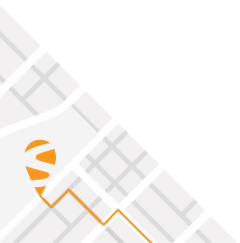
mosaic-G5 Reference Guide

Applicable to version 1.1.1 of the Firmware



mosaic-G5 Reference Guide

2026-06-12

Applicable to version 1.1.1 of the Firmware

© Copyright 2000-2026 Septentrio NV/SA, part of HEXAGON. All rights reserved.

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

http://www.septentrio.com
Phone: +32 16 300 800
@Septentrio
https://www.linkedin.com/company/septentrio
https://github.com/septentrio-gnss/
https://www.youtube.com/@SeptentrioGNSS

List of Contents

Contents 6

Scope 7

List of Acronyms 8

1 How To... 11

1.1 Check the Capabilities of your Receiver 12

1.2 Connect to the Receiver 13
1.2.1 Via COM Ports 13
1.2.2 Via USB 13
1.2.3 Connection Descriptors 13

1.3 Set up a Secure Connection over COM or USB Ports 14
1.3.1 COBS Algorithm 14
1.3.2 Receiver Certificate 15
1.3.3 Root CA 15

1.4 Manage Users 16

1.5 Understand the Output of the Receiver 17
1.5.1 Proprietary Binary Output (SBF) 17
1.5.2 NMEA 17

1.6 Define an SBF Output Stream 19

1.7 Save the Receiver Configuration in Non-Volatile Memory 20

1.8 Configure the Receiver in DGNSS/RTK-Base Mode 21
1.8.1 Static Base Station Mode 21
1.8.2 RTK Moving Base Station Mode 22

1.9 Configure the Receiver in DGNSS/RTK-Rover Mode 24

1.10 Configure the Receiver in Galileo HAS Mode 25

1.11 Determine a GNSS-Based Attitude from the Main and Aux Antennas 26

1.12 Determine a GNSS-Based Attitude from a Moving Base 28

1.13 Log SBF or NMEA 30

1.14 Communicate with External Equipment 31

1.15 Generate a "Pulse Per Second" Signal 32

1.16 Time Tag External Events 33

1.17 Monitor the RF Spectrum 34

1.18 Detect Spoofing Attacks 35
1.19 Use Galileo OSNMA 36
1.20 Upgrade the Receiver 37
1.21 Check the Receiver Permissions 38

2 Operation Details **39**

2.1 Channel Allocation and Signal Selection 39
2.2 Generation of Measurements 39
2.2.1 Pilot vs. Data Component 40
2.3 Time Management 41
2.3.1 Free-Running Clock 41
2.3.2 Clock Steering 42
2.4 Computation of Position, Velocity, and Time (PVT Solution) 43
2.4.1 SBAS Positioning 44
2.4.2 DGNSS Positioning 45
2.4.3 RTK Positioning 45
2.4.3.1 Integer Ambiguities (RTK-fixed) 45
2.4.3.2 Floating Ambiguities (RTK-float) 45
2.4.3.3 Moving Base 46
2.4.4 Precise Point Positioning 46
2.4.4.1 PPP Datum Offset 46
2.4.4.2 Tide Corrections 46
2.4.5 Transition between PVT Modes 47
2.4.6 PVT Latency 47
2.4.7 Datum Transformation 47
2.4.7.1 Transformation to Regional Datum 48
2.5 Antenna Effects 48
2.5.1 Antenna Effects in Rover Mode 49
2.5.2 Antenna Effects in Base Mode 49
2.6 Galileo OSNMA 49
2.6.1 Use of OSNMA in Simulated Scenarios 50

3 Command Line Reference **52**

3.1 Command Line Interface Outline 53
3.1.1 Command Types 53
3.1.2 Command Line Syntax 53
3.1.3 Command Replies 54
3.1.4 Command Syntax Tables 55
3.2 Command Definitions 58
3.2.1 Receiver Administration 58
3.2.2 User Management 76
3.2.3 Tracking and Measurement Generation 83
3.2.4 Frontend and Interference Mitigation 96
3.2.5 Navigation Filter 101
3.2.6 Authentication 128
3.2.7 Attitude Determination 133
3.2.8 Datum Definition 135

3.2.9 Timing and Time Management 140
3.2.10 Station Settings 148
3.2.11 General Input/Output 151
3.2.12 NMEA Configuration 160
3.2.13 SBF Configuration 166
3.2.14 RTCM v3.x Settings 172
3.2.15 CMR Settings 180
3.2.16 Internal Disk Logging 181
3.2.17 MSS/L-Band Configuration 185

4 SBF Reference 188

4.1 SBF Outline 189
4.1.1 SBF Block Header Format 189
4.1.2 SBF Block Names and Numbers 189
4.1.3 SBF Block Time Stamp (TOW and WNc) 190
4.1.4 Sub-blocks 190
4.1.5 Padding Bytes 191
4.1.6 SBF Revision Number 191
4.1.7 Do-Not-Use Value 191
4.1.8 Output Rate 191
4.1.9 SBF Satellite Numbering 192
4.1.10 Signal Type 192
4.1.11 Channel Numbering 193
4.1.12 Decoding of SBF Blocks 193

4.2 SBF Block Definitions 195
4.2.1 Measurement Blocks 195
4.2.2 Navigation Page Blocks 203
4.2.3 GPS Decoded Message Blocks 226
4.2.4 GLONASS Decoded Message Blocks 233
4.2.5 Galileo Decoded Message Blocks 236
4.2.6 BeiDou Decoded Message Blocks 244
4.2.7 QZSS Decoded Message Blocks 255
4.2.8 NavIC/IRNSS Decoded Message Blocks 258
4.2.9 SBAS L1 Decoded Message Blocks 260
4.2.10 GNSS Position, Velocity and Time Blocks 264
4.2.11 GNSS Attitude Blocks 294
4.2.12 Receiver Time Blocks 298
4.2.13 External Event Blocks 300
4.2.14 Correction Blocks 314
4.2.15 L-Band Demodulator Blocks 317
4.2.16 Status Blocks 319
4.2.17 Miscellaneous Blocks 338

4.3 SBF Change Log 345

A Attitude Angles 346

B List of SBF Blocks 348

C List of NMEA Sentences 351

C.1 Proprietary NMEA Sentences 352
C.1.1 HRP : Heading, Roll, Pitch 352

C.1.2 RBD : Rover-Base Direction 353
C.1.3 RBP : Rover-Base Position 353
C.1.4 RBV : Rover-Base Velocity 354
C.1.5 TFM : Used RTCM Coordinate Transformation Messages 354

D List of Differential Correction Messages 355

D.1 CMR Messages 355

D.2 RTCM v3.x Messages 355

Index of Commands 357

Index of SBF Blocks 364

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
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
DGNSS	Differential GNSS
DOP	Dilution Of Precision
DTLS	Datagram Transport Layer Security
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
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
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
PDOP	Position DOP
PLL	Phase Locked Loop
PPP	Precise Point Positioning
PPS	Pulse Per Second
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
TLS	Transport Layer Security
TOW	Time Of Week
UERE	User Equivalent Range Error
UHF	Ultra High Frequency
URA	User Range Accuracy
USB	Universal Serial Bus
UTC	Coordinated Universal Time
VDOP	Vertical DOP
VPL	Vertical Protection Level
VRS	Virtual Reference Station
WAAS	Wide Area Augmentation System
WGS84	World Geodetic System 1984
WN	Week Number
WNc	Week number
XOR	Exclusive OR

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.
- 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 1.2). 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*).

Depending on the capabilities of your particular receiver (see section 1.1), some of the user commands, SBF blocks or communication interfaces described in this document may not be supported.

1.1 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 1.21.

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

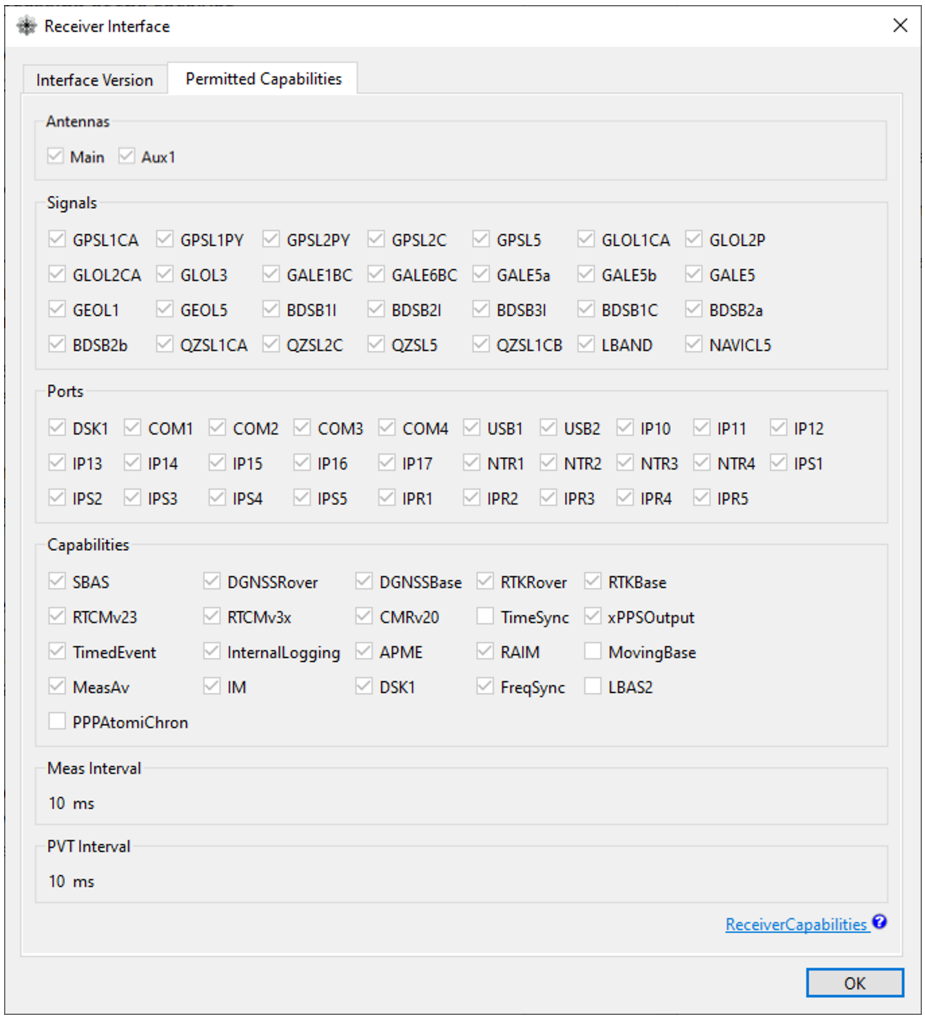


Figure 1-1: Example of receiver capabilities.

1.2 Connect to the Receiver

1.2.1 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. 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.

1.2.2 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.

1.2.3 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	•	•	•
DSK1	the internal disk. See section 1.13		•	

1.3 Set up a Secure Connection over COM or USB Ports

⚠ This section is only applicable to receivers having the AuthIO capability (see section 1.1).

By default, communication over COM and USB ports is plaintext. To enable authentication and encryption, the **`setSerialTlsSettings`** command can be used. When enabled, communication is secured using DTLS (Datagram TLS), with datagrams framed using Consistent Overhead Byte Stuffing (COBS).

To send data (e.g. a user command), the user application generates a datagram with a DTLS library, frames it using the COBS framing algorithm as described below, and sends the resulting frame to the GNSS receiver.

To receive data (e.g. a command reply, an SBF block, a NMEA sentence, ...), the user application reads framed bytes from the GNSS receiver, unframes them and passes the datagram to the DTLS library.

1.3.1 COBS Algorithm

COBS is used to send and receive DTLS datagrams over a medium without packet framing such as a COM or USB port. In Septentrio's COBS implementation, the delimiter byte is 'S' (0x53), every COBS-encoded frame begins and ends with 'S' and all occurrences of 'S' in the datagram are removed and replaced by a *code byte*. The *code byte* specifies the number of bytes until the next occurrence of 'S' in the original datagram.

More specifically, Septentrio's COBS framing/unframing algorithm is described below, where "datagram" is the unencoded DTLS datagram and "frame" is the encoded frame.

COBS Framing Procedure

1. Start the frame with the delimiter byte 'S' (0x53)
2. Determine N as follows: count how many bytes appear before the next 'S' in the datagram. If the count is $\leq$ 253, set N to this value, otherwise set N to 254. If there is no 'S' until the end of the datagram, set N to 254 too. Thus, N is always in the range 0..254.
3. Insert a *code byte* into the frame, with *code byte* = (0x53 + N + 1) mod 256.
4. Copy the next N datagram bytes into the frame (or fewer if the end of the datagram is reached).
5. If N $\leq$ 253, drop the next datagram byte (it must be 'S').
6. If there are still datagram bytes to encode, go back to step 2.
7. Once all datagram bytes are encoded, append a trailing 'S' to close the frame.

COBS Unframing Procedure

1. Wait for the starting delimiter 'S'.
2. Read the first *code byte*, convert it back to N = (*code byte* + 255 - 0x53) mod 256.
3. Read N bytes into the datagram buffer, stopping earlier if 'S' is found (which marks the end of the frame).
4. If N $\leq$ 253, insert an 'S' in the datagram. If N = 254, skip this step.
5. Read the next encoded *code byte* and repeat from step 2 until the closing 'S' is reached.

1.3.2 Receiver Certificate

Receiver certificates can be inspected with the **`lstTLSCertificates`** command. They follow the IEEE 802.1AR standard and can be regarded as an 802.1AR IDevID. The subject has the following structure:

```
O = Septentrio, OU = Devices, serialNumber = SN24310100012345
```

