Hemisphere GNSS Technical Reference Manual

875-0522-10
Current Version: v4.2 **September 28, 2022**

Contents

Introduction and Quick Start

The Hemisphere GNSS Technical Reference Manual (TRM) is a resource for software engineers and system integrators to configure GNSS receivers. It may also be useful to persons with knowledge of the installation and operation of GNSS navigation systems.

The TRM includes information on GNSS technology and platforms, general operations of receiver, and the commands and messages you need to operate your receiver and/or other HGNSS hardware.

Use the following links to navigate quickly throughout the contents of this manual:

Quick Start - the basic information you need to get started using your Hemisphere GNSS receiver.

GNSS Technology and Platforms -an overview the GNSS engine, satellite tracking information, positioning, accuracy, and update rates of NMEA 0183 and binary messages.

DGNSS Solutions

Receiver Operation introduces general operational features of the receiver operation modes, and default operation parameters.

Commands and Messages are grouped by type (General, GNSS, e-Dif, Data, RAIM, etc.). You can find a listing of all commands in the Commands and Message table. For a more detailed description of each message and command, click the link to navigate to that specific command or message.

Firmware

Resources provides resources for additional information.

Quick Start

Quick Start contains basic information to get you started using your Hemisphere GNSS receiver. What is my receiver type? Send the JHTYPE,SHOW command.

How do I load firmware onto my receiver, and why would I do this? To load firmware, use RightARM,

Loading firmware allows you to run application specific capabilities.

What is my current receiver configuration? Send the JSHOW query.

What is my Vector receiver configuration? Send the JATT,SUMMARY query.

What commands are supported by my receiver?
First, send the JHTYPE_SHOW command to identify the GNSS engine in your receiver. Then, go to the Overview (?) topic for a list of commands supported by that GNSS engine.

Overview for Commands Supported by a GNSS engine

How do I send a command to my receiver?

Connect your receiver to a PC and use a terminal program (i.e., HyperTerminal), or Hemisphere GNSS' PocketMax, or SLXMon. For more information, refer to the User Guide for your product. User Guides can be found on the Hemisphere GNSS website.

How do I turn on data messages (such as GPGGA) for a receiver? See **Configuring the Data Message Output**.

Topic Last Updated: v1.11 / November 15, 2018

GNSS Technology and Platforms

GNSS Engine Overview

The GNSS engine is always operating regardless of the DGNSS mode of operation. The following sections describe the general operation of the receiver.

- Satellite Tracking
- Positioning Accuracy
- Update Rates

Both the GNSS and SBAS operation of the receiver module feature automatic operational algorithms. When powered for the first time, the receiver system performs a "cold start," which involves acquiring the available GNSS satellites in view and the SBAS differential service. To do this, the receiver needs a compatible GNSS antenna connected that offers a relatively clear, unobstructed view of the sky. While you can often achieve this indoors with an antenna placed against a window, you may need to place the antenna outside (i.e., on a roof or a short distance away from the building).

If SBAS is not available in a particular area, an external source of RTCM SC-104 differential correction may be used. If an external source of correction data is needed, the external source needs to support an eight data bit, no parity and one stop bit configuration (8-N-1). See also SBAS Overview.

Topic Last Updated: v1.07 / February 16, 2017

Satellite Tracking

The receiver automatically searches for GNSS satellites, acquires the signal, and manages the associated navigation information required for positioning and tracking. This is a hands-free mode of operation. Satellite acquisition quality is described as a signal-to-noise ratio (SNR. The higher the SNR, the better the signal reception quality. SNR information is provided by the receiver through the use of NMEA 0183 data messages available via its multiple serial ports.

Topic Last Updated: v1.07 / February 16, 2017

Positioning Accuracy

The receiver is a sub-meter product with 95% horizontal accuracy under ideal conditions.

To determine the positioning performance of the receiver, Hemisphere GNSS gathers a 24-hour data set of positions in order to log the diurnal environmental effects and full GPS constellation changes. Data sets shorter than 24 hours tend to provide more optimistic results.

The horizontal performance specification of 95% accuracy is, as stated above, based on ideal conditions. In reality, obstruction of satellites, multipath signals from reflective objects, and operating with poor corrections will detract from the receiver's ability to provide accurate and reliable positions. Differential performance can also be compromised if the receiver module is used in a region without sufficient ionospheric coverage.

Further, if external corrections are used, the baseline separation between the remote base station antennas can affect performance.

The estimated positioning precision is accessible through the use of NMEA 0183 command responses as described in Commands and Messages.

Because the receiver cannot determine accuracy with respect to a known location in real time (traditionally performed in post-mission analyses), the precision numbers are relative in nature and are only approximates.

Topic Last Updated: v1.11 / November 15, 2018

Update Rates

The update rate of each NMEA 0183 and binary message of the receiver can be set independently with a maximum that is dependent upon the message type. For example, some messages have a 1 Hz maximum while other messages have a 50 Hz maximum. The higher update rates, such as 20 Hz, are an option and can be obtained with an additional subscription.

Higher update rates are valuable for applications where:

- Higher speeds are present such as in aviation
- You have manual navigational tasks such as in agricultural guidance
- You have an automated or autonomous navigational task such as in robotics or machine control.

Keep the following in mind regarding message rates:

- Some messages can only be OFF or ON (0 or 1Hz) Example: $JASC,RTCM3,1
- Some messages can only be 0 or 1 Hz, but will come out once first, then only if they change

  Example:
  $JASC,BIN95,1

- Messages that are available at other rates can be set to rates SLOWER than 1 Hz *(see* Note 1 below):

- Example: $JASC,GPGGA,0.1

- If the receiver is subscribed to 10 or 20Hz, the receiver can log at rates FASTER than 1 Hz *(See Note 2 below.)*

  Example: $JASC,GPGGA,5
  Note 1: Slower than 1 Hz.

  Use the following guidelines:

To log once every x seconds	**Use JASC,xxxx,**
2	0.5
3	0.3333
4	0.25
5	.2
6	0.1667
7	0.1429
8	0.125
9	0.1111
10	0.1
15	0.0667
20	0.05
25	0.04
40	0.025
50	0.02
100	0.01
120	0.0083

Rates not listed above may be possible but may not log on integer seconds. Users should test to see if the results are acceptable for their application.
Note 2: Faster than 1Hz, if subscribed.

For traditional firmware support is 20 Hz, Acceptable rates are 1, 2, 4, 5, 10 or 20 Hz. Using rates other than those listed will result in data appearing in a rate similar to the rate requested, but the data times will be quantized to 0.05 second resolution. This is due to the receiver's internal computing rate of 20 Hz. Time resolution is 0.05 seconds even if the receiver is only subscribed for 10 Hz data. Quantizing may result in a slightly different number of messages per minute than expected. For example, 3 Hz data produces approximately 172 messages per minute due to quantizing, instead of the expected 180 messages.

Some products and firmware support 50 Hz. Acceptable rates are 1, 2, 5, 10, 25, or 50. Using rates other than a factor of 20 Hz may result in quantized data. Regardless, the data in the message is referenced to the time of the message.

Topic Last Updated: v1.11 / November 15, 2018

Hemisphere GNSS Hardware Platforms

Hemisphere GNSS offers many hardware platforms, some of which include:

- Positioning OEM boards and receivers
- Heading OEM boards and receivers

Topic Last Updated: v1.11 / November 15, 2018

Universal Development Kit

The Universal Development Kit allows you to integrate a Hemisphere GNSS OEM board into your design and includes the following: Main carrier board

- Adapter board
- Power cable and AC power supply
- Serial data cable, power cable and AC power supply, ethernet cable, serial cable null modem, USB-C to USB-A cable

The Universal Development Kit supports the following Hemisphere GNSS OEM boards:

- Phantom 20
- Phantom 34
- Phantom 40
- Vega 28
- Vega 40
- Vega 60

Depending on the Hemisphere GNSS OEM board you purchase with your Universal Development Kit, an Integrator Guide is available for download from the Hemisphere GNSS website.

Last Updated: v4.0 / June 30, 2020

SBAS Overview

The following topics describe the general operation and performance monitoring of the Space-Based Augmentation System (SBAS) demodulator within the receiver module:

- SBAS Constellations
- SBAS Automatic Tracking
- SBAS Performance
- SBAS Corrections and Signal Information
- WAAS Coverage

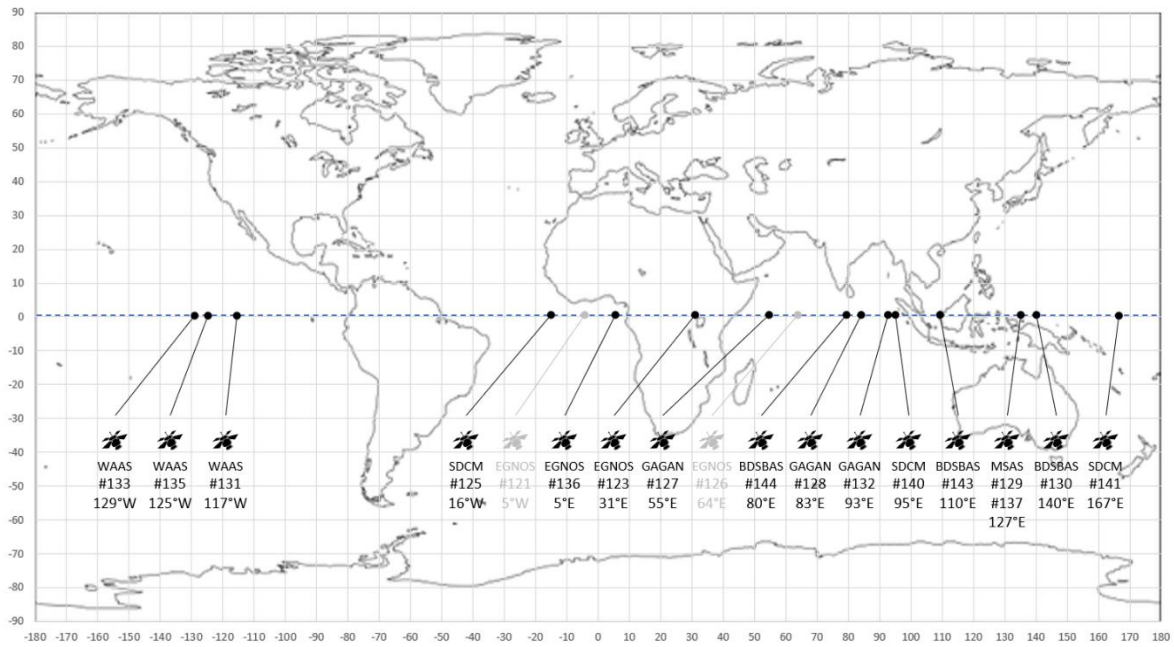
Topic Last Updated: v4.2 / September 13, 2022

SBAS Constellations

SBAS constellations have been set up for use by the aviation administrations for various airspaces globally. At the time of this topic update the following SBAS regions were in operation:

- WAAS, operated by the USA, covering North America and Hawaii
- EGNOS, operated by Europe, covering Europe and parts of northern Africa
- MSAS, operated by Japan, covering Japan and surrounding region
- GAGAN, operated by India, covering the region surrounding India
- SDCM, operated by Russia, covering the Russian region

SBAS corrections can be used within these regions and will have gradually degraded performance as the distance from the nearest region increases.



SBAS satellites are identified by their pseudo-range-number (PRN). In some areas, two or more satellites may be visible. SBAS satellites stay positioned over the equator at a specific longitude, as shown. The gray satellites represent test satellites which may or may not be transmitting SBAS corrections.

Topic Last Updated: v4.2 / September 13, 2022

SBAS Automatic Tracking

The SBAS demodulator features two-channel tracking that enhances the receiver's ability to maintain acquisition on SBAS signals when more than one SBAS satellite is in view. This redundant tracking approach results in more consistent signal acquisition in areas where signal blockage of either satellite is possible. See the JWAASPRN command for information on tuning the receiver to specific SBAS satellites.

Topic Last Updated: v4.2 / September 13, 2022

SBAS Performance

SBAS performance is described in terms of bit error rate (BER). The SBAS receiver requires a line of sight to the SBAS satellite to acquire a signal.

The BER number indicates the number of unsuccessfully decoded symbols in a moving window of 2048 symbols. Due to the use of forward error correction algorithms, one symbol is composed of two bits. The BER value for SBAS receiver channels is available in the RD1 message.

A lower BER indicates data is being successfully decoded with fewer errors, providing more consistent throughput. The BER has a default no-lock of 500 or more. As the receiver begins to successfully acquire a signal, a lower BER results. For best operation, this value should be less than 150 and ideally less than 20.

SBAS broadcasts an ionospheric map on a periodic basis, and it can take up to five minutes to receive the map on startup. Until it downloads the SBAS map the receiver uses the broadcast ionosphere model, which can result in a lower performance compared to when the map has been downloaded. This is the case for any GNSS product supporting SBAS services.

Note: Signal coverage may be present in some areas without either sufficient ionospheric map coverage or satellites with valid orbit and clock corrections. In such cases performance may be degraded compared to areas fully covered by the SBAS ionospheric coverage.

Topic Last Updated: v.4.2 / September 13, 2022

SBAS Corrections and Signal Information

SBAS services take in reference data from a network of base stations to model the sources of error directly, rather than computing the sum impact of errors upon observed ranges. The advantage of this approach is that the error source can be more specifically accounted for during the correction process.

Specifically, SBAS calculates separate corrections for ionospheric, satellite timing, and satellite orbit errors.

SBAS systems transmit correction data on the same frequency as GNSS L1 signals, allowing GNSS receivers to receive the signals without additional equipment or antennas.

Topic Last Updated: v4.2 / September 13, 2022

Atlas

The Atlas signal is a line-of-sight L-band signal that is similar to GNSS. For the Atlas differential receiver to acquire the signal, there must be a line of sight between the antenna and the geostationary communications satellite.

Various Atlas communications satellites are used for transmitting the correction data to Atlas users around the world. When the Atlas receiver has acquired an Atlas signal, the elevation and azimuth are available in the menu system to enable troubleshooting line-of sight problems.

Contact your Atlas service provider for further information on this service.

Topic Last Updated: v1.11/November 15, 2018

Atlas Reception

Atlas services broadcast at a similar frequency to GNSS and as a result is a line-of-sight system; there must be a line of sight between the antenna and the Atlas satellite for reception of the service.

Atlas services use geostationary satellites for communication. The elevation angle to these satellites is dependent upon latitude. For latitudes higher than approximately 55° North or South, the Atlas signal may be blocked more easily by obstructions such as trees, buildings, and terrain.

Topic Last Updated: v1.07/ February 16, 2017

Atlas Automatic Tracking

The Hemisphere GNSS Atlas receiver features an automatic mode that allows it to locate the best spot beam if more than one is available in a particular region. With this function you do not need to adjust the receiver's frequency. The receiver also features a manual tune mode for flexibility.

See the JFREQ command for more information on automatic and manual tuning.

Topic Last Updated: v1.07 / February 16, 2017

Atlas Receiver Performance

Atlas receivers provide both a lock indicator and a BER (bit error rate) to describe the lock status and reception quality. Both these features depend on a line of sight between the antenna and the geostationary communications satellite broadcasting the Atlas correction information.

Atlas capable Hemisphere GNSS antennas are designed with sufficient gain at low elevation angles to perform well at higher latitudes where the signal power is lower and the satellite appears lower on the horizon. The BER number indicates the number of unsuccessfully decoded symbols in a moving

window of 2048 symbols. Because of the use of forward error correction algorithms, one symbol is composed of two bits.

The BER has a default, no-lock value of 500. As the receiver begins to successfully acquire the signal a lower BER results. For best operation this value should be less than 150 and ideally less than 20.

Topic Last Updated: v1.07 / February 16, 2017

DGPS Operation

DGPS Base Station Performance

Base station performance depends primarily on the site location for the base station GNSS antenna. An ideal location would have no obstructions above the height of the antenna, offering a full 180º by 360º view of the sky. In reality, obstructions such as trees, vehicles, people, and buildings nearby both block satellite signals and reflect interfering signals called multipath signals. Multipath degrades the accuracy of the satellite measurements and detracts from the receiver's ability to provide accurate and reliable corrections for the rovers.

For a rover to work optimally, a base station should be near by the rover's area of operation. As distance from the base to the rover increases, the modeling process cannot tune the solution to the exact environmental conditions at the rover's location and the rover's accuracy will not be as good. Best performance is attained when the distance from your base to your rover is less than 50 km (30 miles).

The Hemisphere receiver with an e-Dif subscription can operate in a DGPS base station mode. JRAD commands need to be sent to the receiver to enter this mode. These commands may be automatically issued through customized software or through a simple terminal interface running on a PC or handheld device. DGPS Base Station Commands provide detailed information on the commands supported by the base station application.

Topic Last Updated: v1.11/November 15, 2018

DGPS Base Station Startup

When the receiver running the e-Dif application first starts up, it requires a few minutes to gather enough satellite tracking information to model the errors for the future. Once commands are sent to put the receiver into base station mode, corrections will be generated and can be sent via the serial port to rover receivers. In some more challenging GNSS environments, the time required to model errors can take up to 10 minutes. The receiver must be stationary during this process and the antenna for the base station must be secured in a stable location.

Topic Last Updated: v1.11/November 15, 2018

DGPS Base Station Calibration

Base station calibration is the process of modeling the errors at the base station. Calibration can be performed in either a relative or an absolute sense, depending on positioning needs. Relative positioning provides positions that are accurate to one another but there may be some offset from the true geographical position.

Calibrating for relative positioning is easier than for absolute position since you are not restricted to using a point with known coordinates. Calibrating for absolute positioning mode requires placing the GPS antenna at a known reference location. Care should be taken to use a location that has good sky visibility and is relatively free from obstructions.

Topic Last Updated: v1.11/November 15, 2018

e-Dif

The Hemisphere receiver module is designed to work with Hemisphere GNSS' patented Extended Differential (e-Dif) software. e-Dif is an optional mode where the receiver can perform with differential-like accuracy for extended periods of time without the use of a differential service. It models the effects of ionosphere, troposphere, and timing errors for extended periods by computing its own set of pseudo-corrections.

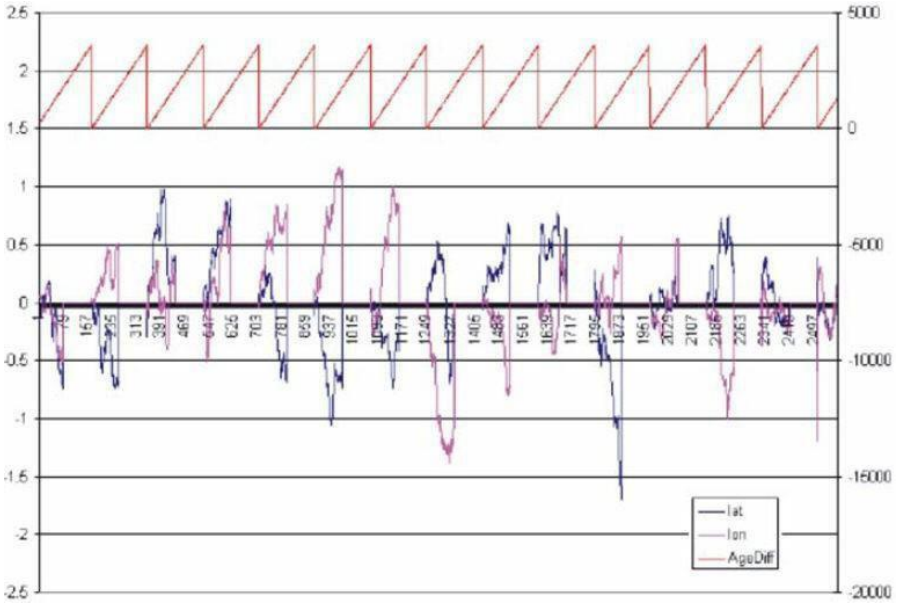
e-Dif may be used anywhere geographically and is especially useful where SBAS networks have not yet been installed, such as South America, Africa, Australia, and Asia. Two things are required to enable e-Dif. First your receiver will require standard MFA application software to be installed on it. A software key, called a subscription code, is needed for the receiver to use e-Dif. Both can be installed in the field using a PC computer. See Using RightARM to Load Firmware if you need to install the application firmware onto your receiver. To install a subscription code, contact Hemisphere GNSS for a JK command which can be issued to your receiver.

