNS server. DynDNS is ready to use. 254: Error detected, the error code is provided in the next field.
`ErrorCode`	u1			DynDNS error code: 0: No error 1: Unspecified error 2: Abusive update 3: User name and password mismatch 4: Not a credited user 5: Hostname is not a fully-qualified domain name 6: Hostname does not exist in this user account 7: Hostname blocked for update abuse 8: Bad agent 9: DNS error 10: DynDNS server problem or maintenance 11: DynDNS server not reachable
`IPAddress`	u1[16]		All elements set to 0	IP address that has been registered at the DynDNS server. For future upgradability, this field can contain a 128-bit IPv6 address. In the current firmware version, the first 12 bytes are always set to 0, and the last 4 bytes contain the IPv4 IP address, or are set to zero if the IP address is not known or not applicable (e.g. because registration failed).
`Padding`	u1[..]			Padding bytes, see 4.1.5

Rev 1

`PowerStatus`	Number:	4101
	`"OnChange"`	interval: 1s

This block contains information on the power supply (source and voltage).

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`PowerInfo`	u2	0.025 V	4095	Bit field: Bits 0-3: Current power source: 0: Unknown 1: Vin: DC supply through the Vin pins 2: PoE: Power over Ethernet 3: USB: Supply through VBUS 4: Battery: Supply from battery Bits 4-15: Voltage at the Vin pins
`Padding`	u1[..]			Padding bytes, see 4.1.5

`QualityInd`	Number:	4082
	`"OnChange"`	interval: 1s

The `QualityInd` block contains quality indicators for the main functions of the receiver. Each quality indicator is a value from 0 to 10, 0 corresponding to poor quality and 10 to very high quality.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of quality indicators contained in this block
`Reserved`	u1			Reserved for future use, to be ignored by decoding software.
`Indicators`	u2[N]		All elements set to 15	N successive quality indicators, coded as follows: Bits 0-7: Quality indicator type: 0: Overall quality 1: GNSS signals from main antenna 2: GNSS signals from aux1 antenna 11: RF power level from the main antenna 12: RF power level from the aux1 antenna 21: CPU headroom 25: OCXO stability (only available on PolaRx5S receivers) 30: Base station measurements. This indicator is only available in RTK mode. A low value could for example hint at severe multipath or interference at the base station, or also at ionospheric scintillation. 31: RTK post-processing. This indicator is only available when the position mode is not RTK. It indicates the likelihood of getting a cm-accurate RTK position when post-processing the current data. Bits 8-11: Value of this quality indicator (from 0 for low quality to 10 for high quality, or 15 if unknown) Bits 12-15: Reserved for future use, to be ignored by decoding software.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`DiskStatus`	Number: 4059 `"OnChange"` interval: 1s

This block reports the size and usage of the disks mounted on the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of `DiskData` sub-blocks this block contains.
`SBLength`	u1	1 byte		Length of one `DiskData` sub-blocks in bytes.
`Reserved`	u1[4]			Reserved for future use
DiskData	...	...		*A succession of N `DiskData` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`DiskData` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`DiskID`	u1			ID of the disk, starting at 1 for the internal SD Memory Card.
`Status`	u1			Bit field: Bit 0: DISK_MOUNTED: bit set when the disk is mounted. Bit 1: DISK_FULL: bit set when the disk is full. A disk is full when it is filled to 95% of its total capacity. Bit 2: DISK_ACTIVITY: bit set for one second each time data is written to the disk. If the logging rate is larger than 1 Hz, set continuously. Bit 3: LOGGING_ENABLED: bit set when at least one file is open on the disk, regardless of the logging rate. Bit 4: MOUNTING: bit set when disk is being mounted. Bit 5: FORMATTING: bit set when disk is being formatted. Bits 6-7: Reserved
`DiskUsageMSB`	u2		65535 (11)	16 MSB of the total disk usage. The disk usage in bytes is given by `DiskUsageMSB`*4294967296+`DiskUsageLSB`.
`DiskUsageLSB`	u4		4294967295 (11)	32 LSB of the total disk usage. The disk usage in bytes is given by `DiskUsageMSB`*4294967296+`DiskUsageLSB`.
`DiskSize`	u4	1 Mbyte	0	Total size of the disk, in megabytes.
`CreateDeleteCount`	u1			Counter incremented by one each time a file or a folder is created or deleted on this disk. This counter starts at zero at receiver start-up and restarts at zero after having reached 255.
`Error`	u1		255	Disk error: 0: No error 1: Disk partition is too large 2: Disk does not have any partition 3: File system check and recovery failed 4: Disk in use over USB 254: Disk mount failed due to unknown error
`Padding`	u1[..]			Padding bytes, see 4.1.5

Rev 1

Rev 1

---

(11) The disk usage is invalid if both `DiskUsageMSB` is 65535 and `DiskUsageLSB` is 4294967295.

`LogStatus`	Number:	4102
	`"OnChange"`	interval: 1s

This block reports the status of the different log sessions, including the status of the FTP push and CloudIt.

There is one `LogSession` sub-block per active log session. Each of these sub-blocks contains a number of `FileUploadStatus` sub-blocks that report the status of the uploads of SBF, RINEX, BINEX, NMEA, RTCMv3 and/or CGGTTS files logged in the corresponding log session.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N1`	u1			Number of `LogSession` sub-blocks this block contains.
`SB1Length`	u1	1 byte		Length of one `LogSession` sub-block, excluding the nested `FileUploadStatus` sub-blocks.
`SB2Length`	u1	1 byte		Length of an `FileUploadStatus` sub-block
`Reserved`	u1[3]			Reserved for future use
LogSessions	...	...		*A succession of N1 `LogSession` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`LogSession` sub-block definition:

Parameter	Type	Units	Description
`SessionID`	u1		ID of the log session of which the status is provided in this sub-block, starting at 1 for the `LOG1` session.
`SessionStatus`	u1		Bit field: Bit 0: DISK_UNMOUNTED: bit set when the disk where the log session is supposed to write is not mounted. Bit 1: DISK_FULL: bit set when the disk where the log session is supposed to write is full. A disk is full when it is filled to 95% of its total capacity. Bit 2: WAITING: bit set when the log session is currently disabled (see the **`setLogSession`** command) or, in the case of scheduled sessions, is waiting for the "start event". Bit 3: DISK_ACTIVITY: bit set for one second each time data is effectively written to the log session. If the logging rate is larger than 1 Hz, set continuously. Bits 4-7: Reserved
`N2`	u1		Number of `FileUploadStatus` sub-blocks included at the end of this `LogSession` sub-block
`Reserved`	u1		Reserved for future use
`Padding`	u1[..]		Padding bytes, see 4.1.5
FileUploadStatus	...	...	*A succession of N2 `FileUploadStatus` sub-blocks, see definition below*

`FileUploadStatus` sub-block definition:

Parameter	Type	Units	Description
`Type`	u1		File upload type: 0: This sub-block contains information on SBF FTP push 1: This sub-block contains information on RINEX FTP push 2: This sub-block contains information on BINEX FTP push 3: This sub-block contains information on NMEA FTP push 4: This sub-block contains information on RTCMv3 FTP push 5: This sub-block contains information on CGGTTS FTP push 10: This sub-block contains information on SBF CloudIt 11: This sub-block contains information on RINEX CloudIt 12: This sub-block contains information on BINEX CloudIt 13: This sub-block contains information on NMEA CloudIt 14: This sub-block contains information on RTCMv3 CloudIt
`ErrorCode`	u1		Latest error code reported by FTP push or CloudIt: 0: No error 1: FTP: Connection to FTP server failed 2: FTP: Wrong credentials 3: FTP: Only part of the file could be transferred 4: FTP: The local file is not available (e.g. it has been deleted by the user before being FTP pushed). 5: FTP: Server permissions prohibit file creation 6: FTP: Server permissions prohibit folder creation 7: FTP: The disk is not accessible. This can for example happen when a past FTP transfer did not succeed and the receiver tries it again but the disk has been unmounted in the meantime. 8: Unexpected error 11: CloudIt: Not authorized to upload files to CloudIt server 12: CloudIt: Timeout while trying to reach server 13: CloudIt: The redirect_uri does not match the one registered in the authentication server 14: CloudIt: Client error server response, see RxMessages for more information 15: CloudIt: Server unreachable or bad response 20: CloudIt: An internal incorrect configuration of CloudIt 21: CloudIt: One or more parameters are invalid 22: CloudIt: One or more parameters are missing 40: CloudIt: The local file is not available (e.g. it has been deleted by the user before being uploaded). 41: CloudIt: Failed to upload file to server
`RetryQueueSize`	u1		Number of files of the type identified in the `Type` field that could not be uploaded but that will be retried by the receiver at a later time.
`NrFailedTransfers`	u1		Number of files of the type identified in the `Type` field that could not be uploaded and for which the receiver will not retry anymore. This number is clipped to 254 if larger than 254.
`Padding`	u1[..]		Padding bytes, see 4.1.5

`RFStatus`	Number:	4092
	`"OnChange"`	interval: 1s

The `RFStatus` block reports on the quality of the radio-frequency (RF) signal received by the antenna(s). The `RFBand` sub-blocks provide a list of the frequency bands where interferences have been detected and/or mitigated, and the `Flags` field contains warnings that the receiver's output may be affected by non-authentic RF signals.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of RF bands for which data is provided in this SBF block, i.e. number of `RFBand` sub-blocks.
`SBLength`	u1	1 byte		Length of one sub-block
`Flags`	u1			Bit field: Bit 0: Set when the receiver determined that its output (position or raw measurements) may be affected by a spoofer, and may therefore be misleading. This bit is based on a set of built-in tests to check the authenticity of the GNSS signals. Note that this bit may be set even if no interference is detected (i.e. with no associated `RFBand` sub-blocks). Bit 1: Set when spoofing is detected by Galileo OSNMA. Bits 2-7: Reserved
`Reserved`	u1[3]			Reserved for future use, to be ignored by decoding software.
RFBand	...	...		*A succession of N `RFBand` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`RFBand` sub-block definition:

Parameter	Type	Units	Description
`Frequency`	u4	1 Hz	Center frequency of the RF band addressed by this sub-block.
`Bandwidth`	u2	1 kHz	Bandwidth of the RF band.
`Info`	u1		Info on this RF band: Bits 0-3: Mode: 1: This RF band is suppressed by a notch filter set manually with the command **`setNotchFiltering`**. 2: The receiver detected interference in this band, and successfully canceled it. 8: The receiver detected interference in this band. No mitigation applied. Bits 4-5: Reserved Bits 6-7: Antenna ID: 0 for main, 1 for *Aux1* and 2 for *Aux2*
`Padding`	u1[..]		Padding bytes, see 4.1.5

`P2PPStatus`	Number: 4238 "`OnChange`" interval: 1s

This block reports the status of the active P2PP (Point-to-Point Protocol) sessions. See the **`setPointToPoint`** command for details.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of active P2PP sessions for which status is provided in this block, i.e. number of `P2PPSession` sub-blocks.
`SBLength`	u1	1 byte		Length of one `P2PPSession` sub-block
P2PPSession	...	...		*A succession of N `P2PPSession` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`P2PPSession` sub-block definition:

Parameter	Type	Units	Description
`SessionID`	u1		Index of the P2PP session (1 for P2PP1, 2 for P2PP2, etc) for which status is provided in this sub-block.
`Port`	u1		Index for the COM port the P2PP session is configured on (1 for COM1, 2 for COM2, etc).
`Status`	u1		Bit field: Bit 0: Mode: Bit set if the P2PP session is in Server mode, and unset if it is in Client mode (future functionality). Bits 1-7: P2PP status: 0: Initializing 1: Waiting for Connection 2: Connected 3: Disconnecting 4: Error, see `ErrorCode` field below
`ErrorCode`	u1		P2PP error: 1: No error 2: Configuration 3: Port Acquisition 4: Port Lock 5: Start Daemon 6: Server Authentication 7: Client Authentication 8: Timeout on Activity 9: Timeout on Negotiation 10: Link Negotiation 255: Unspecified
`Padding`	u1[..]		Padding bytes, see 4.1.5

`CosmosStatus`	Number: 4243 `"OnChange"` interval: 1s

The `CosmosStatus` block provides information on the status of the Cosmos receiver service.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Status`	u1			The status of Cosmos receiver service: 0: Disabled 1: Running
`Padding`	u1[..]			Padding bytes, see 4.1.5

`GALAuthStatus`	Number:	4245
	`"OnChange"`	interval: 1s

The `GALAuthStatus` block contains the current status of the Galileo OSNMA authentication.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`OSNMAStatus`	u2			Bit field: Bits 0-2: status: 0: Disabled 1: Initializing 2: Waiting on NTP 3: Init failed - inconsistent time 4: Init failed - KROOT signature invalid 5: Init failed - invalid param received 6: Authenticating Bits 3-10: OSNMA initialization progress, expressed in percent [0-100]. This value will only be encoded when the OSNMA Status is initializing. Bits 11-15: Reserved
`TrustedTimeDelta`	f4	1 s	$-2 \cdot 10^{10}$	Time difference between external trusted and receiver time, positive when receiver time lags trusted time.
`GalActiveMask`	u8			Bit field indicating the Galileo satellites for which OSNMA results are available. If bit *i* is set, OSNMA authentication is available for Galileo satellite *i*+1.
`GalAuthenticMask`	u8			Bit field indicating the Galileo satellites successfully authenticated by OSNMA. If bit *i* is set, the navigation message from Galileo satellite *i*+1 is authentic. If bit *i* is not set and the corresponding bit is set in `GalActiveMask`, the navigation message from that satellite is non-authentic.
`GpsActiveMask`	u8			Bit field indicating the GPS satellites for which OSNMA results are available. If bit *i* is set, OSNMA authentication is available for GPS satellite *i*+1.
`GpsAuthenticMask`	u8			Bit field indicating the GPS satellites successfully authenticated by OSNMA. If bit *i* is set, the navigation message from GPS satellite *i*+1 is authentic. If bit *i* is not set and the corresponding bit is set in `GpsActiveMask`, the navigation message from that satellite is non-authentic.
`Padding`	u1[..]			Padding bytes, see 4.1.5

4.2.16 Miscellaneous Blocks

`ReceiverSetup`	Number:	5902
	`"OnChange"` interval:	Block generated each time a user-command is entered to change one or more values in the block (e.g. when entering the **`setMarkerParameters`** command)

The `ReceiverSetup` block contains parameters related to the receiver and its installation. When generating RINEX files, this block defines the RINEX file name and the contents of the header.

For all fields containing a string, if the length of the string is lower than the size of the corresponding field, the unused bytes are set to zero.