1.3.3 Root CA

The authenticity of the receiver can be verified by validating the signatures of the certificate chain provided by the receiver. The Root CA should match the following certificate:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 5309884601674937909 (0x49b07fe98926b635)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = BE, ST = Vlaams-Brabant, O = Septentrio, OU = Devices, CN = Septentrio Devices Root CA G1
        Validity
            Not Before: Jan  1 00:00:00 2024 GMT
            Not After : Jan  1 00:00:00 2054 GMT
        Subject: C = BE, ST = Vlaams-Brabant, O = Septentrio, OU = Devices, CN = Septentrio Devices Root CA G1
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:55:3f:28:77:7c:d5:43:d2:11:89:f8:47:01:8a:
                    e6:f5:3c:f5:21:d8:61:9c:84:a7:26:dc:65:42:9a:
                    88:83:4c:b5:4f:6c:e7:6a:ab:f4:69:d0:e4:b6:84:
                    4d:22:4b:7e:8b:6d:79:e5:a5:a2:db:1f:94:f9:23:
                    97:45:6e:84:0b:35:a0:94:9b:9f:7f:91:2c:6a:3b:
                    6e:ba:ed:b5:f3:11:50:66:7a:00:9d:22:d4:60:6e:
                    d2:d2:19:0a:82:44:aa
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                DA:49:03:0D:E3:B5:98:6F:BB:C8:2C:62:B6:6B:74:EB:D6:53:99:E3
            X509v3 Authority Key Identifier:
                DA:49:03:0D:E3:B5:98:6F:BB:C8:2C:62:B6:6B:74:EB:D6:53:99:E3
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:2
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:66:02:31:00:8d:9b:fe:a3:1c:a6:ef:11:c6:db:d5:6b:2f:
        78:71:68:32:44:c6:75:e6:47:7a:fe:e2:cd:cf:d7:f4:75:f1:
        cb:5a:80:34:37:24:03:74:65:2c:96:71:d5:85:1b:38:66:02:
        31:00:b5:75:87:25:ba:16:00:78:35:d2:9d:1b:bd:51:be:bc:
        ac:5d:cf:87:83:01:61:7e:15:bc:f9:a9:c9:b2:af:ca:26:32:
        6e:56:cb:bf:38:72:a1:d3:6f:e5:f2:69:2b:d5
-----BEGIN CERTIFICATE-----
MIIChjCCAgugAwIBAgIISbB/6YkmtjUwCgYIKoZIzj0EAwIwdTELMAkGA1UEBhMC
QkUxFzAVBgNVBAgMDlZsYWFtcy1CcmFiYW50MRMwEQYDVQQKDApTZXB0ZW50cmlv
MRAwDgYDVQQLDAdEZXZpY2VzMSYwJAYDVQQDDB1TZXB0ZW50cmlvIERldmljZXMg
Um9vdCBDQSBHMTAgFw0yNDAxMDEwMDAwMDBaGA8yMDU0MDEwMTAwMDAwMFowdTEL
MAkGA1UEBhMCQkUxFzAVBgNVBAgMDlZsYWFtcy1CcmFiYW50MRMwEQYDVQQKDApT
ZXB0ZW50cmlvMRAwDgYDVQQLDAdEZXZpY2VzMSYwJAYDVQQDDB1TZXB0ZW50cmlv
IERldmljZXMgUm9vdCBDQSBHMTB2MBAGByqGSM49AgEGBSuBBAAiA2IABFU/KHd8
1UPSEYn4RwGK5vU89SHYYZyEpybcZUKaiINMtU9s52qr9GnQ5LaETSJLfotteeWl
otsflPkjl0VuhAs1oJSbn3+RLGo7brrttfMRUGZ6AJ0i1GBu0tIZCoJEqqNmMGQw
HQYDVR0OBBYEFNpJAw3jtZhvu8gsYrZrdOvWU5njMB8GA1UdIwQYMBaAFNpJAw3j
tZhvu8gsYrZrdOvWU5njMBIGA1UdEwEB/wQIMAYBAf8CAQIwDgYDVR0PAQH/BAQD
AgEGMAoGCCqGSM49BAMCA2kAMGYCMQCNm/6jHKbvEcbb1WsveHFoMkTGdeZHev7i
zc/X9HXxy1qANDckA3RlLJZx1YUbOGYCMQC1dYcluhYAeDXSnRu9Ub68rF3Ph4MB
YX4VvPmpybKvyiYyblbLvzhyodNv5fJpK9U=
-----END CERTIFICATE-----
```

1.4 Manage Users

The receiver maintains a list of known users (username and password) and their access level (`Viewer` for read-only access or `User` for full-control access).

Use the **`setUserAccessLevel`** command to manage existing user accounts and create new ones. For example...

1. To check the list of known users:
   `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 obfuscated 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. To log in as `john`, type:
   `login, john, abc123 <CR>`

4. You can define up to eight users. 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>`

`setUserAccessLevel` is a permanent command (see 1.7): it survives resets and power cycles without calling **`exeCopyConfigFile`** and is not removed by a reset to default configuration (see the **`exeResetReceiver`** command).

1.5 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)	`$@`

1.5.1 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 B

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

1.5.2 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 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] 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

2 The receiver can operate as DGNSS and/or RTK base station and output the corresponding RTCM2 messages. The instructions to set the receiver in base station mode can be found in section 1.8.

The list of supported RTCM2 messages can be found in appendix D.

1.6 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 B). 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.

1.7 Save the Receiver Configuration in Non-Volatile Memory

The receiver configuration is the set of user commands required to bring the receiver to a given configuration. The current configuration can be displayed with the following command:
lstConfigFile, Current <CR>

By default, the current configuration is lost after a reset or a power-cycle unless it is explicitly saved to the boot configuration file using:
exeCopyConfigFile,Current,Boot <CR>

The receiver can be restored to its default configuration by copying the `RxDefault` configuration into the `Current` configuration:
exeCopyConfigFile,RxDefault,Current <CR>

Certain user commands related to critical settings have a special status: they survive resets and power cycles without calling **exeCopyConfigFile** and are not overwritten by the `RxDefault` configuration. Examples include the user accounts defined with the **setUserAccessLevel** command. These so-called *permanent commands* are not listed in the reply of **lstConfigFile,Current**, but can be viewed using:
lstConfigFile, Permanent <CR>

At startup, the receiver first applies the permanent configuration followed by the boot configuration.

1.8 Configure the Receiver in DGNSS/RTK-Base Mode

This section is only applicable to receivers having the DGNSSBase and/or the RTKBase capability (see section 1.1).

The receiver can generate and output DGNSS and/or RTK corrections in the RTCM2 format. The list of supported messages can be found in appendix D.

1.8.1 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 **set*xx*Interval** command to specify the message interval (replace *xx* by the format name). 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 **set*xx*Formatting** command 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 RTCM2 messages necessary for DGNSS and RTK operation. If necessary, the set of output messages can be specified with the **set*xx*Output** command. 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 1.2.3. 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.

1.8.2 RTK Moving Base Station Mode

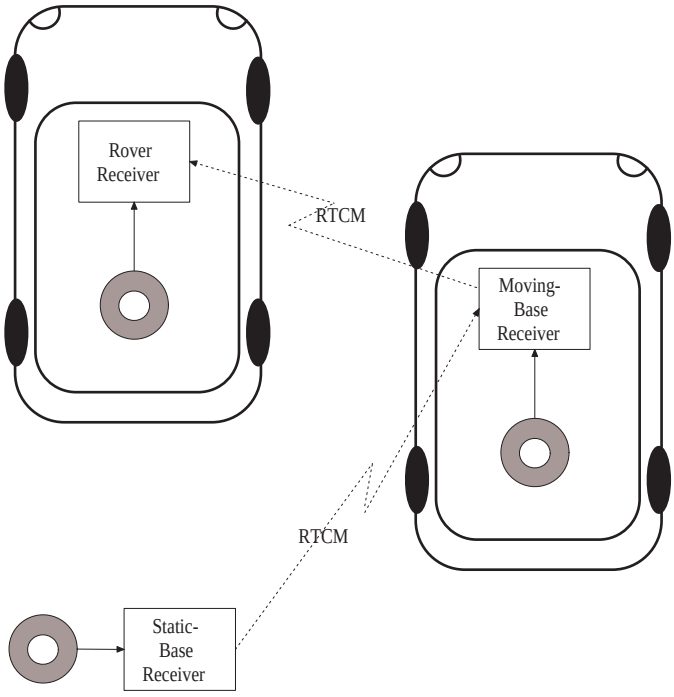


Figure 1-2: Example of a RTK moving-base configuration where the moving base receives RTCM corrections from a static base and transmits RTCM corrections to the rover.

To configure the receiver in RTK moving base, follow the steps below:

1. The PVT engine must be set in one of the rover modes (standalone, DGNSS, SBAS, RTK). The type of the PVT mode at the moving base station will determine the absolute

position accuracy of the RTK rover receiver. On the other hand, the accuracy of the relative position of the rover with respect to the moving base is not influenced by the PVT mode at the moving base station.
For instance, to let the moving base station compute a simple standalone PVT, use the following:
`setPVTMode,Rover,StandAlone <CR>`

If accurate absolute and relative positioning of the rover is required, the moving base can operate in RTK-rover positioning mode and receive RTCM or CMR corrections from a static base station, as illustrated in Figure 1-2. Refer to section 1.9 to configure the moving-base receiver in RTK-rover mode.

2. From now on, follow the same procedure as for a static base station, starting at step 2 of section 1.8.1 and taking into account the following recommendations:
   - RTCM v2.x and CMR are not recommended for moving-base applications, always use RTCM v3.x in the link between the moving base and the rover.
   - To decrease the effect of extrapolation errors, use a short RTCM message interval (see **`setRTCMv3Interval`**). In most cases, it is safe to set the interval to its minimum value of 0.1 seconds. If the RTCM messages are sent through a COM connection, make sure that the baud rate is sufficient to support the high rate. A rate of at least 230400 baud is recommended.
   - In moving base, it is recommended to send the base position and observables at the same rate.

See also section 2.4.3.3 for more details on moving-base operation.

1.9 Configure the Receiver in DGNSS/RTK-Rover Mode

This section is only applicable to receivers having the DGNSSRover and/or the RTKRover capability (see section 1.1).

The receiver computes a DGNSS 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 DGNSS/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 1.2.3 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. By default, the receiver assumes that the base station is static. If it is moving, enter the following command:
   `setDiffCorrUsage,,,,,on <CR>`
   In DGNSS-rover mode, the base station must be static. Moving base stations are only supported in RTK-rover mode.

3. The receiver automatically detects the format of the differential corrections and switches between standalone, DGNSS 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 DGNSS and RTK positioning mode.

1.10 Configure the Receiver in Galileo HAS Mode

This section is only applicable to receivers having the PPPGalileoHAS capability (see section 1.1).

Galileo HAS (High Accuracy Service) is a Precise-Point-Positioning (PPP) service that provides free, real-time, worldwide decimeter-level positioning using precise satellite orbit, clock and bias corrections transmitted by Galileo satellites on their E6 signal.

To enable Galileo HAS in your receiver:

1. Make sure that the receiver is connected to an E6-capable antenna, i.e., an antenna covering the 1278.75MHz frequency.

2. Make sure that tracking of GALE6BC is enabled and that usage of its navigation data (containing the PPP corrections) is enabled too. This is the default, but in case the receiver is not in its default configuration, this can be done with:
   `setSignalTracking, +GALE6BC <CR>`
   `setSignalUsage, ,+GALE6BC <CR>`

3. Enable the PPP rover positioning mode: **`setPVTMode, , +PPP <CR>`**

The receiver automatically decodes and applies the PPP corrections received from the Galileo E6 signal. Note that it can take several minutes for the position accuracy to reach the decimeter level.

In Galileo HAS mode, the positioning mode reported in the `Mode` field of the `PVTCartesian` and `PVTGeodetic` SBF blocks is set to "PPP" and the `TimeSystem` field is set to "Galileo time".

See also section 2.4.4 for more information on PPP positioning.

1.11 Determine a GNSS-Based Attitude from the Main and Aux Antennas

This section is only applicable to receivers having the Aux1 antenna capability (see section 1.1).

The attitude (heading/pitch) can be computed from the orientation of the baseline between the main and the aux1 GNSS antennas. This is illustrated in Figure 1-3.

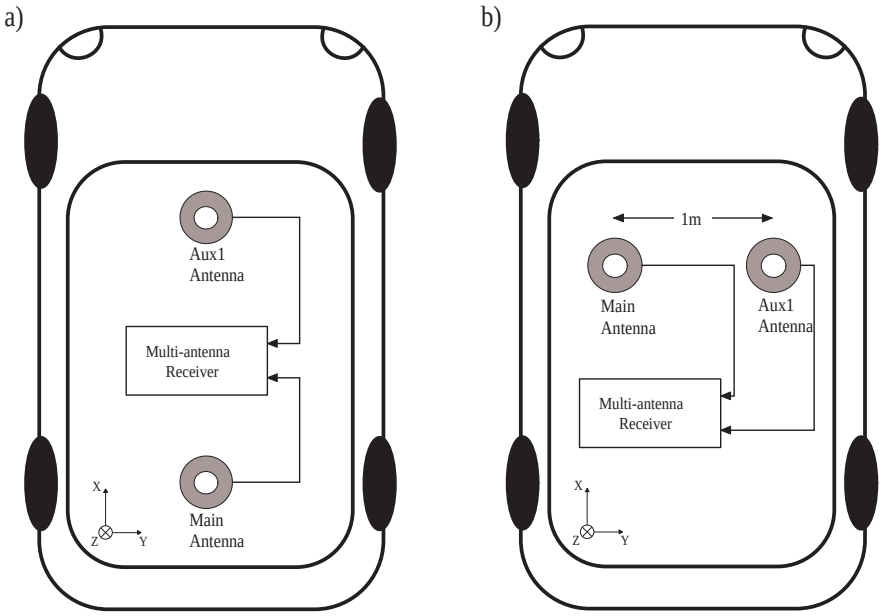


Figure 1-3: Multi-antenna attitude determination setup. a) default configuration. b) alternate configuration.

To enable multi-antenna attitude determination, follow the following procedure:

1. Attach two antennas to your vehicle, using cables of approximately the same length. The default antenna configuration is as depicted in Figure 1-3a. It consists in placing the antennas aligned with the longitudinal axis of the vehicle, main antenna behind aux1. For best accuracy, try to maximize the distance between the antennas, and avoid significant height difference between the antenna ARPs. If the configuration in Figure 1-3a is not feasible, another possibility is to have the antenna baseline perpendicular to the vehicle, as shown in Figure 1-3b. In that configuration, an "attitude offset" needs to be provided with the following command:
   `setAttitudeOffset,90 <CR>` if the aux1 antenna is on the right of the main antenna (Figure 1-3b), or
   `setAttitudeOffset,-90 <CR>` if the aux1 antenna is on the left of the main antenna.

2. In practice, the two antenna ARPs may not be exactly at the same height in the vehicle frame, or the main-aux1 baseline may not be exactly parallel or perpendicular to the longitudinal axis of the vehicle. This leads to offsets in the computed attitude angles. These offsets can be compensated for with the **`setAttitudeOffset`** command. For example, if the heading and pitch angles reported by the receiver have an offset of respectively 3 and 4.5 degrees with respect to their expected value, you can have

the receiver compensate for them by using the following command (assuming the antennas are in the configuration of Figure 1-3a):
setAttitudeOffset,3,4.5 <CR>

3. Specify that the attitude has to be computed in multi-antenna mode:
setGNSSAttitude,MultiAntenna <CR>

The GNSS-based attitude angles (heading and pitch) are available in the `AttEuler` SBF block or in the HDT and HRP NMEA sentences. Note that, in the case where the antenna baseline is perpendicular to the vehicle longitudinal axis, the “pitch” angle is to be interpreted as a “roll” angle.

1.12 Determine a GNSS-Based Attitude from a Moving Base

⚠ This section is only applicable to receivers having the MovingBase capability (see section 1.1).

The heading and pitch of a vehicle can be derived from the orientation of the baseline between a base and a rover antenna when both antennas are attached to the vehicle. The base antenna is connected to a first receiver configured as RTK moving base station. The rover antenna is connected to a second receiver configured as RTK rover and accepting the RTCM stream from the first receiver. This is illustrated in Figure 1-4.

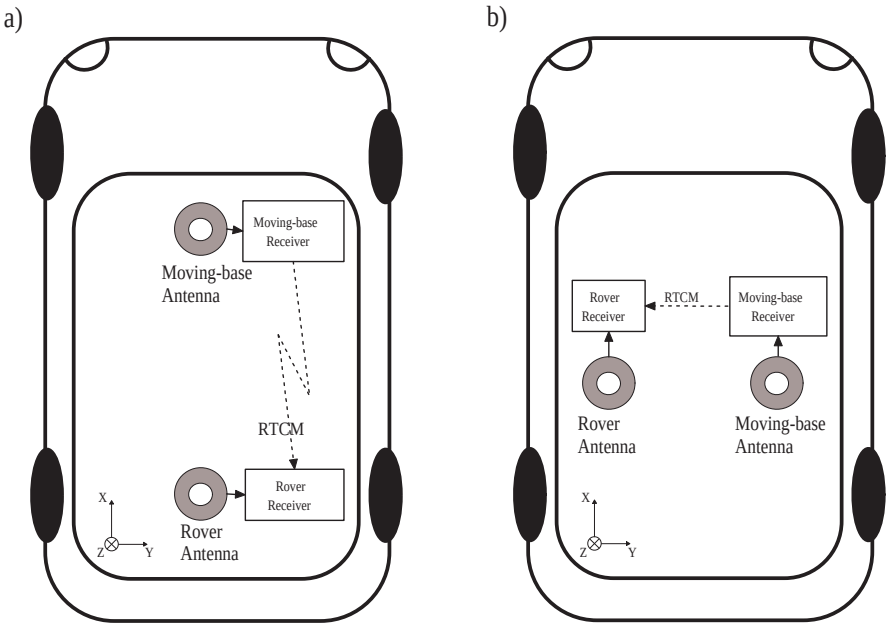


Figure 1-4: Moving-base attitude determination setup. a) default configuration. b) example of non-default configuration.

To enable moving-base attitude determination, follow the following procedure:

1. Attach two antennas to your vehicle. The default antenna configuration is as depicted in Figure 1-4 a). It consists in placing the antennas aligned with the longitudinal axis of the vehicle. If such configuration is not possible, you will have to provide additional information to the receiver, as explained below. For best accuracy, try to maximize the distance between the antennas, and avoid significant height difference between them.

2. Connect one of the antennas (preferably the one at the front of the vehicle) to the receiver that will serve as moving base. Connect the other to the receiver that will serve as rover. That latter receiver is the one where the heading and pitch will be computed.

3. Configure the moving-base receiver to send RTCM 3.x corrections to the rover. The procedure to do so is explained in section 1.8.2.

4. Configure the rover receiver to accept the RTCM corrections from the moving base, by following the steps in section 1.9.

5. By default, the attitude angles are computed assuming that the two antennas are aligned with the longitudinal axis of the vehicle, and that the moving-base antenna is in front of the rover antenna (see Figure 1-4 a)). If you cannot place the antennas in such configuration, the reported attitude angles will be biased. This bias can be compensated for with the **`setAttitudeOffset`** command. For example, if the heading and pitch angles reported by the receiver have an offset of respectively 10 and 12 degrees with respect to their expected value, you can have the receiver compensate for that offset with:
   `setAttitudeOffset,10,12 <CR>`

6. Specify that the attitude has to be computed in moving-base mode by issuing the following command in the rover receiver:
   `setGNSSAttitude, MovingBase <CR>`

The attitude angles are available from the rover receiver in the `AttEuler` SBF block or in the HDT and HRP NMEA sentences.

1.13 Log SBF or NMEA

This section is only applicable to receivers having the InternalLogging capability (see section 1.1).

The connection descriptor (see section 1.2.3) associated to the internal disk is "DSK1". Enabling SBF or NMEA logging on the internal disk involves the following steps:

1. By default, the receiver logs SBF blocks into a file named "log.sbf" and NMEA sentences into a file named "log.nma". Use the **setFileNaming** command to specify another file name or to enable incremental file naming where a new file is created each time logging is stopped and restarted. For instance:
   setFileNaming,DSK1,Incremental,mylog <CR>

2. Use the command **setSBFOutput** to define which SBF blocks need to be logged (for NMEA, use **setNMEAOutput** instead), and at which interval (see also section 1.6). For instance, to log all SBF blocks necessary to build RINEX files, with the measurements and positions being output at a 10-s interval, use:
   setSBFOutput,Stream1,DSK1,rinex,sec10 <CR>

3. Start the logging by enabling SBF and NMEA output to the DSK1 connection (it is enabled by default):
   setDataInOut,DSK1,,+SBF+NMEA <CR>

4. Once the logging session is finished, stop the logging by invoking:
   setDataInOut,DSK1,,-SBF-NMEA <CR>

1.14 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.

You can convert an SBF file containing `ASCIIIn` SBF blocks to RINEX using the `sbf2rin` program or the `SBFConverter` graphical tool. To be able to generate a RINEX file, the output of the meteo sensor must be formatted according to the NMEA XDR sentence.

1.15 Generate a "Pulse Per Second" Signal

The receiver is able to generate two independent x-pulse-per-second (xPPS) signals.

The PPS parameters (rate, polarity, width, time system, ...) are configured with the **`setPPSParameters`** and **`setPPS2Parameters`** commands for the first and second PPS outputs respectively. For instance, to configure the first PPS output to generate one pulse every ten seconds in the UTC time scale, , use:
`setPPSParameters,sec10,,,UTC <CR>`

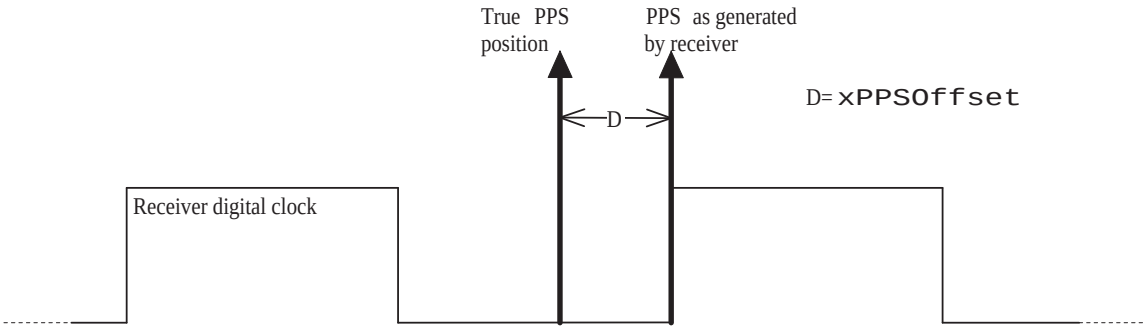


Figure 1-5: xPPS output granularity.

The physical PPS pulse is always generated at a raising edge of the internal receiver digital clock, as illustrated in the figure above. This leaves a time offset (noted "D" in the figure) between the desired xPPS pulse and the one actually generated by the receiver. This offset is very small (< 1 nanosecond) and can often be neglected. If needed, it is available in real-time in the `xPPSOffset` SBF block (only for the first PPS output).

To align its xPPS output with GNSS system time, the receiver relies on a recent GNSS time estimate derived from its PVT solution. During GNSS outages or, if enabled with **`setHoldoverTrigger`**, when non-authentic GNSS signals are suspected, the receiver enters time-holdover mode, in which the xPPS output is maintained based on the last available PVT solution. The PPS accuracy gradually degrades during holdover, and if the last PVT solution becomes older than a configurable timeout (*maxHoldover*), the generation of xPPS pulses is stopped.

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.

1.16 Time Tag External Events

This section is only applicable to receivers having the TimedEvent capability (see section 1.1).

The receiver can accurately time-tag electrical level transitions on its Event*X* inputs.

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.

1.17 Monitor the RF Spectrum

You can monitor the RF spectrum using the spectral analyzer in RxControl (go to the *View* > *Spectral View* 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.

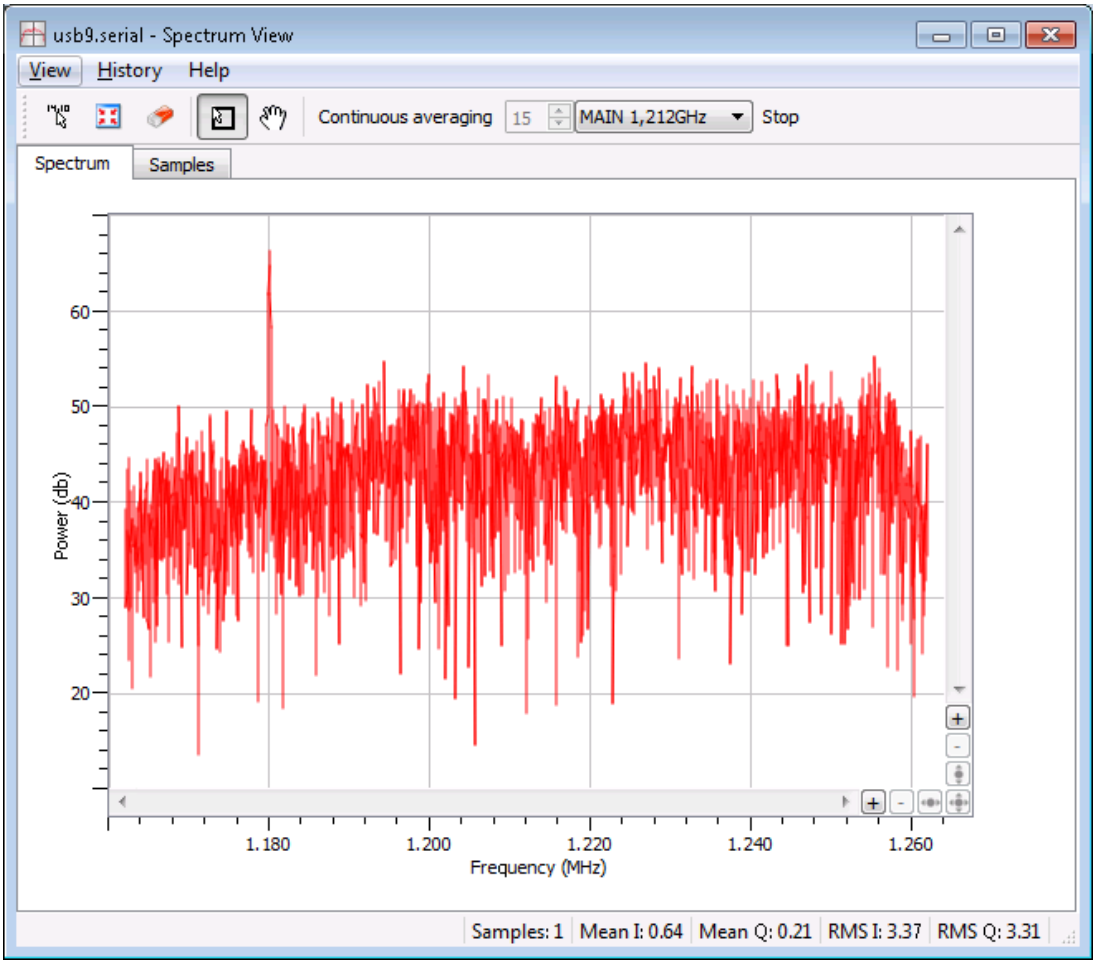
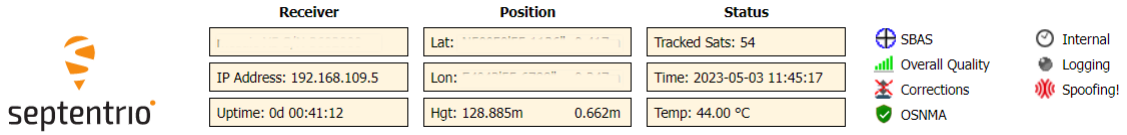


Figure 1-6: Spectral Analyser functionality of RxControl.

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.

1.18 Detect Spoofing Attacks

The receiver continuously checks various parameters to ensure the integrity and authenticity of the received GNSS signals. It also performs Navigation Message Authentication (NMA) using Galileo OSNMA. The outcome of those checks is available in the `Flags` field of the `RFStatus` SBF block and in the `AlertFlag` field of the `PVTGeodetic` and `PVTCartesian` SBF blocks. It is also shown as an icon in the header of each page of the web interface. For instance, the image below shows a red "Spoofing!" icon, indicating spoofing suspicion.



The following guidelines are recommended to maximize the probability of detecting a spoofing attack:

- Make sure that the antenna net gain (antenna LNA gain minus cable loss) is within the operating range of the receiver, as specified in the Hardware Manual. In particular, the automatic gain control (AGC) gain should be lower than 50dB for all frequency bands. The AGC gain can be checked in RxControl (*View* > *AGC Table*).

- Make sure that the AGC is in automatic mode. This is the default, but in case the receiver is not in its default configuration, you can enable auto-AGC mode with:
  `setAGCMode,all,auto <CR>`

- Some built-in spoofing detectors are based on cross-frequency and cross-constellation consistency checks. For optimal detection, it is therefore preferred to enable the tracking of all GNSS satellites and signals, at the cost of a slight increase of the CPU load:
  `setSignalTracking,all <CR>`
  `setSatelliteTracking,all <CR>`

- Additional checks are enabled for static receivers. If the receiver is known to be static, it is a good idea to configure it explicitly in static mode:
  `setPVTMode,Static <CR>`

- Enable Galileo OSNMA following the instructions in section 1.19.

1.19 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 measurements and/or PVT engine.

For the measurements, two OSNMA authentication levels are available: *off* where OSNMA authentication is ignored, *loose* where measurements are provided for satellites passing the authentication test or if their authentication status is unknown. In that mode, measurements from satellites failing OSNMA authentication are discarded from all internal processing (including the PVT engine), output streams and log files.

For the PVT, three OSNMA authentication levels are supported: *off* where OSNMA authentication is ignored, *loose* where satellites are included in the PVT processing if they are successfully authenticated or if their authentication status is unknown, and *strict* where only successfully-authenticated satellites are included. 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 satellites depending on the OSNMA service status).

After enabling OSNMA at measurement and/or PVT level, 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 at PVT and/or measurement level. For example, to discard measurements from non-authentic satellites from the measurements and the PVT engine, use :
   `setGalOSNMAUsage, loose, , loose <CR>`

2. In strict PVT mode, OSNMA authentication requires time information from an external non-GNSS source. In loose mode, this is optional but recommended for enhanced security. The user can enter the current time (in GPS time scale) with the **`exeSetTime`** command:
   `exeSetTime, DateTime, "2025-01-01 00:00:00" <CR>`

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.

1.20 Upgrade the Receiver

Upgrading the receiver is the process of installing a new GNSS firmware, a new permission file (see section 1.21) 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 1.7). 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).

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. 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.

1.21 Check the Receiver Permissions

The permission file lists which optional features (such as dual-antenna, logging, 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 1.1.

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:

- 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_o 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 has been encrypted on at least one satellite in the last 24 hours. To avoid inter-satellite biases, this applies to all Galileo satellites even if encryption is only enabled on some of them.
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.

The synchronization level is given by three status bits (`TOWSET`, `WNSET` and `FINETIME`) available in both 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.

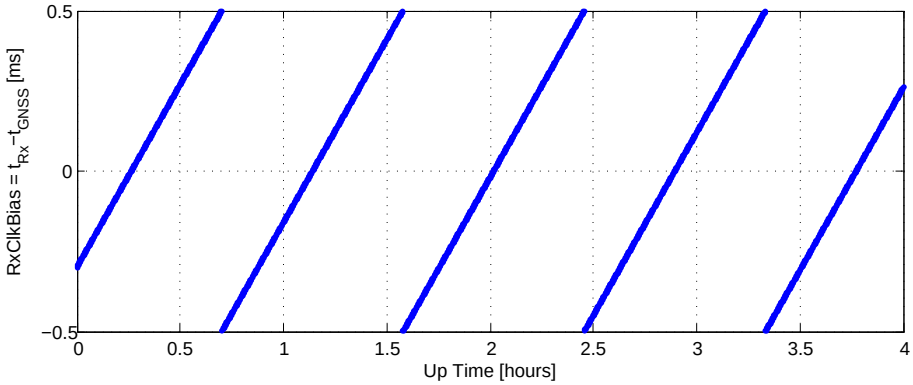


Figure 2-1: Example of the evolution of the receiver time offset with respect to the GNSS time in free-running mode.

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 couple 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 startup of the receiver, the clock bias would have been like in Figure 2-2 below.

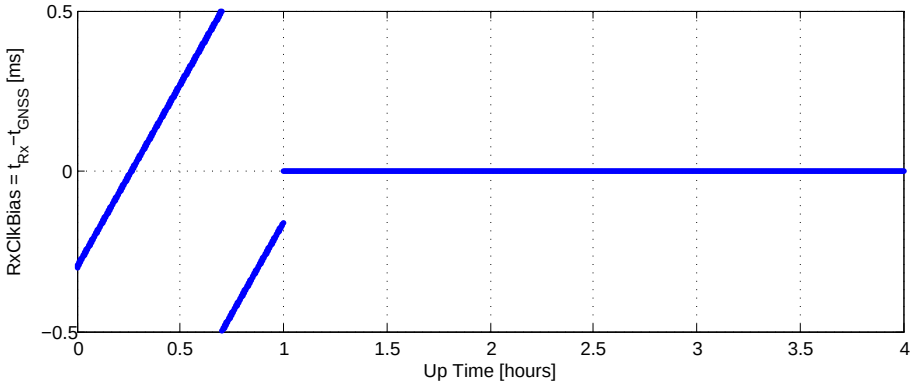


Figure 2-2: Effect of clock steering on the clock bias (clock steering enabled at an up time of 1 hour).

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, it is mandatory to reset the receiver before every (re)start of the simulation and to clear all satellite and PVT data stored in its non-volatile memory. This can be done by entering the **`exeResetReceiver, Hard, PVTData+SatData`** command.

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, 2023. The receiver time will be incorrect before that date.

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 DGNSS/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 DGNSS 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 DGNSS Positioning

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

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 in the RTCM or CMR format. The base station providing these RTK messages can be either static or moving. 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 DGNSS. Over the course of several minutes the positional accuracy will converge from several decimeters to several centimeters as the floating ambiguities become more accurate.

[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.

2.4.3.3 Moving Base

In RTK, the base station does not necessarily need to be static. In some applications, one is interested in the relative positioning of two moving vehicles. In that case, both base and rover receivers are mounted on moving platforms and the RTK engine computes the baseline between them. If both base and rover receivers are mounted on the same vehicle, the baseline can be used to determine the orientation of the vehicle (see section 1.12). If accurate absolute positioning is required in addition to relative positioning, the moving base receiver can operate in RTK mode and get RTK correction from a fixed base station (see section 1.8.2).

With the command **`setDiffCorrUsage`**, the rover receiver must be informed that the base is moving. The baseline coordinates and orientation is contained in the `BaseVectorCart` and `BaseVectorGeod` SBF blocks.

Due to delays in the generation and transmission of the RTK data (base station position and measurements) from the base to the rover, the RTK data has a certain "age" when received by the rover. When operating with a moving base station, the RTK engine is of the "low-latency" type. This means that, when the rover computes its RTK position at time t_0, it extrapolates the most recently received RTK data from the base to time t_0. The accuracy of this extrapolation, and hence the accuracy of the final RTK solution, degrades with the age of the RTK data. Therefore it is essential that the base sends its position and measurements at a sufficient rate.

The default rate of 1 Hz is adequate in the case of a static base station, but is generally too low for a moving base with a non-constant velocity. In moving base operation, it is recommended to set the RTK data rate to its maximum allowed value of 10 Hz. Not only the RTK data rate, but also the communication link latency is important. Especially in moving base, it is essential to have a low-latency communication link between base and rover.

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 provided either by GNSS satellites (e.g. Galileo HAS (High Accuracy Service)) or by 3rd party service providers.

2.4.4.1 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, the regional datum must be provided with the **`setGeodeticDatum`** command.

2.4.4.2 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 DGNSS mode, the position does not exhibit a sudden jump, but slowly degrades from RTK to DGNSS accuracy.

2.4.6 PVT Latency

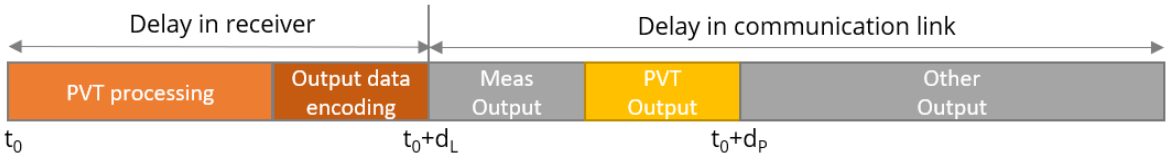


Figure 2-3: PVT latency.

The total PVT latency is the delay between the time of applicability of the PVT (t_0) and its availability to the user application (t_0+d_P). It is the sum of the PVT processing time, the output data encoding time, and the time needed to transmit the data over the communication link. The processing and encoding time (d_L) is measured by the receiver and is reported in the `Latency` field of the `PVTCartesian` and `PVTGeodetic` SBF blocks. The communication latency (grey and yellow bars) entirely depends on the communication link and is unknown to the receiver. Serial ports configured at high baud rate generally yield the shortest latency.

The shortest total PVT latency is obtained when the receiver only needs to output PVT data (e.g. only the `PVTCartesian` SBF block). Latency increases when more data need to be output. Note that the receiver always outputs raw measurements, e.g. the `MeasEpoch` SBF block, before PVT data. Therefore, when using slow communication links, enabling the measurement output can have a significant impact on the total PVT latency.

2.4.7 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 DGNSS or RTK corrections from a DGNSS/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] Sinko, J., A Compact Earth Tides Algorithm for WADGPS. Proceedings of ION GPS-95, Palm Springs, California, September 12-15, 1995, pp. 35-44.

2.4.7.1 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.

2.5 Antenna Effects

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

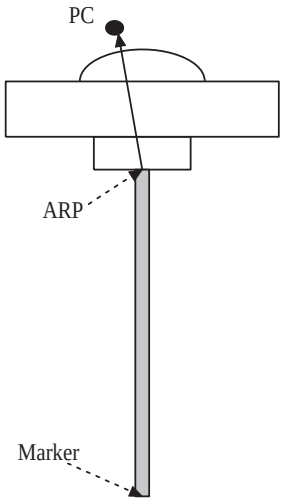


Figure 2-4: Antenna mount.

The GNSS measurements (pseudoranges and carrier phases observables) refer to a theoretical point in space called the phase center (noted PC in Figure 2-4). 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. They contain phase center variation (PCV) and phase center offset (PCO) values 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 1.20.

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 2-4). 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 the marker position unless the position of the antenna phase center is explicitly requested with the **setAntennaOffset** command.

In DGNSS 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.

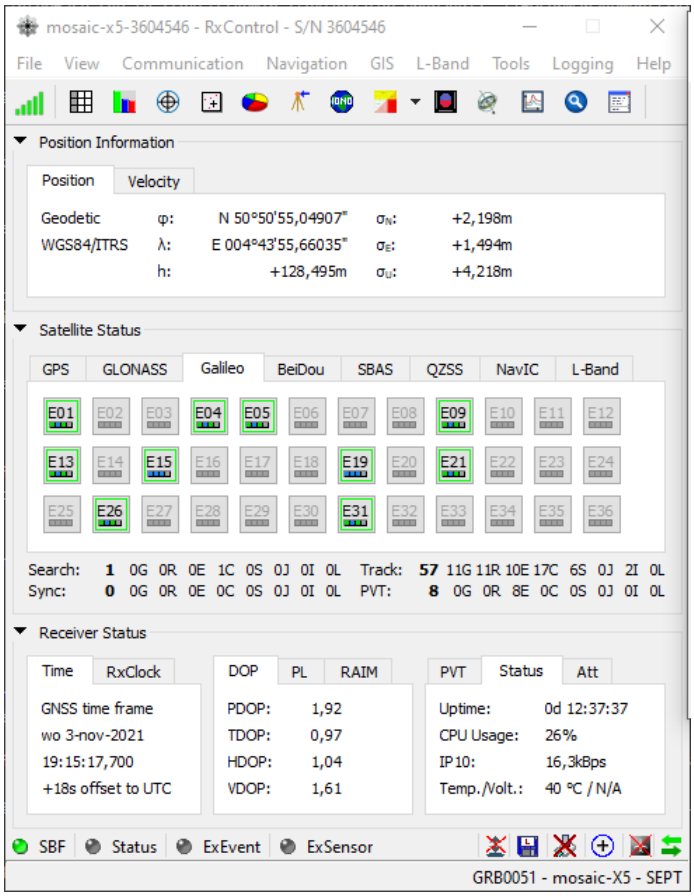


Figure 2-5: Example OSNMA satellite status. The satellites marked with a green square are successfully authenticated.

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.

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 section1.2.3).

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 (40/2)*	*Mode*	*PRN*
		+ Com1 + Com2 all	-20.00 …0.00 …20.00 m	1 …50 sec	Unknown		on off	none + G01 ... G36 + S120 ... S158 + 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 are required, 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 usable length of 20 characters (40/2). Only half of the total password length is available to the user, the other half being reserved by the system. Password arguments are obfuscated by the receiver and they cannot be read back in plaintext in command replies, unless explicitly specified otherwise.
  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+ ... +G36`, `SBAS` is an alias for `S120+ ... +S158` 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 [1]	lstAntennaInfo	*Antenna*								
		Overview Main Aux1 [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` or `Aux1` will return the phase center variation parameters corresponding to the main or auxiliary 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="22.1.0">
  <Antenna ID="3COAG35          NONE"/>
  <Antenna ID="3COAG35H         NONE"/>
  <Antenna ID="3COAT-703        NONE"/>
  <Antenna ID="3COAT-705        NONE"/>
  <Antenna ID="3COAT901         NONE"/>
  ...
  <Antenna ID="WIEGPOT206-003   NONE"/>
</AntennaInfo>
COM1>
```

```
COM1>  lai, "AERAT2775_159 SPKE"<CR>
$R; lai,"AERAT2775_159   SPKE"
<?xml version="1.0" encoding="ISO-8859-1" ?>
<AntennaInfo version="22.1.0">
  <Antenna ID="AERAT2775_159   SPKE"/>
    <L1>
      <offset north="0.5" east="0.1" up="77.8"/>
      <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>
```