Positioning with e-Dif is jump-free compared to a receiver working with just raw GPS provided the receiver consistently maintains a lock on at least four satellites at one time. The accuracy of positioning will have a slow drift that limits use of the e-Dif for approximately 30 to 40 minutes although it depends on how tolerant the application is to drift as e-Dif can be used for longer periods.

This mode of operation should be tested to determine if it is suitable for the application and for how long the user is comfortable with its use. As accuracy will slowly drift, the point at which to recalibrate e-Dif to maintain a certain level of accuracy must be determined.

The figure below displays the static positioning error of e-Dif while it is allowed to age for fourteen consecutive cycles of 30 minutes. The top line indicates the age of the differential corrections. The receiver computes a new set of corrections using e-Dif during the calibration at the beginning of each hour and modifies these corrections according to its models. After the initialization, the age correspondingly increases from zero until the next calibration.

The position excursion from the true position (the lines centered on the zero axis are northing [dark line] and easting [light line]) with increasing correction age is smooth from position to position; however, there is a slow drift to the position. The amount of drift depends on the rate of change of the environmental errors relative to the models used inside the e-Dif software engine.



Note: You decide how long e-Dif is to function before between calibrations, and you should test this operation mode to determine an acceptable level of performance.

Topic Last Updated: v1.11/November 11, 2018

RTK Overview

Real Time Kinematic (RTK) positioning is the highest form of navigational accuracy for GNSS receivers. Hemisphere GNSS offers RTK for multi-frequency platforms. See RTK commands for more information.

Topic Last Updated: v4.2 / September 13, 2022

Post-Processing

Hemisphere receiver modules can output raw measurement data for post processing applications. The raw measurement and ephemeris data are contained in the following messages, which must be logged in a binary file:

Observations: Bin 16 (all GNSS)

Ephemeris: Bin 95 (GPS), Bin 65 (GLONASS), Bin 35 (BEIDOU), Bin 45 (GALILEO)

Time conversion: Bin 94 (GPS), Bin 34 (BEIDOU), Bin 44 (GALILEO)

Depending on the application, the binary data can be logged to a file and then translated to RINEX at a later time on a PC.

Contact Hemisphere GNSS technical support for a RINEX translator.

Because there is limited ability to store station information in the binary file, developers may consider writing their own translator.

Topic Last Updated: v4.2 / September 13, 2022

Evaluating Receiver Performance

Hemisphere GNSS evaluates performance of the receiver with the objective of determining best-case performance in a real-world environment. Our testing has shown that the receiver achieves a performance better than 0.6 m 95% of the time in typical DGPS modes.

The qualifier of 95% is a statistical probability. Manufacturers often use a probability of RMS, one sigma, or one standard deviation. These three terms all mean the same thing and represent approximately 67% probability. Performance measures with these probabilities are not directly comparable to a 95% measure since they are lower probability (less than 70% probability).

Table 1 summarizes the common horizontal statistical probabilities.

Table 1: Horizontal Accuracy Probability Statistics

Accuracy Measure	Probability (%)
rms (root mean square)	63 to 68
CEP (circular error probability)	50
R95 (95% radius)	95 to 98
2drms (twice the distance root)	95

It is possible to convert from one statistic to another using Table 2. Using the value where the 'From' row meets the 'To' column, multiply the accuracy by this conversion value.

Table 2: Accuracy Conversions

	To			
From	CEP	rms	R95	2drms
CEP	1	1.2	2.1	2.4
rms	0.83	1	1.7	2.0
R95	0.48	.59	1	1.2
2drms	0.42	.5	.83	1

For example, Product A, after testing, has an accuracy of 90 cm 95% of the time (R95). To compare this to Product B that has a sub-meter horizontal rms specification of 60 cm:

1. Select the value from where the 'R95' row and the 'rms' column intersect (to convert to rms). This conversion value is 0.59.

2. Multiply the 90 cm accuracy by this conversion factor and the result is 53 cm rms. Compared to Product B's 60cm specification of sub-meter rms, Product A offers better performance.

To properly evaluate one receiver against another statistically, the receivers should be using identical correction input (from an external source) and share the same antenna using a power splitter (equipped with appropriate DC-blocking of the receivers and a bias-T to externally power the antenna). With this setup, the errors in the system are identical with the exception of receiver noise.

Although this is a comparison of the GNSS performance qualities of a receiver, it excludes other performance merits of a GNSS engine. The dynamic ability of a receiver should always be compared in a similar way with the test subjects sharing the same antenna. Unless a receiver is moving, its software filters are not stressed in a similar manner to the final product application. When testing dynamically, a much more accurate reference would need to be used, such as an RTK system, so that a "truth" position per epoch is available.

Further, there are other performance merits of a GNSS engine such as its ability to maintain a lock on GNSS and SBAS satellites. When evaluating this ability, the same GNSS antenna should be shared between the receivers' test subjects. For the sake of comparing the tracking availability of one receiver to another, no accurate "truth" system is required unless performance testing is also to be analyzed. Again, an RTK system would be required; however, it is questionable how its performance will fare with environments where there are numerous obstructions such as foliage. Other methods of providing a truth reference may need to be provided through observation times on surveyed monuments or traversing well-known routes.

Should you look to compare two RTK systems, determining truth can be very complicated. A rigorous dynamic comparison of two competing RTK systems should only be attempted by individuals and organizations familiar with RTK and potentially with inertial navigation equipment. Fortunately, most manufacturer's RTK performance is specified in similar accuracy values, and in general, RTK accuracy is quite similar across different manufacturers.

Topic Last Updated: v1.07 / February 16, 2017

Receiver Operation

When turned on, the receiver goes through an internal startup sequence. It is, however, ready to communicate immediately. Refer to the receiver-specific manual for the power specifications of the product. When its antenna has an unobstructed view of the sky, the receiver provides a position in approximately 60 seconds and acquires SBAS lock in approximately 30 seconds.

Topic Last Updated: v1.11/ November 15, 2018

Communicating with the Receiver

The receiver module features three primary serial ports (A, B, C) that may be configured independently of each other. The ports can be configured to output a combination of data types:

- NMEA 0183
- Hemisphere GNSS proprietary binary and ASCII formats
- RTCM v2.x, 3.x, proprietary ROX and CMR

The usual data output is NMEA 0183 messages because these are the industry standard.

Topic Last Updated: v1.11 / November 15, 2018

Configuring the Receiver

You can configure all aspects of receiver operation through any serial port using NMEA 0183 commands. You can select one of the two on-board applications.

- Two applications may be loaded at the same time, but only one can be active.
- You can select the active application through serial commands or through menu options on products with displays
- Set the baud rate of communication ports
- Select NMEA 0183 data messages to output on the serial ports and select the output rate of each message
- Set the maximum differential age cut-of
- Set the satellite elevation angle cut-off mask

The appropriate commands are described in Commands and Messages.

Topic Last Updated: v1.07 / February 16, 2017

Saving the Receiver Configuration

Each time the configuration of the receiver is changed, the new configuration should be saved so the receiver does not have to be reconsidered for the next power cycle.

To save the settings issue the JSAVE command. The receiver records the current configuration to non-volatile memory. The receiver indicates when the save process, which takes about five seconds is complete.

Topic Last Updated: v1.00 / August 11, 2010

RTCM SC-104 Protocol

RTCM SC-104 is a standard that defines the data structure for differential correction information for a variety of differential correction applications. It was developed by the Radio Technical Commission for Maritime services (RTCM) and has become an industry standard for communication of correction information. RTCM is a binary data protocol and is not readable with a terminal program. Because it is a binary format and not ASCII text, it appears as "garbage" data on screen.

See Reference Documents for RTCM contact information to purchase a copy of the RTCM SC-104 specifications.

Topic Last Updated: v1.07 / February 16, 2017

Hemisphere GNSS Proprietary Binary Interface

Hemisphere GNSS proprietary binary messages may be output from the receiver simultaneously with

NMEA 0183 messages.

Binary messages are inherently more efficient than NMEA 0183 and would be used when maximum communication efficiency is required. Some receiver-specific pieces of information are only available through binary messages, such as raw data for post processing.

Topic Last Updated: v1.06 / March 10, 2015

Subscriptions Codes

This section covers:

- Finding the serial number and inputting a subscription code (e-Dif, RTK, 20 Hz or 10Hz, etc.) into a Hemisphere GNSS receiver
- Viewing the status and interpreting the $JI subscription date codes
- The difference between the receiver's response to the $JK and $JI commands

Topic Last Updated: v1.07/ February 16, 2017

Interpreting the $JK 'Date'/Subscription Codes

Subscription codes enable GNSS differential correction sources on your receiver. When discussing them it is important to understand the following.

The YYYY component of a MM/DD/YYYY formatted date—returned by the JK command—is not always just the year component of that date. When a date's year starts with 30, only the 30 represents the year - and that year is 3000. A subscription expiration date of 01/01/3000 effectively means there is no expiration date.

The last two digits of the 30YY 'date' represent the data output rate and the GNSS differential correction sources that have been subscribed to and are therefore enabled on your receiver. Hemisphere GNSS refers to these two digits as the Additive Code (see Understanding Additive Codes).

The 30 and the 00 in the 'year' 3000, then, represents "Expires 3000 (so effectively does not expire), the data rate is 10 Hz, and SBAS is enabled." The 'year' 3015 indicates "Expires 3000, the data rate is 20 Hz and differential correction sources SBAS/e-Dif/RTK and L-Dif have been subscribed to and are enabled."

Below is an example of the $JK command response, part of which is the subscription start and expiration dates (the Date Code is shaded).

$>JK,01/01/3000,0

Topic Last Updated: v1.09 / January 8, 2018

Understanding Additive Codes

There are several ways to check which activations and subscriptions are on your receiver. Activations and subscriptions are typically shown as a bitmask. The definition of the bitmask is as follows:

Hex Value	Subscription/ Activation	Description
0x01	20HZ	Add the ability to output some messages at rates up to 20Hz
0x02	EDif	Allow for eDif capability
0x04	RTK	Allow for RTK capability
0x08	LDiff	Allows for 15cm RTK (if you do not have a 0x4 activation)
0x10	RAW	Allow for the output of raw binary GNSS observations for converting to Rinex
0x20	mFreq	Allow for multi-frequency GNSS
0x40	mGNSS	Allow for multi-constellation GNSS If mGNSS is activated without mFreq, the receiver can use L1, G1, E1BC, B1; if the receiver has an mFreq activation without mGNSS, the receiver will be activated for L1, L2, L5; if the receiver has both activations, the receiver can use every signal
0x100	Heading	Allow the output of heading messages on Vega boards
0x400	Atlas L-Band tracking	Track the signal and use aRTK; an H10, H30, or Atlas Basic subscription is required to output an Atlas position solution
0x800	Atlas H10	Allow the use of Atlas H10. Atlas H10 codes are provided with 0x800, 0x400, 0x40, and 0x20 as temporary subscriptions.
0x1000	Atlas H30	Allow the use of Atlas H30. Atlas H30 codes are provided with 0x1000, 0x400, 0x40, and 0x20 as temporary subscriptions.
0x2000	Atlas Basic	Allows the use of Atlas Basic. Atlas Basic codes are provided with 0x2000, 0x400, and 0x40 as temporary subscriptions.
0x4000	Atlas Offshore	Atlas H10, Atlas H30, and Atlas Basic are geofenced to within 20km of major landmasses. The Atlas Offshore bit, ordered in conjunction with Atlas H10, H30, or Atlas Basic, removes the geofence and allows for the use of Atlas globally.
0x8000	50Hz	Add the ability to output some messages at rates up to 50Hz

The less common downgrade codes are listed below:

Hex Value	Subscription/ Activation	Description
0x01	1HZ	Restrict the output of messages to 1Hz
0x02	5HZ	Restrict the output of messages to 5Hz
0x04	0HZ	Restrict the output of all messages
0x08	China geofence	Prevent Atlas from working outside of China
0x10	Beidou only	Restrict the solution to only Beidou
0x20	Low Performance	Low Performance Heading (artificially adds noise to heading solution)

Send the $JK,SHOW command to check which activation and subscriptions are on your receiver.

$>JK,SHOW,475,C60,12/31/2025,0,OPT=,20Hz,RTK,RAW_DATA,L2_L5,MULTI_GNSS,ATLAS_LBAND,ATLAS_10cm

This command lists the receiver activations, subscriptions, and downgrades. In the example above, the receiver is activated and/or subscribed for 20Hz, RTK, raw data output, mFreq (L2_L5), mGNSS, Atlas tracking, and H10. However, some are permanent activations and others are temporary subscriptions. To distinguish one from another, check the 3rd, 4th, and 6th fields.

The 3rd field (475 in this case) is a **hexadecimal** bitmask showing what your receiver is activated for permanently. 0x475 = 0x01 | 0x04 | 0x10 | 0x20 | 0x40 | 0x400. Match those with the above, and we have 20Hz, RTK, raw data, mFreq, mGNSS, and Atlas tracking that will never expire.

Next, the following field (C60) is a hexadecimal bitmask of what will expire (the expiration date is in the following field, 31st of December 2025).

0xC60 = 0x800 | 0x400 | 0x40 | 0x20. This receiver has a temporary subscription for Atlas H10, L-band tracking, mGNSS, and mFreq. Those subscriptions will expire. However, please note that even after the subscription expires, the receiver will still have mGNSS, mFreq, and Atlas L-band tracking since those are also permanent activations.

The field after the date (0 in this case) is the downgrade bit field. This receiver doesn't have any downgrade bits.

The $JK command response:

$>JK,12/31/2025,C75

C75 is a hexadecimal bitmask of subscriptions and activations, but it does not specify what will expire on the 31st of December 2025. Therefore, this command is less desirable.

0xC75 = 0xC60 | 0x475.

Another option is the $JI response. This response gives you a **decimal** bitmask + 3000. In the example below, the bitmask is 6189. 6189 – 3000 = 3189 = 0xC75.

$>JI,21000380,20,1,04092019,01/01/1900,01/01/6189,6.0Aa01,0

Comparing the JI and JK Responses

Example 1:

In the following examples, the Date Code is shaded. JI query date code example:

$>JI,311077,1,7,04102005,01/01/1900,01/01/3000,6.8Hx,46

JK query date code example:

$>JK,01/01/3000,0,(1, 2, 5 or no number)

In the JK example the last two digits ('00') of the Date Code ('3000') represent the Hex Code (the second column of Table 2 above).

The last digit to the right (1, 2, 5 or no number) is the Downgrade Code...this is the output rate in Hertz indicating a downgrade from the default of 10 Hz. So, if 1, 2 or 5 does not appear (no number), the output rate is the default 10 Hz.

The Date Codes are identical in either query or are directly related to each other. Also, the last digit in the JK query is the hexadecimal equivalent of the last two digits in the Date Code. The following example further illustrate this (Date Code is shaded).

Note: The JI response provides the decimal Date Code while the JK response provides both the decimal Date Code and the hex Date Code (the Hex Code).

Example 2:

$>JI,311077,1,7,04102005,01/01/1900,01/01/3015,6.8Hx,46

JK query date code example:

$>JK,01/01/3015,F

In this example the last two digits ('15') of the Date Code ('3015') is the decimal equivalent of the last value ('F'), which is the Hex Code (see the last row in Table 1 above). Example shows no downgrade code.

Topic Last Updated: v4.0 / June 30, 2020

Ethernet Configuration

Some receivers have support for Ethernet. It is disabled by default but may be enabled with the $JETHERNET serial command.

Enabling and Disabling Ethernet

To start, the full current state of Ethernet configuration may be checked with the command "**$JETHERNET**". When ethernet is disabled, the following is displayed:

$JETHERNET

$>JETHERNET,MAC,8C-B7-F7-F0-00-01

$>JETHERNET,MODE,OFF

$>JETHERNET,PORTI,OFF

$>JETHERNET,PORTUDP,OFF

$>JETHERNET,NTRIPCLIENT,OFF

$>JETHERNET,IPADDRESS,NONE

To enable Ethernet, you first need to know if you are going to allow the receiver to be assigned an IP address automatically via DHCP, or statically assigned. If you are unsure, please contact the administrator of the network you wish to connect it to.

To enable Ethernet support with a DHCP-assigned IP address, simply use the command "**$JETHERNET,MODE,DHCP**". The receiver will attempt to get an address from the DHCP server on the network. You should be able to see the current IP address reported by a "**$JETHERNET**" query change.

To enable Ethernet support with a statically assigned IP address, use the command "**$JETHERNET,MODE,STATIC,ip,subnet,gateway,dns**" where **ip**/**subnet**/**gateway**/**dns** are each replaced with the relevant IP address. The **gateway**and **dns**parameters are optional, and only useful for allowing outgoing connections from the P328, which are not currently supported anyway. An example command would be "**$JETHERNET,MODE,STATIC,192.168.0.42,255.255.255.0**"

If one wishes to disable Ethernet use the command "**$JETHERNET,MODE,OFF**"

Enabling Network Services

With Ethernet enabled, it should be possible to send an ICMP ping to the receiver from a PC on the same network if one wishes to test that. No actual services are enabled on Ethernet by default however though, so to make practical use of Ethernet support, one must also enable a service.

As of the v6.0.0 firmware, the only services implemented the PORTI virtual serial port, PORTUDP, and NTRIPCLIENT. Additional types of network services may be implemented in future firmware versions.

For details regarding these services, please reference the relevant $JETHERNET,* command documentation for the service in question.

For sake of example, it is possible to enable the PORTI virtual serial port as a TCP server. Once a connection to it is made, it will act just like a local serial port of the receiver would. Only one TCP client may be connected to it at a time.

Important Note: Enabling PORTI as a TCP server should only be done when the network the receiver is connected to is considered to be a trusted network, since it gives full access to the receiver just as a local serial port would and has no authentication mechanism.

To enable the PORTI service as a TCP server, use the command "**$JETHERNET,PORTI,port**" where **port** is replaced with the TCP port number which one wishes to use. Any port in the range 1 to 65535 is allowable, but it is recommended one consider which TCP port numbers are typically reserved for various common protocols and avoid those port numbers.

To disable the PORTI service, use the command "**$JETHERNET,PORTI,OFF**".

As an additional note, when the connected to a network, the receiver can be accessed with a hostname of "hgnss########.local" where ######## is replaced with the receiver's electronic serial number as is reported by the $JI command. This can make it easier to connect to a receiver on a local network that was assigned an IP address by DHCP, so you do not need to check which IP address it was assigned.

Topic Last Updated: v3.0/December 30, 2019

Enabling Ethernet Services

With Ethernet enabled, it should be possible to send an ICMP ping to the P328 receiver from a PC on the same network, if one wishes to test that. No actual services are enabled on Ethernet by default however though, so to make practical use of Ethernet support, one must also enable a service.

As of the writing of this document, the only Ethernet service implemented is the PORTI virtual serial port. Additional types of Ethernet services may be implemented in future firmware versions.

The PORTI virtual serial port allows a listening TCP port to be opened, which will act just like a local serial port of the receiver would. Only one TCP client may be connected at a time.

Important Note: Enabling "PORTI" on Ethernet should only be done with the P328 connected to a trusted network, since it gives full access to the receiver just as a local serial port would and has no authentication or security mechanisms.

To enable the PORTI service, use the command:

$JETHERNET,PORTI,port

where **port** is replaced with the TCP port number which one wishes to use. Any port in the range 1 to 65535 is allowable, but it is recommended one consider which TCP port numbers are typically reserved for various common protocols and avoid those port numbers.

To disable the PORTI service, use the command:

$JETHERNET,PORTI,OFF

Topic Last Updated: v1.07 / February 16, 2017

Commands and Messages

The receiver supports a selection of NMEA 0183 messages, proprietary messages that conform to NMEA 0183 standards, and Hemisphere GNSS proprietary binary messages. It is your decision as a systems designer whether or not to support a NMEA 0183-only software interface or a selection of both NMEA 0183 and binary messages.

All Crescent and Eclipse receivers are configured with NMEA 0183 commands and can output NMEA 0183 messages. In addition to NMEA 0183, some receivers can be configured using NMEA 2000 commands and can output NMEA 2000 messages.

Atlas Commands

The following tables lists the commands accepted by the Atlas-band receiver to configure and monitor the Atlas functionality of the receiver.