	Parameter	Type	Units	Do-Not-Use	Description
	`Sync1`	c1			Block Header, see 4.1.1
	`Sync2`	c1			
	`CRC`	u2			
	`ID`	u2			
	`Length`	u2	1 byte		
	`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
	`WNc`	u2	1 week	65535	
	`Reserved`	u1[2]			2 bytes reserved for future use, to be ignored by decoding software
	`MarkerName`	c1[60]			Marker name (set with **`setMarkerParameters`**).
	`MarkerNumber`	c1[20]			Marker number (set with **`setMarkerParameters`**).
	`Observer`	c1[20]			Observer name (set with **`setObserverParameters`**).
	`Agency`	c1[40]			Observer agency (set with **`setObserverParameters`**).
	`RxSerialNumber`	c1[20]			Receiver serial number.
	`RxName`	c1[20]			Receiver GNSS engine name.
	`RxVersion`	c1[20]			Receiver firmware version.
	`AntSerialNbr`	c1[20]			Serial number of the main antenna (set with **`setAntennaOffset`**).
	`AntType`	c1[20]			Type of the main antenna (set with **`setAntennaOffset`**).
	`deltaH`	f4	1 m		δH offset of the main antenna (set with **`setAntennaOffset`**).
	`deltaE`	f4	1 m		δE offset of the main antenna (set with **`setAntennaOffset`**).
	`deltaN`	f4	1 m		δN offset of the main antenna (set with **`setAntennaOffset`**).
Rev 1	`MarkerType`	c1[20]			Marker type (set with the **`setMarkerParameters`** command).
Rev 2	`GNSSFWVersion`	c1[40]			Version the firmware installed on the receiver.
Rev 3	`ProductName`	c1[40]			Product name.
Rev 4	`Latitude`	f8	1 rad	$-2\cdot10^{10}$	Latitude of the reference position, from $-\pi/2$ to $+\pi/2$, positive North of Equator. Use the **`setPVTMode`** command to set the reference position.
	`Longitude`	f8	1 rad	$-2\cdot10^{10}$	Longitude of the reference position, from $-\pi$ to $+\pi$, positive East of Greenwich. Use the **`setPVTMode`** command to set the reference position.
	`Height`	f4	1 m	$-2\cdot10^{10}$	Ellipsoidal height of the reference position (with respect to WGS84 ellipsoid). Use the **`setPVTMode`** command to set the reference position.
	`StationCode`	c1[10]			Station code (set with **`setMarkerParameters`**). This field can for example contains the four-letter IGS station code assigned to the receiver.
	`MonumentIdx`	u1			Monument index (set with **`setMarkerParameters`**). This index is used to identify the monument when there are multiple monuments at the same station.

Rev 4

`ReceiverIdx`	u1			Receiver index (set with **`setMarkerParameters`**). This index is used to identify the receiver when there are multiple receivers at the same monument.
`CountryCode`	c1[3]			ISO 3-character country code (set with the **`setMarkerParameters`** command).
`Reserved1`	c1[21]			Reserved.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`RxComponents`	Number:	4084
	`"OnChange"`	interval: 10s

This block contains information on various hardware and software components of the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u1			Number of components for which information is provided in this block.
`SBLength`	u1	1 byte		Length of one `Component` sub-block
`Reserved`	u1[4]			Reserved for future use, to be ignored by decoding software
Component	...	...		*A succession of N `Component` sub-blocks, see definition below*
`Padding`	u1[..]			Padding bytes, see 4.1.5

`Component` sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
`Type`	u1			Type of component described in this sub-block: 1: Motherboard 2: GNSS module 3: WiFi module 4: Cellular module 5: Bluetooth module 6: L-Band module 7: UHF module 8: Reserved 9: Reserved 10: Library
`CPULoad`	u1	1 %	255	Load on the component CPU, if applicable.
`Reserved`	u1[2]			Reserved for future use, to be ignored by decoding software
`Name`	c1[40]			Component name.
`SerialNumber`	c1[20]			Component serial number. Empty if not applicable.
`FWVersion`	c1[40]			Component firmware version. Empty if not applicable.
`MACAddress`	u1[6]			MAC address if applicable. The first byte corresponds to the MSB of the address. All bytes set to 0 for components for which no MAC address applies.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`RxMessage`	Number:	4103
	`"OnChange"` interval:	block generated each time a message needs to be sent

The receiver generates ASCII messages to help users follow the progress of processes such as file logging or FTP push (activity log). These messages are output in the `RxMessage` block, and they can also be retrieved from the command line using the `lif,RxMessages` command.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Type`	u1		255	Type of message contained in this block: 1: Asynchronous command reply 2: Message about internal logging 3: Message about FTP push 4: Message about Receiver Status 5: Message from slave GNSS receiver 6: Message about CloudIt
`Severity`	u1		255	Message severity: 1: Info 2: Warning 3: Error
`MessageID`	u4		0	A unique value associated to each message. This is a counter starting at 1 for the first message after boot and incrementing at each message.
`StringLn`	u2			Length of `Message` in characters, including the terminating \0.
`Reserved2`	u1[2]			Reserved, contents to be ignored.
`Message`	c1[StringLn]			Receiver message terminated by \0.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`Commands`	Number:	4015
	`"OnChange"`	interval: each time a user command is entered

Every time the user sends a command, a `Commands` block is output on all ports for which this block is enabled. The `Commands` SBF block is inserted in the SBF stream at the very moment when the command starts to take effect.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`Reserved`	u1[2]			Reserved for future use, to be ignored by decoding software.
`CmdData`	u1[*N*]			Command data, this is the command in the SNMP' format (reserved for maintenance and support only).
`Padding`	u1[..]			Padding bytes, see 4.1.5

`Comment`	Number:	5936
	`"OnChange"` interval:	block generated each time a comment is entered with **`setObserverComment`**

The `Comment` block contains a comment string as entered with the **`setObserverComment`** command.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	Receiver time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`CommentLn`	u2			Length of the `Comment` string, in characters. The maximum length of a comment is 120 characters.
`Comment`	c1[CommentLn]			Comment string, as entered with the **`setObserverComment`** command. Note that this string is not terminated by the "\0" character.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`BBSamples`	Number:	4040
	`"OnChange"` interval:	block generated each time new baseband samples are ready (typically at 2Hz)

The `BBSamples` block contains a series of successive complex baseband samples. These samples can be used for signal monitoring and for spectral analysis of the GNSS bands supported by the receiver.

Parameter	Type	Units	Do-Not-Use	Description
`Sync1`	c1			Block Header, see 4.1.1
`Sync2`	c1			
`CRC`	u2			
`ID`	u2			
`Length`	u2	1 byte		
`TOW`	u4	0.001 s	4294967295	External time stamp, see 4.1.3
`WNc`	u2	1 week	65535	
`N`	u2			Number of complex baseband samples contained in this block
`Info`	u1			Bit field as follows: Bits 0-2: Antenna ID: antenna from which the samples have been taken: 0 for main, 1 for *Aux1* and 2 for *Aux2*. Bits 3-7: Reserved
`Reserved`	u1[3]			Reserved for future use, to be ignored by decoding software.
`SampleFreq`	u4	1 Hz		Sampling frequency in Hz.
`LOFreq`	u4	1 Hz		Frequency of the local oscillator (LO) used to down-convert the RF signal to baseband.
`Samples`	u2[N]			N successive complex baseband samples (I+jQ), coded as follows: Bits 0-7: 8-bit Q component, two's complement. Bits 8-15: 8-bit I component, two's complement.
`Padding`	u1[..]			Padding bytes, see 4.1.5

`ASCIIIn`	Number: 4075
	`"OnChange"` interval: block generated each time an ASCII string is received

The `ASCIIIn` block contains a string that has been received on one of the receiver's connection ports.

More specifically, this block is output each time an end-of-line character is received on a communication port configured to receive `ASCIIIn` input (with the **`setDataInOut`** command). The string reported in this block contains all characters received since the previous occurrence of an end-of-line character.