[1] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

```
</AntennaInfo>
COM1>
```

help	lstCommandHelp	*Action (255)*								
		Overview [command]								

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 Permanent								

Use this command to list the contents of a configuration file. A configuration file contains a list of user commands.

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, which is always empty.
`User1`	A user-defined configuration.
`User2`	A user-defined configuration.
`Permanent`	Permanent configuration. See section 1.7. In the reply, the related commands are prepended with a # sign.

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 configuration 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.

Note that the permanent settings (see 1.7) are not affected by this command. In particular, copying `RxDefault` to `Current` does not overwrite the permanent settings.

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>
```

sgp1 ggp1	**setGPIO1Mode** **getGPIO1Mode**	*Mode*	*GPOLevel*	*GPILevel*						
		GPI GPO	Low High pPVTLED nPVTLED pRTKLED nRTKLED pTRACKLED nTRACKLED pDIFFCORRLED nDIFFCORRLED	(Low) (High)						

RxControl: Navigation > Receiver Operation > GPIO

Use these commands to set the functionality of the GPIO1 pin.

In general-purpose output mode (`GPO`), the pin level is defined by the *GPOLevel* argument. The pin can be set at a fixed low or high level, or it can be configured as a status indicator. In the latter case, the pin is typically connected to a LED. Use the LED configuration prefixed by "`p`" if the LED is wired to light when the pin level is high and use the configuration prefixed by "`n`" if the LED is wired to light when the pin level is low.

In general-purpose input mode (`GPI`), the current pin level is reported in the read-only *GPILevel* argument. That argument is meaningless in `GPO` mode.

Example

```
COM1>  sgp1, GPO, nTRACKLED, (Low)<CR>
$R: sgp1, GPO, nTRACKLED, (Low)
  GPIO1Mode, GPO, nTRACKLED, (Low)
COM1>
```

sgp2 ggp2	**setGPIO2Mode** **getGPIO2Mode**	*Mode*	*GPOLevel*	*GPILevel*						
		GPI GPO	Low High pPVTLED nPVTLED pRTKLED nRTKLED pTRACKLED nTRACKLED pDIFFCORRLED nDIFFCORRLED	(Low) (High)						

RxControl: Navigation > Receiver Operation > GPIO

Use these commands to set the functionality of the GPIO2 pin.

In general-purpose output mode (`GPO`), the pin level is defined by the *GPOLevel* argument. The pin can be set at a fixed low or high level, or it can be configured as a status indicator. In the latter case, the pin is typically connected to a LED. Use the LED configuration prefixed by "`p`" if the LED is wired to light when the pin level is high and use the configuration prefixed by "`n`" if the LED is wired to light when the pin level is low.

In general-purpose input mode (`GPI`), the current pin level is reported in the read-only *GPILevel* argument. That argument is meaningless in `GPO` mode.

Example

```
COM1>  sgp2, GPO, nTRACKLED, (Low)<CR>
$R: sgp2, GPO, nTRACKLED, (Low)
  GPIO2Mode, GPO, nTRACKLED, (Low)
COM1>
```

lif	lstInternalFile	*File*								
		Permissions Identification Debug Error SisError DiffCorrError SetupError RxMessages License								

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.
`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.
`License`	The license file associated with the receiver.

Example

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

lmd	lstMIBDescription	*File (255)*								
		Overview SBFTable [command]								

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.
`Aux1`	First auxiliary antenna.

Signals	Description
`GPSL1CA`	GPS L1 C/A 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.
`GLOL2P`	GLONASS L2 P signal.
`GLOL2CA`	GLONASS L2 C/A signal.
`GLOL3`	GLONASS L3 signal.
`GALE1BC`	Galileo E1 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.
`BDSB2b`	BeiDou B2b signal.

Signals (Continued)	Description
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.
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.
USB1	USB-device virtual serial port 1.
USB2	USB-device virtual serial port 2.

Capabilities	Description
SBAS	Positioning with SBAS corrections.
DGNSSRover	Positioning with DGNSS corrections.
RTKRover	Positioning with RTK corrections.
RTKBase	Generation of RTK corrections.
RTCMv3x	Generation/decoding of RTCM v3.x 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.
MovingBase	Usage of Moving Base.
MeasAv	Measurement availability.
IM	Interference mitigation.
FreqSync	External frequency input.
Integrity	Integrity info
GalOSNMA	Galileo OSNMA
AuthIO	Authenticated IO
SignalAuth	Signal Authentication
UnrestrictedMotion	Motion Fencing

Capabilities (Continued)	Description
NoNatoLimits	No NATO Limits
PPPGalileoHAS-SIS	Galileo High Accuracy Service (HAS) from the E6 signal

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 + USB1 + USB2 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 1.2.3).

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 [2] grst	**exeResetReceiver** **getResetReceiver**	***Level***	***EraseMemory***							
		Soft Hard Upgrade	none + Config + PVTData + SatData + 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 its default settings, with the exception of the settings set by permanent commands (see section 1.7 for details). 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.
`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.

[2] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

Example

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

ltc	lstTLSCertificates	*Certificate*								
		[certificate ID]								

Use this command with the argument *Certificate* set to `Overview` to get a list of all X509 certificates available on the receiver.

Use this command with the argument *Certificate* set to one of the certificate IDs returned by **`lstTLSCertificates,Overview`** to retrieve information about that particular x509 certificate. Note that the private key corresponding to the certificate is never shown.

Examples

```
COM1>  ltc, Overview <CR>
$R; ltc, Overview
<?xml version="1.0" encoding="ISO-8859-1" ?>
<TLSCertificates>
    <certificate ID="1" subject="O=Septentrio, OU=Devices,
       serialNumber=SN23330000000244"/>
    <certificate ID="2" subject="C=BE, ST=Vlaams-Brabant, L=Leuven,
        O=Septentrio, OU=Devices, CN=Septentrio Devices Signing CA
       G1.1.1"/>
    <certificate ID="3" subject="C=BE, ST=Vlaams-Brabant, O=
       Septentrio, OU=Devices, CN=Septentrio Devices Intermediate
       CA G1.1"/>
    <certificate ID="4" subject="C=BE, ST=Vlaams-Brabant, O=
       Septentrio, OU=Devices, CN=Septentrio Devices Root CA G1"/>
</TLSCertificates>
COM1>

COM1>  ltc, 1 <CR>
$R; ltc, 1
<?xml version="1.0" encoding="ISO-8859-1" ?>
<TLSCertificates>
    <certificate ID="1">
        <version value="3"/>
        <serial value="26:2D:BB:45:D1:45:97:B2:2D:8A:59:0A:FB:90:F6
           :A2"/>
        <issuer name="C=BE, ST=Vlaams-Brabant, L=Leuven, O=
           Septentrio, OU=Devices, CN=Septentrio Devices Signing CA
            G1.1.1"/>
        <subject name="O=Septentrio, OU=Devices, serialNumber=
           SN23330000000244"/>
        <validity>
              <not_before date="2025-02-02 08:51:53">
              <not_after date="9999-12-31 23:59:59">
        </validity>
        <basic_contraints CA="false"/>
        <pem>
-----BEGIN CERTIFICATE-----
MIICajCCAfCgAwIBAgIQJi27RdFFl7ItilkK+5D2ojAKBggqhkjOPQQDAzCBjTEL
MAkGA1UEBhMCQkUxFzAVBgNVBAgMDlZsYWFtcy1CcmFiYW50MQ8wDQYDVQQHDAZM
ZXV2ZW4xEzARBgNVBAoMClNlcHRlbnRyaW8xEDAOBgNVBAsMB0RldmljZXMxLTAr
```

```
BgNVBAMMJFNlcHRlbnRyaW8gRGV2aWNlcyBTaWduaW5nIENBIEcxLjEuMTAgFw0y
NTAyMDIwODUxNTNaGA85OTk5MTIzMTIzNTk1OVowQjETMBEGA1UECgwKU2VwdGVu
dHJpbzEQMA4GA1UECwwHRGV2aWNlczEZMBcGA1UEBRMQU04yMzMzMDAwMDAwMDI0
NDB2MBAGByqGSM49AgEGBSuBBAAiA2IABKXGCKHZXODxH+B7T+NSuyF3icQSKnf6
1qZ+brmr5LLyvY8LMUWZkDSslbxcteouN+B0NpXnDtPAJRfVpR6mjGLfLC7PyAr8
J+MRbAaq32Ofje1iw/FhyNEfwpGD5osqVaNdMFswCQYDVR0TBAIwADAfBgNVHSME
GDAWgBSNsXvJ4khacvQSvI2B0dEhq8aXdTAOBgNVHQ8BAf8EBAMCA8gwHQYDVR0l
BBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMAoGCCqGSM49BAMDA2gAMGUCMQDL3aRS
SuBT0PsFm9Vb399ymoviSZ45JDp9KCaVnu6Q4GiAwy2RluEYkMNHuFo5wRQCMCnZ
VhTa0okmlE5Kg8oV3o4cN5S84xFcwwwIZ7zeQFKinhA0zOvZ/DkgSgC+0Pd6Cg==
-----END CERTIFICATE-----
        </pem>
    </certificate>
</TLSCertificates>
COM1>
```

3.2.2 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**	*Com*	*Usb*							
		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, 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.

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.

Example

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

facr	factoryReset									

This command is used to reset the receiver to its factory default state. It will erase all user settings, including the list of known users (see section 1.4), and restore the factory default configuration.

Except for the specific case mentioned below, this command can only be entered by logged-in users (see the **login** command). If there are no users defined yet, it is advised to create one first following the procedure described in section 1.4.

In the specific case that all anonymous accesses to the receiver are blocked (see **setDefaultAccessLevel**) and that the credentials for all users are lost, the following recovery procedure can be followed. It requires physical access to the receiver and can only be executed over USB or COM connections.

1. Physically reset the receiver, either by a power cycle or via its reset pin (not via the **exeResetReceiver** command).
2. Within 30 seconds, connect to the receiver using its USB or one of its COM ports and send the **factoryReset** command.
3. Within a minute, physically reset the receiver again, either by a power cycle or via its reset pin (not via the **exeResetReceiver** command).
4. The receiver restarts, clears its configuration, returns to factory defaults and then resets one last time.

factoryReset is a dangerous command that will erase all user settings and data. Use it with caution, and only when you are sure you want to reset the receiver to its factory defaults. In most cases, the **exeResetReceiver** command is more appropriate.

Example

Log in as user "admin" with password "admin", then issue **factoryReset** to reset the receiver to factory defaults.

```
COM1>  login, admin, admin <CR>
$R! LogIn
  User admin logged in.
COM1>  factoryReset <CR>
$R: factoryReset: Resetting receiver to factory defaults.
COM1>
```

sit git	**setInactivePortTimeout** **getInactivePortTimeout**	***Timeout***								
		0 . . . 60 min								

RxControl: File > User Management

This command specifies the number of minutes of inactivity (no input or output) on a port after which the receiver will automatically log out the authenticated use and revert to anonymous access. Setting the value to 0 disables automatic logout.

Example

```
COM1>  sit, 30<CR>
$R: sit, 30
  InactivePortTimeout, 30
COM1>
```

login	**LogIn**	*UserName (16)*	*Password (32/2)* (3)							

Use this command to authenticate yourself.

When initially connecting to the receiver, a user is considered "anonymous". By default, anonymous users have full control over USB or COM ports. See also the **`setDefaultAccessLevel`** command.

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

The list of known 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 or if *Password* is empty.

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.

Note that users logged into inactive ports are logged out automatically after a certain timeout (see **`setInactivePortTimeout`** command).

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>
```

(3) Only half of the total password length is available to the user (See Section 3.1.4 on page 55)

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*					
		+ 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 (*User-Level*). If *UserLevel* is `User`, the user has full control of the receiver. If it is `Viewer`, the user can only issue **`get`**-commands.

UserName and *Password* are case-sensitive. Note that the *UserName* **"RxAdmin"** and the *Password* **"S3pt3ntr10"** are reserved and cannot be used.

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`**.

⚠ This is a permanent command: it survives resets and power cycles without calling **`exeCopyConfigFile`** and is not removed by a reset to default configuration (see the **`exeResetReceiver`** command).

Example

```
COM1>  sual, User3, Mildred, mypwd, Viewer<CR>
$R: sual, User3, Mildred, mypwd, Viewer
  UserAccessLevel, User3, Mildred, mypwd, Viewer
COM1>
```

3.2.3 Tracking and Measurement Generation

scco gcco	**setCalibCommonDelay** **getCalibCommonDelay**	*CommonDelay*								
		-10000.000 ...0.000 ... 10000.000 ns								

RxControl: Navigation > Advanced User Settings > Delay Calibration

setCalibCommonDelay and **setCalibSignalDelay** are expert-level commands that can be used to compensate pseudoranges for signal delays in the reception chain (antenna, cable, amplifiers, receiver frontend, ...).

setCalibCommonDelay sets a common delay affecting all GNSS signals and **setCalibSignalDelay** sets additional signal-specific delays. The partitioning of delays between the two commands is at the user's discretion. The common delay could for example represent the average delay of all signals, and the signal-specific delay would then contain the per-signal deviation from the average.

The delays are compensated at the source, when generating the pseudoranges. All pseudoranges output by the receiver and used internally are corrected for the sum of the common and the signal-specific delays.

Providing non-zero delays primarily affects the receiver time determination (receiver clock bias and PPS generation). In particular, setting *CommonDelay* to X nanoseconds will cause the PPS pulse to come X nanoseconds earlier. Note that the same effect can be obtained with the *Delay* argument of the **setPPSParameters** command. The difference is that the latter command only moves the PPS pulse without affecting the pseudoranges.

⚠ Using these commands is completely optional. They should only be used by expert users who understand the consequences of modifying the default values. Providing incorrect delay values can significantly degrade the performance of the receiver.

Example

```
COM1>  scco, 65.4<CR>
$R: scco, 65.4
  CalibCommonDelay, 65.400
COM1>
```

scsi gcsi	**setCalibSignalDelay** **getCalibSignalDelay**	***Signal*** *Signal*	*SignalDelay*							
		+ GPSL1CA + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL2P + GLOL2CA + GLOL3 + GALE1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL6 + QZSL1C + QZSL1S + QZSL1CB + NAVICL5 all	-1000.000 . . . 0.000 . . . 1000.000 ns							

RxControl: Navigation > Advanced User Settings > Delay Calibration

setCalibSignalDelay and **setCalibCommonDelay** are expert-level commands that can be used to compensate pseudoranges for signal delays in the reception chain (antenna, cable, amplifiers, receiver frontend, ...).

Refer to the **setCalibCommonDelay** command for the command description.

Using these commands is completely optional. They should only be used by expert users who understand the consequences of modifying the default values. Providing incorrect delay values can significantly degrade the performance of the receiver.

Example

```
COM1>  scsi, GPSL5+GALE5a, -2.56<CR>
$R: scsi, GPSL5+GALE5a, -2.56
  CalibSignalDelay, GPSL5+GALE5a, -2.560
COM1>
```

sca gca	**setChannelAllocation** **getChannelAllocation**	***Channel*** *Channel*	*Satellite*	*Search*	*Doppler*	*Window*				
		+ Ch01 ... Ch80 all	auto G01 ... G36 G38 G39 F01 ... F14 E01 ... E36 S120 ... S158 C01 ... C63 J01 ... J10 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.