Command	Description
$JI	Requests the serial number and firmware version number from the receiver
$JK	Is used to send an authorization code to the receiver
$JK,SHOW	Requests the subscription and activation information from the receiver
$JASC,GPGGA,1	Requests receiver to output GGA positions at 1Hz.
$JASC,RD1,1	Enables Atlas Diagnostic message output
$JDIFF,LBAND,SAVE	Enables Atlas mode for tracking the Atlas communication satellites
$JDIFF,INCLUDE,ATLAS	Enables the Atlas solution in the receiver
$JFREQ,AUTO	Automatically sets the Atlas parameters to track the Atlas communication satellites
$JATLAS,LIMIT	Configure the accuracy threshold for when the NMEA 0183 GPGGA message reports a quality indicator of 4. See $JATLAS,LIMIT, section for more detail
$JSAVE	Saves issued commands

Note: Use the JSAVE command to save changes you need to keep and wait for the $J>SAVE COMPLETE response.

If your Atlas communication is working properly the following should apply:

Bit Error Rate: less than 10-10

Spot Beam Freq:

- **AMERICAS**: 1545.915
- **APAC**: 1545.855
- **EMEA**: 1545.905

Nav Condition: FFFFF

If this is not the case, then enter the following commands in the Receiver Command Page, one at a time:
Command: $JFREQ,AUTO, $JDIFF,LBAND,SAVE

Base IDs:
4715- Atlas with ITRF08 reference frame (default)
4716-Atlas with GDA94 reference frame*
4717-Atlas with User-specific local reference frame (Cartesian)*

*Available on select products

Topic Last Updated: v4.0/ June 30, 2020

DGPS Base Station Commands

The following table lists the commands supported by the base station feature for its control and operation.

Command	Description
JRAD,1	Display the current reference position in e-Dif applications only
JRAD,1,LAT,LON,HEIGHT	Use this command—a derivative of the JRAD,1,P command—when absolute positioning is required in e-Dif applications only
JRAD,1,P	**e-Dif:** Record the current position as the reference with which to compute e-Dif corrections. This would be used in relative mode as no absolute point information is specified. **DGPS Base Station:** Record the current position as the reference with which to compute Base Station corrections in e-Dif applications only. This would be used in relative mode as no absolute point information is specified.
JRAD,9	Initialize the Base Station feature and use the previously entered point, either with $JRAD,1,P or $JRAD,1,LAT,LON,HEIGHT, as the reference with which to compute Base Station corrections in e-Dif applications only. Use this for both relative mode and absolute mode.
JRAD,10	If $JRAD,10,1 is entered, the diff output will be RTCM2.4

Topic Last Updated: v2.0/ April 30, 2019

e-Dif Commands

The following table lists the commands supported by the e-Dif application for its control and operation.

Command	Description
JRAD,1	Display the current reference position in e-Dif applications only
JRAD,1,LAT,LON,HEIGHT	Use this command—a derivative of the JRAD,1,P command—when absolute positioning is required in e-Dif applications only
JRAD,1,P	e-Dif: Record the current position as the reference with which to compute e-Dif corrections. This would be used in relative mode as no absolute point information is specified. DGPS Base Station: Record the current position as the reference with which to compute Base Station corrections in e-Dif applications only. This would be used in relative mode as no absolute point information is specified
JRAD,2	Forces the receiver to use the new reference point (you normally use this command following a JRAD,1 type command)
JRAD,3	Invoke the e-Dif function once the unit has started up with the e-Dif application active, or update the e-Dif solution (calibration) using the current position as opposed to the reference position used by the JRAD,2 command
JRAD,7	Turn auto recalibration on or off

Note: Use the JSAVE command to save changes you need to keep and wait for the $>SAVE COMPLETE response.

Topic Last Updated: v1.02 / January 25, 2011

GALILEO Commands and Messages

The following table lists the commands applicable to GALILEO-capable receivers.

Command	Description
JASC,GAGSV	Enable/disable the data for GALILEO satellites in view. When turning messages on, various update rates are available depending on the requirements.
JASC,GNGNS	Enable/disable fix data for GNSS systems including GALILEO (GAGNS). When turning messages on, various update rates are available depending on the requirements.
JNMEA,GGAALLGNSS	Configure the GGA string to include full GNSS information (the number of used satellites will be included in the GPGGA message) or query the current setting

The following table lists the messages applicable to GALILEO-capable receivers.

Message	Description
Bin45	GALILEO ephemeris information
Bin16	GALILEO GNSS code and phase observation information
Bin44	GALILEO time conversion information

*Note: For observations in tracking status, see GNSS, Bin 16 & Bin 19.

Topic Last Updated: v2.0/ April 30, 2019

General Operation and Configuration Commands

The following table lists the commands related to the general operation and configuration of the receiver.

Command	Description
JAIR	Specify how the receiver will respond to the dynamics associated with airborne applications
JALT	Turn altitude aiding for the receiver on or off
JAPP	Specify or query receiver application firmware
JASC,D1	Set the RD1 diagnostic information message from the receiver to on or off
JASC,VIRTUAL	Configure the receiver to have RTCM data input on one port and output through the other (when using an external correction source)
JBAUD	Specify the baud rates of the receiver or query the current setting
JBOOT	

JBIN	Enable the output of the various binary messages supported by the receiver
JCONN	Create a virtual circuit between the A and B ports to enable communication through the receiver to the device on the opposite port
JDIFF	Specify or query the differential mode of the receiver
JDIFF,AVAILABLE	Query the receiver for the differential types currently being received
JDIFFX,EXCLUDE	Specify the differential sources to be excluded from operating in a multi-diff application
JDIFFX,GNSSOUT	Specify GNSS output in correction formats or query the current setting
JDIFFX,INCLUDE	Specify the differential sources to be allowed to operate in a multi-diff application
JDIFFX,SOURCE	Query the receiver for the differential source
JDIFFX,TYPE	Query the receiver for the differential type
JEPHOUT,PERIODSEC	to allow ephemeris messages (95, 65, 35) to go out a rate other than when they change
JFLASH,DIR	Display the files on a USB flash drive
JFLASH,FILE,CLOSE	Close an open file on a USB flash drive
JFLASH,FILE,NAME	Open a specific file, append to a specific file, or display the file name of the open file on a USB flash drive
JFLASH,FILE,OPEN	Create and open a file with an automatically generated file name on a USB flash drive
JFLASH,FREESPACE	Display the free space in kilobytes (KB) on a USB flash drive
JFLASH,NOTIFY,CONNECT	Enable/disable the automatic response when a USB flash drive is inserted or removed (if port is not specified the response will be sent to the port that issued the command)

Note: Use the JSAVE command to save changes you need to keep and wait for the $>SAVE COMPLETE response.

GLONASS Commands and Messages

The following table lists the commands applicable to GLONASS-capable receivers.

Command	Description
JASC,GL	Enable the GLONASS data messages at a particular update rate to be turned on or off. When turning messages on, various update rates are available depending on the requirements.
JNMEA,GGAALL GNSS	Configure the GGA string to include full GNSS information (the number of used GLONASS satellites will be included in the GPGGA message) or query the current setting

The following table lists the messages applicable to GLONASS-capable receivers.

Message	Description
Bin16	GNSS code and phase observation information
Bin62	GLONASS almanac information
Bin65	GLONASS ephemeris information
Bin66	GLONASS L1 code and carrier phase information
Bin69	GLONASS L1 diagnostic information
GLMLA	GLONASS almanac data - contains complete almanac data for one GLONASS satellite (multiple sentences may be transmitted, one for each satellite in the GLONASS constellation)

Topic Last Updated: v2.0/ April 30, 2019

GNSS Commands

The following table lists the commands supported by the internal GNSS engine for its configuration and operation.

Command	Description
JAGE	Specify maximum DGPS (COAST) correction age (6 to 8100 seconds)
JASC,GN	Enable the GPS data messages at a particular update rate to be turned on or off
JMASK	Specify the elevation cutoff mask angle for the GPS engine
JNMEA,PRECISION	Specify or query the number of decimal places to output in the GPGGA and the GPGLL messages or query the current setting
JNP	Specify the number of decimal places output in the GPGGA and GPGLL messages
JOFF	Turn off all data messages being output through the current port or other port
JOFF,ALL	Turn off all data messages being output through all ports
JSMOOTH	Set the carrier smoothing interval (15 to 6000 seconds) or query the current setting
JTAU,COG	Set the course over ground (COG) time constant (0.00 to 3600.00 seconds) or query the current setting
JTAU,SPEED	Set the speed time constant (0.00 to 3600.00 seconds) or query the current setting
JFLASH,QUERYCONNECT	Manually verify if a USB flash drive is connected or disconnected
JFORCEAPP	Force an application to be used in a multi-application (MFA)
JHTYPE, SHOW	Queries the hardware type
JI	Display receiver information, such as its serial number and firmware version
JK	Subscribe the receiver to various options, such as higher update rates, e-Dif (or base station capability) or L-Dif; or query for the current subscription expiration date when running Atlas application or the receiver subscription code

	when running all other applications
JK,SHOW	Contains authorization information
JLIMIT	Set the threshold of estimated horizontal performance for which the DGPS position LED is illuminated or query the current setting
JMODE	Query receiver for status of JMODE settings
JMODE,BASE	Enable/disable base mode functionality or query the current setting
JMODE,FIXLOC	Set the receiver to not re-average (or re-average) its position or query the current setting
JMODE,FOREST	Turn the higher gain functionality (for tracking under canopy) on/off or query the current setting
JMODE,GLOFIX	Enable/disable use of RTCM v3 (RTK) GLONASS correctors
JMODE,MIXED	Include satellites that do not have differential corrections in the solution
JMODE,NULLNMEA	Enable/disable output of NULL fields in NMEA 0183 messages when no there is no fix (when position is lost)
JMODE,SBASNORTK	Disable/enable the use of SBAS ranging signals (carrier phase) in RTK
JMODE,SBASR	Enable/disable SBAS ranging
JMODE,STRICTRTK	Use this command to invoke stricter checks on whether RTK fix is declared. Forces float of RTK at 30 seconds of Age-of-Diff
JMODE,SURETRACK	Enable/disable SureTrack functionality (default is enabled) or query the current setting
JMODE,SURVEY	Assure RTK fix is not declared when residual errors exceed 10 cm. Also forces use of GLONASS and prevents SureTrack operation.
JMODE,TIMEKEEP	Enable/disable continuous time updating in NMEA 0183 messages when there is no fix (when position is lost)
JMODE,TUNNEL	Enable/disable faster reacquisition after coming out of a tunnel or query the current setting
JPOS	Speed up the initial acquisition when changing continents with the receiver or query the receiver for the current position of the receiver
JPPS,FREQ	Specify the pps frequency of the receiver or query the current setting
JPPS,WIDTH	Specify the pps width of the receiver or query the current setting
JPRN,EXCLUDE	**For advanced users only.** Exclude GPS and/or other GNSS satellites from being used in the positioning solution or query the current setting
JQUERY,GUIDE	Query the receiver for its determination on whether or not it is providing suitable accuracy after both the SBAS, and GPS have been acquired (up to five minutes)
JQUERY,TEMPERATURE	Query the receiver's temperature
JRELAY	Send user-defined text out of a serial port

JRESET	Reset the receiver to its default operating parameters by turning off outputs on all ports, saving the configuration, and setting the configuration to its defaults
JSAVE	Send this command after making changes to the operating mode of the receiver
JSHOW	Query the current operating configuration of the receiver
JSHOW,ASC	Query receiver for current ASCII messages being output
JSHOW,BIN	Query receiver for current Bin messages being output
JSHOW,CONF	Query receiver for configuration settings
JSHOW,GP	Query the receiver for each GP message currently being output through the current port and the update rate for that message
JSHOW,THISPORT	Query to determine which receiver port you are connected to
JSIGNAL, EXCLUDE	Query the receiver for the signals for which you are disabling tracking
JSIGNAL,INCLUDE	Query the receiver for the signals for which you are enabling tracking
JSYSVER	Returns the boot loader version from the GPS card

Note: Use the JSAVE command to save changes you need to keep and wait for the $>SAVE COMPLETE response. The following table lists the messages applicable to GNSS.

Message	Description
Bin16	GNSS code and phase observation information
Bin19	GNSS Tracking Information

Topic Last Updated: v1.07/ February 16, 2017

JASC Command

The JASC command is used to request ASCII messages.

Command	Description
JASC,CMR	Set the proprietary CMR messages to on or off to provide corrections to the rover
JASC,D1 (RD1)	Set the RD1 diagnostic information message from the receiver to on or off
JASC,DFX	Set the proprietary DFX messages to on or off to provide corrections to the rover
JASC,GL	Enable the GLONASS data messages at a particular update rate to be turned on or off. When turning messages on, various update rates are available depending on the requirements.
JASC,GN	Enable the GNSS data messages at a particular update rate to be turned on or off. When turning messages on, various update rates are available depending on the requirements.
JASC,GP	Enable the GPS data messages at a particular update rate to be turned on or off
JASC,HPR	Configure the receiver to output UTC time, heading, pitch, and roll. Pitch and roll will come from the antenna array, one from internal sensor, for more information refer to JATT, ROLL

JASC,INTLT	Configure the receiver to output pitch and roll data
JASC,PASHR	Configure the receiver to output time, true heading, roll, and pitch data in one message
JASC,PSAT,ATTSTAT	Configure the receiver to output the information of secondary antenna
JASC,PSAT,BLV,1	Configure the receiver to output the North,East,Up base-line vector
JASC,PSAT,FVI,1	Configure the receiver to output a message include most position and attitude information
JASC,PSAT,RTKPROG	Configure the receiver to output RTK fix progress
JASC,PSAT,RTKSTAT	Configure the receiver to output the most relevant parameters affecting RTK
JASC,PSAT,VCT,1	Configure the receiver to output the heading, pitch, roll, and master to slave vector
JASC,PTSS1	Configure the receiver to output heave, pitch, and roll in the commonly used TSS1 message format
JASC,ROX	Set the proprietary ROX messages to on or off to provide corrections to the rover
JASC,RTCM	Configure the receiver to output RTCM version 2 DGPS corrections from SBAS or beacon through either receiver serial port
JASC,RTCM3	Set the RTCM version 3 messages to on or off to provide corrections to the rover
JASC,VIRTUAL	Configure the receiver to have RTCM data input on one port and output through the other (when using an external correction source)

JATT Commands

The JATT command is used to define or query attitude settings for Vector products.

Command	Description
JATT,COGTAU	Set the course over ground (COG) time constant (0.0 to 3600.0 seconds) or query the current setting
JATT,CSEP	Query to retrieve the current separation between GPS antennas
JATT,EXACT	Enable/disable internal filter reliance on the entered antenna separation or query the current setting
JATT,FLIPBRD	Allow upside down installation
JATT,GYROAID	Turn on gyro aiding or query the current feature status
JATT,HBIAS	Set the heading bias or query the current setting
JATT,HELP	Show the available commands for GPS heading operation and status
JATT,HIGHMP	Set/query the high multipath setting for use in poor GPS environments
JATT,HRTAU	Set the rate of turn time constant or query the current setting
JATT,HTAU	Set the heading time constant or query the current setting

JATT,LEVEL	Turn on level operation or query the current feature status
JATT,MOVEBASE	Set the auto GPS antenna separation or query the current setting
JATT,MSEP	Set (manually) the GPS antenna separation or query the current setting
JATT,NEGTILT	Turn on the negative tilt feature or query the current setting
JATT,NMEAHE	Instruct the Vector to preface the HDG, HDT, ROT and THS messages with GP or HE, and the HDM message with GP or HC.
JATT,PBIAS	Set the pitch bias or query the current setting
JATT,PTAU	Set the pitch time constant or query the current setting
JATT,ROLL	Configure the Vector for roll or pitch output
JATT,SEARCH	Force a new RTK heading search
JATT,SPDTAU	Set the speed time constant (0.0 to 3600.0 seconds) or query the current setting
JATT,SUMMARY	Show the current configuration of the Vector
JATT,TILTAID	Turn tilt aiding on/off or query the Vector for the current status of this feature
JATT,TILTCAL	Calibrate the internal tilt sensor of the Vector

Topic Last Updated: v1.09 / January 8, 2018

JETHERNET Commands

The JETHERNET command is used to configure Ethernet settings on Ethernet-capable boards.

Command	Description
JETHERNET	Query current Ethernet configuration state
JETHERNET,MODE	Enable/Disable Ethernet and set IP address configuration
JETHERNET,PORTI	Enable/Disable PORTI virtual serial port
JETHERNET,NTRIPCLIENT	Configure receiving RTK corrections over IP from an NTRIP caster
JETHERNET,NTRIPSERVER	Configure sending RTK connections over IP to an NTRIP caster
JETHERNET,WEBUI	Enable/Disable the WebUI interface over HTTP

Topic Last Updated: v4.0 / June 30, 2019

JFLASH Commands

The JFLASH command is used to perform file operations via a USB flash drive on Eclipse and Eclipse II based receivers.

Command	Description
JFLASH,DIR	Display the files on a USB flash drive
JFLASH,FILE,CLOSE	Close an open file on a USB flash drive
JFLASH,FILE,NAME	Open a specific file, append to a specific file, or display the file name of the open file on a USB flash drive

JFLASH,FILE,OPEN	Create and open a file with an automatically generated file name on a USB flash drive
JFLASH,FREESPACE	Display the free space in kilobytes (KB) on a USB flash drive
JFLASH,NOTIFY,CONNECT	Enable/disable the automatic response when a USB flash drive is inserted or removed
JFLASH,QUERYCONNECT	Manually verify if a USB flash drive is connected or disconnected

Topic Last Updated: v1.02 / January 25, 2011

JRAD Commands

This topic provides information related to the NMEA 0183 messages accepted by the receiver's e-Dif application.

The following table provides a brief description of the commands supported by the e-Dif application for its control and operation.

Command	Description
JRAD1	Display the current reference position in e-Dif applications only
JRAD,1,LAT,LON,HEIGHT	Use this command—a derivative of the JRAD,1,P command—when absolute positioning is required in e-Dif applications only
JRAD,1,P	e-Dif: Record the current position as the reference with which to compute e-Dif corrections. This would be used in relative mode as no absolute point information is specified. DGPS Base Station: Record the current position as the reference with which to compute Base Station corrections in e-Dif applications only. This would be used in relative mode as no absolute point information is specified
JRAD,2	Forces the receiver to use the new reference point (you normally use this command following a JRAD,1 type command)
JRAD,3	Invoke the e-Dif function once the unit has started up with the e-Dif application active, or update the e-Dif solution (calibration) using the current position as opposed to the reference position used by the JRAD,2 command
JRAD,7	Turn auto recalibration on or off
JRAD,9	Initialize the Base Station feature and use the previously entered point, either with$JRAD,1,P or $JRAD,1,LAT,LON,HEIGHT, as the reference with which to compute Base Station corrections in e-Dif applications only. Use this for both relative mode and absolute mode.
JRAD,10	Specify BDS message to be transmitted by base station

JRTK Commands

The JRTK commands are used to define or query RTK settings.

Command	Description
JRTK,1	Show the receiver's reference position (can issue command to base station or rover)
JRTK,1,LAT,LON,HEIGHT	Set the receiver's reference position to the coordinates you enter (can issue command to base station or rover)
JRTK,1,P	Set the receiver's reference coordinates to the current calculated position if you do not have known coordinates for your antenna location (can issue command to base station or rover)
JRTK,5	Show the base station's transmission status for RTK applications (can issue command to base station)
JRTK,5,Transmit	Suspend or resume the transmission of RTK (can issue command to base station)
JRTK,6	Display the progress of the base station (can issue command to base station)
JRTK,12	Disable or enable the receiver to go into fixed integer mode (RTK) vs. float mode (L- Dif) - can issue command to rover
JRTK,17	Display the transmitted latitude, longitude, and height of the base station (can issue command to base station or rover)
JRTK,18	Display the distance from the rover to the base station, in meters (can issue command to rover)
JRTK,18,BEARING	Display the bearing from the base station to the rover, in degrees (can issue command to rover)
JRTK,18,NEU	Display the distance from the rover to the base station and the delta North, East, and Up, in meters (can issue command to rover)
JRTK,28	Set the base station ID transmitted in ROX/DFX/CMR/RTCM3 messages (can issue command to base station)

Topic Last Updated: v1.03 / January 11, 2012

JTAU Commands

The JTAU command is used to set the time constants for specific parameters for Crescent, Crescent Vector, and Eclipse products.

Command	Description
JTAU,COG	Set the course over ground time (COG) constant and query the current setting
JTAU,SPEED	Set the speed time constant and query the current setting

Topic Last Updated: v1.00 / August 11, 2010

QZSS Commands and Messages

The following table lists the commands applicable to QZSS-capable receivers.