The maximum length of the string is 2000 characters. If there are more than 2000 characters between the occurrence of two successive end-of-line characters, the string is discarded

<table>
<tr><th>Parameter</th><th>Type</th><th>Units</th><th>Do-Not-Use</th><th>Description</th></tr>
<tr><td>Sync1</td><td>c1</td><td></td><td></td><td>Block Header, see 4.1.1</td></tr>
<tr><td>Sync2</td><td>c1</td><td></td><td></td><td></td></tr>
<tr><td>CRC</td><td>u2</td><td></td><td></td><td></td></tr>
<tr><td>ID</td><td>u2</td><td></td><td></td><td></td></tr>
<tr><td>Length</td><td>u2</td><td>1 byte</td><td></td><td></td></tr>
<tr><td>TOW</td><td>u4</td><td>0.001 s</td><td>4294967295</td><td>Receiver time stamp, see 4.1.3</td></tr>
<tr><td>WNc</td><td>u2</td><td>1 week</td><td>65535</td><td></td></tr>
<tr><td>CD</td><td>u1</td><td></td><td></td><td>Identifier of the connection from which the data has been received:
<table>
<tr><th>Value of CD</th><th>Connection type</th><th>Example</th></tr>
<tr><td>0-31</td><td>COMx, with <i>x</i>=CD</td><td>1: COM1</td></tr>
<tr><td>32-47</td><td>USBx, with <i>x</i>=CD-32</td><td>33: USB1</td></tr>
<tr><td>48-63</td><td>OTGx, with <i>x</i>=CD-48</td><td>49: OTG1</td></tr>
<tr><td>64-95</td><td>IPx, with <i>x</i>=CD-54</td><td>64:IP10</td></tr>
<tr><td>128-159</td><td>NTRx, with <i>x</i>=CD-128 (NTRIP connections)</td><td>129:NTR1</td></tr>
<tr><td>192</td><td>BT01 (Bluetooth connection)</td><td></td></tr>
<tr><td>193</td><td>BT02 (Bluetooth connection)</td><td></td></tr>
<tr><td>196</td><td>UHF1 (UHF Modem)</td><td></td></tr>
<tr><td>200-205</td><td>IPRx, with <i>x</i>=CD-200 (IP receive connections)</td><td>201:IPR1</td></tr>
<tr><td>210</td><td>DCL1 (cellular data-call connection)</td><td></td></tr>
<tr><td>214</td><td>CAN1 (CAN stream interface)</td><td></td></tr>
<tr><td>215-255</td><td>Reserved</td><td></td></tr>
</table></td></tr>
<tr><td>Reserved1</td><td>u1[3]</td><td></td><td></td><td>Reserved, contents to be ignored.</td></tr>
<tr><td>StringLn</td><td>u2</td><td></td><td></td><td>Length of <code>ASCIIString</code> in characters.</td></tr>
<tr><td>SensorModel</td><td>c1[20]</td><td></td><td></td><td>Not supported, reserved for future use.</td></tr>
<tr><td>SensorType</td><td>c1[20]</td><td></td><td></td><td>Not supported, reserved for future use.</td></tr>
<tr><td>Reserved2</td><td>u1[20]</td><td></td><td></td><td>Reserved, contents to be ignored.</td></tr>
<tr><td>ASCIIString</td><td>c1[StringLn]</td><td></td><td></td><td>ASCII string. Note that this string is not terminated by the "\0" character. The string does not include the end-of-line character(s) (carrier return and/or line feed).</td></tr>
<tr><td>Padding</td><td>u1[..]</td><td></td><td></td><td>Padding bytes, see 4.1.5</td></tr>
</table>

4.2.17 Advanced Blocks

`SystemInfo`	Number:	6000
	`"OnChange"` interval: 1s	

This block contains various system parameters that can be used for maintenance and support.

The detailed definition of this block is not available in this document.

4.3 SBF Change Log

Date	Change Description
Oct 01, 2021	Added the `QZSRawL5S` block containing the raw QZSS L5S navigation bits
Apr 10, 2020	Added the `NTRIPServerStatus` block for the NTRIP server connection status
Feb 11, 2020	Added the `BDSRawB2b` block containing the raw BeiDou B2b navigation symbols
May 16, 2019	Added the `P2PPStatus` block to report the status of the Point-to-Point sessions
Apr 26, 2019	Added the `QZSAlm` block containing QZSS almanac parameters
Apr 8, 2019	Renamed `IRNSSRaw` to `NAVICRaw`
Mar 12, 2019	Added the `BDSAlm` block containing BeiDou almanac parameters
Aug 07, 2018	Added the `QZSRawL1S` block containing the raw QZSS L1S navigation bits
Aug 07, 2018	Added the `QZSRawL1C` block containing the raw QZSS L1C navigation symbols
May 31, 2018	Added the `GPSRawL1C` block containing the raw GPS L1C navigation symbols
Apr 19, 2018	Added the `BDSRawB1C` block containing the raw BeiDou B1C navigation symbols
Apr 19, 2018	Added the `BDSRawB2a` block containing the raw BeiDou B2a navigation symbols
Jun 20, 2017	Added the `BDSIon` and `BDSUtc` blocks containing BeiDou ionospheric and UTC offset parameters
Jun 15, 2017	Added the `Meas3PP` and `Meas3MP` blocks to supplement the `Meas3Ranges` block
Mar 6, 2017	Added the `Meas3Ranges`, `Meas3CN0HiRes` and `Meas3Doppler` blocks containing GNSS measurements
Mar 1, 2017	Renamed `CMPNav` to `BDSNav` and `CMPRaw` to `BDSRaw`
Nov 15, 2016	Added the `QZSRawL6` block containing raw QZSS L6 navigation bits
July 13, 2016	Added the `SystemInfo` block containing advanced information about systems
Nov 10, 2015	Added the `RxMessage` block containing the receiver activity log
Nov 5, 2015	Added the `LogStatus` block to report the status of the log sessions
Sep 29, 2015	Added the `PowerStatus` block to report the voltage of the power supply
Feb 04, 2015	Added the `QZSNav` block containing decoded QZSS navigation data
Jan 13, 2015	Added the `PosProjected` block containing plane grid coordinates
Dec 12, 2014	Added the base measurements quality indicator
Nov 6, 2014	Added the `RFStatus` block for interference mitigation monitoring
April 30, 2014	Added new values for the `Datum` field
April 22, 2014	Added the `DiskStatus` block reporting the disk usage and free space of the disks available on the receiver
Feb 21, 2014	Added the `NTRIPClientStatus` block for the NTRIP client connection status
June 24, 2013	Added the `RxComponents` block containing information on the various receiver's components
June 24, 2013	Added the `WiFiAPStatus` block for the WiFi status in access point mode
March 14, 2013	Added the `QualityInd` block containing various quality indicators
Feb 19, 2013	Added the `CMPNav` block containing decoded BeiDou navigation data

Feb 8, 2013	Fixed typo: field `t_oG` of `GALGstGps` changed to type `u4` and units of seconds
Jan 8, 2013	Added fields `HAccuracy`, `VAccuracy` and `Misc` to the `PVTCartesian` and `PVTGeodetic` blocks
Dec 19, 2012	Added PRNs 139 and 140 to the list of SBAS satellites
Oct 25, 2012	Added `RTCMDatum` and `PosLocal` blocks
Oct 19, 2012	Added `GEORawL5` block
Oct 1, 2012	Added new signal type for L-band and SBAS L5 signals (value 23 and 25)
Sep 29, 2012	Added `LBandBeams` block and added `SVID` field to `LBandTrackerStatus` block
Sep 20, 2012	Added field `PPPInfo` to the `PVTCartesian` and `PVTGeodetic` blocks
Jun 27, 2012	Added fields to the `LBAS1DecoderStatus` block to report various service subscription parameters
Feb 28, 2012	Added `GALSARRLM` block
Feb 6, 2012	Added QZSS signals and `QZSRawL1CA`, `QZSRawL2C` and `QZSRawL5` blocks

Appendix A

List of SBF Blocks

The following table provides the list of the SBF block names and numbers available on PolaRx5TR and a short description of the associated contents. The block number is contained in bits 0 to 12 of the block `ID` field (see section 4.1.1).