In forced allocation mode (*Satellite* argument different from `auto`), it is possible to specify the Doppler window in which the receiver will 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 + Reserved2 + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL2P + GLOL2CA + GLOL3 + GALE1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL6 + QZSL1C + QZSL1S + 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 ... G36 + G38 + G39 + R01 ... R30 + E01 ... E36 + S120 ... S158 + C01 ... C63 + J01 ... J10 + 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+G33+G34+G35+G36
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>
```

snho [4] gnho	**setSignalHealthOverride** **getSignalHealthOverride**	*Measurements*	*PVT*							
		none	none							
		+ GPSL1CA	+ GPSL1CA							
		+ GPSL2PY	+ GPSL2PY							
		+ GPSL2C	+ GPSL2C							
		+ GPSL5	+ GPSL5							
		+ GPSL1C	+ GPSL1C							
		+ GLOL1CA	+ GLOL1CA							
		+ GLOL2P	+ GLOL2P							
		+ GLOL2CA	+ GLOL2CA							
		+ GLOL3	+ GLOL3							
		+ GALE1BC	+ GALE1BC							
		+ 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							
		+ QZSL1CB	+ QZSL1CB							
		+ NAVICL5	+ NAVICL5							
		+ GPS	+ GPS							
		+ GLONASS	+ GLONASS							
		+ GALILEO	+ GALILEO							
		+ SBAS	+ SBAS							
		+ BEIDOU	+ BEIDOU							
		+ QZSS	+ QZSS							
		+ NAVIC	+ NAVIC							
		all	all							

RxControl: Navigation > Receiver Operation > Masks

This advanced command can be used to override the health flags for selected GNSS signals. This feature is mainly intended for experimental use: ignoring the health flags may lead to unpredictable results.

By default, the receiver discards GNSS signals for which the health flag is set to "unhealthy" in the navigation message. This command instructs the receiver to ignore the health flags for the specified set of signals. The health flags can be ignored for the measurement generation only, or for both the measurement generation and the PVT computation.

If a signal is listed in the *Measurements* argument, measurements (pseudoranges, carrier phases, ...) are produced for that signal regardless of its health flag. These measurements become available in all output formats (SBF, RINEX, RTCM, ...). This applies to all satellites transmitting that signal.

If a signal is listed in both the *Measurements* and the *PVT* arguments, measurements from that signal are allowed to be used in PVT engine, regardless of their health status. This applies to all PVT modes (standalone, RTK, ...).

The signals in the following table do not transmit a health flag or the receiver does not decode it. For those signals, the health status is taken from a different signal, as indicated in the table.

[4] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

Signal	Health flag source
GPSL1C	GPSL1CA
GALE5	GALE5a and GALE5b
GALE6BC	GALE1BC
GPSL2PY	GPSL1CA
GLOL2P	GLOL2CA
GLOL3	GLOL1CA
QZSL1S	QZSL1CA or QZSL1CB

The GPS L5 signal is handled a bit differently. When overriding the health flag for GPS L5, measurements from that signal are produced or used in the PVT engine if the L1CA signal of the corresponding satellite is flagged as healthy.

Note that health flags are assumed "healthy" if unknown or if not received yet.

Example

```
COM1>  snho, GPSL5, GPSL5<CR>
$R: snho, GPSL5, GPSL5
  SignalHealthOverride, GPSL5, GPSL5
COM1>
```

snt [5] gnt	**setSignalTracking** **getSignalTracking**	*Signal*								
		+ GPSL1CA + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL2P + GLOL2CA + GLOL3 + GALE1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL6 + QZSL1C + QZSL1S + 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 `GPSL2PY` has no effect unless `GPSL1CA` 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.

Example

To configure the receiver in dual-frequency GPS L1CA+L2C mode, use:

```
COM1>  snt, GPSL1CA+GPSL2C <CR>
$R: snt, GPSL1CA+GPSL2C
  SignalTracking, GPSL1CA+GPSL2C
COM1>
```

[5] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

ssi [6] gsi	**setSmoothingInterval** **getSmoothingInterval**	***Signal*** *Signal*	*Interval*	*Alignment*						
		+ GPSL1CA + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL2P + GLOL2CA + GLOL3 + GALE1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL6 + QZSL1C + QZSL1S + 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
```

[6] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

```
  SmoothingInterval, GPSL1CA, 300, 0
COM1>

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

3.2.4 Frontend and Interference Mitigation

sam [7] gam	**setAGCMode** **getAGCMode**	***Band*** *Band*	*Mode*	*Gain*						
		+ L1 + B3E6 + L2 + L5 + LBand 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.

Note that the AGC configuration is applied to both the main and the aux1 antennas.

Example

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

[7] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

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. All frequency bands are sampled in turn.
`AfterIM`	The samples in the `BBSamples` SBF block are taken after interference mitigation. All frequency bands are sampled in turn.

Example

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

sfm [8] gfm	**setFrontendMode** **getFrontendMode**	*Mode*								
		Nominal SingleAnt DualAnt								

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

Use this command to define the frontend operating mode. The following modes are available.

Mode	Description
`Nominal`	Nominal operation. This option will choose the best operating mode for your product. Normally this is `SingleAnt`, but for products with heading as main purpose, this is `DualAnt`.
`SingleAnt`	Single-antenna mode.
`DualAnt`	Dual-antenna mode. This option forces the receiver into dual-antenna mode, but only if the receiver has dual-antenna capability (see the **`getReceiverCapabilities`** command).

This command only takes effect at the next reset or reboot. It must be stored in the boot configuration file with the **`exeCopyConfigFile`** command.

Example

```
COM1>  sfm, Nominal<CR>
$R: sfm, Nominal
  FrontendMode, Nominal
COM1>
```

[8] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

smnf gmnf	**setManualNotchFilter** **getManualNotchFilter**	***Notch*** *Notch*	*Enable*	*CenterFreq*	*Bandwidth*					
		+ Notch1 + Notch2 + Notch3 all	off on	1100.000 ...1700.000 MHz	45...3000 kHz					

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

Use these commands to manually configure one or more notch filter(s) in the receiver's frontend. Notch filters are used to cancel narrowband interferences.

To enable a manual notch, the *Enable* argument must be `on` and the region of the spectrum to be blanked by the notch filter 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.

Manual notches are applied to both antennas.

In some cases, enabling or disabling a notch may cause a short GNSS signal tracking interruption.

Example

```
COM1>  smnf, Notch1, on, 1227.0, 45<CR>
$R: smnf, Notch1, on, 1227.0, 45
  ManualNotchFilter, Notch1, on, 1227.000, 45
COM1>
```

sitm [9] gitm	**setRFInterferenceMitigation** **getRFInterferenceMitigation**	*JammingClass*								
		none + Class1 + Class2 all								

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

By default, radio-frequency interference mitigation is enabled for the following classes of interferences:

JammingClass	Description
`Class1`	Mitigation of relatively narrow-band interferences such as carrier-wave (CW), modulated CW, FM & FSK modulations, harmonic interference, ...
`Class2`	Mitigation of relatively wide-band interferences such as sweep-type (chirp) jammers, pulsed jammers, frequency-hopping signals, interference from Distance Measurement Equipment (DME), ...

This command can be used to check the effect of turning interference mitigation on or off. It is provided for experimental purposes only, as it is recommended to keep interference mitigation enabled at all times.

Note that the status of the interference mitigation can be monitored with the `RFStatus` SBF block.

Invoking this command may cause a short interruption of the signal tracking.

Example

```
COM1>  sitm, Class1<CR>
$R: sitm, Class1
  RFInterferenceMitigation, Class1
COM1>
```

[9] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

3.2.5 Navigation Filter

sao [10] gao	**setAntennaOffset** **getAntennaOffset**	***Antenna*** *Antenna*	*DeltaE*	*DeltaN*	*DeltaU*	*Type (20)*	*SerialNr (20)*	*SetupID*	*APCOutput*	
		+ Main + Aux1 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	off on	

RxControl: Navigation > Receiver Setup > Antennas

Use these commands to define/inquire the parameters that are associated with the antennas 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. The purpose is to take into account the fact that the antenna may not be located directly on the point of interest. Each antenna can have its own marker. When *APCOutput* is `off` (default), all absolute positions reported by the receiver are marker positions, obtained by subtracting this offset from the ARP. When *APCOutput* is `on`, the `PVTGeodetic` and `PVTCartesian` SBF blocks and the NMEA sentences contain the position of the average L1 phase center instead of the marker.

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 message types 1007, 1008 and 1033 of RTCM3. It has otherwise no effect on the receiver operation. In multi-antenna receivers, this parameter is only relevant for the `Main` antenna, as RTCM messages are applicable to the main antenna only.

Example

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

[10] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

```
COM1>
```

sbcp gbcp	**setBaseCPB** **getBaseCPB**	*Mode*	*L1CPB*	*L2CPB*						
		auto manual Ashtech Javad Leica Magellan Novatel Septentrio Topcon Trimble Unicore	-1000.00 ... 0.00 ... 1000.00 m	-1000.00 ... 0.00 ... 1000.00 m						

RxControl: Navigation > Advanced User Settings > PVT > Base Station Parameters

Use this command to specify the code-phase bias value to be applied to the measurements received from the RTK base station.

The code phase bias values are used to remove the inter-frequency carrier phase biases from the GLONASS measurements received from the base station.

By default, code-phase bias compensation is performed automatically. For older base stations, it may be useful to overrule the automatic compensation by forcing manual values with this command.

The L1 and L2 code phase biases follow the definition of RTCM v3.2, message type 1230.

Example

```
COM1>  sbcp, manual, 142.4, 142.4 <CR>
$R: sbcp, manual, 142.4, 142.4
  BaseCPB, manual, 142.40, 142.40
COM1>
```

sdca gdca	**setDiffCorrMaxAge** **getDiffCorrMaxAge**	*DGNSSCorr*	*RTKCorr*	*PPPCorr*						
		0.0 ...400.0 ...3600.0 s	0.0 ...20.0 ...600.0 s	0.0 ...360.0 ...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 *DGNSSCorr* defines the timeout of the range corrections when the PVT is computed in DGNSS 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).

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*	*MovingBase*				
		LowLatency	0.1 . . . 3600.0 s	auto manual	0 . . . 4095	off on				

RxControl: Navigation > Positioning Mode > PPP and Differential Corrections

Use these commands to define/inquire the usage of incoming differential corrections in DGNSS 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 (DGNSS, 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 to be used. If `auto` is selected the receiver will automatically select the preferred base station. If `manual` is selected, the receiver will only use the corrections from the base station defined by the *BaseID* argument (in both DGNSS and RTK modes).

The *MovingBase* argument defines whether the base station is static or moving.

Example

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

sem gem	**setElevationMask** **getElevationMask**	***Engine*** *Engine*	*Mask*							
		+ Tracking + PVT all	-90 . . . 5 . . . 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.

Satellites under the PVT mask are excluded from the PVT computation. Note that it does not imply that all satellites above the mask are included: in addition to the elevation mask, satellites must pass internal sanity checks to be effectively included 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, Tracking, 15<CR>
$R: sem, Tracking, 15
  ElevationMask, Tracking, 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>
```

sim gim	**setIonosphereModel** **getIonosphereModel**	*Model*								
		auto off KlobucharGPS SBAS MultiFreq KlobucharBDS								

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.
`KlobucharGPS`	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.
`KlobucharBDS`	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>
```

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 [11] 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>
```

[11] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

spm [12] gpm	**setPVTMode** **getPVTMode**	*Mode*	*RoverMode*	*RefPos*						
		Static Rover	+ StandAlone + SBAS + DGNSS + RTKFloat + RTKFixed + PPP + RTK all	auto Geodetic1 Geodetic2 Geodetic3 Geodetic4 Geodetic5 Cartesian1 Cartesian2 Cartesian3 Cartesian4 Cartesian5						

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 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
```

[12] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

```
  PVTMode, Static, StandAlone+RTK, Geodetic1
COM1>
```

srd grd	**setReceiverDynamics** **getReceiverDynamics**	*Level*	*Motion*							
		High Moderate Low	Static Quasistatic Pedestrian Automotive RaceCar HeavyMachinery UAV 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 *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.

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.
`RaceCar`	High speed terrestrial vehicle. E.g. race cars, ...
`HeavyMachinery`	Construction equipment, tractors, ...
`UAV`	Unmanned Aerial Vehicle.
`Unlimited`	Unconstrained motion.

Example

```
COM1>  srd, High, Automotive<CR>
$R: srd, High, Automotive
  ReceiverDynamics, High, Automotive
COM1>
```

ernf [13] grnf	**exeResetNavFilter** **getResetNavFilter**	***Level***								
		+ PVT + AmbRTK + GNSSAttitude + AmbGNSSAttitude 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.
`GNSSAttitude`	Reset the whole attitude filter such that all previous attitude information is discarded and the filter is re-initialized, including the attitude ambiguities.
`AmbGNSSAttitude`	Only reset the ambiguities used in attitude computation to float status.

Example

```
COM1>  ernf, PVT <CR>
$R: ernf, PVT
  ResetNavFilter, PVT
COM1>
```

[13] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

ssho gsho	**setSatelliteHealthOverride** **getSatelliteHealthOverride**	*Measurements*	*PVT*							
		none + G01 ... G36 + G38 + G39 + R01 ... R30 + E01 ... E36 + S120 ... S158 + C01 ... C63 + J01 ... J10 + I01 ... I14 + GPS + GLONASS + GALILEO + SBAS + BEIDOU + QZSS + NAVIC all	none + G01 ... G36 + G38 + G39 + R01 ... R30 + E01 ... E36 + S120 ... S158 + C01 ... C63 + J01 ... J10 + I01 ... I14 + GPS + GLONASS + GALILEO + SBAS + BEIDOU + QZSS + NAVIC all							

RxControl: Navigation > Receiver Operation > Masks

This advanced command can be used to override the health flags for selected satellites. This feature is mainly intended for experimental use: ignoring the health flags may lead to unpredictable results.

By default, the receiver discards GNSS signals for which the health flag is set to "unhealthy" in the navigation message. This command instructs the receiver to ignore the health flags for the specified set of satellites. The health flags can be ignored for the measurement generation only, or for both the measurement generation and the PVT computation.

If a satellite is listed in the *Measurements* argument, measurements (pseudoranges, carrier phases, ...) are produced for all signals of that satellite regardless of their health flag. These measurements become available in all output formats (SBF, RINEX, RTCM, ...).

If a satellite is listed in both the *Measurements* and the *PVT* arguments, measurements from all its signals are allowed to be used in PVT engine, regardless of their health status. This applies to all PVT modes (standalone, RTK, ...).

Some signals do not transmit a health flag, or the receiver does not decode it. For those signals, the health status is taken from a different signal. See the description of **`setSignalHealthOverride`** for details.

Note that health flags are assumed "healthy" if unknown or if not received yet, except for GPS satellites G33 and above. Those satellites are considered experimental and assumed unhealthy by default.

Example

```
COM1>  ssho, E14+E18, E14+E18<CR>
$R: ssho, E14+E18, E14+E18
  SatelliteHealthOverride, E14+E18, E14+E18
COM1>
```

ssu gsu	**setSatelliteUsage** **getSatelliteUsage**	*Satellite*								
		none + G01 ... G32 + R01 ... R24 + R25 + R26 + R27 + R28 + R29 + R30 + E01 ... E36 + C01 ... C63 + J01 ... J10 + GPS + GLONASS + GALILEO + 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`, `Cxx` and `Rxx` refer to a GPS, Galileo, BeiDou and GLONASS satellite respectively. GLONASS satellites must be referenced by their slot number in this command.

Note that this command only allows the usage of the selected satellites in the PVT. It does not mean that the PVT will effectively use all of them: the PVT engine may only use a subset of the allowed satellites depending on its internal quality checks.

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 exclude the G03 measurements from the PVT, use:

```
COM1>  ssu, -G03 <CR>
$R: ssu, -G03
  SatelliteUsage, G01+G02+G04+G05+G06+G07+G08+G09+G10+G11+G12
  +G13+G14+G15+G16+G17+G18+G19+G20+G21+G22+G23+G24+G25+G26+G27
  ...
CM1>
```

ssbc gsbc	**setSBASCorrections** **getSBASCorrections**	*Satellite*	*SISMode*							
		auto EGNOS WAAS MSAS GAGAN SDCM BDSBAS KASS ANGA SouthPAN S120 ... S158	Test Operational							

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.

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.

Example

```
COM1>  ssbc, S122, Test<CR>
$R: ssbc, S122, Test
  SBASCorrections, S122, Test
COM1>
```

estm [14] gstm	exeSetTime getSetTime	*Format*	*TimeSpec (40)*							
		DateTime								

RxControl: Navigation > Receiver Initialization > Set Trusted Time

Use this command to provide the receiver with the current time from an external non-GNSS source. It allows the receiver to compare its own time, obtained from the GNSS satellites, with the time provided in the *TimeSpec* argument. In case of mismatch, the GNSS signals are considered non-authentic. The time must be provided in the GPS/Galileo time scale, which, in 2025, was running 18 seconds ahead of UTC. It must be accurate to a few seconds.

Entering this command is fully optional, except when Galileo OSNMA authentication is enabled in strict mode (see the **`setGalOSNMAUsage`** command).

Note that this command has no effect on the receiver time keeping described in section 2.3. In particular, it cannot be used to set the receiver time. It is only used to provide a trusted external time stamp, allowing the receiver to validate the time it retrieved from the GNSS satellites.

It is not necessary to enter this command at regular intervals. A single call is enough as the receiver keeps track of the elapsed time since the command was entered.

Example

```
COM1>  estm, DateTime, "2024-12-31 00:00:00"<CR>
$R: estm, DateTime, "2024-12-31 00:00:00"
  SetTime, DateTime, "2024-12-31 00:00:00"
COM1>
```

[14] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

snu [15] gnu	**setSignalUsage** **getSignalUsage**	*PVT*	*NavData*							
		+ GPSL1CA + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL2P + GLOL2CA + GLOL3 + GALE1BC + GALE6BC + GALE5a + GALE5b + GALE5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL6 + QZSL1C + QZSL1S + QZSL1CB + NAVICL5 all	+ GPSL1CA + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL2P + GLOL2CA + GLOL3 + GALE1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL6 + QZSL1C + QZSL1S + QZSL1CB + NAVICL5 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. Note that enabling a signal does not force the PVT to use it; it only means that the PVT is allowed to use it.

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>
```

[15] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sss [16] gss	**setSolutionSelectivity** **getSolutionSelectivity**	*Level*								
		Loose Medium Strict								

RxControl: Navigation > Positioning Mode > PVT Mode

Use this command to adjust the selectivity level of the PVT engine. This level determines how strictly the PVT engine filters satellite signals and transitions between PVT modes. Setting the *Level* argument to `Loose` can enhance the availability of the PVT solution in challenging environments, though it may result in a slightly noisier output compared to the `Medium` or `Strict` levels.

Note that the availability of this command depends on the multi-frequency RTK permission. Use the **`lstInternalFile,permissions`** command to check your permission file.

Example

```
COM1>  sss, Loose<CR>
$R: sss, Loose
  SolutionSelectivity, Loose
COM1>
```

[16] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sspc gspc	**setStaticPosCartesian** **getStaticPosCartesian**	***Position*** *Position*	*X*	*Y*	*Z*	*Datum*				
		+ Cartesian1 + Cartesian2 + Cartesian3 + Cartesian4 + Cartesian5 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+DGNSS+RTKFloat+RTKFixed,
     Cartesian1
COM1>
```

sspg gspg	**setStaticPosGeodetic** **getStaticPosGeodetic**	***Position*** *Position*	*Latitude*	*Longitude*	*Altitude*	*Datum*				
		+ Geodetic1 + Geodetic2 + Geodetic3 + Geodetic4 + Geodetic5 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+DGNSS+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>
```

3.2.6 Authentication

lopk [17]	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>
```

[17] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sopk [18] gopk	**setGalOSNMAPublicKeys** **getGalOSNMAPublicKeys**	***ID*** *ID*	***Key (213)***							
		+ Key0 ... Key15 all								

RxControl: Navigation > Advanced User Settings > Authentication > Galileo OSNMA

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>
```

[18] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sou [19] gou	**setGalOSNMAUsage** **getGalOSNMAUsage**	*PVTLevel*	*MTRoot (65)*	*MeasLevel*						
		off loose strict		off loose						

RxControl: Navigation > Advanced User Settings > Authentication > Galileo OSNMA

Use this command to configure the Galileo OSNMA processing of the receiver.

The *PVTLevel* argument defines the level at which the PVT engine applies OSNMA authentication. If set to `off`, the engine ignores OSNMA authentication checks and may use satellites flagged as non-authentic.

In `loose` mode, 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 computation.

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 need for an additional accurate and independent time source. In `loose` mode, this time source is optional. In `strict` mode, it is required. See also the **`exeSetTime`** 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).

The *MeasLevel* argument specifies whether measurements (pseudoranges, carrier phases, ...) should be produced for satellites for which Galileo authentication failed. When `off`, OSNMA does not affect the generation of measurements. When `loose`, measurements from non-authentic satellites are discarded in all internal processing and in all receiver output streams and log files. Measurements from satellites of which the authenticity could not be determined are not discarded.

Note that satellites excluded at measurement level are de-facto also excluded at PVT level, regardless of the *PVTLevel* setting. In other words, when *MeasLevel* is `loose`, there is no distinction between *PVTLevel*=`off` and *PVTLevel*=`loose`.

Example

```
COM1>  sou, loose,
   AB20435CB64A837C8EACC6FF427416433DFE715F1607D361236A2C163B9E0A5A,
   loose<CR>
$R: sou, loose,
   AB20435CB64A837C8EACC6FF427416433DFE715F1607D361236A2C163B9E0A5A
   , loose
  GalOSNMAUsage, loose,
     AB20435CB64A837C8EACC6FF427416433DFE715F1607D361236A2C163B9E0A5A
     , loose
```

[19] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

```
COM1>
```

sna [20] gna	setSignalAuthentication getSignalAuthentication	*Level*								
		Detection Mitigation								

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

In addition to navigation message authentication (see the **setGalOSNMAUsage** command), the receiver continuously performs a series of autonomous checks to verify the integrity and authenticity of the GNSS signals, with the result reported in the SIG_AUTH_ALERT flag of the RFStatus SBF block. This command sets the operation level of these autonomous checks.

The following values for *Level* are defined:

Level	Description
Detection	Suspicion of non-authentic GNSS signals are flagged by raising the SIG_AUTH_ALERT flag.
Mitigation	In addition to raising the SIG_AUTH_ALERT flag, the receiver attempts to identify the affected signals and excludes them from the PVT solution.

Example

```
COM1>  sna, Mitigation<CR>
$R: sna, Mitigation
  SignalAuthentication, Mitigation
COM1>
```

[20] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

3.2.7 Attitude Determination

sto [21] gto	**setAttitudeOffset** **getAttitudeOffset**	*Heading*	*Pitch*							
		-360.000 ... 0.000 ... 360.000 deg	-90.000 ... 0.000 ... 90.000 deg							

RxControl: Navigation > Positioning Mode > GNSS Attitude

Use this command to specify the offsets that the receiver applies to the computed attitude angles.

The attitude of a vehicle can be determined from the orientation of the baseline between two antennas attached to the vehicle. By default, the receiver determines the attitude angles assuming that the baseline between the antenna ARPs is parallel to the longitudinal axis of the vehicle. Attitude biases appear when this is not the case. The user can use this command to provide the value of the biases, such that the receiver can compensate for them before outputting the attitude.

The receiver subtracts the provided biases from the attitude angles before encoding them in NMEA or in the `AttEuler` SBF block.

Example

```
COM1>  sto, 93.2, -0.4<CR>
$R: sto, 93.2, -0.4
  AttitudeOffset, 93.200, -0.400
COM1>
```

[21] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sga [22] gga	**setGNSSAttitude** **getGNSSAttitude**	*Source*	*MultiAntennaMode*							
		none MovingBase MultiAntenna	+ Float + Fixed							

RxControl: Navigation > Positioning Mode > GNSS Attitude

Use this command to define/inquire the way GNSS-based attitude is computed.

The attitude of a vehicle (more precisely the heading and pitch angles) can be determined from the orientation of the baseline between two antennas attached to the vehicle.

The *Source* argument specifies how to compute the GNSS-based attitude:

Source	Description
`none`	GNSS attitude computation is disabled.
`MovingBase`	Attitude is computed from the baseline between antennas connected to two receivers configured in moving-base RTK operation (moving-base attitude).
`MultiAntenna`	Attitude is computed from the baseline between antennas connected to a single multi-antenna receiver (multi-antenna attitude).

When *Source* is `MultiAntenna`, the second argument specifies which type of attitude solution is allowed for output (fixed or float ambiguities). For highest accuracy, fixed ambiguitiy mode should be selected.

Example

```
COM1>  sga, none, Float<CR>
$R: sga, none, Float
  GNSSAttitude, none, Float
COM1>
```

[22] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

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:

- In standalone or SBAS or PPP mode, 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).
- In DGNSS or RTK mode, the datum depends on the type of differential corrections, as can be found in the `WACorrInfo` field of the `PVTCartesian` and `PVTGeodetic` SBF blocks. For State-Space Representation (SSR) corrections, the coordinates refer to WGS84/ITRF. Otherwise, the coordinates refer to the regional datum of the DGNSS/RTK provider.

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 all WGS84/ITRF coordinates 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 regional DGNSS/RTK provider.

In static PVT mode (see **setPVTMode**) and when the user has specified a position explicitly with the **setStaticPosGeodetic** or the **setStaticPosCartesian** command, the datum specified by the user in those commands is maintained, and the **setGeodeticDatum** command has no effect.

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	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 or TimeSync signal is fed into the receiver. It also has no effect when the receiver is configured in dual-antenna mode.

Refer to section 2.3 for a more detailed description of the time keeping in your receiver.

Example

```
COM1>  scst, msec1, off<CR>
$R: scst, msec1, off
  ClockSyncThreshold, msec1, off
COM1>
```

sep [23] gep	**setEventParameters** **getEventParameters**	***Event*** *Event*	*Polarity*	*Delay*	*CurrentLevel*					
		+ EventA + EventB all	Low2High High2Low	-500.000000 ...0.000000 ...500.000000 ms	(Low) (High)					

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 last argument (*CurrentLevel*) is read-only. It reports the current pin level in the reply of the command.

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 and the attitude angles in the `ExtEventAttEuler` SBF block. Beware that, when using large *Delay* values in high-dynamics conditions, the position accuracy may degrade.

Note that the *Delay* argument has no effect when an event pin is configured in TimeSync mode with the **`setTimeSyncSource`** command.