Command	Description
JASC,GQGSV	Enable/disable the data for QZSS satellites in view.
JASC,GNGNS	Enable/disable fix data for GNSS systems.
JASC,GNGSA	DOP and active satellite information

The following table lists the binary messages applicable to QZSS-capable receivers.

Message	Description
Bin16	GNSS code and phase observation information
Bin19	GNSS diagnostic information

Topic Last Updated: v1.07 / February 16, 2017

RAIM Commands

RAIM (Receiver Autonomous Integrity Monitoring) is a GNSS integrity monitoring scheme that uses redundant ranging signals to detect a satellite malfunction resulting in a large range error. The Hemisphere GNSS products use RAIM to alert users when errors have exceeded a user-specified tolerance. RAIM is available for SBAS, and Beacon applications.

The following table lists the available RAIM commands.

Command	Description
JRAIM	Specify the parameters of the RAIM scheme that affect the output of the PSAT,GBS message or query the current setting

Topic Last Updated: v1.07 / February 16, 2017

RTK Commands and Messages

The following table lists the commands supported by RTK features for its control and operation.

Command	Description
JASC,CMR	Set the proprietary CMR messages to on or off to provide corrections to the rover (only applies to an Eclipse base station receiver when using GPS dual frequency RTK mode)
JASC,DFX	Set the proprietary DFX messages to on or off to provide corrections to the rover (only applies to a Crescent base receiver when using L-Dif or RTK mode)
JASC,ROX	Set the proprietary ROX messages to on or off to provide corrections to the rover (only applies to an Eclipse base station receiver when using GPS dual frequency RTK mode)
JASC,RTCM3	Set the RTCM version 3 messages to on or off to provide corrections to the rover (only applies to an Eclipse base station

	receiver when using GPS dual frequency RTK mode)
JASC,PSAT,BLV,1	Configure the receiver to output the North,East,Up base-line vector
JASC,PSAT,FVI,1	Configure the receiver to output a message include most position and attitude information
JASC,PSAT,RTKPROG	Configure the receiver to output RTK fix progress
JASC,PSAT,RTKSTAT	Configure the receiver to output the most relevant parameters affecting RTK
JASC,PSAT,VCT,1	Configure the receiver to output the heading, pitch, roll, and master to slave vector
JMODE,BASE	Enable/disable base mode functionality or query the current setting
JNMEA,PRECISION	Specify or query the number of decimal places to output in the GPGGA And the GPGLL messages or query the current setting
JNP	Specify the number of decimal places output in the GPGGA and GPGLL messages
JQUERY,RTKPROG	Perform a one-time query of RTK fix progress information
JQUERY,RTKSTAT	Perform a one-time query of the most relevant parameters that affect RTK
JRTK,1	Show the receiver's reference position (can issue command to base station or rover)
JRTK,1,LAT,LON,HEIGHT	Set the receiver's reference position to the coordinates you enter (can issue command to base station or rover)
JRTK,1,P	Set the receiver's reference coordinates to the current calculated position if you do not have known coordinates for your antenna location (can issue command to base station or rover)
JRTK,5	Show the base station's transmission status for RTK applications (can issue command to base station)
JRTK,5,Transmit	Suspend or resume the transmission of RTK (can issue command to base station)
JRTK,6	Display the progress of the base station (can issue command to base station)
JRTK,12	Disable or enable the receiver to go into fixed integer mode (RTK) vs. float mode (L- Dif) - can issue command to rover
JRTK,17	Display the transmitted latitude, longitude, and height of the base station (can issue command to base station or rover)
JRTK,18	Display the distance from the rover to the base station, in meters (can issue command to rover)
JRTK,18,BEARING	Display the bearing from the base station to the rover, in degrees (can issue command to rover)
JRTK,18,NEU	Display the distance from the rover to the base station and the delta North, East, and Up, in meters (can issue command to rover)
JRTK,28	Set the base station ID transmitted in ROX/DFX/CMR/RTCM3 messages (can issue command to base station)
JRTCM3, ANTNAME	Specify the antenna name that is transmitted in various RTCM3 messages from the base

JRTCM3, EXCLUDE	Specify RTCM3 message types to not be transmitted (excluded) by base station
JRTCM3, INCLUDE	Specify RTCM3 message types to be transmitted by base station
JRTCM3, NULLANT	Specify the antenna name as null (no name) that is transmitted in various RTCM3 messages from the base

The following table lists RTK messages.

Message	Description
PSAT,RTKPROG	Contains RTK fix progress information
PSAT,RTKSTAT	Contains the most relevant parameters affecting RTK

Topic Last Updated: v1.07 / October 13, 2016

SBAS Commands

The following table lists the commands supported by the SBAS demodulator for its control and operation.

Command	Description
JASC,D1	Set the RD1 diagnostic information message from the receiver to on or off
JASC,RTCM	Configure the receiver to output RTCM version 2 DGPS corrections from SBAS or beacon through either receiver serial port
JGEO	Display information related to the current frequency of SBAS and its location in relation to the receiver's antenna
JWAASPRN	Change the SBAS PRNs in memory or query the receiver for current PRNs in memory

Note: Use the JSAVE command to save changes you need to keep and wait for the $>SAVE COMPLETE response.

Topic Last Updated: v1.00 / August 11, 2010

Vector Commands and Messages

The following table lists the commands related to the GPS heading aspect of the Vector OEM heading system.

Command	Description
JASC	Turn on different messages
JASC,INTLT	Configure the receiver to output pitch and roll data (pitch and roll are factory calibrated over temperature to be accurate to ±3°)
JASC,PASHR	Configure the receiver to output time, true heading, roll, and pitch data in one message
JASC,PTSS1	Configure the receiver to output heave, pitch, and roll in the commonly used TSS1 message format
$JATT,ACC90	Refer to the User Guide for your product

$JATT, ACC180	Refer to the User Guide for your product
JATT,COGTAU	Set the course over ground (COG) time constant (0.0 to 3600.0 seconds) or query the current setting
JATT,CSEP	Query for the current separation between GPS antennas
JATT,EXACT	Enable/disable internal filter reliance on the entered antenna separation or query the current setting
JATT,FLIPBRD	Turn the flip feature on/off (allowing you to install the Crescent Vector board upside down) or query the current feature status
JATT,GYROAID	Turn gyro aiding on or off or query the current setting
JATT,HBIAS	Set the heading bias or query the current setting
JATT,HELP	Show the available commands for GPS heading operation and status
JATT,HIGHMP	Set/query the high multipath setting for use in poor GPS environments
JATT,HRTAU	Set the heading rate time constant or query the current setting
JATT,HTAU	Set the heading time constant or query the current setting
JATT,LEVEL	Turn level operation on or off or query the current setting
JATT,MOVEBASE	Set the auto GPS antenna separation or query the current setting
JATT,MSEP	Manually set the GPS antenna separation or query the current setting
JATT,NEGTILT	Turn the negative tilt feature on or off or query the current setting
JATT,NMEAHE	Instruct the Crescent Vector to preface the HDG, HDM, HDT, and ROT messages with GP or HE
JATT,PBIAS	Set the pitch/roll bias or query the current setting
JATT,PTAU	Set the pitch time constant or query the current setting
JATT,ROLL	Configure the Crescent Vector for roll or pitch GPS antenna orientation
JATT,SEARCH	Force the Crescent Vector to reject the current GPS heading solution and begin a new search
JATT,SPDTAU	Set the speed time constant (0.0 to 3600.0 seconds) or query the current setting
JATT,SUMMARY	Display a summary of the current Crescent Vector settings
JATT,TILTAID	Turn tilt aiding on or off or query the current setting
JATT,TILTCAL	Calibrate tilt aiding or query the current feature status

The following table lists Vector messages.

Message	Description
GNGSA	GNSS DOP and active satellites
GPDTM	Datum reference
GPGGA	GPS fix data
GPGLL	Geographic position - latitude/longitude
GPGNS	GNSS fix data
GPGRS	GNSS range residuals
GPGST	GNSS pseudorange error statistics

GPGSV	GPS satellites in view
GLGSV	GLONASS satellites in view
GAGSV	Galileo satellites in view
GBGSV	BeiDou satellites in view
GPHDG/HEHDG	Provide magnetic deviation and variation for calculating magnetic or true heading
GPHDM/HEHDM	Provide magnetic heading of the vessel derived from the true heading calculated
GPROT/HEROT	Contains the vessel's rate of turn (ROT) information
GPRRE	Range residual message
GPVTG	Course over ground and ground speed
GPZDA	Time and date
PASHR	Time, true heading, roll, and pitch data in one message
PSAT,GBS	Satellite fault detection used for RAIM
PSAT,HPR	Proprietary NMEA sentence that provides the true heading, pitch/roll information and time in a single message
PSAT,INTLT	Proprietary NMEA sentence that provides the title measurement from the internal inclinometer (in degrees)
TSS1	Heave, pitch, and roll message in the commonly used TSS1 message format

Topic Last Updated: v2.0/ April 30, 2019

Binary Messages

The binary messages supported by the receiver are in an Intel Little Endian format for direct read in a PC environment. More information on this format at the following web site:
http://www.cs.umass.edu/~verts/cs32/endian.html

Each binary message begins with an 8-byte header and ends with a carriage return, line feed pair (0x0D, 0x0A). The first four characters of the header is the ASCII sequence $BIN.

The following table provides the general binary message structure.

Component	Description	Type	Bytes
Header	Synchronization String	4-byte string	4
	Block ID - type of binary message	Unsigned short	2
	DataLength - the length of the binary message	Unsigned short	2
Data	Binary Data - varying fields of data with a total length of DataLength bytes	Mixed fields	Data Length bytes
Epilogue	Checksum - sum of all bytes of the data (all DataLength bytes); the sum is placed in a 2- byte integer	Unsigned short	2
	CR- Carriage return	Byte	1
	LF - Line feed	Byte	1

Data Messages

Note: Output rates greater than 1Hz may require a subscription. Output rates greater than 20 Hz are not available for all products. Please refer to your product's documentation for the supported output rates.

For messages supporting rates greater than 1 Hz, see the following table:

Firmware Version	Support Output Rates
50 Hz	50, 25, 10, 5, 2, 1, .2, 0
20 Hz	20, 10, 5, 4, 2, 1, .2, .5, 0

For message descriptions and maximum rates see the following table:

Message	Maximum Rate	Description
GNGSA	1 Hz	GPS DOP and active satellite information
GPALM	1 Hz	GPS almanac data
GPGGA	50 Hz	Detailed GPS position information
GPGLL	50 Hz	Latitude and longitude data
GPGNS	50 Hz	Fixes data for single or combined satellite navigation systems
GPGRS	50 Hz	Supports Receiver Autonomous Integrity Monitoring (RAIM)
GPGST	1 Hz	GNSS pseudo range error statistics
GPGSV	20 Hz	GPS satellites in view
GLGSV	20 Hz	GLONASS satellites in view
GAGSV	20 Hz	Galileo satellites in view
GBGSV	20 Hz	BeiDou satellites in view
GPHDG/HEHDG	50 Hz	Magnetic deviation and variation for calculating magnetic or true heading
GPHDM/HEHDM	50 Hz	Magnetic heading of the vessel derived from the true heading calculated
GPHDT/HEHDT	50 Hz	True heading of the vessel
GPHEV	50 Hz	Heave value in meters
GPRMC	50 Hz	Recommended minimum specific GNSS data
GPROT/HEROT	50 Hz	Vessel's rate of turn (ROT) information
GPRRE	1 Hz	Range residual message
GPVTG	50 Hz	Course over ground and ground speed
GPZDA	50 Hz	UTC time and date information

PASHR	1 Hz	Time, true heading, roll, and pitch data in one message
PSAT,ATTSTAT	1HZ	
PSAT,GBS	1 Hz	Used to support Receiver Autonomous Integrity Monitoring (RAIM)
PSAT,HPR	50 Hz	Proprietary NMEA message that provides the true heading, pitch, roll, and time in a single message
PSAT,INTLT	1 Hz	Proprietary NMEA message that provides the tilt measurements from the internal inclinometers (in degrees)
PSAT,RTKPROG	1 Hz	Contains RTK fix progress information
PSAT,RTKSTAT	1 Hz	Contains the most relevant parameters affecting RTK
RD1	1 Hz	SBAS diagnostic information
TSS1	50 Hz	Heave, pitch, and roll message in the commonly used TSS1 message format

Topic Last Updated: v1.11/ November 15, 2018

NMEA 0183 Messages

NMEA 0183 is a communications standard established by the National Marine Electronics Association (NMEA). NMEA 0183 provides data definitions for a variety of navigation instruments and related equipment such as gyrocompasses, Loran receivers, echo sounders, and GNSS receivers.

NMEA 0183 functionality is virtually standard on all GNSS equipment available. NMEA 0183 has an ASCII character format that enables the user to read the data via a receiving device with terminal software.

The following is an example of one second of NMEA 0183 data from the receiver:

$GPGGA,144049.0,5100.1325,N,11402.2729,W,1,07,1.0,1027.4,M,0,M,,010*61

$GPVTG,308.88,T,308.88,M,0,0.04,N,0.08,K*42

$GPGSV,3,1,10,02,73,087,54,04,00,172,39,07,66,202,54,08,23,147,48,*7 9

$GPGSV,3,2,10,09,23,308,54,11,26,055,54,15,00,017,45,21,02,353,45*78

$GPGSV,3,3,10,26,29,257,51,27,10,147,45,45,,,,,,,*74

The NMEA 0183 standard allows manufacturers to define proprietary custom commands and to combine data into proprietary custom messages. Proprietary NMEA 0813 messages are likely to be supported only by specific manufacturers. All messages and ports can be configured independently (see example below).

Port	Baud Rate	Messages
A	9600	GPGGA, one every 1 second GPGSV, one every 5 seconds

B	19200	GPGGA, one every 2 seconds Bin1, one every 1 second Bin2, one every 1 second

A selection of NMEA 0183 data messages can be configured at various update rates with each message having a maximum update rate. A different selection of NMEA 0183 messages with different rates can be configured on another port.

Commands and Messages Overview presents information about the NMEA 0183 interface of the receiver smart antenna. See
Reference Documents for contact information if you need to purchase a copy of the NMEA 0183 standard.

Topic Last Updated: v1.07 / February 16, 2017

NMEA 0183 Message Format

NMEA 0183 messages (sentences) have the following format:

$XXYYY,ZZZ,ZZZ,ZZZ...*CC<CR><LF>

where:

Element	Description
$	Message header character
XX	NMEA 0183 talker field (GP = GPS, GL = GLONASS, GA = GALILEO, GB =BEIDOU, GN = All constellations)
YYY	Type of GPS NMEA 0183 message
ZZZ	Variable length message fields
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Null (empty) fields occur when there is no information for that field. You can use the JNP command to specify the number of decimal places output in the GPGGA and GPGLL messages.

What does <CR><LF> mean?

The literal translation means "Carriage Return, Line Feed." These are terms used in computer programming languages to describe the end of a line or string of text. If you are writing your own communication software for a receiver, see some of the examples below. If you are already using a program such as Hemisphere GNSS' PocketMax, when you click to send a command to the receiver, the program adds the carriage return and line feed to the end of the text string for you. If you are using HyperTerminal or other terminal software, typically the Enter key on your keyboard is set to send the <CR><LF> pair. You may need to define this in the setup section of the terminal software. Some software may treat the Enter key on your numeric keypad differently than the main Enter key in the main QWERTY section of the keyboard – use the main Enter key for best results.

Electronics use different ways to represent the <CR><LF> characters. In ASCII numbers, <CR> is represented as 13 in decimal, or 0D in hexadecimal. ASCII for <LF> is 10 decimal, or 0A hexadecimal. Some computer languages use different ways to represent <CR><LF>. Unix and C language can use

"\x0D\x0A". C language can also use "\r\n" in some instances. Java may use CR+LF. In Unicode, carriage return is U+000D, and line feed is U+000A. It is advised to clearly understand how to send these characters if you are writing your own interface software.

Topic Last Updated: v2.0/ April 30, 2019

NMEA 2000 CAN Messages

Refer to the NMEA Specification Appendix A & B. The following NMEA 2000 CAN messages are supported by HGNSS:

PGN	Description	Default Rate
126992	System Time	1 Hz
129025	Position Rapid update	10 Hz
129026	COG & SOG, Rapid update	4 Hz
129027	Position Delta, High Precision Rapid update	10 Hz
129028	Altitude Delta, High Precision Rapid update	10 Hz
129029	GNSS Position Data	1 Hz
129033	Local Time Offset	1 Hz
129539	GNSS DOPs	1 Hz
129540	GNSS Sats in View	1 Hz
129542	GNSS Pseudorange Noise Statistics	1 Hz
129545	GNSS RAIM Output	Off
127250	Vessel Heading	10 Hz*
127251	Rate of Turn	10 Hz*
127258	Magnetic Variation	1 Hz*
127257	Attitude, Yaw, Pitch, Roll	1 Hz*

* These messages may be off when heading is not supported.
For a list of Hemisphere GNSS Proprietary Commands, refer to the NMEA 2000 Reference Manual on the HGNSS website.
Note: Not all products support the messages listed above.

Topic Last Updated: v1.10 / June 1, 2018

General Operation and Configuration Commands

JAGE Command

Command Type:	GPS
Description:	Specify maximum DGPS (COAST) correction age (6 to 8100 seconds). Using COAST technology, the receiver can use old correction data for extended periods of time. If using aRTK, the parameter must be set to higher than 601 seconds. The default setting for the receiver is 2700 seconds. If you select a maximum correction age older than 1800 seconds (30 minutes) test the receiver to ensure the new setting meets the requirements, as accuracy will slowly drift with increasing time.
Command Format:	$JAGE,age<CR><LF> where 'age' is the maximum differential age time out
Receiver Response:	$>
Example:	To set the DGPS correction age to 60 seconds issue the following command: **$JAGE,60<CR><LF>**
Additional Information:	To query the receiver for the current DGPS correction age, issue the JSHOW command. What does <CR><LF> mean?
Related Commands and Messages:	

Topic Last Updated: v2.0/ April 30, 2019

JAIR Command

Command Type:	General Operation and Configuration
Description:	Specify how the receiver will respond to the dynamics associated with airborne applications or query the current setting.
Command Format:	Specify how the receiver responds: $JAIR,r<CR><LF> where 'r' is the AIR mode: NORM - normal track and nav filter bandwidth HIGH - highest track and nav filter bandwidth (receiver is optimized for the high dynamic environment associated with airborne platforms) LOW - lowest track and nav filter bandwidth AUTO - default track and nav filter bandwidth, similar to NORM but automatically goes to HIGH above 30m/sec Query the current setting: $JAIR<CR><LF>
Receiver Response:	Receiver response when specifying how the receiver responds or querying the current setting: $>JAIR,MAN,NORM $>JAIR,MAN,HIGH $>JAIR,MAN,LOW

	$>JAIR,AUTO,NORM
Example:	To set the AIR mode to LOW issue the following command: $JAIR,LOW<CR><LF> The response is then: $>JAIR,MAN,LOW<CR><LF>
Additional Information:	Defaults to normal (NORM) which is recommended for most applications. The AUTO option enables the receiver to decide when to turn JAIR to HIGH. CAUTION: Setting AIR mode to HIGH is not recommended for Crescent Vector operation. On the HIGH setting, the receiver tolerates larger and sudden drops in the SNR value before it discards the data as being invalid. This additional tolerance is beneficial in applications such as crop dusting where an aircraft is banking rapidly. As the aircraft banks, the antenna position shifts from upright and having a clear view of the sky to being tipped slightly, with a possibly obscured view of the sky, and then back to upright. This sudden tipping of the antenna causes the SNR value to drop. If the tolerance is not set as HIGH, the receiver views the data recorded while banking as invalid and discards it. As a result, the GPS position will not be accurate. The status of this command is also output in the JSHOW message.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JAPP Command