The "Flex Rate" column indicates whether a given block can be output at a user-defined rate and the "esoc" column whether it can be used as an argument of the **`exeSBFOnce`** command (see also section 4.1.8). The "Time stamp" column indicates which type of time is encoded in the block time stamp (see section 4.1.3 for details).

Block name	**Block No**	**Content description**	**Flex Rate**	**esoc**	**Time Stamp**
Measurement Blocks					
`MeasEpoch`	4027	Measurement set of one epoch	●	●	R
`MeasExtra`	4000	Additional info such as observable variance	●	●	R
`Meas3Ranges`	4109	Code, phase and CN0 measurements	●	●	R
`Meas3CN0HiRes`	4110	Extension of Meas3Ranges containing fractional C/N0 values	●	●	R
`Meas3Doppler`	4111	Extension of Meas3Ranges containing Doppler values	●	●	R
`Meas3PP`	4112	Extension of Meas3Ranges containing proprietary flags for data post-processing.	●	●	R
`Meas3MP`	4113	Extension of Meas3Ranges containing multipath corrections applied by the receiver.	●	●	R
`EndOfMeas`	5922	Measurement epoch marker	●	●	R
Navigation Page Blocks					
`GPSRawCA`	4017	GPS CA navigation subframe			S
`GPSRawL2C`	4018	GPS L2C navigation frame			S
`GPSRawL5`	4019	GPS L5 navigation frame			S
`GPSRawL1C`	4221	GPS L1C navigation frame			S
`GLORawCA`	4026	GLONASS CA navigation string			S
`GALRawFNAV`	4022	Galileo F/NAV navigation page			S
`GALRawINAV`	4023	Galileo I/NAV navigation page			S
`GALRawCNAV`	4024	Galileo C/NAV navigation page			S
`GEORawL1`	4020	SBAS L1 navigation message			S
`GEORawL5`	4021	SBAS L5 navigation message			S
`BDSRaw`	4047	BeiDou navigation page			S
`BDSRawB1C`	4218	BeiDou B1C navigation frame			S
`BDSRawB2a`	4219	BeiDou B2a navigation frame			S
`BDSRawB2b`	4242	BeiDou B2b navigation frame			S
`QZSRawL1CA`	4066	QZSS L1C/A or L1C/B navigation frame			S
`QZSRawL2C`	4067	QZSS L2C navigation frame			S
`QZSRawL5`	4068	QZSS L5 navigation frame			S
`QZSRawL6`	4069	QZSS L6 navigation message			S
`QZSRawL1C`	4227	QZSS L1C navigation frame			S
`QZSRawL1S`	4228	QZSS L1S navigation message			S
`QZSRawL5S`	4246	QZSS L5S navigation message			S
`NAVICRaw`	4093	NavIC/IRNSS subframe			S
GPS Decoded Message Blocks					
`GPSNav`	5891	GPS ephemeris and clock		●	S

Block name	Block No	Content description	Flex Rate	esoc	Time Stamp
`GPSAlm`	5892	Almanac data for a GPS satellite		●	S
`GPSIon`	5893	Ionosphere data from the GPS subframe 5		●	S
`GPSUtc`	5894	GPS-UTC data from GPS subframe 5		●	S
`GPSCNav`	4042	CNAV Ephemeris data for one satellite.		●	S
GLONASS Decoded Message Blocks					
`GLONav`	4004	GLONASS ephemeris and clock		●	S
`GLOAlm`	4005	Almanac data for a GLONASS satellite		●	S
`GLOTime`	4036	GLO-UTC, GLO-GPS and GLO-UT1 data		●	S
Galileo Decoded Message Blocks					
`GALNav`	4002	Galileo ephemeris, clock, health and BGD		●	S
`GALAlm`	4003	Almanac data for a Galileo satellite		●	S
`GALIon`	4030	NeQuick Ionosphere model parameters		●	S
`GALUtc`	4031	GST-UTC data		●	S
`GALGstGps`	4032	GST-GPS data		●	S
`GALSARRLM`	4034	Search-and-rescue return link message			S
BeiDou Decoded Message Blocks					
`BDSNav`	4081	BeiDou ephemeris and clock		●	S
`BDSCNav2`	4252	BeiDou B-CNAV2 ephemeris data for one satellite.		●	S
`BDSAlm`	4119	Almanac data for a BeiDou satellite		●	S
`BDSIon`	4120	BeiDou Ionospheric delay model parameters		●	S
`BDSUtc`	4121	BDT-UTC data		●	S
QZSS Decoded Message Blocks					
`QZSNav`	4095	QZSS ephemeris and clock		●	S
`QZSAlm`	4116	Almanac data for a QZSS satellite		●	S
NavIC/IRNSS Decoded Message Blocks					
`NavICLNav`	4254	NavIC/IRNSS ephemeris and clock		●	S
SBAS L1 Decoded Message Blocks					
`GEOMT00`	5925	MT00 : SBAS Don't use for safety applications			S
`GEOPRNMask`	5926	MT01 : PRN Mask assignments			S
`GEOFastCorr`	5927	MT02-05/24: Fast Corrections			S
`GEOIntegrity`	5928	MT06 : Integrity information			S
`GEOFastCorrDegr`	5929	MT07 : Fast correction degradation factors			S
`GEONav`	5896	MT09 : SBAS navigation message		●	S
`GEODegrFactors`	5930	MT10 : Degradation factors			S
`GEONetworkTime`	5918	MT12 : SBAS Network Time/UTC offset parameters			S
`GEOAlm`	5897	MT17 : SBAS satellite almanac		●	S
`GEOIGPMask`	5931	MT18 : Ionospheric grid point mask			S
`GEOLongTermCorr`	5932	MT24/25 : Long term satellite error corrections			S
`GEOIonoDelay`	5933	MT26 : Ionospheric delay corrections			S
`GEOServiceLevel`	5917	MT27 : SBAS Service Message			S
`GEOClockEphCovMatrix`	5934	MT28 : Clock-Ephemeris Covariance Matrix			S
GNSS Position, Velocity and Time Blocks					
`PVTCartesian`	4006	GNSS position, velocity, and time in Cartesian coordinates	●	●	R
`PVTGeodetic`	4007	GNSS position, velocity, and time in geodetic coordinates	●	●	R
`PosCovCartesian`	5905	Position covariance matrix (X,Y, Z)	●	●	R
`PosCovGeodetic`	5906	Position covariance matrix (Lat, Lon, Alt)	●	●	R
`VelCovCartesian`	5907	Velocity covariance matrix (X, Y, Z)	●	●	R
`VelCovGeodetic`	5908	Velocity covariance matrix (North, East, Up)	●	●	R
`DOP`	4001	Dilution of precision	●	●	R
`PosCart`	4044	Position, variance and baseline in Cartesian coordinates	●	●	R
`PosLocal`	4052	Position in a local datum	●	●	R
`PosProjected`	4094	Plane grid coordinates	●	●	R
`PVTSatCartesian`	4008	Satellite positions	●	●	R
`PVTResiduals`	4009	Measurement residuals	●	●	R
`RAIMStatistics`	4011	Integrity statistics	●	●	R
`GEOCorrections`	5935	Orbit, Clock and pseudoranges SBAS corrections	●	●	R
`BaseVectorCart`	4043	XYZ relative position and velocity with respect to base(s)	●	●	R
`BaseVectorGeod`	4028	ENU relative position and velocity with respect to base(s)	●	●	R
`PVTSupport`	4076	Internal parameters for maintenance and support	●	●	R
`PVTSupportA`	4079	Internal parameters for maintenance and support	●	●	R
`EndOfPVT`	5921	PVT epoch marker	●	●	R
Receiver Time Blocks					
`ReceiverTime`	5914	Current receiver and UTC time	●	●	R
`xPPSOffset`	5911	Offset of the xPPS pulse with respect to GNSS time			R
External Event Blocks					
`ExtEvent`	5924	Time at the instant of an external event			E
`ExtEventPVTCartesian`	4037	Cartesian position at the instant of an event			E
`ExtEventPVTGeodetic`	4038	Geodetic position at the instant of an event			E