Example

```
COM1>  sep, EventA, High2Low, 10<CR>
$R: sep, EventA, High2Low, 10
  EventParameters, EventA, High2Low, 10.000000, (Low)
COM1>
```

[23] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sht [24] ght	**setHoldoverTrigger** **getHoldoverTrigger**	*AuthAlert*								
		off on								

RxControl: Navigation > Receiver Operation > Timing

The receiver enters holdover mode during GNSS outages and, if enabled by this command, also when an authentication alert is raised. An authentication alert is raised when the receiver determines that the GNSS signals may not be authentic (see the `SIG_AUTH_ALERT` and the `NAV_MSG_AUTH_ALERT` bits in the `RFStatus` SBF block for details). In critical applications, it may be safer to avoid updating the xPPS position based on potentially misleading GNSS signals. This behaviour can be enabled by setting *AuthAlert* to `on`.

During holdover, the xPPS output is maintained based on the last valid GNSS solution. The maximum duration for which the xPPS output is maintained can be configured independently for the two PPS signals with the *maxHoldOver* argument of the **`setPPSParameters`** and **`setPPS2Parameters`** commands. Once this duration is exceeded, the generation of xPPS pulses is stopped.

This command depends on the `TimeSync` capability (see the **`getReceiverCapabilities`** command).

Example

```
COM1>  sht, on<CR>
$R: sht, on
  HoldoverTrigger, on
COM1>
```

[24] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sps2 gps2	**setPPS2Parameters** **getPPS2Parameters**	*Interval*	*Polarity*	*Delay*	*TimeScale*	*MaxHoldover*	*PulseWidth*			
		MHz10 MHz1 kHz100 kHz10 kHz5 kHz2 kHz1 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	0.000001 . . . 5.000000 . . . 1000.000000 ms			

RxControl: Navigation > Receiver Operation > Timing

Use these commands to define/inquire the parameters of the second x-pulse-per-second (xPPS) output.

The second xPPS output can be configured independently from the first xPPS output, to provide two independent pulse trains. The first xPPS output is configured with the **setPPSParameters** command, and the second xPPS output is configured with this command.

Refer to the **setPPSParameters** command for a description of the arguments.

Example

```
COM1>  sps2, sec1, Low2High, 23.40, GPS, 60, 0.1<CR>
$R: sps2, sec1, Low2High, 23.40, GPS, 60, 0.1
  PPS2Parameters, sec1, Low2High, 23.40, GPS, 60, 0.100000
COM1>
```

spps gpps	**setPPSParameters** **getPPSParameters**	*Interval*	*Polarity*	*Delay*	*TimeScale*	*MaxHoldover*	*PulseWidth*			
		MHz10 MHz1 kHz100 kHz10 kHz5 kHz2 kHz1 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	0.000001 . . . 5.000000 . . . 1000.000000 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 1.15 for additional information on the xPPS functionality.

The *Interval* argument specifies the pulse rate, either as the time interval between pulses, or as a frequency. 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. Note that the *Delay* argument of **`setPPSParameters`** only changes the position of the xPPS pulse with no other effect on the receiver. An alternative way of compensating for signal delays is by using the **`setCalibCommonDelay`** command. With that command, the delay is compensated at pseudorange level, indirectly affecting the position of the xPPS pulse.

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 receiver enters time-holdover mode, in which the xPPS output is maintained based on the last available GNSS measurements. In addition, time-holdover can be entered when non-authentic GNSS signals are suspected (see **`setHoldoverTrigger`**).

The maximum time that the receiver can remain in time-holdover mode is set with the *MaxHoldover* argument. After that time, the generation of xPPS pulses is stopped. If *MaxHoldover* is set to 0, or if *TimeScale* is set to `RxClock`, the xPPS pulses will continue indefinitely.

The *PulseWidth* argument sets the duration of the PPS pulse. Note that the receiver will always stop the pulse shortly before the start of the next pulse, even if the requested *PulseWidth* is longer than the interval between pulses.

Note that, when *Interval* is `off`, the PPS behaves as a general-purpose output. Its level is low if *Polarity* is `Low2High`, and high if it is `High2Low`. The other arguments have no effect.

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, 0.100000
COM1>
```

stss [25] gtss	**setTimeSyncSource** **getTimeSyncSource**	*Source*								
		none EventA EventB								

RxControl: Navigation > Receiver Operation > Timing

Use this command to set which event input pin to be used as TimeSync input. It only takes effect at the next reset or reboot, so do not forget to save the configuration in the boot configuration file with **`exeCopyConfigFile`** after entering the **`setTimeSyncSource`** command.

When an event pin is configured as TimeSync input, the receiver expects to get a one-per-second time pulse on that pin. At startup, it will synchronize its internal time base (i.e. the time at which GNSS measurements are taken) to those pulses, with an offset of a few tens of nanoseconds plus an integer number *k* of milliseconds. *k* is set such that the difference between the internal time base and GNSS time does not exceed the limit set with the **`setClockSyncThreshold`** command (0.5 ms by default). An `ExtEvent` SBF block is generated for each pulse, and *k* can be obtained from the `TOW` and the `Offset` fields of that SBF block, using the formula:

k = round(round(`TOW`*0.001 + `Offset`) * 1000 - (`TOW` + `Offset`*1000)).

If *k* is positive, the receiver time is in advance of the time pulses by *k* milliseconds. If it is negative, it lags the time pulses by *k* milliseconds.

The expected time pulse polarity (rising or falling edge) is set with the **`setEventParameters`** command.

A typical use case of the TimeSync input is in conjunction with the REF IN input: the receiver is fed with a 10 MHz frequency reference on its REF IN input, and to a one-per-second time pulse derived from the same frequency reference on its TimeSync input.

Example

```
COM1>  stss, EventB<CR>
$R: stss, EventB
  TimeSyncSource, EventB
COM1>
```

[25] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

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. In particular, choosing a specific time system can prevent the PVT engine from entering PPP mode (see **`setPVTMode`**), as PPP clock corrections do not always relate to a GNSS time system.

Example

```
COM1>  sts, GPS<CR>
$R: sts, GPS
  TimingSystem, GPS
COM1>
```

3.2.10 Station Settings

smp gmp	**setMarkerParameters** **getMarkerParameters**	*MarkerName (60)*	*MarkerNumber (20)*	*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`**) 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>
```

3.2.11 General Input/Output

lad	lstAsciiDisplay									

Use this command to output the current ASCII Display. The ASCII Display is a textual report of the tracking and PVT status. It can be used to get a quick overview of the receiver operation.

This command outputs a single ASCII Display. Use the **`setDataInOut`** command to enable the ASCII Display at a 1-Hz rate.

Example

```
COM1>  lstAsciiDisplay <CR>
$R; lstAsciiDisplay
---->
$-- BLOCK 1 / 1
$TD
# -- TRACKING 12 sats ------------------------------------------
#Ch PRN CorAg El     Res    DRes|Sg    LT C/N0 |Sg    LT C/N0
#         [s] [d]    [m]   [m/s]|     [s][dBHz]|     [s][dBHz]
#                               |   ---MAIN--- |   ---MAIN---
# 1 G30        7+  -0.43 +0.009|CA  2589 33.3 |2P  2568 20.5
#
# ...
#
# -- PVT: StandAlone ---------- 10 sats (10G, 0R, 0E, 0C, 0J) ---
# POS N 50o50'55.121" E   4o43'55.715" H +127.989 m (datum:WGS84)
# VEL N   +0.000 m/s  E    +0.000 m/s  U   +0.000 m/s
# CLK B +113.415 usec D  -346.038 ppb  ADEV1s 1E-10  GPS
# DOP P    1.822      H     1.014      V    1.513
# -- RX INFO ------------------ 12 sats/ 40 sigs TRACKING -------
# 08:49:50  16 Jun 2025 - sec:118190 week:2371 - uptime:  2d22:05
# CPU: 16% (tracking:  5%)         INTLTCY:  5ms    RX: 45oC
# ERROR: none
# RECEIVER:                   (SEPT) - firmware: 0.0.0
COM1>
```

scs gcs	**setCOMSettings** **getCOMSettings**	***Cd*** *Cd*	*Rate*	*DataBits*	*Parity*	*StopBits*	*FlowControl*			
		+ COM1 + COM2 all	baud1200 baud2400 baud4800 baud9600 baud19200 baud38400 baud57600 baud115200 baud230400 baud460800 baud500000 baud576000 baud921600 baud1000000 baud1152000 baud1500000 baud2000000 baud2500000 baud3000000 baud3500000 baud4000000	bits8	No	bit1	none RTS|CTS			

RxControl: Communication > Serial 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>
```

sdcm gdcm	**setDaisyChainMode** **getDaisyChainMode**	***DC*** *DC*	*Mode*							
		+ DC1 + DC2 all	Raw ASCII RTCMv3							

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. Similarly, if *Mode* is set to `RTCMv3`, recognized RTCMv3 messages are transmitted at once.

Example

```
COM1>  sdcm, DC1, ASCII<CR>
$R: sdcm, DC1, ASCII
  DaisyChainMode, DC1, ASCII
COM1>
```

sdio [26] gdio	**setDataInOut** **getDataInOut**	***Cd*** *Cd*	*Input*	*Output*	*Show*					
		+ DSK1 + COM1 + COM2 + USB1 + USB2 all	none CMD RTCMv3 CMRv2 DC1 DC2 ASCIIIN auto	none + RTCMv3 + SBF + NMEA + ASCIIDisplay + DC1 + DC2 + LBandBeam1 + LBandBeam2 + LBandBeam3 + LBandBeam4 + LBandBeam5	(off) (on) (waiting)					

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 1.2.3).

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, 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. `ASCIIIN` is used for connections receiving free-formatted ASCII messages, e.g. from an external meteo sensor.

In `auto` mode, the receiver automatically detects user commands and incoming differential corrections in RTCM or CMR format. Other types of input 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).

`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.

[26] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

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 1.2.3 for details.

Note that not all input connections can accept user commands, check section 1.2.3 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>
```

eecm gecm	exeEchoMessage getEchoMessage	*Cd*	*Message (242)*	*EndOfLine*						
		COM1 COM2 USB1 USB2 DC1 DC2	A:Unknown	none + CR + LF all						

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>
```

spe gpe	**setPeriodicEcho** **getPeriodicEcho**	***Cd*** *Cd*	*Message (201)*	*Interval*						
		+ COM1 + COM2 all	A:Unknown	off once msec100 msec200 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>
```

ssts gsts	**setSerialTlsSettings** **getSerialTlsSettings**	***Cd*** *Cd*	***Enable***							
		+ COM1 + COM2 + USB1 + USB2 all	off on							

RxControl: Communication > Serial Port Settings

Use these commands to define/inquire if the receiver should use TLS on a COM or USB port.

The *Enable* argument is used to tell the receiver if TLS is switched on for the connection *Cd*.

In `on` mode, the receiver will only accept/send data on the connection *Cd* if there is an active TLS connection.

3.2.12 NMEA Configuration

enoc [27] gnoc	**exeNMEAOnce** **getNMEAOnce**	***Cd***	***Messages***							
		DSK1 COM1 COM2 USB1 USB2	+ ALM + DTM + GBS + GGA + GLL + GNS + GRS + GSA + GST + GSV + HDT + RMC + ROT + VTG + ZDA + HRP + RBP + RBV + RBD + AVR + GGAaux1 + GGK + GFA + GGQ + TFM + THS							

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 1.2.3) 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>
```

[27] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

sno [28] gno	**setNMEAOutput** **getNMEAOutput**	***Stream*** *Stream*	*Cd*	*Messages*	*Interval*					
		+ Stream1 ... Stream10 all	none DSK1 COM1 COM2 USB1 USB2	none + ALM + DTM + GBS + GGA + GLL + GNS + GRS + GSA + GST + GSV + HDT + RMC + ROT + VTG + ZDA + HRP + RBP + RBV + RBD + AVR + GGAaux1 + GGK + GFA + GGQ + TXTbase + TFM + THS	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 1.2.3) 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
```

[28] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

```
  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 . . . 2 . . . 3	Nominal Mode1 Mode2	off	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 has no effect.

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 "GB" 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>
```

3.2.13 SBF Configuration

ssgp [29] gsgp	**setSBFGroups** **getSBFGroups**	***Group*** *Group*	*Messages*							
		+ Group1 + Group2 + Group3 + Group4 all	none [SBF List] + Measurements + RawNavBits + GPS + GLO + GAL + GEO + BDS + QZS + NavIC + PVTCart + PVTGeod + PVTExtra + Attitude + Time + Events + DiffCorr + Status + LBand + PostProcess + Rinex + 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 B.

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 1Hz, 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>
```

[29] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

esoc [30] gsoc	exeSBFOnce getSBFOnce	*Cd*	*Messages*							
		DSK1 COM1 COM2 USB1 USB2	[SBF List] + Measurements + GPS + GLO + GAL + GEO + BDS + QZS + NavIC + PVTCart + PVTGeod + PVTExtra + Attitude + Time + Status + LBand + UserGroups + PostProcess + Rinex + 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 1.2.3) 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 B.

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
`GPS`	+GPSNav +GPSCNav +GPSAlm +GPSIon +GPSUtc
`GLO`	+GLONav +GLOAlm +GLOTime
`GAL`	+GALNav +GALAlm +GALIon +GALUtc +GALGstGps
`GEO`	+GEONav +GEOAlm
`BDS`	+BDSAlm +BDSNav +BDSCNav1 +BDSCNav2 +BDSCNav3 +BDSIon +BDSUtc
`QZS`	+QZSNav +QZSAlm
`NavIC`	+NavICLNav

[30] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

Messages (Continued)	Description
PVTCart	+PVTCartesian +PosCovCartesian +VelCovCartesian +BaseVectorCart +NavCart
PVTGeod	+PVTGeodetic +PosCovGeodetic +VelCovGeodetic +BaseVectorGeod +NavGeod
PVTExtra	+DOP +PVTSupport +PVTSupportA +EndOfPVT
Attitude	+AuxAntPositions +AttEuler +AttCovEuler +EndOfAtt
Time	+ReceiverTime
Status	+SatVisibility +ChannelStatus +ReceiverStatus +InputLink +OutputLink +QualityInd +DiskStatus +RFStatus
LBand	+LBandTrackerStatus
UserGroups	+Group1 +Group2 +Group3 +Group4
PostProcess	+MeasEpoch +MeasExtra +GPSNav +GPSCNav +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 +BDSCNav1 +BDSCNav2 +BDSCNav3 +QZSNav +NavICLNav +PVTGeodetic +ReceiverSetup +Comment
Support	+MeasEpoch +MeasExtra +EndOfMeas +GPSNav +GPSCNav +GPSAlm +GPSIon +GPSUtc +GLONav +GLOAlm +GLOTime +GALNav +GALAlm +GALIon +GALUtc +GALGstGps +GEONav +GEOAlm +BDSAlm +BDSNav +BDSCNav1 +BDSCNav2 +BDSCNav3 +BDSIon +BDSUtc +QZSNav +PVTGeodetic +PosCovGeodetic +BaseVectorGeod +AuxAntPositions +AttEuler +DOP +PVTSupport +PVTSupportA +EndOfPVT +ChannelStatus +ReceiverStatus +InputLink +OutputLink +ReceiverSetup +Commands +RxMessage +LBandTrackerStatus +QualityInd +DiskStatus +RFStatus

Example

To output the next MeasEpoch block, use:

```
COM1>  esoc, COM1, MeasEpoch <CR>
$R: esoc, COM1, MeasEpoch
  SBFOnce, COM1, MeasEpoch
COM1>
```

sso [31] gso	setSBFOutput getSBFOutput	*Stream* *Stream*	*Cd*	*Messages*	*Interval*					
		+ Stream1 ... Stream10 + Res1 + Res2 + Res3 + Res4 all	none DSK1 COM1 COM2 USB1 USB2	none [SBF List] + Measurements + RawNavBits + GPS + GLO + GAL + GEO + BDS + QZS + NavIC + PVTCart + PVTGeod + PVTExtra + Attitude + Time + Event + DiffCorr + Status + LBand + UserGroups + PostProcess + Rinex + 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 1.2.3). 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 B.

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
`RawNavBits`	+GPSRawCA +GPSRawL2C +GPSRawL5 +GPSRawL1C +GLORawCA +GALRawFNAV +GALRawINAV +GALRawCNAV +GEORawL1 +GEORawL5 +BDSRaw +BDSRawB1C +BDSRawB2a +BDSRawB2b +NAVICRaw +QZSRawL1CA +QZSRawL2C +QZSRawL5 +QZSRawL6D +QZSRawL6E +QZSRawL1C +QZSRawL1S +QZSRawL5S
`GPS`	+GPSNav +GPSCNav +GPSAlm +GPSIon +GPSUtc
`GLO`	+GLONav +GLOAlm +GLOTime
`GAL`	+GALNav +GALAlm +GALIon +GALUtc +GALGstGps +GALSARRLM
`GEO`	+GEONav +GEOAlm +GEOIGPMask +GEOIonoDelay
`BDS`	+BDSAlm +BDSNav +BDSCNav1 +BDSCNav2 +BDSCNav3 +BDSIon +BDSUtc

[31] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

Messages (Continued)	Description
QZS	+QZSNav +QZSAlm
NavIC	+NavICLNav
PVTCart	+PVTCartesian +PosCovCartesian +VelCovCartesian +BaseVectorCart +NavCart
PVTGeod	+PVTGeodetic +PosCovGeodetic +VelCovGeodetic +BaseVectorGeod +NavGeod
PVTExtra	+DOP +PVTSupport +PVTSupportA +EndOfPVT
Attitude	+AuxAntPositions +AttEuler +AttCovEuler +EndOfAtt
Time	+ReceiverTime +xPPSOffset
Event	+ExtEvent +ExtEventPVTCartesian +ExtEventPVTGeodetic +ExtEventBaseVectGeod +ExtEventAttEuler
DiffCorr	+DiffCorrIn +BaseStation
Status	+SatVisibility +ChannelStatus +ReceiverStatus +InputLink +OutputLink +QualityInd +DiskStatus +RFStatus +GALAuthStatus
LBand	+LBandTrackerStatus +LBandRaw
UserGroups	+Group1 +Group2 +Group3 +Group4
PostProcess	+MeasEpoch +MeasExtra +GALRawCNAV +GEORawL1 +GPSNav +GPSCNav +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 +BDSCNav1 +BDSCNav2 +BDSCNav3 +QZSNav +NavICLNav +PVTGeodetic +ReceiverSetup +Comment
Support	+MeasEpoch +MeasExtra +EndOfMeas +GPSRawCA +GPSRawL2C +GPSRawL5 +GPSRawL1C +GLORawCA +GALRawFNAV +GALRawINAV +GALRawCNAV +GEORawL1 +GEORawL5 +BDSRaw +BDSRawB1C +BDSRawB2a +BDSRawB2b +NAVICRaw +QZSRawL1CA +QZSRawL2C +QZSRawL5 +QZSRawL6D +QZSRawL6E +QZSRawL1C +QZSRawL1S +QZSRawL5S +GPSNav +GPSAlm +GPSIon +GPSUtc +GLONav +GLOAlm +GLOTime +GALNav +GPSCNav +GALAlm +GALIon +GALUtc +GALGstGps +GALAuthStatus +GEONav +GEOAlm +BDSAlm +BDSNav +BDSCNav1 +BDSCNav2 +BDSCNav3 +BDSIon +BDSUtc +QZSNav +QZSAlm +PVTGeodetic +PosCovGeodetic +BaseVectorGeod +AuxAntPositions +AttEuler +DOP +PVTSupport +PVTSupportA +EndOfPVT +ExtEvent +DiffCorrIn +BaseStation +ChannelStatus +ReceiverStatus +InputLink +OutputLink +ReceiverSetup +Commands +RxMessage +LBandTrackerStatus +LBandRaw +QualityInd +DiskStatus +RFStatus

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 B (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>
```

3.2.14 RTCM v3.x Settings

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)*	*MaxMSMSats*				
		0 . . . 4095	+ GPSL1CA + GPSL2PY + GPSL2C + GPSL5 + GPSL1C + GLOL1CA + GLOL2P + GLOL2CA + GLOL3 + GALE1BC + GALE6BC + GALE5a + GALE5b + GALE5 + GEOL1 + GEOL5 + BDSB1I + BDSB2I + BDSB3I + BDSB1C + BDSB2a + BDSB2b + QZSL1CA + QZSL2C + QZSL5 + QZSL1C + QZSL1CB + NAVICL5 all	L2CA L2P	default	1 . . . 128				

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 *MaxMSMSats* argument specifies the maximum number of satellites to be encoded in the MSM messages of the same epoch.

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,
   GPSL1CA+GPSL2PY+GLOL1CA+GLOL2CA+GALE1BC+GALE5b+BDSB1I+BDSB3I,
   L2CA, default, 28<CR>
```

```
$R: sr3f, 345, GPSL1CA+GPSL2PY+GLOL1CA+GLOL2CA+GALE1BC+GALE5b+
   BDSB1I+BDSB3I, L2CA, default, 28
  RTCMv3Formatting, 345, GPSL1CA+GPSL2PY+GLOL1CA+GLOL2CA+GALE1BC+
     GALE5b+BDSB1I+BDSB3I, L2CA, default, 28
COM1>
```

sr3i gr3i	**setRTCMv3Interval** **getRTCMv3Interval**	***Message*** *Message*	*Interval*							
		+ 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 + USB1 + USB2 all	none + RTCM1001 + RTCM1002 + RTCM1003 + RTCM1004 + RTCM1005 + RTCM1006 + RTCM1007 ... RTCM1013 + RTCM1019 + RTCM1020 + RTCM1029 + RTCM1033 + RTCM1041 + RTCM1042 + RTCM1044 + RTCM1045 + RTCM1046 + RTCM1071 + RTCM1072 + RTCM1073 + RTCM1074 + RTCM1075 + RTCM1076 + RTCM1077 + RTCM1081 + RTCM1082 + RTCM1083 + RTCM1084 + RTCM1085 + RTCM1086 + RTCM1087 + RTCM1091 + RTCM1092 + RTCM1093 + RTCM1094 + RTCM1095 + RTCM1096 + RTCM1097 + RTCM1101 ... RTCM1107 + RTCM1111 ... RTCM1117 + RTCM1121 + RTCM1122 + RTCM1123 + RTCM1124 + RTCM1125 + RTCM1126 + RTCM1127 + RTCM1131 ... RTCM1137 + RTCM1230 + MSM1 + MSM2 + MSM3 + MSM4 + MSM5 + MSM6 + MSM7 + Nav 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 1.2.3). 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. `Nav` enables the Ephemeris Data messages for all constellations (RTCM1019, RTCM1020, ...).

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 + RTCM1019 + RTCM1020 + RTCM1029 + RTCM1033 + RTCM1041 + RTCM1042 + RTCM1044 + RTCM1045 + RTCM1046 + RTCM1071 ... RTCM1077 + RTCM1081 ... RTCM1087 + RTCM1091 ... RTCM1097 + RTCM1111 ... RTCM1117 + RTCM1121 ... RTCM1127 + RTCM1230 + MSM1 + MSM2 + MSM3 + MSM4 + MSM5 + MSM6 + MSM7 + Nav 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. `Nav` is an alias for all the Ephemeris Data messages (RTCM1019, RTCM1020, ...).

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>
```

3.2.15 CMR Settings

sc2u gc2u	**setCMRv2Usage** **getCMRv2Usage**	*MsgUsage*								
		none + CMR0 + CMR1 + CMR2 + CMR3 + CMR0p + CMR0w all								

RxControl: Communication > Input Settings > Differential Corrections > CMR

Use this command to restrict the list of incoming CMR 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 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>
```

3.2.16 Internal Disk Logging

sdfa gdfa	**setDiskFullAction** **getDiskFullAction**	***Disk*** *Disk*	*Action*							
		+ DSK1 all	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.

The currently supported actions are as follows:

Action	Description
`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 all								

Use this command to retrieve information about the disk identified by the *Disk* argument. 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)*						
		+ DSK1 all	FileName Incremental	log						

RxControl: Logging > Internal Logging Settings > SBF Logging and Upload

Use these commands to define/inquire the file naming convention for the internal SBF, NMEA or user-message log files.

If *NamingType* is `FileName`, the file name is given by the third argument *FileName*, followed by the extension `.sbf`, `.nma` or `.ecm` for SBF, NMEA and user-message files respectively. User-message files contain messages entered by the command **`exeEchoMessage`** prefixed with the GPS week number and time of week in seconds.

If *NamingType* is `Incremental`, the file name is given by the first five characters of the *FileName* argument (right padded with "_" if necessary), followed by a modulo-1000 counter incrementing each time logging is stopped and restarted. The file name extension is `.sbf`, `.nma` or `.ecm` as described above. If the auto-incremented file name already exists on the disk, the receiver takes action as specified by the **`setDiskFullAction`** command.

The set of allowed characters for the *FileName* argument is:

`_0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`

Note that the actual file name on the disk is case insensitive and only contains lower-case characters even if the user entered upper-case characters in the *FileName* argument.

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`", use:

```
COM1>  sfn, DSK1, FileName, mytest <CR>
$R: sfn, DSK1, FileName, mytest
  FileNaming, DSK1, FileName, "mytest"
COM1>
```

erf grf	exeRemoveFile getRemoveFile	*Disk*	*FileName (200)*							
		DSK1	none all							

RxControl: Logging > Remove Internal File

Use this command to remove one file or an entire directory (if your receiver supports directories, see for example IGS file naming mode (see the **`setFileNaming`** command)) from the disk identified by the *Disk* argument.

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 when supported.

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>
```

3.2.17 MSS/L-Band Configuration

slbb [32] glbb	**setLBandBeams** **getLBandBeams**	***Beam*** *Beam*	*Frequency*	*Rate*	*Name (8)*	*Region (8)*	*Usage*			
		+ User1 + User2 + User3 + User4 + User5 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>
```

[32] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

slcs [33] glcs	**setLBandCustomServiceID** **getLBandCustomServiceID**	*ServiceID (4)*	*ScramblingVector (4)*	*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>
```

[33] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

slsm [34] glsm	setLBandSelectMode getLBandSelectMode	*Mode*	*Service*	*Beam1*	*Beam2*	*Beam3*	*Beam4*	*Beam5*		
		off manual	Inmarsat	User1 User2 User3 User4 User5	User1 User2 User3 User4 User5	User1 User2 User3 User4 User5	User1 User2 User3 User4 User5	User1 User2 User3 User4 User5		