Command Type:	General Operation and Configuration
Description:	Specify which of the installed applications should be utilized or query the receiver for the currently installed applications. All modern versions of Hemisphere receivers have MFA firmware. However, you can use this command if you have 2 different versions of firmware installed. For example, if you update the firmware on application 1 and your receiver still shows you have the previous version of firmware installed, check to see if you are in application 2, or vice versa. Specify receiver application firmware (when two applications are present)
Command Format:	$JAPP,OTHER<CR><LF> or $JAPP,O<CR><LF> (The second command uses the letter O, not a zero) or $JAPP,x<CR><LF> where ‘x’ is either 1 (application in slot 1) or 2 (application in slot 2) Query receiver application firmware: $JAPP<CR><LF>
Receiver Response:	For example, if WAAS (SBAS) and AUTODIFF (e-Dif) are the two installed applications (WAAS in slot1 and AUTODIFF in slot2) and WAAS is the current application, if you issue the JAPP,OTHER<CR><LF> command on a receiver, the response to $JAPP<CR><LF> will be $>JAPP,AUTODIFF,WAAS,2,1, indicating that application slot 2 (e- Dif) is currently being used. Hemisphere GNSS recommends that you follow up the sending of these commands with a $JAPP query to see which application is 1 or 2. It is best to

	use these two commands when upgrading the firmware inside the receiver, because the firmware upgrading utility uses the application number to designate which application to overwrite. Response to querying the current setting: $>JAPP,CURRENT,OTHER,[1 OR 2],[2 OR 1] where: •'CURRENT' indicates the current application in use •'OTHER' indicates the secondary application that is not currently in use •1 and 2 indicate in which application slots the applications reside
Example:	If the response to: $JAPP<CR><LF> is $>JAPP,WAAS,AUTODIFF,1,2, this indicates: WAAS (SBAS) is the current application and is in application slot 1 e-Dif is the other application (not currently used)and is in application slot 2
Additional Information:	When querying the current setting, the following application names may appear (depending on your product): • Crescent • WAAS – Changes to the SBAS application. For the sake of the application names, the SBAS application is referred to as WAAS by the receiver's internal firmware • AUTODIFF – Changes to the e-Dif application. Referred to as "AUTODIFF" in the receiver's internal firmware • LOCRTK – Changes to the local differential rover application • RTKBAS – Changes to the local differential base application • LBAND – Changes to Atlas DGPS service • Eclipse • WAASRTKB – Changes to the SBAS/RTK Base application • LBAND – Changes to Atlas DGPS service • RTK – Changes to the RTK Rover application • Eclipse II • SBASRTKB – Changes to the SBAS/L-band/RTK Base application • AUTODIFF – Changes to the e-Dif application, referred to as "AUTODIFF" in the firmware • RTK – Changes to the RTK Rover application • MFA - Multi-function application • miniEclipse • WAASRTKB – Changes to the SBAS/RTK Base application • AUTODIFF – Changes to the e-Dif application, referred to as "AUTODIFF" in the firmware • RTK – Changes to the RTK Rover application • MFA - Multi-function application
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JASC,CMR Command

Command Type:	Local Differential and RTK
Description:	Set the proprietary CMR messages to on or off to provide corrections to the rover. This command only applies to an Eclipse base station receiver when using GPS dual frequency RTK mode. RTK is relative to the reference position (base only).
Command Format:	where: $JASC,CMR,r[,OTHER]<CR><LF> 'r' = correction status variable (0 = turn corrections Off, 1 = turn corrections On) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on CMR messages on the OTHER port issue the following command: $JASC,CMR,1,OTHER<CR><LF>
Additional Information:	To query the receiver for the current setting, issue the JSHOW command. To change the broadcast station ID, use JRTK,28.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JASC,D1 Command

Command Type:	General Operation and Configuration, SBAS
Description:	Set the RD1 diagnostic information message from the receiver to on or off There is currently only an (R)D1 message. This contains diagnostic information for L-band.
Command Format:	where: $JASC,D1,r[,OTHER]<CR><LF> •'r' = message rate (0 = Off, 1 = On at 1Hz) ',OTHER' = optional field, enacts a change in the RD1 message on the current port when you send the command without it (and without the brackets)and enacts a change in the RD1 message on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To output the RD1 message once per second from THIS port issue the following command: $JASC,D1,1<CR><LF> ...and the output will look similar to the following:

	$RD1,410213,1052,1551.489,1,0,39,- 611.5,0,1F,1F,0,999999 $RD1,410214,1052,1551.489,1,0,40,- 615.1,0,1F,1F,0,999999 $RD1,410215,1052,1551.489,1,0,40,- 607.1,0,1F,1F,0,999999 See RD1 message for a description of each field in the response.
Additional Information:	Although you request D1 through this command the responding message is RD1. To query the receiver for the current setting, issue the JSHOW command. For example, if you issue the following command: $JASC,D1,1<CR><LF> ...then issuing the JSHOW command displays the following as part of its output: $>JSHOW,ASC,D1,1\
Related Commands and Messages:	

Topic Last Updated: v2.0/ April 30, 2019

JASC,DFX Command

Command Type:	Local Differential and RTK
Description:	Set the proprietary DFX messages to on or off to provide corrections to the rover This command only applies to a Crescent base receiver when using L-Dif or RTK mode. Differential is relative to the reference position (base only). See the JASC,ROX command for the equivalent message for the Eclipse series of products.
Command Format:	$JASC,DFX,r[,OTHER]<CR><LF> where: 'r' = correction status variable (0 = turn corrections Off, 1 = turn corrections On) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on DFX messages on THIS port issue the following command: $JASC,DFX,1<CR><LF>
Additional Information:	To query the receiver for the current setting, issue the JSHOW command. To change the broadcast station ID, use JRTK,28.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JASC,GL Command

Command Type:	GLONASS
Description:	Enable the GLONASS data messages at a particular update rate to be turned on or off. When turning messages on, various update rates are available depending on the requirements.
Command Format:	$JASC,msg,r[,OTHER]<CR><LF>

<table>
<tr><td></td><td>where:

•'msg' = name of the data message
•'r' = message rate (see table below)
•',OTHER' = optional field, enacts a change on the current port (THIS port) when you send the command without it (and without the brackets) and enacts a change on the other port (OTHER port) when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.

Send a command with a zero value for the 'R' field to turn off a message.

<table>
<tr><th>MSG</th><th>R (rate in Hz)</th><th>Description</th></tr>
<tr><td>GLMLA</td><td>1 (on) or 0 (off)</td><td>GLONASS almanac data</td></tr>
<tr><td></td><td>When set to on the message is sent once (one message for each tracked satellite) and then sent again whenever satellite information changes</td><td></td></tr>
<tr><td>GLGSV</td><td>1 or 0</td><td>GLONASS satellite in view</td></tr>
</table>
</td></tr>
<tr><td>Receiver Response:</td><td>$></td></tr>
<tr><td>Example:</td><td>To output the GLGNS message through the OTHER port at a rate of 20 Hz, issue the following command:

$JASC,GLGNS,20,OTHER<CR><LF></td></tr>
<tr><td>Additional Information:</td><td>The status of this command is also output in the JSHOW message. What does <CR><LF> mean?</td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v1.02 / January 25, 2011

JASC,GA Command

Command Type:	GALILEO
Description:	Enable the GALILEO data messages at a particular update rate to be turned on or off. When turning messages on, various update rates are available depending on the requirements.
Command Format:	$JASC,msg,r[,OTHER]<CR><LF> where: •'msg' = name of the data message •'r' = message rate (see table below) •',OTHER' = optional field, enacts a change on the current port (THIS port) when you send the command without it (and without the brackets) and enacts a change on the other port (OTHER port) when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.

	Send a command with a zero value for the 'R' field to turn off a message. MSG: GNGNS — R (rate in Hz): 20, 10, 2, 1, 0 or .2 — Description: All GNSS fix data (GAGNS output is GALILEO) MSG: GAGSV — R (rate in Hz): 1 or 0 — Description: GALILEO satellites in view
Receiver Response:	$>
Example:	
Additional Information:	The status of this command is also output in the JSHOW message. What does <CR><LF> mean?
Related Commands and Messages:	

Topic Last Updated: v1.07 / February 16, 2017

JASC,GQ Command

Command Type:	QZSS
Description:	Enable the QZSS data messages at a particular update rate to be turned on or off.
Command Format:	$JASC,msg,r[,OTHER]<CR><LF> where: •'msg' = name of the data message •'r' = message rate (see table below) •',OTHER' = optional field, enacts a change on the current port (THIS port) when you send the command without it (and without the brackets) and enacts a change on the other port (OTHER port) when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology. Send a command with a zero value for the 'R' field to turn off a message. MSG: GQGSV — R (rate in Hz): 1 or 0 — Description: QZSS satellites in view
Receiver Response:	$>
Example:	To output the GQGSV message through the OTHER port, issue the following command: $JASC,GQGSV,1,OTHER<CR><LF>
Additional Information:	The status of this command is also output in the JSHOW message. What does <CR><LF> mean?
Related Commands and Messages:	

Topic Last Updated: :2.0/ April 30, 2019

JASC,GN Command

Command Type:	GPS, Vector
Description:	Enable the GNSS data messages at a particular update rate to be turned on or

<table>
<tr><td></td><td>off. When turning messages on, various update rates are available depending on the requirements.</td></tr>
<tr><td>Command Format:</td><td>$JASC,msg,r[,OTHER]<CR><LF>

where:

•'msg' = name of the data message
•'r' = message rate (see table below)
•',OTHER' = optional field, enacts a change on the current port (THIS port) when you send the command without it (and without the brackets) and enacts a change on the other port (OTHER port) when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.

Send a command with a zero value for the 'R' field to turn off a message.

<table>
<tr><th>MSG</th><th>R (rate in Hz)</th><th>Description</th></tr>
<tr><td>GNGGA</td><td>20, 10, 2, 1, 0 or .2</td><td>GNSS fix data</td></tr>
<tr><td>GNGLL</td><td>20, 10, 2, 1, 0 or .2</td><td>Geographic position - latitude/longitude</td></tr>
<tr><td>GNGNS</td><td>20, 10, 2, 1, 0 or .2</td><td>GNSS fix data</td></tr>
<tr><td>GNGSA</td><td>1 or 0</td><td>GNSS DOP and active satellites</td></tr>
</table>
</td></tr>
<tr><td>Receiver Response:</td><td>$></td></tr>
<tr><td>Example:</td><td>To output the GNGNS message through the OTHER port at a rate of 20 Hz, issue the following command:

$JASC,GNGNS,20,OTHER<CR><LF></td></tr>
<tr><td>Additional Information:</td><td>The status of this command is also output in the JSHOW message. What does <CR><LF> mean?</td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v1.07 / February 16, 2017

JASC,GP Command

Command Type:	GPS, Vector
Description:	Enable the GPS data messages at a particular update rate to be turned on or off. When turning messages on, various update rates are available depending on the requirements.
Command Format:	$JASC,msg,r[,OTHER]<CR><LF> where: •'msg' = name of the data message •'r' = message rate (see table below) •',OTHER' = optional field, enacts a change on the current port (THIS port) when

you send the command without it (and without the brackets)and enacts a change on the other port (OTHER port) when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.

Send a command with a zero value for the 'R' field to turn off a message.

MSG	R (rate in Hz)	Description
GPALM	1 or 0	GPS almanac data
GPDTM	1 or 0	Datum reference
GPGBS	1 or 0	Satellite fault detection used for RAIM
GPGGA	20,10,2,1,0 OR 2	Detailed GPS position information
GPGLL	20,10,2,1,0 OR 2	Latitude and longitude data
GPGNS	20,10,2,1,0 OR 2	Fixes data for single or combined satellite navigation systems
GPGRS	1,0 OR 2	GNSS rage residuals
GNGSA	1 OR 0	GPS DOP and active satellite information
GPGST	1 OR 0	GNSS pseudorange error statistics
GPGSV	20,10,1,0 OR 2	GPS satellites in view
GPHDG OR HEHDG	20,10,2,0,OR 2,1,0 OR 2	Magnetic deviation and variation for calculating magnetic or true heading
GPHDM OR HEHDM	20,10,2,0,OR 2,1,0 OR 2	Magnetic heading of the vessel derived from the true heading calculated
GPDHT OR HEHDT	20,10,2,0,OR 2,1,0 OR 2	True heading of the vessel
GPHEV	20,10,2,0,OR 2,1,0 OR 2	Heave value in meters
GPHPR	20,10,2,0,OR 2,1,0 OR 2	Proprietary NMEA message that provides the true heading, pitch, roll, and time in a single message
GPRMC	10,2,1,0 OR 2	Recommended minimum specific GNSS data
GPROT OR HEROT	20,10,2,1,0 OR 2	Vessel's rate of turn (ROT) information
GPRRE	1 OR 0	Range residual message
GPVTG	20,10,2,1,0 OR 2	Course over ground and ground speed
GPZDA	20,10,2,1,0 OR 2	UTC Time and date information

Receiver Response:	$>
Example:	To output the GPGGA message through the OTHER port at a rate of 20 Hz, issue the following command: $JASC,GPGGA,20,OTHER<CR><LF>
Additional Information:	The status of this command is also output in the JSHOW message. What does <CR><LF> mean?
Related Commands and Messages:	

Topic Last Updated: v1.11 / November 15, 2019

JASC,INTLT Command

Command Type:	Vector
Description:	Configure the receiver to output pitch and roll data (pitch and roll are factory calibrated over temperature to be accurate to ±3°C) directly from the internal tilt sensor Saved with JSAVE.
Command Format:	$JASC,INTLT,r[,OTHER]<CR><LF> where: •'r' = message rate (0 = Off, 1 = On at 1Hz) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$PSAT,INTLT,pitch,roll*CC<CR><LF> where pitch and roll are in degrees
Example:	
Additional Information:	PSAT,INTLT message
Related Commands and Messages:	

Topic Last Updated: v2.0/ April 30, 2019

JASC,PASHR Command

Command Type:	Vector
Description:	Configure the receiver to output time, true heading, heave, roll, and pitch data in one message
Command Format:	$JASC,PASHR,r[,OTHER]<CR><LF> where: •'r' = message rate (0 = Off, 1 = On at 1Hz) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS 'and 'OTHER' port terminology.
Receiver Response:	$PASHR,hhmmss.ss,HHH.HH,T,RRR.RR,PPP.PP,heave,rr.rrr,pp.ppp,hh.hhh,QF*CC<CR> where:

Message Component	Description
hhmmss.ss	UTC time
HHH.HH	Heading value in decimal degrees
T	True heading (T displayed if heading is relative to true north)
RRR.RR	Roll in decimal degrees (- sign will be displayed when applicable)
PPP.PP	Pitch in decimal degrees (- sign will be displayed when applicable)
heave	Heave, in meters
rr.rrr	Roll standard deviation in decimal degrees

	pp.ppp	Pitch standard deviation in decimal degrees
	hh.hhh	Heading standard deviation in decimal degrees
	QF	Quality Flag •0 = No position •1 = All non-RTK fixed integer positions 2 = RTK fixed integer position
	*CC	Checksum
	<CR>	Carriage return
	<LF>	Line feed

Example:	To turn on the PASHR message on THIS port issue the following command: $JASC,PASHR,1<CR><LF> ...and the message output appears similar to the following: $PASHR,162930.00,,T,2.48,3.92,-0.64,0.514,0.514,0.000,1*05 $PASHR,162931.00,,T,2.38,3.93,-0.70,0.508,0.508,0.000,1*07 $PASHR,162932.00,,T,2.67,4.00,-0.66,0.503,0.503,0.000,1*04
Additional Information:	
Related Commands and Messages:	PASHR message

Topic Last Updated: v1.06 / March 10, 2015

JASC,PSAT,ATTSTAT Command

Command Type:	Local Differential and RTK
Description:	The information of secondary antenna.
Command Format:	$JASC,PSAT,ATTSTAT,r[,OTHER]<CR><LF> where: •'r' = message rate (0 = Off, 1 = On at 1Hz) • ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on this message on the THIS port issue the following command: $JASC,PSAT,ATTSTAT,1<CR><LF>
Additional Information:	Issuing the JSAVE command after setting JASC,PSAT,ATTSTAT to 1 (message on at 1Hz) does not save this setting. You must enable JASC,PSAT,ATTSTAT (set it to 1) each time you power on the receiver.
Related Commands and Messages:	PSAT,ATTSTAT message

Topic Last Updated: v. 1.07/ October 13, 2016

JASC,PSAT,BLV Command

Command Type:	Local Differential and RTK
Description:	Configure the receiver to output the North, East, Upbase-line vector
Command Format:	$JASC,PSAT,BLV,r[,OTHER]<CR><LF> where: 'r' = message rate 0,1,2,5,10,20 (0 = Off, 1 = On at 1Hz) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on this message on the THIS port issue the following command: $JASC,PSAT,BLV,1<CR><LF>
Additional Information:	
Related Commands and Messages:	PSAT, BLV message

Topic Last Updated: v.1.07/October 13, 2016

JASC,PSAT,FVI Command

Command Type:	Local Differential and RTK
Description:	Contains information on position, standard deviation of position, heading, pitch, and roll along with standard deviations of the previous, horizontal and vertical velocities, as well as general position quality information.
Command Format:	$JASC,PSAT,FVI,r[,OTHER]<CR><LF> where: •'r' = message rate 0,1,2,5,10,20 (0 = Off, 1 = On at 1Hz) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on this message on the THIS port issue the following command: $JASC,PSAT,FVI,1<CR><LF>
Additional Information:	
Related Commands and Messages:	PSAT, FVI message

Topic Last Updated: v2.0/ April 30, 2019

JASC,PSAT,RTKPROG Command

Command Type:	Local Differential and RTK
Description:	Configure the receiver to output RTK fix progress
Command Format:	$JASC,PSAT,RTKPROG,r[,OTHER]<CR><LF> where: •'r' = message rate (0 = Off, 1 = On at 1Hz) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology. You can also perform a one-time query of the message information by issuing the JQUERY,RTKPROG command.
Receiver Response:	$>
Example:	To turn on this message on the THIS port issue the following command: $JASC,PSAT,RTKPROG,1<CR><LF>
Additional Information:	Issuing the JSAVE command after setting JASC,PSAT,RTKPROG to 1 (message on at 1Hz) does not save this setting. You must enable JASC,PSAT,RTKPROG (set it to 1) each time you power on the receiver.
Related Commands and Messages:	PSAT,RTKPROG message.

Topic Last Updated: v2.0/ April 30, 2019

JASC,PSAT,RTKSTAT Command

Command Type:	Local Differential and RTK
Description:	Configure the receiver to output the most relevant parameters affecting RTK
Command Format:	$JASC,PSAT,RTKSTAT,r[,OTHER]<CR><LF> where: •'r' = message rate (0 = Off, 1 = On at 1Hz) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology. You can also perform a one-time query of the message information by issuing the JQUERY,RTKSTAT command.
Receiver Response:	$>
Example:	To turn on this message on the THIS port issue the following command: $JASC,PSAT,RTKSTAT,1<CR><LF>
Additional Information:	Issuing the JSAVE command after setting JASC,PSAT,RTKSTAT to 1 (message on at 1Hz) does not save this setting. You must enable JASC,PSAT,RTKSTAT (set it to 1) each time you power on the receiver.
Related Commands	JQUERY,RTKSTAT command PSAT,RTKSTAT message.

and Messages:	

Topic Last Updated: v1.05 / January 18, 2013

JASC,PSAT,VCT Command

Command Type:	Local Differential and RTK
Description:	Configure the receiver to output the most relevant parameters affecting RTK
Command Format:	$JASC,PSAT,VCT,r[,OTHER]<CR><LF> where: •'r' = message rate 0,1,2,5,10,20 (0 = Off, 1 = On at 1Hz) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on this message on the THIS port issue the following command: $JASC,PSAT,VCT,1<CR><LF>
Additional Information:	
Related Commands and Messages:	PSAT, VCT message.