Block name	Block No	Content description	Flex Rate	esoc	Time Stamp
`ExtEventBaseVectGeod`	4217	ENU relative position with respect to base(s) at the instant of an event			E
Differential Correction Blocks					
`DiffCorrIn`	5919	Incoming RTCM or CMR message			R
`BaseStation`	5949	Base station coordinates			R
`RTCMDatum`	4049	Datum information from the RTK service provider			R
L-Band Demodulator Blocks					
`LBandTrackerStatus`	4201	Status of the L-band signal tracking	●	●	R
`LBAS1DecoderStatus`	4202	Status of the LBAS1 L-band service			R
`LBAS1Messages`	4203	LBAS1over-the-air message			R
`LBandBeams`	4204	L-band satellite/beam information		●	R
Status Blocks					
`ChannelStatus`	4013	Status of the tracking for all receiver channels	●	●	R
`ReceiverStatus`	4014	Overall status information of the receiver	●	●	R
`SatVisibility`	4012	Azimuth/elevation of visible satellites	●	●	R
`InputLink`	4090	Statistics on input streams	●	●	R
`OutputLink`	4091	Statistics on output streams	●	●	R
`NTRIPClientStatus`	4053	NTRIP client connection status	●	●	R
`NTRIPServerStatus`	4122	NTRIP server connection status	●	●	R
`IPStatus`	4058	IP address, gateway and MAC address of Ethernet interface		●	R
`WiFiAPStatus`	4054	WiFi status in access point mode	●	●	R
`WiFiClientStatus`	4096	WiFi status in client mode	●	●	R
`DynDNSStatus`	4105	DynDNS status	●	●	R
`PowerStatus`	4101	Power supply source and voltage	●	●	R
`QualityInd`	4082	Quality indicators	●	●	R
`DiskStatus`	4059	Internal logging status	●	●	R
`LogStatus`	4102	Log sessions status	●	●	R
`RFStatus`	4092	Radio-frequency interference mitigation status	●	●	R
`P2PPStatus`	4238	P2PP client/server status	●	●	R
`CosmosStatus`	4243	Cosmos receiver service status	●	●	R
`GALAuthStatus`	4245	Galileo OSNMA authentication status		●	R
Miscellaneous Blocks					
`ReceiverSetup`	5902	General information about the receiver installation		●	R
`RxComponents`	4084	Information on various receiver components	●	●	R
`RxMessage`	4103	Receiver message		●	R
`Commands`	4015	Commands entered by the user		●	R
`Comment`	5936	Comment entered by the user		●	R
`BBSamples`	4040	Baseband samples			E
`ASCIIIn`	4075	ASCII input from external sensor			R
Advanced Blocks					
`SystemInfo`	6000	System parameters for maintenance and support	●	●	R

Appendix B

List of BINEX Records

The following table provides a list of BINEX records supported by your receiver. The first column is the record identifier to be used in the **`setBINEXOutput`** and the **`exeBINEXOnce`** commands.

The "OnChange Rate" column indicates the rate at which each record is output when the OnChange interval is selected with the **`setBINEXOutput`** command, and the "Flex Rate" column indicates whether the record can be output at a different rate than its OnChange rate.

Message Identifier	Short Description	OnChange Rate	Flex Rate
Rec00	Site metadata	at the beginning of each log file, and when contents is changed by user with the **`setAntennaOffset`**, **`setMarkerParameters`** or the **`setObserverParameters`** command	
Rec0101	Decoded GPS Ephemeris	when a new GPS dataset is decoded	
Rec0102	Decoded GLONASS Ephemeris	when a new GLONASS dataset is decoded	
Rec0103	Decoded SBAS Ephemeris	when a new SBAS dataset is decoded	
Rec0114	Decoded Galileo Ephemeris	when a new Galileo dataset is decoded	
Rec0105	Decoded BeiDou Ephemeris	when a new BeiDou dataset is decoded	
Rec0106	Decoded QZSS Ephemeris	when a new QZSS dataset is decoded	
Rec0107	Decoded NavIC/IRNSS Ephemeris	when a new NavIC dataset is decoded	
Rec0141	Raw GPS navigation subframe	at the end of each subframe	
Rec0142	Raw GLONASS navigation string	at the end of each string	
Rec0143	Raw SBAS navigation block	at the end of each block	
Rec0144	Raw Galileo navigation page	at the end of each page	
Rec0145	Raw BeiDou navigation subframe	at the end of each subframe	
Rec0146	Raw QZSS navigation subframe	at the end of each subframe	
Rec0147	Raw NavIC/IRNSS navigation subframe	at the end of each subframe	
Rec05Geod	Geodetic position and velocity	0.1s	•
Rec7D00	Receiver internal state	1s	•
Rec7E01	Raw ASCII string from ancillary device	when a new string is received	
Rec7F05	GNSS measurements (empty epochs are encoded as record 0x7F-04)	0.1s	•

Appendix C

List of NMEA Sentences

The following table provides a list of the NMEA messages supported by your receiver. The first column is the message identifier to be used in the **setNMEAOutput** and the **exeNMEAOnce** commands.

For a full description of the NMEA messages, please refer to the NMEA 0183 standard.

Message Identifier	NMEA Formatter	Short Description	Comment
ALM	ALM	GPS Almanac Data	
AVR	AVR	Trimble Navigation proprietary `$PTNL,AVR` sentence	
DTM	DTM	Datum Reference	
GBS	GBS	GNSS Satellite Fault Detection	
GFA	GFA	GNSS Fix Accuracy and Integrity	
GGA	GGA	GPS Fix Data	GPS Quality Indicator field is set to 5 in PPP mode
GGK	GGK	Trimble Navigation proprietary `$PTNL,GGK` sentence	
GGQ	GGQ	Leica Real-Time Position with CQ	
GLL	GLL	Geographic Position - Latitude/Longitude	
GMP	GMP	GNSS Map Projection Fix Data	
GNS	GNS	GNSS Fix Data	
GRS	GRS	GNSS Range Residuals	
GSA	GSA	GNSS DOP and Active Satellites	
GST	GST	GNSS Pseudorange Error Statistics	
GSV	GSV	GNSS Satellites in View	
HDT	HDT	Heading, True	
LLK	LLK	Leica Local Position and GDOP	
LLQ	LLQ	Leica Local Position and Quality	
PUMRD	PUMRD	Septentrio proprietary	Septentrio proprietary, not documented here
RBD	RBD	Rover-Base Direction	Septentrio proprietary, see section C.1.1
RBP	RBP	Rover-Base Position	Septentrio proprietary, see section C.1.2
RBV	RBV	Rover-Base Velocity	Septentrio proprietary, see section C.1.3
RMC	RMC	Recommended Minimum Specific GNSS Data	
ROT	ROT	Rate of Turn	
SDI	SDI	Disk Status	Septentrio proprietary, see section C.1.4
SNC	SNC	NTRIP Client Status	Septentrio proprietary, see section C.1.5
SPW	SPW	Power Status	Septentrio proprietary, see section C.1.6
SRX	SRX	Receiver Status	Septentrio proprietary, see section C.1.7

Message Identifier	NMEA Formatter	Short Description	Comment
TFM	TFM	Used Coordinate Transformation Messages	Septentrio proprietary, see section C.1.8
TXTbase	TXT	Text Transmission	Text from a base station in RTCM message type 1029. The text identifier is set to 1, and the text message is in the form "`nnnn:`*<base txt>*", where `nnnn` is the base station ID.
VTG	VTG	Course Over Ground and Ground Speed	
ZDA	ZDA	Time and Date	

Note: in sentences containing satellite-per-satellite data, data for BeiDou satellites are encoded using System ID 4 (BD) and satellite ID 1-36. Data for NavIC/IRNSS, QZSS and SBAS satellites with a PRN>151 are not encoded in NMEA.