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
`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. 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, User2, User1,
   User1<CR>
$R: slsm, manual, Inmarsat, User1, User2, User2, User1, User1
  LBandSelectMode, manual, Inmarsat, User1, User2, User2, User1,
     User1
COM1>
```

[34] This command and/or some of its arguments depend on the enabled capabilities of your receiver. See section 1.1 on how to check the capabilities

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 B.

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 startup, 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 B.

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 SBF Satellite Numbering

In SBF, satellites are numbered according to the table below, unless specified otherwise in the block description.

SBF sat nr	Description	RINEX sat code
0	Do-Not-Use value	NA
1-37	PRN number of a GPS satellite	G01 to G37
38-61	Slot number of a GLONASS satellite with an offset of 37	R01 to R24
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
71-106	PRN number of a GALILEO satellite with an offset of 70	E01 to E36
107-119	LBand (MSS) satellite. Corresponding satellite name can be found in the LBandBeams block.	NA
120-140	PRN number of an SBAS satellite with an offset of 100	S20 to S40
141-180	PRN number of a BeiDou satellite with an offset of 140	C01 to C40
181-190	PRN number of a QZSS satellite with an offset of 180	J01 to J10
191-197	PRN number of a NavIC/IRNSS satellite with an offset of 190	I01 to I07
198-215	PRN number of an SBAS satellite with an offset of 157	S41 to S58
216-222	PRN number of a NavIC/IRNSS satellite with an offset of 208	I08 to I14
223-245	PRN number of a BeiDou satellite with an offset of 182	C41 to C63
250-251	PRN number of a GPS satellite with an offset of 212	G38 to G39

To ensure compatibility with future SBF upgrades, decoding software must ignore SBF blocks and sub-blocks containing a satellite number that is not defined in the above table.

4.1.10 Signal Type

Some sub-blocks contain a signal type field, which identifies the type of signal and modulation the sub-block applies to. The signal numbering is defined in the following table.

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+*FN**9/16, with *FN* the frequency number (-7 to +6)	1C
9	L1P	GLONASS	1602.00+*FN**9/16	1P
10	L2P	GLONASS	1246.00+*FN**7/16	2P
11	L2CA	GLONASS	1246.00+*FN**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
17	E1	Galileo	1575.42	1C
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
32	L1C	QZSS	1575.42	1L
33	L1S	QZSS	1575.42	1Z
34	B2b	BeiDou	1207.14	7D
37	L1	NavIC/IRNSS	1575.42	1P
38	L1CB	QZSS	1575.42	1E
39	L5S	QZSS	1176.45	5P

(*) See the `E6B Used` bit in the `MeasEpoch` SBF block.

To ensure compatibility with future SBF upgrades, decoding software must ignore SBF blocks and sub-blocks containing a signal number that is not defined in the above table.

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

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 startup, 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: Half-Cycle: 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: Half-Cycle: 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 pseudo-range, 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 preprocessing, 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, modulo 256. Use the following formula to compute the number of sub-blocks from the N, Length and SBLength fields: NrSB = ((Length / SBLength - N) / 256) * 256 + N where the divisions are integer divisions.
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: $\sigma^2_{\text{Doppler}}[\text{mHz}^2]$ = CarrierVariance * DopplerVarFactor, where CarrierVariance can be found for each measurement type in the MeasExtraChannelSub sub-blocks.
ChannelSub	...	...		*A succession of NrSB 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 startup, and is incremented at each initial lock after signal (re)acquisition, or when a cycle slip is detected.
Rev 1	`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: Bit 0: Reserved. Bit 1: Reserved. Bits 2-7: 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

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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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. The receiver does not output navigation frames of which the BCH code in subframe 1 is invalid, so the validity of subframe 1 does not need to be checked.

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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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. For example, a value of 1 corresponds to frequency number -7.
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 E1BC. In that case, bits 0-4 are set to E1BC. 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. The receiver does not output navigation frames of which the BCH codes in subframe 1 are invalid, so the validity of subframe 1 does not need to be checked.

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

QZSRawL6D	Number: 4270 "OnChange" interval: 1s

This block contains the 2000 bits of a QZSS L6D 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. Always set to 1 (L6D).
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

QZSRawL6E	Number: 4271 "OnChange" interval: 1s

This block contains the 2000 bits of a QZSS L6E 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. Always set to 2 (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. The receiver does not output navigation frames of which the BCH code in subframe 1 is invalid, so the validity of subframe 1 does not need to be checked.

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 L5 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 L5 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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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 ionospheric parameters are received from a GPS satellite

The GPSIon block contains ionosphere data (the Klobuchar coefficients) decoded from GPS satellites. Refer to the GPS ICDs 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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 4.1.9.
Type	u1			Bit field: Bits 0-3: Signal type from which these coefficients have been decoded: 0: Unknown 1: L1CA 2: L2C 3: L5 4: L1C Bits 4-7: Reserved
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			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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, as transmitted on the L2C and/or L5 signals. The ephemeris and health parameters are extracted from message types (MT) 10 and 11. The clock and the inter-signal corrections 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	
PRNidx	u1			PRN number of the satellite in the GPS constellation (1 for G01, 2 for G02, etc...). **Warning**: this is the index within the constellation and not the global PRN number defined in 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, 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	$-2 \cdot 10^{10}$	Group delay differential (T_{GD}, 13 bits from MT30)
ISC_L1CA	f4	1 s	$-2 \cdot 10^{10}$	Inter-Signal Correction for L1C/A ($ISC_{L1C/A}$, 13 bits from MT30)
ISC_L2C	f4	1 s	$-2 \cdot 10^{10}$	Inter-Signal Correction for L2C (ISC_{L2C}, 13 bits from MT30)
ISC_L5I5	f4	1 s	$-2 \cdot 10^{10}$	Inter-Signal Correction for L5I (ISC_{L5I5}, 13 bits from MT30)
ISC_L5Q5	f4	1 s	$-2 \cdot 10^{10}$	Inter-Signal Correction for L5Q (ISC_{L5Q5}, 13 bits from MT30)
Padding	u1[..]			Padding bytes, see 4.1.5

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, with an offset of 8. For example, a value of 1 corresponds to frequency number -7.
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		$\gamma_n(t_b)$:relative deviation of predicted carrier frequency
tau	f4	1 s		$\tau_n(t_b)$: time correction to GLONASS time
dtau	f4	1 s		$\Delta\tau_n$: 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 t_b
P2	u1			1-bit odd/even flag of 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 t_b
Rev 1 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, with an offset of 8. For example, a value of 1 corresponds to frequency number -7.
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		$\Delta\ i_n^A$: correction to inclination
lambda	f4	1 semi-circle		λ_n^A: Longitude of first ascending node
t_ln	f4	1 s		$t_{\lambda\ n}^A$: time of first ascending node passage
omega	f4	1 semi-circle		ω_n^A: argument of perigee
Delta_T	f4	1 s / orbit-period		$\Delta\ T_n^A$: correction to mean Draconian period
dDelta_T	f4	1 s / orbit-period^2		$d\Delta\ T_n^A$: 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, with an offset of 8. For example, a value of 1 corresponds to frequency number -7.
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_{DVS}, L1-B_{HS}, E5b_{DVS},E5b_{HS}, SISA_L1E5b and BGD_L1E5b are available
16 (F/NAV)	At least E5a_{DVS},E5a_{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_{DVS} Bits 2-3: 2-bit L1-B_{HS} Bit 4: If set, bits 5 to 7 are valid, otherwise they must be ignored. Bit 5: 1-bit E5b_{DVS} Bits 6-7: 2-bit E5b_{HS} Bit 8: If set, bits 9 to 11 are valid, otherwise they must be ignored. Bit 9: 1-bit E5a_{DVS} Bits 10-11: 2-bit E5a_{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 and L1-B signals: Bit 0: If set, bits 1 and 2 are valid, otherwise they must be ignored. Bits 1-2: 2-bit L1-B_{HS} Bit 3: If set, bits 4 and 5 are valid, otherwise they must be ignored. Bits 4-5: 2-bit E5b_{HS} Bit 6: If set, bits 7 and 8 are valid, otherwise they must be ignored. Bits 7-8: 2-bit E5a_{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	$-2 \cdot 10^{10}$	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

BDSCNav1	Number: 4251 "OnChange" interval: block generated when a new data set is received

The BDSCNav1 block contains the B-CNAV1 navigation data decoded from the B1C 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 (5-bit raw value without sign interpretation)

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 (5-bit raw value without sign interpretation)
SISMAI	u1			Signal in space monitoring accuracy index
HealthIF	u1			Health and integrity flags: Bit 0: Accuracy integrity flag ($AIF_{(B1C)}$) Bit 1: Signal integrity flag ($SIF_{(B1C)}$) Bit 2: Data integrity flag ($DIF_{(B1C)}$) Bits 3-5: Reserved Bits 6-7: Satellite health status (0 if healthy)
IODE	u1			Issue Of Data Ephemeris
IODC	u2			Issue Of Data Clock
ISC_B1Cd	f4	1 s		Group delay differential between the B1C data and pilot components
T_GDB1Cp	f4	1 s		Group delay differential of the B1C pilot component
T_GDB2ap	f4	1 s		Group delay differential of the B2a pilot component
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 (5-bit raw value without sign interpretation)

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 (5-bit raw value without sign interpretation)
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	$-2 \cdot 10^{10}$	Group delay differential between the B2a data and pilot components
T_GDB2ap	f4	1 s	$-2 \cdot 10^{10}$	Group delay differential of the B2a pilot component
T_GDB1Cp	f4	1 s	$-2 \cdot 10^{10}$	Group delay differential of the B1C pilot component
Padding	u1[..]			Padding bytes, see 4.1.5

BDSCNav3	Number: 4253 "OnChange" interval: block generated when a new data set is received

The BDSCNav3 block contains the B-CNAV3 navigation data decoded from the B2b_I 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 (5-bit raw value without sign interpretation)

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 (5-bit raw value without sign interpretation)
SISMAI	u1			Signal in space monitoring accuracy index
HealthIF	u1			Health and integrity flags: Bit 0: B2b accuracy integrity flag (AIF_I) Bit 1: B2b signal integrity flag (SIF_I) Bit 2: B2b data integrity flag (DIF_I) Bits 3-5: Reserved Bits 6-7: Satellite health status (0 if healthy)
Reserved	u1[3]			Reserved for future use, to be ignored by decoding software
T_GDB2bI	f4	1 s		Group delay differential of the B2b_I signal
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			This is a bit field containing bits 71 and 72 from subframe 1: Bit 0: Bit 72 of subframe 1 (Ephemeris Status Flag). Bit 1: Bit 71 of subframe 1 (Reserved). Bits 2-7: Reserved Note that these bits were previously (in QZSS ICD is-qzss-pnt-004 and earlier) defined as "C/A or P on L2".
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

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

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

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

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 9: Reserved 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
Undulation	f4	1 m	$-2\cdot10^{10}$	Geoid undulation. See the **setGeoidUndulation** command.
Vx	f4	1 m / s	$-2\cdot10^{10}$	Velocity in the X direction
Vy	f4	1 m / s	$-2\cdot10^{10}$	Velocity in the Y direction
Vz	f4	1 m / s	$-2\cdot10^{10}$	Velocity in the Z 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 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: $t_{sys} = t_{rx}$ - 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 to which the offset is provided in this sub-block: 0: GPS time 1: Galileo time (GST) 3: GLONASS time 4: BeiDou time 5: QZSS time 100: Fugro AtomiChron 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 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 5-6: Type of RTK or DGNSS differential corrections: 0: unknown or not in differential positioning mode (DGNSS or RTK) 1: corrections from a physical base 2: corrections from a virtual base (VRS) 3: SSR corrections (RTK using SSR corrections is often referred to as RTK-SSR or PPP-RTK) Bit 7: Reserved
ReferenceID	u2		65535	This field indicates the reference ID of the differential information used. In case of DGNSS 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 DGNSS 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.

Rev 3	AlertFlag	u1			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 Bit 5: SIG_AUTH_ALERT: see description in the Flags field of the RFStatus block Bit 6: NAV_MSG_AUTH_ALERT: see description in the Flags field of the RFStatus block Bit 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 DGNSS 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.
Rev 2	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 (PCO) and/or the phase center variation (PCV) are compensated at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating which position is reported in this block: the APC, the ARP or the marker (see **setAntennaOffset** command) 0: Unknown 1: The ARP is reported (the ARP-to-marker offset is zero) 2: The marker is reported (the ARP-to-marker offset is not zero) 3: APC is reported. This value is used if the *APCOutput* argument of the **setAntennaOffset** command is available and is set to on.
	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 9: Reserved 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 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: $t_{sys} = t_{rx}$ - 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 to which the offset is provided in this sub-block: 0: GPS time 1: Galileo time (GST) 3: GLONASS time 4: BeiDou time 5: QZSS time 100: Fugro AtomiChron 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 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 5-6: Type of RTK or DGNSS differential corrections: 0: unknown or not in differential positioning mode (DGNSS or RTK) 1: corrections from a physical base 2: corrections from a virtual base (VRS) 3: SSR corrections (RTK using SSR corrections is often referred to as RTK-SSR or PPP-RTK) Bit 7: Reserved
ReferenceID	u2		65535	This field indicates the reference ID of the differential information used. In case of DGNSS 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 DGNSS 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.

Rev 3	AlertFlag	u1			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 Bit 5: SIG_AUTH_ALERT: see description in the `Flags` field of the `RFStatus` block Bit 6: NAV_MSG_AUTH_ALERT: see description in the `Flags` field of the `RFStatus` block Bit 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 DGNSS 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.
Rev 2	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 (PCO) and/or the phase center variation (PCV) are compensated at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating which position is reported in this block: the APC, the ARP or the marker (see **`setAntennaOffset`** command) 0: Unknown 1: The ARP is reported (the ARP-to-marker offset is zero) 2: The marker is reported (the ARP-to-marker offset is not zero) 3: APC is reported. This value is used if the *APCOutput* argument of the **`setAntennaOffset`** command is available and is set to `on`.
	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.

Note that the availability of the off-diagonal elements depends on the "Integrity" capability (see section 1.1).

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 9: Reserved 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:

$$\begin{pmatrix} \sigma_\phi^2 & \sigma_{\phi\lambda} & \sigma_{\phi h} & \sigma_{\phi b} \\ \sigma_{\lambda\phi} & \sigma_\lambda^2 & \sigma_{\lambda h} & \sigma_{\lambda b} \\ \sigma_{h\phi} & \sigma_{h\lambda} & \sigma_h^2 & \sigma_{hb} \\ \sigma_{b\phi} & \sigma_{b\lambda} & \sigma_{bh} & \sigma_b^2 \end{pmatrix}$$

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.

Note that the availability of the off-diagonal elements depends on the "Integrity" capability (see section 1.1).

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 9: Reserved 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	$-2 \cdot 10^{10}$	Variance of the latitude estimate
Cov_lonlon	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of the longitude estimate
Cov_hgthgt	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of the height estimate
Cov_bb	f4	1 m^2	$-2 \cdot 10^{10}$	Variance of the clock-bias estimate
Cov_latlon	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the latitude and longitude estimates
Cov_lathgt	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the latitude and height estimates
Cov_latb	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the latitude and clock-bias estimates
Cov_lonhgt	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the longitude and height estimates
Cov_lonb	f4	1 m^2	$-2 \cdot 10^{10}$	Covariance between the longitude and clock-bias estimates
Cov_hb	f4	1 m^2	$-2 \cdot 10^{10}$	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:

$$\begin{pmatrix} \sigma^2_{v_x} & \sigma_{v_x v_y} & \sigma_{v_x v_z} & \sigma_{v_x d} \\ \sigma_{v_y v_x} & \sigma^2_{v_y} & \sigma_{v_y v_z} & \sigma_{v_y d} \\ \sigma_{v_z v_x} & \sigma_{v_z v_y} & \sigma^2_{v_z} & \sigma_{v_z d} \\ \sigma_{dv_x} & \sigma_{dv_y} & \sigma_{dv_z} & \sigma^2_{d} \end{pmatrix}$$

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.

Note that the availability of the off-diagonal elements depends on the "Integrity" capability (see section 1.1).

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 9: Reserved 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.

Note that the availability of the off-diagonal elements depends on the "Integrity" capability (see section 1.1).

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 9: Reserved 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	$-2\cdot10^{10}$	Variance of the north-velocity estimate
Cov_VeVe	f4	1 m^2/s^2	$-2\cdot10^{10}$	Variance of the east-velocity estimate
Cov_VuVu	f4	1 m^2/s^2	$-2\cdot10^{10}$	Variance of the up-velocity estimate
Cov_DtDt	f4	1 m^2/s^2	$-2\cdot10^{10}$	Variance of the clock drift estimate
Cov_VnVe	f4	1 m^2/s^2	$-2\cdot10^{10}$	Covariance between the north- and east-velocity estimates
Cov_VnVu	f4	1 m^2/s^2	$-2\cdot10^{10}$	Covariance between the north- and up-velocity estimates
Cov_VnDt	f4	1 m^2/s^2	$-2\cdot10^{10}$	Covariance between the north-velocity and clock drift estimates
Cov_VeVu	f4	1 m^2/s^2	$-2\cdot10^{10}$	Covariance between the east- and up-velocity estimates
Cov_VeDt	f4	1 m^2/s^2	$-2\cdot10^{10}$	Covariance between the east-velocity and clock drift estimates
Cov_VuDt	f4	1 m^2/s^2	$-2\cdot10^{10}$	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^tA). 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	$-2\cdot10^{10}$	Horizontal Protection Level (see the DO 229 standard).
VPL	f4	1 m	$-2\cdot10^{10}$	Vertical Protection Level (see the DO 229 standard).
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 the PVT engine used corrections 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 9: Reserved 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	$-2\cdot10^{10}$	X baseline component (from rover to base)
`DeltaY`	f8	1 m	$-2\cdot10^{10}$	Y baseline component (from rover to base)
`DeltaZ`	f8	1 m	$-2\cdot10^{10}$	Z baseline component (from rover to base)
`DeltaVx`	f4	1 m / s	$-2\cdot10^{10}$	X velocity of base with respect to rover
`DeltaVy`	f4	1 m / s	$-2\cdot10^{10}$	Y velocity of base with respect to rover
`DeltaVz`	f4	1 m / s	$-2\cdot10^{10}$	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 the PVT engine used corrections 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 9: Reserved 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

NavCart	Number: 4272 "OnChange" interval: default PVT output rate (see 4.1.8)

This block combines fields taken from the PVTCartesian, AttEuler, DOP and ReceiverTime SBF blocks, and as such provides the combination of position, velocity, attitude, DOP and UTC time.

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			See corresponding field in the PVTCartesian block description.
Error	u1			See corresponding field in the PVTCartesian block description.
X	f8	1 m	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
Y	f8	1 m	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
Z	f8	1 m	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
Undulation	f4	1 m	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
Vx	f4	1 m / s	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
Vy	f4	1 m / s	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
Vz	f4	1 m / s	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
COG	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
RxClkBias	f8	1 ms	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
RxClkDrift	f4	1 ppm	$-2 \cdot 10^{10}$	See corresponding field in the PVTCartesian block description.
TimeSystem	u1		255	See corresponding field in the PVTCartesian block description.
Datum	u1		255	See corresponding field in the PVTCartesian block description.
NrSV	u1		255	See corresponding field in the PVTCartesian block description.
WACorrInfo	u1		0	See corresponding field in the PVTCartesian block description.
ReferenceID	u2		65535	See corresponding field in the PVTCartesian block description.
MeanCorrAge	u2	0.01 s	65535	See corresponding field in the PVTCartesian block description.
SignalInfo	u8		0	See corresponding field in the PVTCartesian block description, except that this is a 64-bit field.

Rev 1

AlertFlag	u1			Bit field indicating integrity related information: Bits 0-1: See corresponding field in PVTCartesian. Bit 2: See corresponding field in PVTCartesian. Bit 3: See corresponding field in PVTCartesian. Bit 4: Reserved Bit 5: See corresponding field in PVTCartesian. Bit 6: See corresponding field in PVTCartesian. Bit 7: Reserved
NrBases	u1		0	See corresponding field in the PVTCartesian block description.
PPPInfo	u2		0	See corresponding field in the PVTCartesian block description.
Latency	u2	0.0001 s	65535	See corresponding field in the PVTCartesian block description.
PosHAcc	u2	0.001 m	65535	2DRMS horizontal position 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 PosHAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m
PosVAcc	u2	0.001 m	65535	2-sigma vertical position accuracy. The vertical distance between the true position and the computed position is expected to be lower than PosVAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m.
VelHAcc	u2	0.001 m / s	65535	2DRMS horizontal velocity accuracy: twice the root-mean-square of the horizontal velocity error. The difference between the true horizontal velocity and the computed velocity is expected to be lower than VelHAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m/s.
VelVAcc	u2	0.001 m / s	65535	2-sigma vertical velocity accuracy. The difference between the true vertical velocity and the computed velocity is expected to be lower than VelVAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m/s.
Misc	u1			See corresponding field in the PVTCartesian block description.
Reserved	u1			Reserved for future use, to be ignored by decoding software
ModeAtt	u2			See corresponding Mode field in the AttEuler block description.
ErrorAtt	u1			See corresponding Error field in the AttEuler block description.
NrSVAtt	u1		255	See corresponding NrSV field in the AttEuler block description.
Heading	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding Heading field in the AttEuler block description.
Pitch	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding Pitch field in the AttEuler block description.
Roll	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding Roll field in the AttEuler block description.
HeadingAcc	u2	0.001 degrees	65535	2-sigma heading accuracy. The difference between the true heading and the computed heading is expected to be lower than HeadingAcc with a probability of at least 95%. The value is clipped to 65534 =65.534 degrees.
PitchAcc	u2	0.001 degrees	65535	2-sigma pitch accuracy. The difference between the true pitch and the computed pitch is expected to be lower than PitchAcc with a probability of at least 95%. The value is clipped to 65534 =65.534 degrees.
RollAcc	u2	0.001 degrees	65535	2-sigma roll accuracy. The difference between the true roll and the computed roll is expected to be lower than RollAcc with a probability of at least 95%. The value is clipped to 65534 =65.534 degrees.
PDOP	u2	0.01	0	See corresponding field in the DOP block description.

UTCHour	i1	1 hour	−128	See corresponding field in the ReceiverTime block description.
UTCMin	i1	1 minute	−128	See corresponding field in the ReceiverTime block description.
UTCmsec	u2	1 ms	65535	Current millisecond in the UTC minute. From 0 to 59999, or 65535 if not available
UTCYear	i1	1 year	−128	See corresponding field in the ReceiverTime block description.
UTCMonth	i1	1 month	−128	See corresponding field in the ReceiverTime block description.
UTCDay	i1	1 day	−128	See corresponding field in the ReceiverTime block description.
Padding	u1[..]			Padding bytes, see 4.1.5

NavGeod	Number: 4275 "OnChange" interval: default PVT output rate (see 4.1.8)

This block combines fields taken from the PVTGeodetic, AttEuler, DOP and ReceiverTime SBF blocks, and as such provides the combination of position, velocity, attitude, DOP and UTC time.

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			See corresponding field in the PVTGeodetic block description.
Error	u1			See corresponding field in the PVTGeodetic block description.
Latitude	f8	1 rad	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
Longitude	f8	1 rad	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
Height	f8	1 m	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
Undulation	f4	1 m	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
Vn	f4	1 m / s	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
Ve	f4	1 m / s	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
Vu	f4	1 m / s	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
COG	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
RxClkBias	f8	1 ms	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
RxClkDrift	f4	1 ppm	$-2 \cdot 10^{10}$	See corresponding field in the PVTGeodetic block description.
TimeSystem	u1		255	See corresponding field in the PVTGeodetic block description.
Datum	u1		255	See corresponding field in the PVTGeodetic block description.
NrSV	u1		255	See corresponding field in the PVTGeodetic block description.
WACorrInfo	u1		0	See corresponding field in the PVTGeodetic block description.
ReferenceID	u2		65535	See corresponding field in the PVTGeodetic block description.
MeanCorrAge	u2	0.01 s	65535	See corresponding field in the PVTGeodetic block description.
SignalInfo	u8		0	See corresponding field in the PVTGeodetic block description.
AlertFlag	u1			Bit field indicating integrity related information: Bits 0-1: See corresponding field in PVTGeodetic. Bit 2: See corresponding field in PVTGeodetic. Bit 3: See corresponding field in PVTGeodetic. Bit 4: Reserved Bit 5: See corresponding field in PVTGeodetic. Bit 6: See corresponding field in PVTGeodetic. Bit 7: Reserved
NrBases	u1		0	See corresponding field in the PVTGeodetic block description.
PPPInfo	u2		0	See corresponding field in the PVTGeodetic block description.
Latency	u2	0.0001 s	65535	See corresponding field in the PVTGeodetic block description.
PosHAcc	u2	0.001 m	65535	2DRMS horizontal position 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 PosHAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m

Rev 1

PosVAcc	u2	0.001 m	65535	2-sigma vertical position accuracy. The vertical distance between the true position and the computed position is expected to be lower than PosVAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m.
VelHAcc	u2	0.001 m / s	65535	2DRMS horizontal velocity accuracy: twice the root-mean-square of the horizontal velocity error. The difference between the true horizontal velocity and the computed velocity is expected to be lower than VelHAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m/s.
VelVAcc	u2	0.001 m / s	65535	2-sigma vertical velocity accuracy. The difference between the true vertical velocity and the computed velocity is expected to be lower than VelVAcc with a probability of at least 95%. The value is clipped to 65534 =65.534m/s.
Misc	u1			See corresponding field in the PVTGeodetic block description.
Reserved	u1			Reserved for future use, to be ignored by decoding software
ModeAtt	u2			See corresponding Mode field in the AttEuler block description.
ErrorAtt	u1			See corresponding Error field in the AttEuler block description.
NrSVAtt	u1		255	See corresponding NrSV field in the AttEuler block description.
Heading	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding Heading field in the AttEuler block description.
Pitch	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding Pitch field in the AttEuler block description.
Roll	f4	1 degree	$-2 \cdot 10^{10}$	See corresponding Roll field in the AttEuler block description.
HeadingAcc	u2	0.001 degrees	65535	2-sigma heading accuracy. The difference between the true heading and the computed heading is expected to be lower than HeadingAcc with a probability of at least 95%. The value is clipped to 65534 =65.534 degrees.
PitchAcc	u2	0.001 degrees	65535	2-sigma pitch accuracy. The difference between the true pitch and the computed pitch is expected to be lower than PitchAcc with a probability of at least 95%. The value is clipped to 65534 =65.534 degrees.
RollAcc	u2	0.001 degrees	65535	2-sigma roll accuracy. The difference between the true roll and the computed roll is expected to be lower than RollAcc with a probability of at least 95%. The value is clipped to 65534 =65.534 degrees.
PDOP	u2	0.01	0	See corresponding field in the DOP block description.
UTCHour	i1	1 hour	−128	See corresponding field in the ReceiverTime block description.
UTCMin	i1	1 minute	−128	See corresponding field in the ReceiverTime block description.
UTCmsec	u2	1 ms	65535	Current millisecond in the UTC minute. From 0 to 59999, or 65535 if not available
UTCYear	i1	1 year	−128	See corresponding field in the ReceiverTime block description.
UTCMonth	i1	1 month	−128	See corresponding field in the ReceiverTime block description.
UTCDay	i1	1 day	−128	See corresponding field in the ReceiverTime block description.
Padding	u1[..]			Padding bytes, see 4.1.5

4.2.11 GNSS Attitude Blocks

AttEuler	Number: 5938 "OnChange" interval: default PVT output rate (see 4.1.8)

The AttEuler block contains the Euler angles (pitch, roll and heading) at the time specified in the TOW and WNc fields (in the receiver time frame).

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		255	The average over all antennas of the number of satellites currently included in the attitude calculations.
Error	u1			Bit field providing error information. For each antenna baseline, two bits are used to provide error information: Bits 0-1: Error code for Main-Aux1 baseline: 0: No error 1: Not enough measurements 2: Reserved 3: Reserved Bits 2-3: Error code for Main-Aux2 baseline, same definition as bit 0-1. Bits 4-6: Reserved Bit 7: Set when GNSS-based attitude not requested by user. In that case, the other bits are all zero.
Mode	u2			Attitude mode code: 0: No attitude 1: Heading, pitch (roll = 0), aux antenna positions obtained with float ambiguities 2: Heading, pitch (roll = 0), aux antenna positions obtained with fixed ambiguities 3: Heading, pitch, roll, aux antenna positions obtained with float ambiguities 4: Heading, pitch, roll, aux antenna positions obtained with fixed ambiguities
Info	u1			Bit field providing extra information: Bits 0-1: Attitude source, as requested with the command **setGNSSAttitude**: 0: Attitude source unknown 1: Attitude computed from multiple antennas 2: Attitude computed from moving-base Bits 2-7: Reserved
Reserved	u1			Reserved for future use, to be ignored by decoding software
Heading	f4	1 degree	$-2\cdot10^{10}$	Heading
Pitch	f4	1 degree	$-2\cdot10^{10}$	Pitch
Roll	f4	1 degree	$-2\cdot10^{10}$	Roll
PitchDot	f4	1 degree / s	$-2\cdot10^{10}$	Rate of change of the pitch angle
RollDot	f4	1 degree / s	$-2\cdot10^{10}$	Rate of change of the roll angle
HeadingDot	f4	1 degree / s	$-2\cdot10^{10}$	Rate of change of the heading angle
Padding	u1[..]			Padding bytes, see 4.1.5

AttCovEuler	Number: 5939 "OnChange" interval: default PVT output rate (see 4.1.8)

This block contains the elements of the symmetric variance-covariance matrix of the attitude angles reported in the AttEuler block

$$\begin{pmatrix} \sigma_\phi^2 & \sigma_{\phi\theta} & \sigma_{\phi\psi} \\ \sigma_{\theta\phi} & \sigma_\theta^2 & \sigma_{\theta\psi} \\ \sigma_{\psi\phi} & \sigma_{\psi\theta} & \sigma_\psi^2 \end{pmatrix}$$

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).

In case the receiver is in heading and pitch mode only, only the heading and pitch variance values will be valid. All other components of the variance-covariance matrix are set to their Do-Not-Use value.

Note that the availability of the off-diagonal elements depends on the "Integrity" capability (see section 1.1).

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			Reserved for future use, to be ignored by decoding software
Error	u1			Bit field providing error information. For each antenna baseline, two bits are used to provide error information: Bits 0-1: Error code for Main-Aux1 baseline: 0: No error 1: Not enough measurements 2: Reserved 3: Reserved Bits 2-3: Error code for Main-Aux2 baseline, same definition as bit 0-1. Bits 4-6: Reserved Bit 7: Set when GNSS-based attitude not requested by user. In that case, the other bits are all zero.
Cov_HeadHead	f4	1 degree^2	$-2\cdot10^{10}$	Variance of the heading estimate
Cov_PitchPitch	f4	1 degree^2	$-2\cdot10^{10}$	Variance of the pitch estimate
Cov_RollRoll	f4	1 degree^2	$-2\cdot10^{10}$	Variance of the roll estimate
Cov_HeadPitch	f4	1 degree^2	$-2\cdot10^{10}$	Covariance between Euler angle estimates. Future functionality. The values are currently set to their Do-Not-Use values.
Cov_HeadRoll	f4	1 degree^2	$-2\cdot10^{10}$	Covariance between Euler angle estimates. Future functionality. The values are currently set to their Do-Not-Use values.