Topic Last Updated: v1.07 / October 13, 2016

JASC,PTSS1 Command

Command Type:	Vector
Description:	Configure the receiver to output heave, pitch, and roll in the commonly used TSS1 message format
Command Format:	$JASC,PTSS1,r[,OTHER]<CR><LF> where: •'r' = messagerate (in Hz) of 0 (off), 0.25,0.5, 1, 2, 4, 5, 10, or 20 (if subscribed) ·',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	XXAAAASMHHHHQMRRRRSMPPPP*CC<CR><LF> where:

Message Component	Description
XX	Horizontal acceleration
AAAA	Vertical acceleration
HHHH	Heave, in centimeters
S	S = space character
M	Space if positive; minus if negative
Q	Status flag
h	Heading aided mode (settling) -

		The System is receiving heading aiding signals from a gyrocompass but is still awaiting the end of the three minutes settling period after power-on or a change of mode or heave bandwidth. The gyrocompass takes approximately five minutes to settle after it has been powered on. During this time, gyrocompass aiding of the System will not be perfect. The status flag does NOT indicate this condition.
	F	Full aided mode (settled condition) - The System is receiving and using aiding signals from a gyrocompass and from a GPS receiver or a Doppler log.
	M	Space if positive; minus if negative
	RRRR	Roll, in units of 0.01 degrees (ex: 1000 = 10°)
	S	S = space character
	M	Space if positive; minus if negative
	PPPP	Pitch, in units of 0.01 degrees (ex: 1000 = 10°)
	<CR>	Carriage return
Example:		
Additional Information:		
Related Commands and Messages:	TSS1 message	

Topic Last Updated: v1.06 / March 10, 2015

JASC,ROX Command

Command Type:	Local Differential and RTK
Description:	Set the proprietary ROX messages to on or off to provide corrections to the rover This command only applies to an Eclipse base station receiver when using GPS dual frequency RTK mode. RTK is relative to the reference position (base only).
Command Format:	$JASC,ROX,r[,OTHER]<CR><LF> where: •'r' = correction status variable (0 = turn corrections Off, 1 = turn corrections On) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on ROX messages on the OTHER port issue the following command: $JASC,ROX,1,OTHER<CR><LF>
Additional Information:	To query the receiver for the current setting, issue the JSHOW command. To change the broadcast station ID, use JRTK,28.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JASC,RTCM Command

Command Type:	SBAS
Description:	Configure the receiver to output RTCM version 2 DGPS corrections from SBAS or beacon through either receiver serial port. The correction data output is RTCM SC-104,even though SBAS uses a different over-the-air protocol (RTCA).
Command Format:	$JASC,RTCM,r[,OTHER]<CR><LF> where: •'r' = message status variable (0 = Off, 1 = On) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To output RTCM corrections from SBAS or beacon on THIS port (current port) issue the following command: $JASC,RTCM,1<CR><LF>
Additional Information:	To verify the current setting is on, issue the JSHOW command. You will see output similar to the following: $>JSHOW,ASC,RTCM,1.0 If the current setting is off, the JSHOW command will not show any information for this setting.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JASC,RTCM3 Command

Command Type:	Local Differential and RTK
Description:	Set the RTCM version 3 messages to on or off to provide corrections to the rover. This command only applies to an Eclipse base station receiver when using GPS dual frequency RTK mode. RTK is relative to the reference position (base only).
Command Format:	$JASC,RTCM3,r[,OTHER]<CR><LF> where: •'r' = correction status variable (0 = turn corrections Off, 1 = turn corrections On) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To turn on RTCM3 messages on the OTHER port issue the following command: $JASC,RTCM3,1,OTHER<CR><LF>
Additional	To query the receiver for the current setting, issue the JSHOW command. To change

Information:	the broadcast station ID, use JRTK,28.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JASC,VIRTUAL Command

Command Type:	General Operation and Configuration
Description:	Configure the receiver to have RTCM data input on one port and output through the other (when using an external correction source). For example, if RTCM is input on Port B, the data will be output through Port A having corrected the receiver position. he receiver acts as a pass-through for the RTCM data. Either port may be configured to accept RTCM data input; this command enables the opposite port to output the RTCM data.
Command Format:	$JASC,VIRTUAL,r[,OTHER]<CR><LF> where: •'r' = message status variable (0 = Off, 1 = On) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.
Receiver Response:	$>
Example:	To configure THIS port to output RTCM messages that are being input through the OTHER port issue the following command: $JASC,VIRTUAL,1
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JATLAS Commands

Command Type:	L-band
Description:	When using Atlas, configure the accuracy threshold for when the GPGGA quality indicator reports a Fix.
Command Format:	$JATLAS,LIMIT,[OPTION],[THRESHOLD],SAVE<CR><LF> where: ·[THRESHOLD] is in meters ·The SAVE field is optional. However, if omitted this setting will not survive a power cycle. $JSAVE does not save this setting. ·Options are 3D, HORI, or VERT

	To configure the receiver so that it reports an RTK fix when the Atlas solution has converged to 3D accuracy of 30cm, send: $JATLAS,LIMIT,3D,0.3,SAVE<CR><LF> Query the current setting: $JATLAS,LIMIT<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

$JATLAS,POS,PRESENT [,OTHER] Command

Command Type:	ATLAS
Description:	Saves the current location and associated standard deviations into nonvolatile memory (provided that the present position is sufficiently stable), to be used with Atlas position seeding. Use of “OTHER” saves the position that is used by the Atlas Autoseed algorithm; otherwise, saves the position that is used for manual position seeding.
Command Format:	$JATLAS,POS,PRESENT[,OTHER] <CR><LF> Query the current setting: See $JATLAS,POS[,OTHER]
Receiver Response:	If the present position is stable, the response is: $> If the present position is not stable, the command is ignored and the response is: Present Location Not Stable
Example:	
Additional Information:	See $JATLAS,MODE,AUTOSEED and $JATLAS,SEED for additional information about position seeding.
Related Commands and Messages:	

Topic Last Updated: v.3.0/ December 30, 2019

$JATLAS,POS[,OTHER]

Command Type:	ATLAS
Description:	Query the receiver for the stored position and standard deviations to be used with Atlas position seeding. Use of “OTHER” displays the position that is used by the Atlas Autoseed algorithm; otherwise, displays the position that is used for manual position seeding.
Command Format:	$JATLAS,POS[,OTHER] <CR><LF>

<table>
<tr><td>Receiver Response:</td><td>$>JATLAS,POS,lat,lon,hgt,(LatStDev,LonStDev,HgtStDev)

where:

<table>
<tr><th>Command Component</th><th>Description</th></tr>
<tr><td>lat</td><td>Latitude in decimal degrees</td></tr>
<tr><td>lon</td><td>Longitude in decimal degrees</td></tr>
<tr><td>hgt</td><td>Ellipsoidal height in meters.
Ellipsoidal height can be calculated by adding the altitude and the geoidal separation, both available from the GPGGA message.
Example:
$GPGGA,173309.00,5101.04028,N,11402.38289,W,2,07,1
.4,1071.0,
M,- 17.8,M,6.0, 0122*48
ellipsoidal height = 1071.0 + (-17.8) = 1053.2 meters</td></tr>
<tr><td>LatStDev</td><td>Standard deviation of latitude in meters</td></tr>
<tr><td>LonStDev</td><td>Standard deviation of longitude in meters</td></tr>
<tr><td>HgtStDev</td><td>Standard deviation of height in meters</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td>A response to querying the saved position would look like:
$>JATLAS,POS,33.64334383,-111.89596094,455.244,(0.062,0.086,0.156)</td></tr>
<tr><td>Additional Information:</td><td>See $JATLAS,MODE,AUTOSEED and $JATLAS,SEED for additional information about position seeding.</td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v.3.0/ December 30, 2019

$JATLAS,POS,lat,lon,hgt[,LatStDev,LonStDev,HgtStDev][,OTHER]

<table>
<tr><td>Command Type:</td><td>ATLAS</td></tr>
<tr><td>Description:</td><td>Saves the input position and optionally the corresponding standard deviation into non-volatile memory, to be used with Atlas position seeding.

Use of “OTHER” saves the position that is used by the Atlas Autoseed algorithm; otherwise, saves the position that is used for manual position seeding.</td></tr>
<tr><td>Command Format:</td><td>$JATLAS,POS[,OTHER] <CR><LF>

where:

<table>
<tr><th>Command Component</th><th>Description</th></tr>
<tr><td>lat</td><td>Latitude in decimal degrees</td></tr>
<tr><td>lon</td><td>Longitude in decimal degrees</td></tr>
<tr><td>hgt</td><td>Ellipsoidal height in meters.
Ellipsoidal height can be calculated by adding the altitude and the geoidal separation, both available from the GPGGA message.
Example:
$GPGGA,173309.00,5101.04028,N,11402.38289,</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td></td><td>W,2,07,1.4,1071.0, M,- 17.8,M,6.0, 0122*48 ellipsoidal height = 1071.0 + (-17.8) = 1053.2 meters</td></tr><tr><td>LatStDev</td><td>Standard deviation of latitude in meters</td></tr><tr><td>LonStDev</td><td>Standard deviation of longitude in meters</td></tr><tr><td>HgtStDev</td><td>Standard deviation of height in meters</td></tr></table> Query the current setting: See $JATLAS,POS[,OTHER]
Receiver Response:	$>
Example:	$JATLAS,POS,33.64334383,-111.89596094,455.244,0.062,0.086,0.156<CR><LF>
Additional Information:	See $JATLAS,MODE,AUTOSEED and $JATLAS,SEED for additional information about position seeding.
Related Commands and Messages:	

Topic Last Updated: v.3.0/ December 30, 2019

$JATLAS,SEED[,OTHER]

Command Type:	ATLAS
Description:	Manually seed the Atlas solution with the saved position and standard deviations. Use of “OTHER” seeds the position using the location stored for the Atlas Autoseed algorithm; otherwise, seeds using the location stored for manual position seeding.
Command Format:	$JATLAS,SEED[,OTHER] <CR><LF> Query the current setting:
Receiver Response:	$> If the seed position is not close enough to the current location, the response is: $>JATLAS,SEED,Current Position Too Far From Seed
Example:	
Additional Information:	Position seeding can reduce Atlas convergence time by supplying the engine with a known position at initialization. Warning: Manual seeding should be used with caution, as any errors entered here will affect the future accuracy of the position solution. The seed position coordinates should generally be known to within several centimeters before attempting to seed the position. See also $JATLAS,MODE,AUTOSEED, which handles the Atlas position seeding automatically.
Related Commands and Messages:	

Topic Last Updated: v.3.0/ December 30, 2019

$JATLAS,SEED,lat,lon,hgt[,LatStDev,LonStDev,HgtStDev]

<table>
<tr><td>Command Type:</td><td>ATLAS</td></tr>
<tr><td>Description:</td><td>Manually seed the Atlas solution with the saved position and standard deviations.</td></tr>
<tr><td>Command Format:</td><td>$JATLAS,SEED,lat,lon,hgt,[LatStDev,LonStDev,HgtStDev]<CR><LF>
where:

<table>
<tr><th>Command Component</th><th>Description</th></tr>
<tr><td>lat</td><td>Latitude in decimal degrees</td></tr>
<tr><td>lon</td><td>Longitude in decimal degrees</td></tr>
<tr><td>hgt</td><td>Ellipsoidal height in meters.
Ellipsoidal height can be calculated by adding the altitude and the geoidal separation, both available from the GPGGA message.

Example:
$GPGGA,173309.00,5101.04028,N,11402.38289,W,2,07,1.4,
1071.0,
M,- 17.8,M,6.0, 0122*48
ellipsoidal height = 1071.0 + (-17.8) = 1053.2 meters</td></tr>
<tr><td>LatStDev</td><td>Standard deviation of latitude in meters</td></tr>
<tr><td>LonStDev</td><td>Standard deviation of longitude in meters</td></tr>
<tr><td>HgtStDev</td><td>Standard deviation of height in meters</td></tr>
</table>

Query the current setting:</td></tr>
<tr><td>Receiver Response:</td><td>$> If the input coordinates are not close enough to the current location, the response is:

$>JATLAS,SEED,Current Position Too Far From Seed</td></tr>
<tr><td>Example:</td><td>$JATLAS,SEED,33.64334383,-111.89596094,455.244,0.062,0.086,0.156<CR><LF></td></tr>
<tr><td>Additional Information:</td><td>Position seeding can reduce Atlas convergence time by supplying the engine with a known position at initialization.

Warning: Manual seeding should be used with caution, as any errors entered here will affect the future accuracy of the position solution. The seed position coordinates should generally be known to within several centimeters before attempting to seed the position. See also $JATLAS,MODE,AUTOSEED, which handles the Atlas position seeding automatically.</td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v.3.0/ December 30, 2019

$JATLAS,MODE,AUTOSEED[,YES/NO]

Command Type:	ATLAS
Description:	Enable or disable the Atlas AUTOSEED feature or query the current setting.
Command Format:	To enable the AUTOSEED feature:

	$JATLAS,MODE,AUTOSEED,YES<CR><LF> To disable the AUTOSEED feature: $JATLAS,MODE,AUTOSEED,NO<CR><LF> Query the current setting: $JATLAS,MODE,AUTOSEED<CR><LF>
Receiver Response:	Response to issuing command to enable/disable AUTOSEED feature: $> Response to querying the current setting: $>JATLAS,MODE,AUTOSEED,[YES/NO]
Example:	
Additional Information:	Position seeding can reduce Atlas convergence time by supplying the engine with a known position at initialization. When AUTOSEED is enabled, receiver locations are automatically saved to memory. The last saved position will then automatically be used to seed the solution when the receiver is powered back on (under appropriate conditions—see below). The setting for AUTOSEED mode is automatically saved to memory. Warning: The AUTOSEED position can only be saved if the receiver has not detected motion for 5 s. It is therefore recommended that the user allow sufficient stationary time before powering off. Warning: The antenna must not be moved after being powered off. The antenna must continue to remain stationary when powered back on and until the seeding process completes. The AUTOSEED feature may not function properly if the antenna has moved more than several centimeters during this time.
Related Commands and Messages:	

Topic Last Updated: v.3.0/ December 30, 2019

$JATLAS,RESET,ENGINE

Command Type:	ATLAS
Description:	The $JATLAS,RESET,ENGINE command resets the Atlas engine. Reset the Atlas engine, forcing the solution to re-converge.
Command Format:	$JATLAS,RESET,ENGINE Query the current setting:
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and	

Messages:	

Topic Last Updated: v.3.0/ December 30, 2019

$JATLAS,STATUS,AUTOSEED

Command Type:	ATLAS
Description:	The $JATLAS,STATUS,AUTOSEED command displays the status of the AUTOSEED initialization process. Displays the status of the Atlas AUTOSEED initialization process.
Command Format:	$JATLAS,STATUS,AUTOSEED<CR><LF> Query the current setting:
Receiver Response:	$>JATLAS,STATUS,AUTOSEED,status where 'status' is one of following:

Status	Description
NoAtlas	Autoseeding cannot occur because the Atlas solution is not available.
Disabled	Autoseed mode is not enabled.
Seeding	Autoseeding is in process.
Failed_NoSeed	Autoseeding failed because no seed position is available
Failed_Moved	Autoseeding failed because receiver motion was detected during the seeding process.
Failed_Timeout	Autoseeding failed to complete within the required time.
Success	Autoseeding was successful.

Example:	
Additional Information:	See also $JATLAS,MODE,AUTOSEED for additional information about Atlas Autoseed
Related Commands and Messages:	

Topic Last Updated: v.3.0/ December 30, 2019

JATT,COGTAU Command

Command Type:	Vector
Description:	The $JATLAS,STATUS,AUTOSEED command displays the status of the AUTOSEED initialization process. Displays the status of the Atlas AUTOSEED initialization process. Set the course over ground (COG) time constant (0.0 to 200.0seconds) or query the current setting. This command allows you to adjust the level of responsiveness of the COG measurement provided in the GPVTG message. The default value is 0.0 seconds of smoothing. Increasing the COG time constant increases the level of COG smoothing. COG is computed using only the primary GPS antenna (when using a multi-antenna system) and its accuracy depends upon the speed of the vessel (noise is

	proportional to 1/speed). This value is invalid when the vessel is stationary, as tiny movements due to calculation inaccuracies are not representative of a vessel's movement. Note: The JTAU,COG command provides identical functionality but works with positioning and heading products.
Command Format:	Set the COG time constant: $JATT,COGTAU,cogtau<CR><LF> where 'cogtau' is the new COG time constant that falls within the range of 0.0 to 200.0 seconds The setting of this value depends upon the expected dynamics of the Crescent. If the Crescent will be in a highly dynamic environment, this value should be set lower because the filtering window would be shorter, resulting in a more responsive measurement. However, if the receiver will be in a largely static environment, this value can be increased to reduce measurement noise. Query the current setting: $JATT,COGTAU<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	You can use the following formula to determine the COG time constant: cogtau (in seconds) = 10 / maximum rate of change of course (in °/s). If you are unsure about the best value for this setting, it is best to be conservative and leave it at the default setting of 0.0 seconds.
Related Commands and Messages:	

Topic Last Updated: v2.0/ April 30, 2019

JATT,CSEP Command

Command Type:	Vector
Description:	Query the Vector for the current calculated separation between antennas, as solved for by the attitude algorithms.
Command Format:	$JATT,CSEP<CR><LF>
Receiver Response:	$> $>JATT,X,CSEP where 'X' is the antenna separation in meters
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,EXACT Command

Command Type:	Vector
Description:	Enabling forces, the heading calculation to rely on the MSEP value (see $JATT,MSEP). Disabling forces, the heading calculation to rely on the CSEP and the MSEP values.
Command Format:	Enable/disable internal filter reliance To enable internal filter reliance: $JATT,EXACT,YES<CR><LF> To disable internal filter reliance: $JATT,EXACT,NO<CR><LF> Query the current setting: $JATT,EXACT<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v2.0/April 30, 2019

JATT,FLIPBRD Command

Command Type:	Vector
Description:	Turn the flip feature on/off or query the current feature status Allow the Vector OEM board to be installed upside down. You should use this command only with the Vector Sensor and the Vector OEM board because flipping the OEM board does not affect the antenna array that needs to remain facing upwards. When using this command, the board needs to be flipped about roll so the front still faces the front of the vessel. For all OEM heading boards starting the H328 and H220, this command is replaced with $JATT,ACC180 and $JATT,ACC90.
Command Format:	Turn the flip feature on/off To turn the flip feature on: $JATT,FLIPBRD,YES<CR><LF> To turn the flip, feature off (return to default mode - right side up): $JATT,FLIPBRD,NO<CR><LF> Query current the current setting: $JATT,FLIPBRD<CR><LF>
Receiver Response:	$>

Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

JATT,GYROAID Command

Command Type:	Vector
Description:	Turn gyro aiding on or off or query the current setting The Vector's internal gyro—enabled by default when shipped—offers two benefits. It shortens reacquisition times when a GPS heading is lost because of obstruction of satellite signals. It does this by reducing the search volume required for solution of the RTK. It provides an accurate substitute heading for a short period (depending on the roll and pitch of the vessel) ideally seeing the system through to reacquisition. For these two benefits, Hemisphere GNSS highly recommend leaving gyro aiding on. Exceeding rates of 90°/sec is not recommended because the gyro cannot measure rates beyond this point. This is a new recommendation since Hemisphere GNSS now uses gyro measurements to obtain a heading rate measurement.
Command Format:	Turn gyro aiding on/off To turn gyro aiding on: $JATT,GYROAID,YES<CR><LF> To turn gyro aiding off: $JATT,GYROAID,NO<CR><LF> Query the current setting: $JATT,GYROAID<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	Every time you power up the Vector the gyro goes through a warm-up procedure and calibrates itself. You cannot save the resulting calibration, so the self-calibration takes place every time the Vector is power cycled. This self-calibration procedure takes several minutes and is the equivalent of the following manual calibration procedure. With the Vector unit installed: 1.Apply power and wait several minutes until it has acquired a GPS signal and is computing heading. 2.Ensure gyro aiding is on by issuing the following command: $JATT,GYROAID<CR><LF> 3.Slowly spin the unit for one minute at no more than 15°/sec.