Appendix C.1 Proprietary NMEA Sentences

C.1.1 RBD : Rover-Base Direction

Field	Description
$PSSN,RBD,	Start of sentence
hhmmss.ss,	UTC of RBD (HoursMinutesSeconds.DecimalSeconds)
xxxxxx,	Date: ddmmyy
x.x,	Azimuth of the base as seen from rover (0 to 360 increasing towards east), degrees True
x.x,	Elevation of the base as seen from rover (-90 to 90), degrees
xx,	Number of satellites used for baseline computation
x,	Quality indicator: 0: Invalid 2: DPGS 4: RTK 5: Float RTK
x,	Base motion indicator: 0: Static base 1: Moving base
x.x,	Correction Age, seconds
c–c,	Rover serial number
xxxx	Base station ID
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

C.1.2 RBP : Rover-Base Position

Field	Description
$PSSN,RBP,	Start of sentence
hhmmss.ss,	UTC of RBP (HoursMinutesSeconds.DecimalSeconds)
xxxxxx,	Date: ddmmyy
x.x,	North (True) baseline component (positive when base is north of rover), meters
x.x,	East baseline component (positive when base is east of rover), meters
x.x,	Up baseline component (positive when base is higher than rover), meters
xx,	Number of satellites used for baseline computation
x,	Quality indicator: 0: Invalid 2: DPGS 4: RTK 5: Float RTK
x,	Base motion indicator: 0: Static base 1: Moving base
x.x,	Correction Age, seconds
c–c,	Rover serial number
xxxx	Base station ID
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

C.1.3 RBV : Rover-Base Velocity

Field	Description
$PSSN,RBV,	Start of sentence
hhmmss.ss,	UTC of RBV (HoursMinutesSeconds.DecimalSeconds)
xxxxxx,	Date: ddmmyy
x.x,	Rate of change of baseline vector (rover to base), north component, m/s
x.x,	Rate of change of baseline vector (rover to base), east component, m/s
x.x,	Rate of change of baseline vector (rover to base), up component, m/s
xx,	Number of satellites used for baseline computation
x,	Quality indicator: 0: Invalid 2: DPGS 4: RTK 5: Float RTK
x,	Base motion indicator: 0: Static base 1: Moving base
x.x,	Correction Age, seconds
c–c,	Rover serial number
xxxx	Base station ID
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

C.1.4 SDI : Disk Status

This proprietary sentence is the NMEA equivalent of the `DiskStatus` SBF block.

Field	Description
$PSSN,SDI,	Start of sentence
[	
x,	message revision
xxxxxxxxx,	time of week, milliseconds
xxxx,	week number
,	Reserved
`<SDISub>`	a succession of `SDISub` sub-messages, see definition below
]	
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

`SDISub` definition:

Field	Description
[	
x.x,	`DiskID` field of the `DiskStatus` SBF block
x.x,	`Status` field of the `DiskStatus` SBF block
x.x,	Disk usage in bytes
x.x,	`DiskSize` field of the `DiskStatus` SBF block, in MBytes
x.x,	Percentage of disk used
x.x,	`CreateDeleteCounter` field of the `DiskStatus` SBF block
x.x	`Error` field of the `DiskStatus` SBF block
]	

Example:
`$PSSN,SDI,[1,314558000,1985,,[1,13,14616641536,15472,90,5,0]]*77`

C.1.5 SNC : NTRIP Client Status

This proprietary sentence is the NMEA equivalent of the `NTRIPClientStatus` SBF block.

Field	Description
$PSSN,SNC,	Start of sentence
[	
x,	message revision
xxxxxxxxx,	time of week, milliseconds
xxxx,	week number
`<SNCSub>`	a succession of `SNCSub` sub-messages, see definition below
]	
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

`SNCSub` definition:

Field	Description
[	
x,	`CDIndex` field of the `NTRIPClientStatus` SBF block
x,	`Status` field of the `NTRIPClientStatus` SBF block
x,	`ErrorCode` field of the `NTRIPClientStatus` SBF block
x	`Info` field of the `NTRIPClientStatus` SBF block
]	

Example:
`$PSSN,SNC,[0,379359000,1840,[1,2,0,0]]*68`

C.1.6 SPW : Power Status

This proprietary sentence is the NMEA equivalent of the `PowerStatus` SBF block.

Field	Description
$PSSN,SPW,	Start of sentence
[	
x,	message revision
xxxxxxxxx,	time of week, milliseconds
xxxx,	week number
x.x,	`PowerSource`. See `PowerInfo` field of the `PowerStatus` SBF block
x.x	Voltage at the Vin pins, in Volts
]	
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

Example:

```
$PSSN,SPW,[0,314389000,1985,1,12.25]*56
```

C.1.7 SRX : Receiver Status

This proprietary sentence is the NMEA equivalent of the `ReceiverStatus` SBF block.

Field	Description
$PSSN,SRX,	Start of sentence
[	
x,	message revision
xxxxxxxxx,	time of week, milliseconds
xxxx,	week number
x.x,	`CPULoad` field of the `ReceiverStatus` SBF block
x.x,	`ExtError` field of the `ReceiverStatus` SBF block
x.x,	`UpTime` field of the `ReceiverStatus` SBF block
x.x,	`RxState` field of the `ReceiverStatus` SBF block
x.x,	`RxError` field of the `ReceiverStatus` SBF block
x.x,	`CmdCount` field of the `ReceiverStatus` SBF block
x.x,	`Temperature` field of the `ReceiverStatus` SBF block, in degC
`<SRXSub>`	a succession of `SRXSub` sub-messages, see definition below
]	
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

`SRXSub` definition:

Field	Description
[	
x.x,	`FrontendID` field of the `ReceiverStatus` SBF block
x.x,	`Gain` field of the `ReceiverStatus` SBF block, in dB
x.x,	`SampleVar` field of the `ReceiverStatus` SBF block
x.x	`BlankingStat` field of the `ReceiverStatus` SBF block, in percent
]	

Example:

```
$PSSN,SRX,[1,316618000,1985,41,0,25,754,8,6,43.00,[0,49,102,0],
[1,52,96,0],[11,52,104,0],[10,36,98,0],[5,48,96,0],[6,45,98,0],
[3,38,93,0],[4,39,100,0],[12,36,102,0],[2,36,104,0]]*4B
```

C.1.8 TFM : Used RTCM Coordinate Transformation Messages

This proprietary sentence indicates which RTCM coordinate transformation messages have been received and used in the position computation.

Field	Description
$PSSN,TFM,	Start of sentence
hhmmss.ss,	UTC time (HoursMinutesSeconds.DecimalSeconds)
x,	Height indicator, a copy of the Height Indicator field in RTCM message 1021 or 1022. Null if unknown.

xxxx,	Message 1021/1022 usage (they are exclusive). Possible field values:
	1021: Message type 1021 used;
	1022: Message type 1022 used;
	null: neither 1021 nor 1022 used.

xxxx,	Message 1023/1024 usage (they are exclusive). Possible field values:
	1023: Message type 1023 used;
	1024: Message type 1024 used;
	null: neither 1023 nor 1024 used.

xxxx	Message 1025/1026/1027 usage (they are exclusive). Possible field values:
	1025: Message type 1025 used;
	1026: Message type 1026 used;
	1027: Message type 1027 used;
	null: neither 1025 nor 1026 nor 1027 used.

*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

Example:

```
$PSSN,TFM,104751.00,2,1021,1023,1025*5F
```

Appendix D

List of CMR and RTCM Messages

This appendix provides a list of all the CMR and RTCM (v2.x and v3.x) messages supported by the receiver.