Cov_PitchRoll	f4	1 degree^2	$-2\cdot10^{10}$	Covariance between Euler angle estimates. Future functionality. The values are currently set to their Do-Not-Use values.
Padding	u1[..]			Padding bytes, see 4.1.5

AuxAntPositions	Number: 5942 "OnChange" interval: default PVT output rate (see 4.1.8)

The AuxAntPositions block contains the relative position and velocity of the different antennas in a multi-antenna receiver. The coordinates are expressed in the local-level ENU reference frame.

When the antenna positions cannot be estimated, the baseline vectors 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 AuxAntPositionSub sub-blocks in this AuxAntPositions block
SBLength	u1	1 byte		Length of one sub-block in bytes
AuxAntPosition	...	...		*A succession of N AuxAntPositionSub sub-blocks, see definition below*
Padding	u1[..]			Padding bytes, see 4.1.5

AuxAntPositionSub sub-block definition:

Parameter	Type	Units	Do-Not-Use	Description
NrSV	u1		255	Total number of satellites tracked by the antenna identified by the AuxAntID field and used in the attitude computation.
Error	u1			Aux antenna position error code: 0: No error 1: Not enough measurements 2: Reserved 3: Reserved If error is not 0, the coordinates reported later in this block are all set to their Do-Not-Use value.
AmbiguityType	u1		255	Aux antenna positions obtained with 0: Fixed ambiguities 1: Float ambiguities
AuxAntID	u1			Auxiliary antenna ID: 1 for the first auxiliary antenna, 2 for the second, etc...
DeltaEast	f8	1 m	$-2\cdot10^{10}$	Position in East direction (relative to main antenna)
DeltaNorth	f8	1 m	$-2\cdot10^{10}$	Position in North direction (relative to main antenna)
DeltaUp	f8	1 m	$-2\cdot10^{10}$	Position in Up direction (relative to main antenna)
EastVel	f8	1 m / s	$-2\cdot10^{10}$	Velocity in East direction (relative to main antenna)
NorthVel	f8	1 m / s	$-2\cdot10^{10}$	Velocity in North direction (relative to main antenna)
UpVel	f8	1 m / s	$-2\cdot10^{10}$	Velocity in Up direction (relative to main antenna)
Padding	u1[..]			Padding bytes, see 4.1.5

EndOfAtt	Number: 5943 "OnChange" interval: default PVT output rate (see 4.1.8)

This block marks the end of transmission of all GNSS-attitude 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

4.2.12 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 time 3: Receiver time 4: GLONASS time 5: Galileo time (GST) 6: BeiDou time 100: Fugro AtomiChron 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 1.15 for an explanation of the xPPS generation principle, and for a description of the xPPS offset.
Padding	u1[..]			Padding bytes, see 4.1.5

4.2.13 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: $t_{ext,rx}$ [s] = TOW/1000 + Offset $t_{ext,rx}$ 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: $t_{ext,GNSS}$ [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 and clock bias 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 9: Reserved 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 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: $t_{sys} = t_{rx}$ - RxClkBias
RxClkDrift	f4	1 ppm	$-2 \cdot 10^{10}$	Not applicable
TimeSystem	u1		255	Time system to which the offset is provided in this sub-block: 0: GPS time 1: Galileo time (GST) 3: GLONASS time 4: BeiDou time 5: QZSS time 100: Fugro AtomiChron 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 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 5-6: Type of RTK or DGNSS differential corrections: 0: unknown or not in differential positioning mode (DGNSS or RTK) 1: corrections from a physical base 2: corrections from a virtual base (VRS) 3: SSR corrections (RTK using SSR corrections is often referred to as RTK-SSR or PPP-RTK) Bit 7: Reserved
ReferenceID	u2		65535	This field indicates the reference ID of the differential information used. In case of DGNSS 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 DGNSS 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			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 Bit 5: SIG_AUTH_ALERT: see description in the Flags field of the RFStatus block Bit 6: NAV_MSG_AUTH_ALERT: see description in the Flags field of the RFStatus block Bit 7: Reserved
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 DGNSS 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.

Rev 3

Rev 1

Rev 2

Rev 2

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 (PCO) and/or the phase center variation (PCV) are compensated at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating which position is reported in this block: the APC, the ARP or the marker (see **setAntennaOffset** command) 0: Unknown 1: The ARP is reported (the ARP-to-marker offset is zero) 2: The marker is reported (the ARP-to-marker offset is not zero) 3: APC is reported. This value is used if the *APCOutput* argument of the **setAntennaOffset** command is available and is set to on.
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 and clock bias 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 9: Reserved 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 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: $t_{sys} = t_{rx}$ - RxClkBias
RxClkDrift	f4	1 ppm	$-2\cdot10^{10}$	Not applicable
TimeSystem	u1		255	Time system to which the offset is provided in this sub-block: 0: GPS time 1: Galileo time (GST) 3: GLONASS time 4: BeiDou time 5: QZSS time 100: Fugro AtomiChron 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 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 5-6: Type of RTK or DGNSS differential corrections: 0: unknown or not in differential positioning mode (DGNSS or RTK) 1: corrections from a physical base 2: corrections from a virtual base (VRS) 3: SSR corrections (RTK using SSR corrections is often referred to as RTK-SSR or PPP-RTK) Bit 7: Reserved
ReferenceID	u2		65535	This field indicates the reference ID of the differential information used. In case of DGNSS 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 DGNSS 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			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 Bit 5: SIG_AUTH_ALERT: see description in the Flags field of the RFStatus block Bit 6: NAV_MSG_AUTH_ALERT: see description in the Flags field of the RFStatus block Bit 7: Reserved
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 DGNSS 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.

Rev 3

Rev 1

Rev 2

Rev 2

`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 (PCO) and/or the phase center variation (PCV) are compensated at the rover, unset if not or unknown. Bit 2: Proprietary. Bit 3: Proprietary. Bits 4-5: Proprietary. Bits 6-7: Flag indicating which position is reported in this block: the APC, the ARP or the marker (see **`setAntennaOffset`** command) 0: Unknown 1: The ARP is reported (the ARP-to-marker offset is zero) 2: The marker is reported (the ARP-to-marker offset is not zero) 3: APC is reported. This value is used if the *APCOutput* argument of the **`setAntennaOffset`** command is available and is set to `on`.
`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 the PVT engine used corrections 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 9: Reserved 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

ExtEventAttEuler	Number: 4237 "OnChange" interval: each time an external event is detected

This block contains the Euler angles (pitch, roll and heading) applicable at the time of an external event.

This block has the same structure and description as the AttEuler 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	
NrSV	u1		255	The average over all antennas of the number of satellites currently included in the attitude calculations.
Error	u1			Bit field providing error information. For each antenna baseline, two bits are used to provide error information: Bits 0-1: Error code for Main-Aux1 baseline: 0: No error 1: Not enough measurements 2: Reserved 3: Reserved Bits 2-3: Error code for Main-Aux2 baseline, same definition as bit 0-1. Bits 4-6: Reserved Bit 7: Set when GNSS-based attitude not requested by user. In that case, the other bits are all zero.
Mode	u2			Attitude mode code: 0: No attitude 1: Heading, pitch (roll = 0), aux antenna positions obtained with float ambiguities 2: Heading, pitch (roll = 0), aux antenna positions obtained with fixed ambiguities 3: Heading, pitch, roll, aux antenna positions obtained with float ambiguities 4: Heading, pitch, roll, aux antenna positions obtained with fixed ambiguities
Reserved	u1			Reserved for future use, to be ignored by decoding software
Heading	f4	1 degree	$-2\cdot10^{10}$	Heading
Pitch	f4	1 degree	$-2\cdot10^{10}$	Pitch
Roll	f4	1 degree	$-2\cdot10^{10}$	Roll
PitchDot	f4	1 degree / s	$-2\cdot10^{10}$	Not applicable
RollDot	f4	1 degree / s	$-2\cdot10^{10}$	Not applicable
HeadingDot	f4	1 degree / s	$-2\cdot10^{10}$	Not applicable
Padding	u1[..]			Padding bytes, see 4.1.5

4.2.14 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: CMR 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 19: SSR2OBS (message generated internally by SSR2OBS conversion)
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 (Msg 1) 8: RTCM 3.x (Msg 1005 or 1006) 9: RTCMV (Msg 3) 10: CMR+ (Type 2)
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
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

4.2.15 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

LBandRaw	Number: 4212 "OnChange" interval: "output each time new data is available"

This block contains the raw user data from an L-band beam. Note that the block may be empty for beams transmitting proprietary data.

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 User Data bytes in the UserData field.
Frequency	u4	1 Hz	0	Nominal frequency of the beam for which data is provided in this block.
UserData	u1[N]			The User Data bytes
Rev 1 | Channel	u1			The channel number of the LBR channel to which the data belongs.
Padding	u1[..]			Padding bytes, see 4.1.5

4.2.16 Status Blocks

ChannelStatus	Number: 4013 "OnChange" interval: 1s

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		E5ab		E5b		E5a		E6BC		Reserved		E1BC		Reserved	
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		L1		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		0	Satellite ID, see 4.1.9. If zero, the satellite ID is not available in this field, but can be found in the SVIDFull field instead.
FreqNr	u1		0	For GLONASS satellites, this is the frequency number, with an offset of 8. It ranges from 1 (corresponding to an actual frequency number of -7) to 14 (corresponding to an actual frequency number of 6). Otherwise, FreqNr is reserved and must be ignored by the decoding software.
SVIDFull	u2			If the SVID field is zero, this field contains the satellite ID (see 4.1.9). If SVID is not zero, this field should be ignored.
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 startup 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. 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 TimeSync input. 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 caused the last reboot. 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: OUTOFFENCE: set if the receiver is currently out of its permitted region of operation. For Geo-fencing this happens when moving out of the defined region. For Motion-fencing it occurs when moving too far from the initial position. The RxMessages will show if this flag was set due to Geo-fencing or Motion-fencing (see the **lif,RxMessage** command). 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/B2a 6: E5b/B2b 7: E5 8: Combined GPS/GLONASS/SBAS/Galileo L1 9: Combined GPS/GLONASS L2 10: MSS/L-band 11: B1 12: B3 13: S-band 14: Combined BeiDou B3 and Galileo E6 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		0	Satellite ID, see 4.1.9. If zero, the satellite ID is not available in this field, but can be found in the SVIDFull field instead.
FreqNr	u1		0	For GLONASS satellites, this is the frequency number, with an offset of 8. It ranges from 1 (corresponding to an actual frequency number of -7) to 14 (corresponding to an actual frequency number of 6). 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
Rev 1 SVIDFull	u2			If the SVID field is zero, this field contains the satellite ID (see 4.1.9). If SVID is not zero, this field should be ignored.
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:

- startup 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 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>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 setDataInOut command)
35: RINEX
36: CGGTTS
40: BINEX
64: NMEA
96: RTCMv2
97: RTCMv3
98: CMR
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
122: raw LBAND data from Beam5
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 (6)</td></tr>
</table>

NrBytesAccepted	u4	1 byte	4294967295	Total number of bytes [6] 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

[6] 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:

- startup 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:

Parameter	Type	Units	Description
`CD`	u1		Identifier of the connection to which this information applies: **Value of CD** / **Connection type** / **Example** 0-31 / COMx, with *x*=CD / 1: COM1 32-47 / USBx, with *x*=CD-32 / 33: USB1 48-63 / OTGx, with *x*=CD-48 / 49: OTG1 64-95 / IPx, with *x*=CD-54 / 64:IP10 96-127 / DSKx, with *x*=CD-96 / 97:DSK1 128-159 / NTRx, with *x*=CD-128 (NTRIP connections) / 129:NTR1 160-191 / IPSx, with *x*=CD-160 (IP server connections) / 161:IPS1 192 / BT01 (Bluetooth connection) / 193 / BT02 (Bluetooth connection) / 196 / UHF1 (UHF Modem) / 200-205 / IPRx, with *x*=CD-200 (IP receive connections) / 201:IPR1 210 / DCL1 (cellular data-call connection) / 214 / CAN1 (CAN stream interface) / 215-219 / Reserved / 220 / SPI1 (SPI interface) / 221-255 / Reserved /
`N2`	u1		Number of `OutputTypeSub` sub-blocks included at the end of this `OutputStatsSub` sub-block
`AllowedRate`	u2	1 kbyte / s	Maximum datarate recommended on this connection
`NrBytesProduced`	u4	1 byte	Total number of bytes produced by the receiver. See also the `NrBytesSent` field.
`NrBytesSent`	u4	1 byte	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.
`NrClients`	u1		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.
`Reserved`	u1[3]		Reserved for future use
`Padding`	u1[..]		Padding bytes, see 4.1.5
OutputType	...	...	*A succession of N2 `OutputTypeSub` sub-blocks, see definition below*

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: CMR 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 122: raw LBAND data from Beam5
`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

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) 29: Scintillation score. This score is inversely proportional to the severity of the detected ionosphere scintillation. A high value indicates the absence of scintillation. 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 (7)	16 MSB of the total disk usage. The disk usage in bytes is given by DiskUsageMSB*4294967296+DiskUsageLSB.
DiskUsageLSB	u4		4294967295 (7)	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 startup and restarts at zero after having reached 255.
Rev 1 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

(7) The disk usage is invalid if both DiskUsageMSB is 65535 and DiskUsageLSB is 4294967295.

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: SIG_AUTH_ALERT: Set when the receiver determined that the GNSS signals at its RF connector may not be authentic and that its output (position or raw measurements) may therefore be misleading. The receiver may be connected to a GNSS simulator, or be subject to a spoofing attack. This bit is based on a set of built-in tests to check the authenticity of the GNSS signals. In addition to those tests, Navigation Message Authentication (NMA) is performed. If NMA verification fails, bit NAV_MSG_AUTH_ALERT is set instead. Note that bit SIG_AUTH_ALERT may be set even if no interference is detected (i.e. with no associated RFBand sub-blocks). Bit 1: NAV_MSG_AUTH_ALERT: Set when a non-authentic navigation message is detected by NMA checks (e.g. Galileo OSNMA or Fugro AtomiChron NMA). 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	Do-Not-Use	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, or 0 in case of short pulsed interferences of which the bandwidth cannot be determined.
Info	u1			Info on this RF band: Bits 0-3: Mode: 1: This RF band is suppressed by a notch filter set manually by user command. 2: The receiver detected interference in this band, and 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*

Power	i1	1 dBm	0	Estimated interference power in this band. Set to 0 if the power could not be estimated or for bands corresponding to manual notch filters.
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 for trusted time information 3: Init failed - inconsistent time 4: Init failed - KROOT signature invalid 5: Init failed - invalid param received 6: Authenticating 255 Bits 3-10: OSNMA initialization progress, expressed in percent [0-100]. This value will only be encoded when the OSNMA Status is initializing. A value of 255 indicates an alert condition of the OSNMA operation resulting in OSNMA not being available. 7 Bits 11-13: Trusted time source: 0: NTP 1: L-Band 2: Command 7: Unknown Bit 14: Indicates if the Merkle tree renewal is in progress: 0: No 1: Yes Bit 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

4.2.17 Miscellaneous Blocks

ReceiverSetup	Number:	5902
	"OnChange" interval:	Block generated every 60 seconds and 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**).
MarkerType	c1[20]			Marker type (set with the **setMarkerParameters** command).
GNSSFWVersion	c1[20]			Version the firmware installed on the receiver.
ProductName	c1[40]			Product name.
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

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). Beware that strings of characters are encoded as plaintext. No obfuscation takes place.
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.
TOWDelta	f4	1 s	$-2 \cdot 10^{10}$	Time offset from TOW. The time tag of the first sample in GNSS time scale, expressed in seconds, is given by TOW*0.001 + TOWDelta.
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

Parameter	Type	Units	Do-Not-Use	
Sync1	c1			
Sync2	c1			
CRC	u2			Block Hea
ID	u2			
Length	u2	1 byte		
TOW	u4	0.001 s	4294967295	Receiver ti
WNc	u2	1 week	65535	
CD	u1			Identifier c received:
Reserved1	u1[3]			Reserved,
StringLn	u2			Length of
SensorModel	c1[20]			Not suppo
SensorType	c1[20]			Not suppo
Reserved2	u1[20]			Reserved,
ASCIIString	c1[StringLn]			ASCII strin "\0" chara character(
Padding	u1[..]			Padding b

Value of CD
0-31
32-47
48-63
64-95
128-159
192
193
196
200-205
210
214
215-255

4.3 SBF Change Log

Date	Change Description
Oct 01, 2021	Added the QZSRawL5S block containing the raw QZSS L5S navigation bits
Feb 11, 2020	Added the BDSRawB2b block containing the raw BeiDou B2b navigation symbols
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
Sep 21, 2017	Added the LBandRaw block containing raw L-Band data bytes
Jun 20, 2017	Added the BDSIon and BDSUtc blocks containing BeiDou ionospheric and UTC offset parameters
Mar 1, 2017	Renamed CMPNav to BDSNav and CMPRaw to BDSRaw
Nov 10, 2015	Added the RxMessage block containing the receiver activity log
Feb 04, 2015	Added the QZSNav block containing decoded QZSS navigation data
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
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 19, 2012	Added GEORawL5 block
Oct 1, 2012	Added new signal type for L-band and SBAS L5 signals (value 23 and 25)
Sep 20, 2012	Added field PPPInfo to the PVTCartesian and PVTGeodetic blocks
Feb 28, 2012	Added GALSARRLM block
Feb 6, 2012	Added QZSS signals and QZSRawL1CA, QZSRawL2C and QZSRawL5 blocks

Appendix A

Attitude Angles

The attitude of the vehicle is defined as the angles between the vehicle reference frame and the local-level reference frame (defined by the East, North and Up directions). The vehicle reference frame is defined as follows. It is attached to the vehicle and has its X axis pointing along the longitudinal vehicle axis, the Y axis pointing towards the vehicle starboard (right) side and the Z axis pointing down, as illustrated in Figure A-1.

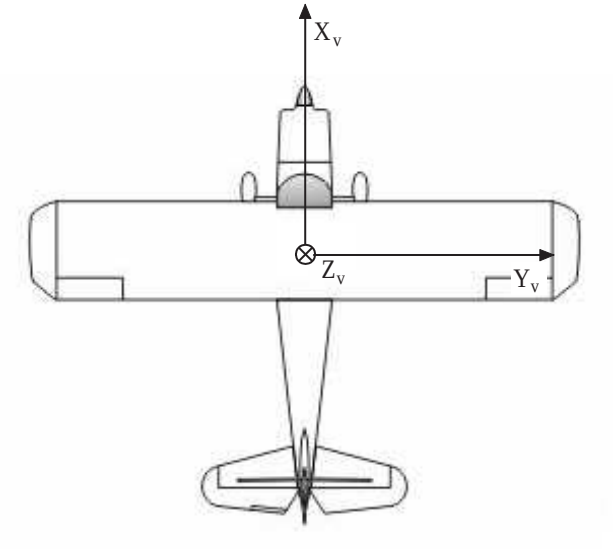


Figure A-1: Vehicle reference frame.

Septentrio receivers express the vehicle attitude in Euler angles using the heading-pitch-roll rotation sequence. More specifically, Euler angles are defined as successive rotations of the vehicle frame (X, Y, Z axes) relative to the local-level East-North-Up reference frame. The rotation sequence is shown in Figure A-2. The heading (ψ) of the vehicle is defined as the right-handed rotation of the vehicle about the Z axis ($0^o \leq \psi \leq 360^o$). The pitch (θ) of the vehicle is defined as the right-handed rotation about the vehicle Y axis ($-90^o \leq \theta \leq 90^o$). The roll ($\phi$) of the vehicle is defined as the right-handed rotation about the vehicle X axis ($-180^o \leq \phi \leq 180^o$).

Starting from the situation where X points to the North, Y to the East and Z down, the following successive rotations define the attitude of the vehicle. Note that the order of the rotations is important.

1. Rotate through angle ψ about Z axis;
2. Rotate through angle θ about new Y axis;
3. Rotate through angle ϕ about new X axis;

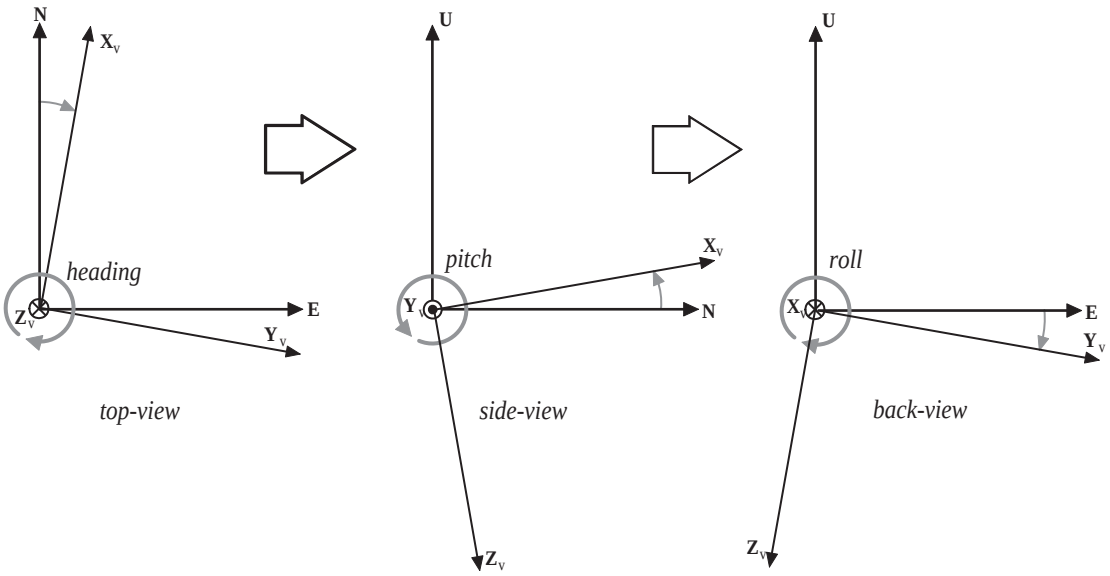


Figure A-2: Euler angle sequence.

Appendix B

List of SBF Blocks

The following table provides the list of the SBF block names and numbers available on mosaic-G5 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
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
QZSRawL6D	4270	QZSS L6D navigation message			S
QZSRawL6E	4271	QZSS L6E 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 L5 subframe			S
GPS Decoded Message Blocks					
GPSNav	5891	GPS ephemeris and clock		•	S
GPSAlm	5892	Almanac data for a GPS satellite		•	S
GPSIon	5893	Ionosphere data from GPS		•	S
GPSUtc	5894	GPS-UTC data from GPS subframe 5		•	S
GPSCNav	4042	CNAV Ephemeris data for one GPS satellite		•	S
GLONASS Decoded Message Blocks					

Block name	Block No	Content description	Flex Rate	esoc	Time Stamp
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
BDSCNav1	4251	BeiDou B-CNAV1 ephemeris data for one satellite		•	S
BDSCNav2	4252	BeiDou B-CNAV2 ephemeris data for one satellite		•	S
BDSCNav3	4253	BeiDou B-CNAV3 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					
GEONav	5896	MT09 : SBAS navigation message		•	S
GEOAlm	5897	MT17 : SBAS satellite almanac		•	S
GEOIGPMask	5931	MT18 : Ionospheric grid point mask			S
GEOIonoDelay	5933	MT26 : Ionospheric delay corrections			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
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	•	•	E
PVTSupportA	4079	Internal parameters for maintenance and support	•	•	E
EndOfPVT	5921	PVT epoch marker	•	•	R
NavCart	4272	Full GNSS position, velocity, attitude, DOP and UTC time in Cartesian coordinates.	•	•	R
NavGeod	4275	Full GNSS position, velocity, attitude, DOP and UTC time in Geodetic co-ordinates.	•	•	R
GNSS Attitude Blocks					
AttEuler	5938	GNSS attitude expressed as Euler angles	•	•	R
AttCovEuler	5939	Covariance matrix of attitude	•	•	R
AuxAntPositions	5942	Relative position and velocity estimates of auxiliary antennas	•	•	R
EndOfAtt	5943	GNSS attitude 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
ExtEventBaseVectGeod	4217	ENU relative position with respect to base(s) at the instant of an event			E
ExtEventAttEuler	4237	GNSS attitude expressed as Euler angles at the instant of an event			E
Correction Blocks					
DiffCorrIn	5919	Incoming RTCM or CMR message			R
BaseStation	5949	Base station coordinates			R
L-Band Demodulator Blocks					
LBandTrackerStatus	4201	Status of the L-band signal tracking	•	•	R
LBandRaw	4212	L-Band raw user data			E
Status Blocks					
ChannelStatus	4013	Status of the tracking for all receiver channels	•	•	R
ReceiverStatus	4014	Overall status information of the receiver	•	•	R

Block name	Block No	Content description	Flex Rate	esoc	Time Stamp
SatVisibility	4012	Azimuth/elevation of visible satellites	●	●	R
InputLink	4090	Statistics on input streams	●	●	R
OutputLink	4091	Statistics on output streams	●	●	R
QualityInd	4082	Quality indicators	●	●	R
DiskStatus	4059	Internal logging status	●	●	R
RFStatus	4092	Radio-frequency interference mitigation status	●	●	R
GALAuthStatus	4245	Galileo OSNMA authentication status	●	●	R
Miscellaneous Blocks					
ReceiverSetup	5902	General information about the receiver installation		●	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

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 `$PTNL,AVR` sentence	
DTM	DTM	Datum Reference	
GBS	GBS	GNSS Satellite Fault Detection	
GFA	GFA	GNSS Fix Accuracy and Integrity	
GGA	GGA	GPS Fix Data	GPS Quality Indicator field is set to 5 in PPP mode.
GGAaux1	GGA	GPS Fix Data	GGA sentence containing the position of the aux1 antenna
GGK	GGK	Trimble Navigation proprietary `$PTNL,GGK` sentence	
GGQ	GGQ	Leica Real-Time Position with CQ	
GLL	GLL	Geographic Position - Latitude/Longitude	
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	
HRP	HRP	Heading, Roll, Pitch	Septentrio proprietary, see section C.1.1
RBD	RBD	Rover-Base Direction	Septentrio proprietary, see section C.1.2
RBP	RBP	Rover-Base Position	Septentrio proprietary, see section C.1.3
RBV	RBV	Rover-Base Velocity	Septentrio proprietary, see section C.1.4
RMC	RMC	Recommended Minimum Specific GNSS Data	
ROT	ROT	Rate of Turn	
TFM	TFM	Used Coordinate Transformation Messages	Septentrio proprietary, see section C.1.5
THS	THS	True Heading and Status	

Message Identifier	NMEA For- matter	Short Description	Comment
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 HRP : Heading, Roll, Pitch

Field	Description
$PSSN,HRP,	Start of sentence
hhmmss.ss,	UTC of HRP (HoursMinutesSeconds.DecimalSeconds)
xxxxxx,	Date: ddmmyy
x.x,	Heading, degrees True
x.x,	Roll, degrees
x.x,	Pitch, degrees
x.x,	Heading standard deviation, degrees
x.x,	Roll standard deviation, degrees
x.x,	Pitch standard deviation, degrees
xx,	Number of satellites used for attitude computation
x,	Mode indicator: 0: No attitude available 1: Heading, Pitch with float ambiguities 2: Heading, Pitch with fixed ambiguities 3: Heading, Pitch, Roll with float ambiguities 4: Heading, Pitch, Roll with fixed ambiguities 5: Heading, Pitch from velocity (dead-reckoning) 6: Heading, Pitch, Roll from non-RTK INS 7: Heading, Pitch, Roll from RTK INS 8: Heading, Pitch, Roll from INS coasting
x.x,a	Magnetic variation, degrees (E=East, W=West, see also the **setMagneticVariance** command)
*hh	Checksum delimiter and checksum field
<CR><LF>	End of sentence

C.1.2 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.1.3 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.1.4 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.1.5 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 Differential Correction Messages

This appendix provides a list of all the CMR and RTCM 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 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
RTCM1019	GPS Satellite Ephemeris Data
RTCM1020	Glonass Satellite Ephemeris Data
RTCM1029	Unicode Text String
RTCM1033	Receiver and Antenna Descriptors
RTCM1041	NavIC/IRNSS Satellite Ephemeris Data
RTCM1042	BDS Satellite Ephemeris Data
RTCM1044	QZSS Satellite Ephemeris Data

Message Identifier	Short Description
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
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

A

AGCMode
- setAGCMode, getAGCMode
  - sam, gam, 96

AntennaInfo
- lstAntennaInfo
  - lai, 58

AntennaOffset
- setAntennaOffset, getAntennaOffset
  - sao, gao, 101

AsciiDisplay
- lstAsciiDisplay
  - lad, 151

AttitudeOffset
- setAttitudeOffset, getAttitudeOffset
  - sto, gto, 133

B

BaseCPB
- setBaseCPB, getBaseCPB
  - sbcp, gbcp, 103

BBSamplingMode
- setBBSamplingMode, getBBSamplingMode
  - sbbs, gbbs, 97

C

CalibCommonDelay
- setCalibCommonDelay, getCalibCommonDelay
  - scco, gcco, 83

CalibSignalDelay
- setCalibSignalDelay, getCalibSignalDelay
  - scsi, gcsi, 84

ChannelAllocation
- setChannelAllocation, getChannelAllocation
  - sca, gca, 85

ClockSyncThreshold
- setClockSyncThreshold, getClockSyncThreshold
  - scst, gcst, 140

CMRv2Usage
- setCMRv2Usage, getCMRv2Usage
  - sc2u, gc2u, 180

CN0Mask
setCN0Mask, getCN0Mask
scm, gcm, 87
CommandHelp
lstCommandHelp
help, 60
COMSettings
setCOMSettings, getCOMSettings
scs, gcs, 152
ConfigFile
lstConfigFile
lcf, 61
CopyConfigFile
exeCopyConfigFile, getCopyConfigFile
eccf, gccf, 62
CurrentUser
lstCurrentUser
lcu, 76

D

DaisyChainMode
setDaisyChainMode, getDaisyChainMode
sdcm, gdcm, 153
DataInOut
setDataInOut, getDataInOut
sdio, gdio, 154
DefaultAccessLevel
setDefaultAccessLevel, getDefaultAccessLevel
sdal, gdal, 77
DiffCorrMaxAge
setDiffCorrMaxAge, getDiffCorrMaxAge
sdca, gdca, 104
DiffCorrUsage
setDiffCorrUsage, getDiffCorrUsage
sdcu, gdcu, 105
DiskFullAction
setDiskFullAction, getDiskFullAction
sdfa, gdfa, 181
DiskInfo
lstDiskInfo
ldi, 182

E

EchoMessage
exeEchoMessage, getEchoMessage
eecm, gecm, 156
ElevationMask
setElevationMask, getElevationMask
sem, gem, 106
EventParameters
setEventParameters, getEventParameters
sep, gep, 141

F

factoryReset
  factoryReset
    facr, 78
FileNaming
  setFileNaming, getFileNaming
    sfn, gfn, 183
FrontendMode
  setFrontendMode, getFrontendMode
    sfm, gfm, 98

G

GalOSNMAPublicKeys
  lstGalOSNMAPublicKeys
    lopk, 128
  setGalOSNMAPublicKeys, getGalOSNMAPublicKeys
    sopk, gopk, 129
GalOSNMAUsage
  setGalOSNMAUsage, getGalOSNMAUsage
    sou, gou, 130
GeodeticDatum
  setGeodeticDatum, getGeodeticDatum
    sgd, ggd, 135
GeoidUndulation
  setGeoidUndulation, getGeoidUndulation
    sgu, ggu, 107
GNSSAttitude
  setGNSSAttitude, getGNSSAttitude
    sga, gga, 134
GPIO1Mode
  setGPIO1Mode, getGPIO1Mode
    sgp1, ggp1, 63
GPIO2Mode
  setGPIO2Mode, getGPIO2Mode
    sgp2, ggp2, 64

H

HoldoverTrigger
  setHoldoverTrigger, getHoldoverTrigger
    sht, ght, 142

I

InactivePortTimeout
  setInactivePortTimeout, getInactivePortTimeout
    sit, git, 79
InternalFile
  lstInternalFile
    lif, 65
IonosphereModel
  setIonosphereModel, getIonosphereModel
    sim, gim, 108

L

LBandBeams
setLBandBeams, getLBandBeams
slbb, glbb, 185
LBandCustomServiceID
setLBandCustomServiceID, getLBandCustomServiceID
slcs, glcs, 186
LBandSelectMode
setLBandSelectMode, getLBandSelectMode
slsm, glsm, 187
LogIn
LogIn
login, 80
LogOut
LogOut
logout, 81

M

MagneticVariance
setMagneticVariance, getMagneticVariance
smv, gmv, 110
ManualNotchFilter
setManualNotchFilter, getManualNotchFilter
smnf, gmnf, 99
MarkerParameters
setMarkerParameters, getMarkerParameters
smp, gmp, 148
MIBDescription
lstMIBDescription
lmd, 66
MultipathMitigation
setMultipathMitigation, getMultipathMitigation
smm, gmm, 88

N

NetworkRTKConfig
setNetworkRTKConfig, getNetworkRTKConfig
snrc, gnrc, 111
NMEAOnce
exeNMEAOnce, getNMEAOnce
enoc, gnoc, 160
NMEAOutput
setNMEAOutput, getNMEAOutput
sno, gno, 161
NMEAPrecision
setNMEAPrecision, getNMEAPrecision
snp, gnp, 163
NMEATalkerID
setNMEATalkerID, getNMEATalkerID
snti, gnti, 164
NMEAVersion
setNMEAVersion, getNMEAVersion

snv, gnv, 165

O

ObserverComment
setObserverComment, getObserverComment
soc, goc, 149
ObserverParameters
setObserverParameters, getObserverParameters
sop, gop, 150

P

PeriodicEcho
setPeriodicEcho, getPeriodicEcho
spe, gpe, 157
PPS2Parameters
setPPS2Parameters, getPPS2Parameters
sps2, gps2, 143
PPSParameters
setPPSParameters, getPPSParameters
spps, gpps, 144
PVTMode
setPVTMode, getPVTMode
spm, gpm, 112

R

ReceiverCapabilities
getReceiverCapabilities
grc, 67
ReceiverDynamics
setReceiverDynamics, getReceiverDynamics
srd, grd, 114
ReceiverInterface
getReceiverInterface
gri, 70
RegisteredApplications
exeRegisteredApplications, getRegisteredApplications
era, gra, 71
RemoveFile
exeRemoveFile, getRemoveFile
erf, grf, 184
ResetNavFilter
exeResetNavFilter, getResetNavFilter
ernf, grnf, 115
ResetReceiver
exeResetReceiver, getResetReceiver
erst, grst, 72
RFInterferenceMitigation
setRFInterferenceMitigation, getRFInterferenceMitigation
sitm, gitm, 100
RTCMv3Delay
setRTCMv3Delay, getRTCMv3Delay
sr3d, gr3d, 172
RTCMv3Formatting

setRTCMv3Formatting, getRTCMv3Formatting
sr3f, gr3f, 173
RTCMv3Interval
setRTCMv3Interval, getRTCMv3Interval
sr3i, gr3i, 175
RTCMv3Message1029
setRTCMv3Message1029, getRTCMv3Message1029
sr3m, gr3m, 176
RTCMv3Output
setRTCMv3Output, getRTCMv3Output
sr3o, gr3o, 177
RTCMv3Usage
setRTCMv3Usage, getRTCMv3Usage
sr3u, gr3u, 179

S

SatelliteHealthOverride
setSatelliteHealthOverride, getSatelliteHealthOverride
ssho, gsho, 116
SatelliteTracking
setSatelliteTracking, getSatelliteTracking
sst, gst, 89
SatelliteUsage
setSatelliteUsage, getSatelliteUsage
ssu, gsu, 117
SBASCorrections
setSBASCorrections, getSBASCorrections
ssbc, gsbc, 118
SBFGroups
setSBFGroups, getSBFGroups
ssgp, gsgp, 166
SBFOnce
exeSBFOnce, getSBFOnce
esoc, gsoc, 167
SBFOutput
setSBFOutput, getSBFOutput
sso, gso, 169
SerialTlsSettings
setSerialTlsSettings, getSerialTlsSettings
ssts, gsts, 159
SetTime
exeSetTime, getSetTime
estm, gstm, 119
SignalAuthentication
setSignalAuthentication, getSignalAuthentication
sna, gna, 132
SignalHealthOverride
setSignalHealthOverride, getSignalHealthOverride
snho, gnho, 91
SignalTracking
setSignalTracking, getSignalTracking
snt, gnt, 93

SignalUsage
setSignalUsage, getSignalUsage
snu, gnu, 120
SmoothingInterval
setSmoothingInterval, getSmoothingInterval
ssi, gsi, 94
SolutionSelectivity
setSolutionSelectivity, getSolutionSelectivity
sss, gss, 121
StaticPosCartesian
setStaticPosCartesian, getStaticPosCartesian
sspc, gspc, 122
StaticPosGeodetic
setStaticPosGeodetic, getStaticPosGeodetic
sspg, gspg, 124

T

TimeSyncSource
setTimeSyncSource, getTimeSyncSource
stss, gtss, 146
TimingSystem
setTimingSystem, getTimingSystem
sts, gts, 147
TLSCertificates
lstTLSCertificates
ltc, 74
TroposphereModel
setTroposphereModel, getTroposphereModel
stm, gtm, 125
TropsphereParameters
setTropsphereParameters, getTropsphereParameters
stp, gtp, 127

U

UserAccessLevel
setUserAccessLevel, getUserAccessLevel
sual, gual, 82
UserDatum
setUserDatum, getUserDatum
sud, gud, 137
UserDatumVel
setUserDatumVel, getUserDatumVel
sudv, gudv, 138
UserEllipsoid
setUserEllipsoid, getUserEllipsoid
sue, gue, 139

Index of SBF Blocks

ASCIIIn, 344
AttCovEuler, 295
AttEuler, 294
AuxAntPositions, 296

BaseStation, 316
BaseVectorCart, 281
BaseVectorGeod, 284
BBSamples, 343
BDSAlm, 252
BDSCNav1, 246
BDSCNav2, 248
BDSCNav3, 250
BDSIon, 253
BDSNav, 244
BDSRaw, 213
BDSRawB1C, 214
BDSRawB2a, 215
BDSRawB2b, 216
BDSUtc, 254

ChannelStatus, 319
Commands, 341
Comment, 342

DiffCorrIn, 314
DiskStatus, 334
DOP, 280

EndOfAtt, 297
EndOfMeas, 202
EndOfPVT, 288
ExtEvent, 301
ExtEventAttEuler, 313
ExtEventBaseVectGeod, 310
ExtEventPVTCartesian, 302
ExtEventPVTGeodetic, 306

GALAlm, 239
GALAuthStatus, 337
GALGstGps, 242
GALIon, 240

GALNav, 236
GALRawCNAV, 210
GALRawFNAV, 208
GALRawINAV, 209
GALSARRLM, 243
GALUtc, 241
GEOAlm, 261
GEOIGPMask, 262
GEOIonoDelay, 263
GEONav, 260
GEORawL1, 211
GEORawL5, 212
GLOAlm, 234
GLONav, 233
GLORawCA, 207
GLOTime, 235
GPSAlm, 228
GPSCNav, 231
GPSIon, 229
GPSNav, 226
GPSRawCA, 203
GPSRawL1C, 206
GPSRawL2C, 204
GPSRawL5, 205
GPSUtc, 230

InputLink, 327

LBandRaw, 318
LBandTrackerStatus, 317

MeasEpoch, 195
MeasExtra, 200

NavCart, 289
NavGeod, 292
NavICLNav, 258
NAVICRaw, 225

OutputLink, 330

PosCovCartesian, 272
PosCovGeodetic, 274
PVTCartesian, 264
PVTGeodetic, 268
PVTSupport, 286
PVTSupportA, 287

QualityInd, 333
QZSAlm, 257
QZSNav, 255
QZSRawL1C, 222
QZSRawL1CA, 217
QZSRawL1S, 223

QZSRawL2C, 218
QZSRawL5, 219
QZSRawL5S, 224
QZSRawL6D, 220
QZSRawL6E, 221

ReceiverSetup, 338
ReceiverStatus, 322
ReceiverTime, 298
RFStatus, 335
RxMessage, 340

SatVisibility, 326

VelCovCartesian, 276
VelCovGeodetic, 278

xPPSOffset, 299