	4.Keep the unit stationary for four minutes. Both the manual and the self-calibration Procedures calibrate the Crescent Vector's gyro to the same effect.
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,HBIAS Command

Command Type:	Vector
Description:	Set the heading output from the Vector to calibrate the true heading of the antenna array to reflect the true heading of the vessel or query the current setting.
Command Format:	Set the heading output: $JATT,HBIAS,x<CR><LF> where 'x' is a bias that will be added to the Vector's heading in degrees. The acceptable range for the heading bias is - 180.0° to 180.0°. The default value of this feature is 0.0°. Query the current setting (current compensation angle): $JATT,HBIAS<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,HELP Command

Command Type:	Vector
Description:	Show the available commands for GPS heading operation and status
Command Format:	$JATT,HELP<CR><LF>
Receiver Response:	$>JATT,HELP,CSEP,MSEP,EXACT,LEVEL,HTAU,HRTAU,HBIASPBIAS,NEGTILT,ROLL,TILTAID, TILTCAL,MAGAID,MAGCAL,MAGCLR,GYROAID,COGTAU,SPDTAU,SEARCH, SUMMARY
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,HIGHMP Command

Command Type:	Vector

Description:	Enable/disable the high multipath setting for use in poor GPS environments or query the current setting Enabling HIGHMP mode may result in longer heading acquisition times in high multipath environments. In HIGHMP mode, the Vector will not output heading until it has good confidence in the result. In very poor environments, this may take a few minutes or more; in normal environments, there is only a slight increase in heading acquisition time.
Command Format:	Set the high multipath setting To enable the high multipath setting: $JATT,HIGHMP,YES<CR><LF> To disable the high multipath setting: $JATT,HIGHMP,NO<CR><LF> Query the current setting: $JATT,HIGHMP<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,HRTAU Command

Command Type:	Vector
Description:	Set the rate of turn (ROT) time constant to adjust the level of responsiveness of the ROT measurement provided in the GPROT message or query the current setting The default value of this constant is 2.0 seconds of smoothing. Increasing the time constant increases the level of ROT smoothing.
Command Format:	Set the heading rate time constant: $JATT,HRTAU,hrtau<CR><LF> where 'hrtau' is the new time constant that falls within the range of 0.0 to seconds The setting of this value depends upon the expected dynamics of the vessel. For example, if the vessel is very large and cannot turn quickly, increasing this time is reasonable. The resulting heading would have reduced ‘noise’, resulting in consistent values with time. However, artificially increasing this value such that it does not agree with a more dynamic vessel could create a lag in the ROT measurement with higher rates of turn. Query the current setting: $JATT,HRTAU<CR><LF>

Receiver Response:	$>
Example:	
Additional Information:	You can use the following formula to determine the level of smoothing: hrtau (in seconds) = 10 / maximum rate of the rate of turn (in °/s2) Note: If you are unsure about the best value for the setting, leave it at the default setting of 2.0 seconds.
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,HTAU Command

Command Type:	Vector
Description:	Set the heading time constant to adjust the level of responsiveness of the true heading measurement provided in the GPHDT message or query the current setting. For OEM boards the default value of this constant is 0.5 seconds of smoothing (regardless of whether the gyro is enabled or disabled). For finished products that implement an OEM board the default value may be different—check your product's documentation for this value. Although the gyro is enabled by default, you can disable it. Increasing the heading time constant increases the level of heading smoothing and increases lag only if the gyro is disabled.
Command Format:	Set the heading time constant: $JATT,HTAU,htau<CR><LF> where 'htau' is the new time constant that falls within the range of 0.0 to seconds The setting of this value depends upon the expected dynamics of the vessel. If the vessel is very large and cannot turn quickly, increasing this time is reasonable. The resulting heading would have reduced ‘noise’ resulting in consistent values with time. However, artificially increasing this value such that it does not agree with a more dynamic vessel could create a lag in the heading measurement with higher rates of turn. Query the current setting: $JATT,HTAU<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	You can use the following formula to determine level of heading smoothing required when the gyro is in use: Gyro on htau (in seconds) = 40 / maximum rate of turn (in °/s) Gyro off htau (in seconds) = 10 / maximum rate of turn (in °/s) If you are unsure about the best value for the setting, leave it at the default setting of 2.0 seconds when the gyro is on and at 0.5 seconds when the gyro is off.

Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,LEVEL Command

Command Type:	Vector
Description:	Turn level operation on or off or query the current setting If the Vector will be operated within ±10° of level, you may use this mode of operation for increased robustness and faster acquisition times of the heading solution.
Command Format:	Turn level operation on/off To turn level operation on: $JATT,LEVEL,YES<CR><LF> To turn level operation off: $JATT,LEVEL,NO<CR><LF> Query the current setting: $JATT,LEVEL<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.05 / January 18, 2013

JATT,MOVEBASE Command

Command Type:	Vector
Description:	Set the auto GPS antenna separation or query the current setting If the operation is turned on ,you do not need to set the GPS antenna separation manually . Only multi-frequency boards are supported.
Command Format:	Turn move base on/off To turn move base operation on: $JATT,MOVEBASE,YES<CR><LF> To turn move base operation off: $JATT,MOVEBASE,NO<CR><LF> Query the current setting:

	$JATT,MOVEBASE<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

JATT,MSEP Command

Command Type:	Vector
Description:	Manually enter a custom separation between antennas (must be accurate to within 2 cm) or query the current setting.
Command Format:	Set the antenna separation: Using the new center-to-center measurement, issue the following command: $JATT,MSEP,sep<CR><LF> where 'sep' is the measured antenna separation entered in meters Query the current setting: $JATT,MSEP<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,NEGTILT Command

This command was intentionally removed.

JATT,NMEAHE Command

Command Type:	Vector
Description:	Instruct the Vector to preface the following messages with GP or HE. •HDG •HDM •HDT •ROT
Command Format:	$JATT,NMEAHE,x<CR><LF> where 'x' is either 1 for HE or 0 for GP To preface specific messages with GP: $JATT,NMEAHE,0<CR><LF>

	To preface specific messages with HE: $JATT,NMEAHE,1<CR><LF>
Receiver Response:	$> $>JATT,NMEAHE,OK
Example:	
Additional Information:	The HDM message is for a magnetic compass. The message will be HCHDM when requesting with $JATT,NMEAHE,1 specified.
Related Commands and Messages:	

Topic Last Updated: v1.06 /

JATT,PBIAS Command

Command Type:	Vector
Description:	Set the pitch/roll output from the Vector to calibrate the measurement if the antenna array is not installed in a horizontal plane or query the current setting.
Command Format:	Set the pitch/roll output: $JATT,PBIAS,x<CR><LF> where 'x' is a bias that will be added to the Vector's pitch/roll measure, in degrees The acceptable range for the pitch bias is -15.0° to 15.0°. The default value is 0.0°. Query the current setting: $JATT,PBIAS<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	Note: The pitch/roll bias is added after the negation of the pitch/roll measurement (if invoked with the JATT,NEGTILT command). Use PBIAS to describe any angular differences between the level of the two GPS antennas. Pitch is the default, but if the antennas are mounted in the roll direction, you can still enter the roll bias in PBIAS (make sure JATT,ROLL,YES is set).
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JATT,PTAU Command

Command Type:	Vector
Description:	Set the level of responsiveness of the pitch measurement provided in the PSAT,HPR message or query the current setting. For OEM boards the default value of this constant is 0.5 seconds of smoothing (regardless of whether the gyro is enabled or disabled). For finished products that implement an OEM board the default value may be different—check your product's documentation for this value. Increasing the pitch time constant increases, the level of pitch smoothing and increases lag.
Command Format:	Set the pitch time constant: $JATT,PTAU,ptau<CR><LF>

	where 'ptau' is the new time constant that falls within the range of 0.0 to 3600.0 seconds. The setting of this value depends upon the expected dynamics of the vessel. For instance, if the vessel is very large and cannot pitch quickly, increasing this time is reasonable. The resulting pitch would have reduced 'noise', resulting in consistent values with time. However, artificially increasing this value such that it does not agree with a more dynamic vessel could create a lag in the pitch measurement. Query the current setting: $JATT,PTAU<CR><LF> Note: If you are unsure about the best value for the setting, leave it at the default setting of 0.5 seconds.
Receiver Response:	$>
Example:	
Additional Information:	You can use the following formula to determine the level of pitch smoothing required: ptau (in seconds) = 10 / maximum rate of pitch (in °/s)
Related Commands and Messages:	

Topic Last Updated: : v1.06 / March 10, 2015

JATT,ROLL Command

Command Type:	Vector
Description:	Configure the Vector for roll or pitch GPS antenna orientation.
Command Format:	Configure the Vector for pitch or roll GPS antenna orientation: To configure the Vector for roll GPS antenna orientation (the Antenna Array must be installed perpendicular to the vessel's axis): $JATT,ROLL,YES<CR><LF> To configure the Vector for pitch GPS antenna orientation (default): $JATT,ROLL,NO<CR><LF> Query the current setting: $JATT,ROLL<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	You can use the following formula to determine the level of pitch smoothing required: ptau (in seconds) = 10 / maximum rate of pitch (in °/s)
Related Commands and Messages:	

Topic Last Updated: : v1.06

JATT,SEARCH Command

Command Type:	Vector
Description:	Force the Vector to reject the current GPS heading solution and begin a new search.
Command Format:	$JATT,SEARCH<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	The SEARCH function will not work if you have enabled the gyroaid feature (using the GYROAID command). In this case you must cycle power to the receiver to have a new GPS solution computed.
Related Commands and Messages:	

Topic Last Updated: : v1.06 / March 10, 2015

JATT,SPDTAU Command

Command Type:	Vector
Description:	Note: The JTAU,SPEED command provides identical functionality but works with Crescent and Eclipse products in addition to Crescent Vector products. Set the speed time constant (0.0 to 3600.0seconds) or query the current setting. This command allows you to adjust the level of responsiveness of the speed measurement provided in the GPVTG message. The default value is 0.0 seconds of smoothing. Increasing the speed time constant increases, the level of speed measurement smoothing.
Command Format:	Set the speed time constant: $JATT,SPDTAU,spdtau<CR><LF> where 'spdtau' is the new time constant that falls within the range of 0.0 to 200.1 seconds The setting of this value depends upon the expected dynamics of the receiver. If the receiver will be in a highly dynamic environment, you should set this to a lower value, since the filtering window will be shorter, resulting in a more responsive measurement. However, if the receiver will be in a largely static environment, you can increase this value to reduce measurement noise. Query the current setting: $JATT,SPDTAU<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	You can use the following formula to determine the COG time constant (Hemisphere GNSS recommends testing how the revised value works in practice): spdtau (in seconds) = 10 / maximum acceleration (in m/s2) If you are unsure about the best value for this setting, it is best to be conservative

	and leave it at the default setting of 0.00 seconds.
Related Commands and Messages:	

Topic Last Updated: : v1.06 / March 10, 2015

JATT,SUMMARY Command

Command Type:	Vector
Description:	Display a summary of the current Vector settings.
Command Format:	$JATT,SUMMARY<CR><LF> Receiver Response: $>JATT,SUMMARY,htau,hrtau,ptau,cogtau,spdtau,hbias,pbias,hexflag<CR><LF> where:

Command Component	Description
htau	Current heading time constant, in seconds
hrtau	Current heading rate time constant, in seconds
ptau	Current pitch time constant, in seconds
cogtau	Current course over ground time constant, in seconds
spdtau	Current speed time constant, in seconds
hbias	Current heading bias, in degrees
pbias	Current pitch/roll bias, in degrees
hexflag	Hex code that summarizes the heading feature status:

Flag	On	Off
Gyro aiding	02	0
Negative tilt	01	0
Roll	08	0
Tilt aiding	02	0
Level	01	0

The 'hexflag' field is two separate hex flags:

- •'GN' - Value is determined by computing the sum of the gyro aiding and negative tilt values, depending on whether they are on or off:
- •If the feature is on, their value is included in the sum
- •If the feature is off, it has a value of zero when computing the sum
- •'RTML'- Value is determined in much the same way, but by adding the value of roll, tilt aiding, and level operation

For example, if gyro aiding, roll, and tilt aiding features were each on, the values of 'GN' and 'RMTL' would be:

•'GN' = hex (02 + 0) = hex (02) = 2

•'RMTL' = hex (08 + 02) = hex (10) = A
•'GN-RMTL' = 2A

The following tables summarize the possible feature configurations for the first 'GN' character and the second 'RMTL' character.

JATT, SUMMARY 1st GN Character Configurations		
GN Value	**Gyro Value**	**Negative Tilt**
0	Off	Off
1	Off	On
2	On	Off
3	On	On

JATT,SUMMARY 2nd RMTL Character Configurations			
RMTL Value	**Roll**	**Tilt Aiding**	**Level**
0	Off	Off	Off
1	Off	Off	On
2	Off	On	Off
3	Off	On	On
8	On	Off	Off
9	On	Off	On
A	On	On	Off
B	On	On	On

Receiver Response:	$>
Example:	$>JATT,SUMMARY,TAU:H=0.50,HR=2.00,COG=0.00,SPD=0.00,BIAS:H=0.00,P=0.00, FLAG_HEX:HF-RMTL=01
Additional Information:	You can use the following formula to determine the COG time constant (Hemisphere GNSS recommends testing how the revised value works in practice): spdtau (in seconds) = 10 / maximum acceleration (in m/s2) If you are unsure about the best value for this setting, it is best to be conservative and leave it at the default setting of 0.00 seconds.
Related Commands and Messages:	

Topic Last Updated: : v1.06 / March 10, 2015

JATT,TILTAID Command

Command Type:	Vector
Description:	Turn tilt aiding on or off or query the current setting. The Vector's internal tilt sensors (accelerometers) may be enabled by default (see your specific product manuals for further information). The sensors act to reduce the RTK search volume, which improves heading startup and reacquisition times. This improves the reliability and accuracy of selecting the correct heading solution by eliminating other possible, erroneous solutions.
Command Format:	Turn tilt aiding on/off Turn tilt aiding on: $JATT,TILTAID,YES<CR><LF> Turn tilt aiding off: $JATT,TILTAID,NO<CR><LF> Query the current setting: $JATT,TILTAID<CR><LF>
Receiver Response:	$> Response to querying the current setting: If setting is currently ON the response is: $>JATT,TILTAID,ON If setting is currently OFF the response is: $>JATT,TILTAID,OFF
Example:	
Additional Information:	Tilt aiding is required to increase the antenna separation of the Vector OEM beyond the default 0.5 m length.
Related Commands and Messages:	

Topic Last Updated: : v.1. 06/ March 10, 2015

JATT,TILTCAL Command

Command Type:	Vector
Description:	Turn tilt aiding on or off or query the current setting. The Vector's internal tilt sensors (accelerometers) may be enabled by default (see your specific product manuals for further information). The sensors act to reduce the RTK search volume, which improves heading startup

	and reacquisition times. This improves the reliability and accuracy of selecting the correct heading solution by eliminating other possible, erroneous solutions.
Command Format:	Turn tilt aiding on/off Turn tilt aiding on: $JATT,TILTAID,YES<CR><LF> Turn tilt aiding off: $JATT,TILTAID,NO<CR><LF> Query the current setting: $JATT,TILTAID<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: : v.1. 06/ March 10, 2015

JBAUD Command

Command Type:	General Operation and Configuration
Description:	Specify the baud rates of the receiver or query the current setting.
Command Format:	Specify the baud rates: $JBAUD,r[,OTHER][,SAVE]<CR><LF> where: ·'r' = baud rate (4800, 9600, 19200, 38400, 57600, or 115200) • ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) •',SAVE' = optional field, saves the baud rate into flash memory so that if you reset power the receiver will boot at the new baud rate (it may take several seconds to save the baud rate to flash memory) Query the current setting: $JBAUD[,OTHER]<CR><LF> where: •',OTHER' = optionalfield, queries the current port when you sendthe command without it (and withoutthe brackets) and queries the other port when you send the command with it (without the brackets)

Receiver Response:	$>JBAUD,R[,OTHER] The response format is the same whether you specify the baud rates or query the current settings.
Example:	Issue the following command to set the baud rate to 19200 on the current port: $JBAUD,19200<CR><LF> ...the response is then: $>JBAUD,19200 Issue the following command to set the baud rate to 9600 on the OTHER port and save it into memory: $JBAUD,9600,OTHER,SAVE<CR><LF> ...the response is then: $>JBAUD,9600,OTHER
Additional Information:	Note: When saving the baud rate wait until you see the SAVE COMPLETE message before powering off the receiver. See the JSAVE command for an example of this output. The status of this command is also output when issuing the JSHOW command.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JBIN Command

Command Type:	General Operation and Configuration
Description:	Enable the output of the various binary messages
Command Format:	$JBIN,msg,r<CR><LF> where: •'msg' = binary message you want to output •'r' = message rate as shown in the following table

Message Name	Msg	R (Hz)	Description
Bin1	1	20, 10, 2, 1, 0, or.2	GPS position message (position and velocity data)
Bin2	2	1 or 0	GPS DOPs (Dilution of Precision)
Bin3	3	20, 10, 2, 1, 0, or.2	Lat/Lon/Hgt, Covariances, RMS, DOPs and COG, Speed, Heading
Bin5	5	1 or 0	Base station information
Bin16	16		All constellation code and phase observation data
Bin19			GNSS diagnostic information
Bin35	35	1 or 0	BeiDou ephemeris information

	Bin36	36	1 or 0	BeiDou code and carrier phase information (all frequencies)
	Bin44	44		GALILEO time conversion
	Bin45	45		GALILEO ephemeris
	Bin62	62	1 or 0	GLONASS almanac information
	Bin65	65	1 or 0	GLONASS ephemeris information
	Bin66	66	20, 10, 2, 1, or 0	GLONASS L1/L2 code and carrier phase information
	Bin69	69	1 or 0	GLONASS L1/L2 diagnostic information
	Bin76	76	20, 10, 2, 1, 0, or.2	GPS L1/L2 code and carrier phase information
	Bin80	80	1 or 0	SBAS data frame information
	Bin89	89	1 or 0	SBAS satellite tracking information
	Bin93	93	1 or 0	SBAS ephemeris information
	Bin94	94	1 or 0	Ionospheric and UTC conversion parameters
	Bin95	95	1 or 0	GPS ephemeris information
	Bin96	96	20, 10, 2, 1, or 0	GPS L1 code and carrier phase information
	Bin97	97	20, 10, 2, 1, 0, or.2	Processor statistics
	Bin98	98	1 or 0	GPS satellite and almanac information
	Bin99	99	1 or 0	GPS L1 diagnostic information
	Bin100	100	1 or 0	GPS L2 diagnostic information
	Bin209	209	1 or 0	SNR and status for all GNSS tracks
Receiver Response:	$>			
Example:	To output the Bin76 message at a rate of 10 Hz, issue the following command: $JBIN,76,10<CR><LF>			
Additional Information:	Higher update rates may be available on select binary message.			
Related Commands and Messages:				

Topic Last Updated v3.0 / December 30, 2019

JDIFFX,GNSSOUT Command

Command Type:	General Operation and Configuration
Description:	Specify the GNSS systems to be output in the differential or query the current setting
Command Format:	Specify the GNSS systems to be output in the differential: $JDIFFX,GNSSOUT,gnss,x<CR><LF> where:

	'gnss' = GNSS system to be outputin the differential (GPS , GLONASS, BEIDOU, GALILEO) 'x' = NO (do not output specified GNSS system in the differential) or YES (output specified GNSS system in the differential) Query the current setting Query what GNSS systems are output in the differential: $JDIFFX,GNSSOUT<CR><LF Query if a specific GNSS system is output in the differential: $JDIFFX,GNSSOUT,gnss<CR><LF where 'gnss' is the GNSS system
Receiver Response:	Receiver response when specifying the GNSS systems to be output in the differential. $> Receiver response when querying the current setting, see Example section below:
Example:	Specify that GPS is output in correction formats: $JDIFFX,GNSSOUT,GPS,YES<CR><LF> Receiver Response: $> Query what GNSS systems are output in the differential: $JDIFFX,GNSSOUT<CR><LF> Response if just GPS: $>JDIFFX,GNSSOUT,GPS Response if all GPS and GLONASS: $>JDIFFX,GNSSOUT,GPS,GLONASS Query if a specific GNSS system is output in the differential (example uses GLONASS) $JDIFFX,GNSSOUT,GLONASS<CR><LF> Response if GLONASS is output: $>JDIFFX,GNSSOUT,GLONASS,YES Response if GLONASS is not output:

	$>JDIFFX,GNSSOUT,GLONASS,NO
Additional Information:	
Related Commands and Messages:	

Topic Last Updated : v1.07 / February 16, 2017

JDIFFX,GNSSOUT Command

Command Type:	General Operation and Configuration
Description:	Specify the GNSS systems to be output in the differential or query the current setting
Command Format:	Specify the GNSS systems to be output in the differential: $JDIFFX,GNSSOUT,gnss,x<CR><LF> where: ·'gnss' = GNSS system to be outputin the differential (GPS , GLONASS, BEIDOU, GALILEO) ·'x' = NO (do not output specified GNSS system in the differential) or YES (output specified GNSS system in the differential) Query the current setting Query what GNSS systems are output in the differential: $JDIFFX,GNSSOUT<CR><LF Query if a specific GNSS system is output in the differential: $JDIFFX,GNSSOUT,gnss<CR><LF where 'gnss' is the GNSS system
Receiver Response:	Receiver response when specifying the GNSS systems to be output in the differential. $> Receiver response when querying the current setting, see Example section below:
Example:	Specify that GPS is output in correction formats: $JDIFFX,GNSSOUT,GPS,YES<CR><LF> Receiver Response: $> Query what GNSS systems are output in the differential: $JDIFFX,GNSSOUT<CR><LF>

	Response if just GPS: $>JDIFFX,GNSSOUT,GPS Response if all GPS and GLONASS: $>JDIFFX,GNSSOUT,GPS,GLONASS Query if a specific GNSS system is output in the differential (example uses GLONASS) $JDIFFX,GNSSOUT,GLONASS<CR><LF> Response if GLONASS is output: $>JDIFFX,GNSSOUT,GLONASS,YES Response if GLONASS is not output: $>JDIFFX,GNSSOUT,GLONASS,NO
Additional Information:	
Related Commands and Messages:	

Topic Last Updated : v1.07 / February 16, 2017

JBOOT Command

Command Type:	General Operation and Configuration
Description:	Power cycles the receiver.
Command Format:	$JBOOT<CR><LF>
Receiver Response:	The response is similar to the following: $>STARTED,MFA,Ver=5.9 Aa08 If any application other than MFA is the current application and you send the $JBOOT command, the response is similar to the following:$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated v3.0 / December 30, 2019

JCONN Command

Command Type:	General Operation and Configuration
Description:	Create a virtual circuit between two ports to enable communication through the

	receiver to the device on the opposite port.
Command Format:	To connect two ports virtually: $JCONN,P1,P2<CR><LF> where P1 and P2 are a pair of the following: A,B,C,D or PortA,PortB,PortC,PortD
Receiver Response:	The response is similar to the following: $>STARTED,MFA,Ver=5.9 Aa08 If any application other than MFA is the current application and you send the $JBOOT command, the response is similar to the following: $>
Example:	$JCONN,A,B<CR><LF> $JCONN,PortA,PortB<CR><LF> To disconnect virtual connection: $JCONN,X<CR><LF>
Additional Information:	Caution: Hemisphere GNSS receivers with menus, such as an R Series, use JCONN within the menu application. Any settings you make with JCONN on these products may disable the menu functions until power is cycled.
Related Commands and Messages:	

Topic Last Updated v1.06 / March 10, 2015

JDIFF Command

Command Type:	General Operation and Configuration
Description:	Specify or query the differential source of the receiver. Forces the system to use “diff” as the source (see table in Command Format section below).
Command Format:	Specify the differential mode: $JDIFF,diff[,SAVE]<CR><LF> where: ·'diff' (differential source) may be one of the following:

Diff	Description
OTHER	Instruct the receiver to use external corrections input through the opposite port that is communicating
THIS	Instruct the receiver to use external corrections input through the same port that is communicating
PORTA or PORTB or PORTC or PORTD	Instruct the receiver to: •Use external corrections input through the specified port. Allow RTCM2 (DGPS) inputs to receiver.