Appendix D.1 CMR Messages

Message Identifier	Short Description
CMR0	Observables
CMR1	Reference Station Coordinates
CMR2	Reference Station Description
CMR3	GLONASS Observables
CMR0p	CMR+ variant
CMR0w	CMR-W variant

Appendix D.2 RTCM v2.x Messages

Message Identifier	Short Description
RTCM1	Differential GPS Corrections
RTCM3	GPS Reference Station Parameters
RTCM9	GPS Partial Correction Set
RTCM15	Ionospheric Delay
RTCM16	GPS Special Message
RTCM17	GPS Ephemerides Message
RTCM18	RTK Uncorrected Carrier Phases
RTCM19	RTK Uncorrected Pseudoranges
RTCM20	RTK Carrier Phase Corrections
RTCM21	RTK/Hi-Accuracy Pseudorange Corrections
RTCM22	Extended Reference Station Parameters
RTCM23	Antenne Type Definition Record
RTCM24	Antenna Reference Point (ARP)
RTCM31	Differential GLONASS Corrections
RTCM32	GLONASS Reference Station Parameters
RTCM34	GLONASS Partial Correction Set
RTCM59	Proprietary Message

Appendix D.3 RTCM v3.x Messages

Message Identifier	Short Description
RTCM1001	L1-Only GPS RTK Observables
RTCM1002	Extended L1-Only GPS RTK Observables
RTCM1003	L1&L2 GPS RTK Observables
RTCM1004	Extended L1&L2 GPS RTK Observables
RTCM1005	Stationary RTK Reference Station ARP
RTCM1006	Stationary RTK Reference Station ARP with Antenna Height
RTCM1007	Antenna Descriptor
RTCM1008	Antenna Descriptor and Serial Number
RTCM1009	L1-Only GLONASS RTK Observables
RTCM1010	Extended L1-Only GLONASS RTK Observables
RTCM1011	L1&L2 GLONASS RTK Observables
RTCM1012	Extended L1&L2 GLONASS RTK Observables
RTCM1013	System Parameters
RTCM1015	Network RTK (MAC), GPS Ionospheric Correction Differences
RTCM1016	Network RTK (MAC), GPS Geometric Correction Differences
RTCM1017	Network RTK (MAC), GPS Combined Geometric and Ionospheric Correction Differences
RTCM1019	GPS Satellite Ephemeris Data
RTCM1020	Glonass Satellite Ephemeris Data
RTCM1021	Helmert-Abridged Molodenski Transformation Parameters
RTCM1022	Molodenski-Badekas Transformation Parameters
RTCM1023	Residuals, Ellipsoidal Grid Representation
RTCM1024	Residuals, Plane Grid Representation
RTCM1025	Projection Parameters, Projection Types other than Lambert Conic Conformal (2 SP) and Oblique Mercator
RTCM1026	Projection Parameters, Projection Type LCC2SP (Lambert Conic Conformal (2 SP))
RTCM1027	Projection Parameters, Projection Type OM (Oblique Mercator)
RTCM1029	Unicode Text String
RTCM1033	Receiver and Antenna Descriptors
RTCM1037	Network RTK (MAC), GLONASS Ionospheric Correction Differences
RTCM1038	Network RTK (MAC), GLONASS Geometric Correction Differences
RTCM1039	Network RTK (MAC), GLONASS Combined Geometric and Ionospheric Correction Differences
RTCM1041	NavIC/IRNSS Satellite Ephemeris Data
RTCM1042	BDS Satellite Ephemeris Data
RTCM1044	QZSS Satellite Ephemeris Data
RTCM1045	Galileo F/NAV Satellite Ephemeris Data
RTCM1046	Galileo I/NAV Satellite Ephemeris Data
RTCM1071	GPS MSM1, Compact Pseudoranges
RTCM1072	GPS MSM2, Compact PhaseRanges
RTCM1073	GPS MSM3, Compact Pseudoranges and PhaseRanges
RTCM1074	GPS MSM4, Full Pseudoranges and PhaseRanges plus CNR
RTCM1075	GPS MSM5, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR
RTCM1076	GPS MSM6, Full Pseudoranges and PhaseRanges plus CNR (high resolution)
RTCM1077	GPS MSM7, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR (high resolution)
RTCM1081	GLONASS MSM1, Compact Pseudoranges
RTCM1082	GLONASS MSM2, Compact PhaseRanges
RTCM1083	GLONASS MSM3, Compact Pseudoranges and PhaseRanges
RTCM1084	GLONASS MSM4, Full Pseudoranges and PhaseRanges plus CNR
RTCM1085	GLONASS MSM5, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR
RTCM1086	GLONASS MSM6, Full Pseudoranges and PhaseRanges plus CNR (high resolution)
RTCM1087	GLONASS MSM7, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR (high resolution)
RTCM1091	Galileo MSM1, Compact Pseudoranges
RTCM1092	Galileo MSM2, Compact PhaseRanges
RTCM1093	Galileo MSM3, Compact Pseudoranges and PhaseRanges
RTCM1094	Galileo MSM4, Full Pseudoranges and PhaseRanges plus CNR
RTCM1095	Galileo MSM5, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR
RTCM1096	Galileo MSM6, Full Pseudoranges and PhaseRanges plus CNR (high resolution)
RTCM1097	Galileo MSM7, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR (high resolution)
RTCM1101	SBAS MSM1, Compact Pseudoranges

Message Identifier	Short Description
RTCM1102	SBAS MSM2, Compact PhaseRanges
RTCM1103	SBAS MSM3, Compact Pseudoranges and PhaseRanges
RTCM1104	SBAS MSM4, Full Pseudoranges and PhaseRanges plus CNR
RTCM1105	SBAS MSM5, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR
RTCM1106	SBAS MSM6, Full Pseudoranges and PhaseRanges plus CNR (high resolution)
RTCM1107	SBAS MSM7, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR (high resolution)
RTCM1111	QZSS MSM1, Compact Pseudoranges
RTCM1112	QZSS MSM2, Compact PhaseRanges
RTCM1113	QZSS MSM3, Compact Pseudoranges and PhaseRanges
RTCM1114	QZSS MSM4, Full Pseudoranges and PhaseRanges plus CNR
RTCM1115	QZSS MSM5, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR
RTCM1116	QZSS MSM6, Full Pseudoranges and PhaseRanges plus CNR (high resolution)
RTCM1117	QZSS MSM7, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR (high resolution)
RTCM1121	BeiDou MSM1, Compact Pseudoranges
RTCM1122	BeiDou MSM2, Compact PhaseRanges
RTCM1123	BeiDou MSM3, Compact Pseudoranges and PhaseRanges
RTCM1124	BeiDou MSM4, Full Pseudoranges and PhaseRanges plus CNR
RTCM1125	BeiDou MSM5, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR
RTCM1126	BeiDou MSM6, Full Pseudoranges and PhaseRanges plus CNR (high resolution)
RTCM1127	BeiDou MSM7, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR (high resolution)
RTCM1131	NavIC/IRNSS MSM1, Compact Pseudoranges
RTCM1132	NavIC/IRNSS MSM2, Compact PhaseRanges
RTCM1133	NavIC/IRNSS MSM3, Compact Pseudoranges and PhaseRanges
RTCM1134	NavIC/IRNSS MSM4, Full Pseudoranges and PhaseRanges plus CNR
RTCM1135	NavIC/IRNSS MSM5, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR
RTCM1136	NavIC/IRNSS MSM6, Full Pseudoranges and PhaseRanges plus CNR (high resolution)
RTCM1137	NavIC/IRNSS MSM7, Full Pseudoranges, PhaseRanges, PhaseRangeRate and CNR (high resolution)
RTCM1230	GLONASS L1&L2 Code-Phase Biases

Index of Commands

Index of SBF Blocks