	<table><tr><td>BEACON</td><td>Instruct the receiver to use RTCM corrections entering Port C at a fixed rate of 9600 baud. This input does not have to be from a beacon receiver, such as SBX. However, this is a common source of corrections.</td></tr><tr><td>WAAS</td><td>Instruct the receiver to use SBAS. This is also the response when running the local dif application as the base.</td></tr><tr><td>RTK</td><td>Response when running the local dif or rover RTK application for the rover.</td></tr><tr><td>LBAND</td><td>Instruct the receiver to turn on theAtlas module and useAtlas. Setting diff to anything other thanAtlas turns off theAtlas module.</td></tr><tr><td>X</td><td>Instruct the receiver to use e-Dif mode</td></tr><tr><td>NONE</td><td>Instruct the receiver to operate in autonomous mode. This turns off the use of SBAS,Atlas, and RTCM2 (DGPS); however, RTK is still allowed.</td></tr></table> ,SAVE' = optional field, saves the differential source into flash memory so that if you reset power the receiver will boot with the new differential source (it may take several seconds to save the differential source to flash memory). Using $JDIFF with SBAS, RTCM2, or Atlas assigns the priority in the MFA. For example, RTCM2is a higher priority if the assigned diff port is PORTA. See MFA for more information. Query the current DIFF setting: $JDIFF<CR><LF>
Receiver Response:	$> Receiver response when querying the differential source: $>JDIFF,SOURCE,TYPE where: •'SOURCE' is the port/source as issued with the JDIFF command •TYPE' is the differential type actually being used •'AUTO' is the response when queried in e-Dif
Example:	Issue the following command to query the receiver: $JDIFF<CR><LF> ...and if the differential source is WAAS, the response is: $>JDIFF,WAAS
Additional Information:	
Related Commands and Messages:	

Topic Last Updated v1.07/ February 16, 2017

JDIFFX,EXCLUDE Command

Command Type:	General Operation and Configuration
Description:	Specify the differential sources to be excluded from operating in a multi-differential application or query the receiver for excluded differential sources
Command Format:	Specify the differential sources to be excluded: $JDIFFX,EXCLUDE[,SBAS] [,ARTK] [,ATLAS] [,RTCM2][,EDIF][,DFX][,CMR] [,RTCM3][,ROX] [,RTCM_23] [,BEIDOU]<CR><LF> Query the current setting: $JDIFFX,EXCLUDE<CR><LF>
Receiver Response:	$> Response to querying the current setting: $JDIFFX,EXCLUDE[,SOURCE1][,SOURCE2]...[,SOURCEn]<CR><LF> where SOURCE1 through SOURCEn represent each excluded source
Example:	Issue the following command to exclude RTCM3: $JDIFFX,EXCLUDE,RTCM3<CR><LF> If you then issue $JDIFFX,EXCLUDE<CR><LF> to query the current setting the response is (if RTCM3 is the only excluded source): $>JDIFFX,EXCLUDE,RTCM3<CR><LF>
Additional Information:	
Related Commands and Messages:	

Topic Last Updated : v1.10 / June 1, 2018

JDIFFX,GNSSOUT Command

Command Type:	General Operation and Configuration
Description:	Specify the GNSS systems to be output in the differential or query the current setting
Command Format:	Specify the GNSS systems to be output in the differential: $JDIFFX,GNSSOUT,gnss,x<CR><LF> where: 'gnss' = GNSS system to be outputin the differential (GPS , GLONASS, BEIDOU, GALILEO) 'x' = NO (do not output specified GNSS system in the differential) or YES (output specified GNSS system in the differential) Query the current setting

	Query what GNSS systems are output in the differential: $JDIFFX,GNSSOUT<CR><LF Query if a specific GNSS system is output in the differential: $JDIFFX,GNSSOUT,gnss<CR><LF where 'gnss' is the GNSS system
Receiver Response:	Receiver response when specifying the GNSS systems to be output in the differential. $> Receiver response when querying the current setting, see Example section below:
Example:	Specify that GPS is output in correction formats: $JDIFFX,GNSSOUT,GPS,YES<CR><LF> Receiver Response: $> Query what GNSS systems are output in the differential: $JDIFFX,GNSSOUT<CR><LF> Response if just GPS: $>JDIFFX,GNSSOUT,GPS Response if all GPS and GLONASS: $>JDIFFX,GNSSOUT,GPS,GLONASS Query if a specific GNSS system is output in the differential (example uses GLONASS) $JDIFFX,GNSSOUT,GLONASS<CR><LF> Response if GLONASS is output: $>JDIFFX,GNSSOUT,GLONASS,YES Response if GLONASS is not output: $>JDIFFX,GNSSOUT,GLONASS,NO
Additional Information:	
Related Commands and Messages:	

Topic Last Updated : v1.07 / February 16, 2017

JDIFFX,INCLUDE Command

Command Type:	General Operation and Configuration
Description:	Specify the differential sources to be allowed to operate in a multi-differential application or query the receiver for included differential sources.
Command Format:	Specify the differential sources to be included: $JDIFFX,INCLUDE[,SBAS] [,ARTK] [,ATLAS] [,RTCM2][,EDIF][,DFX][,CMR] [,RTCM3][,ROX] [,RTCM_23] [,BEIDOU]<CR><LF> Query the current setting $JDIFFX,INCLUDE<CR><LF>
Receiver Response:	Response to issuing command to include differential sources: $> Response to querying the current setting: $JDIFFX,INCLUDE[,SOURCE1][,SOURCE2]...[,SOURCEn]<CR><LF> where SOURCE1 through SOURCEn represent each included source
Example:	Issue the following command to include CMR: $JDIFFX,INCLUDE,CMR<CR><LF> If you then issue $JDIFFX,INCLUDE<CR><LF> to query the current setting the response may be (showing all included sources including CMR): $>JDIFFX,INCLUDE,SBAS,RTCM2,EDIF,DFX,CMR,RTCM3,ROX Additional Information: For example, if an Eclipse II receiver with SBAS,Atlas, and RTK-base in the same application (multi-diff) has no active Atlas subscription: 1. The receiver tries Atlas high precision services and when it is not found, falls back to Atlas DGPS service. 2. The receiver tries Atlas DGPS service and when it is not found, falls back to WAAS. 3. No warnings when subscription has expired – user expects a certain level of accuracy with Atlas services, not SBAS level accuracy. If you do not actively watch the Atlas service end date, you could potentially use SBAS without knowing it. This command limits the differential sources to ensure a certain level of accuracy is retained.
Additional Information:	
Related Commands and Messages:	

Topic Last Updated : : v1.10 / June 1, 2018

JDIFFX,TYPE Command

Command Type:	General Operation and Configuration
Description:	Query the receiver for the differential type
Command Format:	$JDIFFX,TYPE<CR><LF>
Receiver Response:	$>JDIFFX,TYPE,type where 'type' is one of the following differential types: NONE (no differential corrections) CMR DFX EDIF ROX RTCM2 RTCM3 SBAS
Example:	Response if SBAS is the differential type: $>JDIFFX,TYPE,SBAS Response if RTK (ROX) is the differential type: $>JDIFFX,TYPE,ROX
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JDISNAVMODE Command

Command Type:	General Operation and Configuration
Description:	Enable/disable Athena nav mode reporting in BIN1 and BIN3 messages.
Command Format:	$JDISNAVMODE<CR><LF>
Receiver Response:	Response to issuing command to enable/disable detailed nav mode display: $> Response to querying the current setting: $> JDISNAVMODE[,DEFAULT][,PHOENIX]
Example:	
Additional Information:	This setting is automatically saved and can be reset to default by sending $JRESET
Related Commands and Messages:	

Topic Last Updated: v1.08 / June 21, 2017

JEPHOUT, PERIODSEC Command

Command Type:	General Operation and Configuration
Description:	To allow ephemeris messages (95, 65, 35) to go out a rate other than when they change. This also does the same rate for the message 94. This is a global message and applies to all ephemeris messages on all ports.
Command Format:	Enable/disable the command To enable this command $JEPHOUT,1<CR><LF> To disable this command: $JEPHOUT,0<CR><LF> Query the current setting: $JEPHOUT<CR><LF>
Receiver Response:	Response to issuing command to enable/disable command $> Response to querying the current setting If setting is currently enabled the response is: $>JEPHOUT,1 If setting is currently disabled the response is: $>JEPHOUT,0
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JETHERNET,MODE Command

Command Type:	General Operation and Configuration
Description:	On receivers with Ethernet support, this command allows configuring how the receiver connects to a network on the Ethernet interface.
Command Format:	$JETHERNET,MODE,OFF<CR><LF> $JETHERNET,MODE,DHCP<CR><LF> $JETHERNET,MODE,STATIC,IP,SUBNET[,GATEWAY[,DNS]]<CR><LF> Where IP, SUBNET, GATEWAY, and DNS are the ip address, subnet mask, gateway ip, and dns server ip respectively, in the standard decimal notation.

Receiver Response:	$>JETHERNET,MODE,...<CR><LF>
Example:	To disable Ethernet support, one would use the command. $JETHERNET,MODE,OFF<CR><LF> To enable Ethernet support in DHCP (automatic IP address assignment by the network) mode, use the following command: $JETHERNET,MODE,DHCP<CR><LF> To enable Ethernet support with a fixed IP address of 192.168.1.5, use the following command: $JETHERNET,MODE,STATIC,192.168.1.5,255.255.255.0<CR><LF>
Additional Information:	
Related Commands and Messages:	

Topic Last Updated v.1.07 / : February 16, 2017

JETHERNET,PORTI Command

Command Type:	General Operation and Configuration
Description:	This command configures the virtual serial port 'PORTI', which may be accessible via the Ethernet interface. By default, PORTI is disabled, but may be enabled on a specified TCP port using this command. This interface supports acting as either TCP server or TCP client, depending on whether you specify a destination host or not. Messages can be enabled on the port with commands such as $JASC and $JBIN by specifying 'PORTI' as the destination port. Note that PORTI provides full access just as a local serial port would and does not have an authentication mechanism. As such, care should be taken for what networks it is enabled on, especially if behaving as a TCP server.
Command Format:	$JETHERNET,PORTI,OFF<CR><LF> To turn off the PORTI interface. $JETHERNET,PORTI,PORT<CR><LF> Where 'PORT' is replaced with a port number to listen for incoming TCP connections on, behaving as a TCP server. $JETHERNET,PORTI,HOST,PORT<CR><LF> Where 'HOST' and 'PORT' are replaced with the host (IP address or domain name) and port to make an outgoing TCP connection to, behaving as a TCP client. $JETHERNET,PORTI,HOST1,PORT1,HOST2,PORT2<CR><LF> Same as the above, except allowing two host/port pairs. The second one is can be switched to if an outgoing connection to the first fails, or vice versa. Only one connection will be active at a time.
Receiver Response:	$>JETHERNET,PORTI,...<CR><LF>

	Where the response reflects the current configuration.
Example:	To disable the PORTI virtual serial port, one may use the command: $>JETHERNET,PORTI,OFF<CR><LF> To enable PORTI listening on TCP port 5000, one may use the following command: $>JETHERNET,PORTI,5000<CR><LF>
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

$JETHERNET,PORTUDP Command

Command Type:	General Operation and Configuration
Description:	The $JETHERNET,PORTUDP command allows configuring a virtual serial port for transmitting messages via UDP packets. Up to four destination host/port pairs may be specified, and messages will be sent to all of them at once. This is for outgoing data only, and incoming data or commands via UDP are not accepted. Messages can be enabled on the port with commands such as $JASC and $JBIN by specifying 'PORTJ' as the destination port.
Command Format:	$JETHERNET,PORTUDP,OFF<CR><LF> To turn off the PORTJ transmission. $JETHERNET,PORTUDP,HOST,PORT<CR><LF> Where 'HOST' and 'PORT' are replaced with the host (IP address or domain name) and port to transmit UDP messages to. $JETHERNET,PORTUDP,HOST1,PORT1,HOST2,PORT2,…<CR><LF> Up to four hosts/port pairs may be specified.
Receiver Response:	$>JETHERNET,PORTUDP,...<CR><LF> Where the response reflects the current configuration.
Example:	To disable the PORTI virtual serial port, one may use the command: $>JETHERNET,PORTI,OFF<CR><LF> To enable PORTI listening on TCP port 5000, one may use the following command: $>JETHERNET,PORTI,5000<CR><LF>
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

$JETHERNET,NTRIPCLIENT Command

Command Type:	General Operation and Configuration
Description:	The $JETHERNET,NTRIPCLIENT command allows configuring a simple NTRIP client for the receive correction messages from.
Command Format:	$JETHERNET,NTRIPCLIENT,OFF<CR><LF> To turn off the NTRIP client transmission. $JETHERNET,NTRIPCLIENT,HOST,PORT,MOUNTPOINT,USERNAME,PASSWORD<CR><LF> Where 'HOST', 'PORT', 'MOUNTPOINT', 'USERNAME', and 'PASSWORD' are all replaced with the relevant configuration parameters for connecting to the NTRIP caster. The username and password fields can be omitted if the NTRIP caster in question does not require authentication
Receiver Response:	$>JETHERNET,NTRIPCLIENT,...<CR><LF> $>JETHERNET,NTRIPSTATUS,Connecting,0.0KB,0.0 seconds<CR><LF> Where the first line indicates the current configuration, and the second line indicates the status of the NTRIP client connection. The second line will be omitted if the NTRIP client is turned off.
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

$JETHERNET,NTRIPSERVER Command

Command Type:	General Operation and Configuration
Description:	The $JETHERNET,NTRIPSERVER command allows configuring a simple NTRIP server for allowing sending correction messages to an NTRIP caster.
Command Format:	$JETHERNET,NTRIPCLIENT,OFF<CR><LF> To turn off the NTRIP client transmission. $JETHERNET,NTRIPCLIENT,HOST,PORT,MOUNTPOINT,USERNAME,PASSWORD<CR><LF> Where 'HOST', 'PORT', 'MOUNTPOINT', 'USERNAME', and 'PASSWORD' are all replaced with the relevant configuration parameters for connecting to the NTRIP caster. The username and password fields can be omitted if the NTRIP caster in question does not require authentication
Receiver Response:	$>JETHERNET,NTRIPSERVER,...<CR><LF> $>JETHERNET,NTRIPSTATUS,Connecting,0.0KB,0.0 seconds<CR><LF> Where the first line indicates the current configuration, and the second line indicates the status of the NTRIP server connection. The second line will be omitted if the NTRIP server is turned off.
Example:	
Additional Information:	
Related Commands	

and Messages:	

Topic Last Updated: v4.0/June 30, 2020

$JETHERNET,NTRIPSERVER Command

Command Type:	General Operation and Configuration
Description:	The $JETHERNET,WEBUI command enables/disables the WebUI interface over HTTP.
Command Format:	$JETHERNET,WEBUI,ON<CR><LF> To enable the WebUI interface. $JETHERNET,WEBUI,OFF<CR><LF> To disable the WebUI interface. Query the current setting: $JETHERNET,WEBUI<CR><LF>
Receiver Response:	$JETHERNET,WEBUI,...<CR><LF> The response reflects whether the WebUI interface is enabled or disabled.
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v4.0/June 30, 2020

JFLASH,DIR Command

Command Type:	General Operation and Configuration
Description:	Display the files on a USB flash drive You can only display files at the root level of the flash drive (you cannot navigate into subdirectories).
Command Format:	$JFLASH,DIR<CR><LF>
Receiver Response:	$>JFLASH,file1 $>JFLASH,file2 $>JFLASH,file3 $>JFLASH,filen One line appears for each file at the root level of the flash drive.
Example:	If you issue the $JFLASH,DIR command and the root level of the flash drive contains the following files: hemi_1.bin, hemi_2.bin, hemi_3.bin the response is: $>JFLASH,hemi_1.bin $>JFLASH,hemi_2.bin $>JFLASH,hemi_3.bin
Additional Information:	

Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JFLASH,FILE,CLOSE Command

Command Type:	General Operation and Configuration
Description:	Close an open file on a USB flash drive Closing a file does not turn off the messages being written to the flash drive; it just closes the file so you can safely remove the flash drive. Caution: Close the file before removing the flash drive. Failure to do so may corrupt the file.
Command Format:	$JFLASH,FILE,CLOSE<CR><LF>
Receiver Response:	$>JFLASH,CLOSE mass_storage:0:\filename
Example:	If you issue the $JFLASH,FILE,CLOSE command and the 'hemi_4.bin' file on the flash drive is currently open, the response is: $>JFLASH,CLOSE mass_storage:0:\HEMI_4.BIN
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JFLASH,FILE,NAME Command

Command Type:	General Operation and Configuration
Description:	Open a specific file, append to a specific file, or display the file name of the open file on a USB flash drive.
Command Format:	Open a specific file (overwrite or append): $JFLASH,FILE,NAME,filename[,APPEND]<CR><LF> where: •'filename' is the name of the file and it must be a legal 8.3 file name •',APPEND' is an optional field that allows you to append data to the file Warning: Using this command without the ",Append" option overwrites the existing file without warning. Display the name of the open file: $JFLASH,FILE,NAME<CR><LF>
Receiver Response:	Response from issuing command to open an existing file or append to an existing file: $>JFLASH, OPEN mass_storage:0:\filename Response from issuing command to display the name of the open file $>JFLASH, mass_storage:0:\filename

	If you attempt to display the name of the open file and no file is actually open the response is: $>JFLASH, NO FILE OPEN
Example:	If you issue the following command to open file hemi_4.bin on a USB flash drive: $JFLASH,FILE,NAME,hemi_4.bin<CR><LF> the response is: $>JFLASH, mass_storage:0:\HEMI_4.BIN
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JFLASH,FILE,OPEN Command

Command Type:	General Operation and Configuration
Description:	Create and open a file with an automatically generated file name (hemi_1.bin… hemi_99.bin) on a USB flash drive (only 8.3 file format is allowed)
Command Format:	$JFLASH,FILE,OPEN<CR><LF>
Receiver Response:	$>JFLASH,OPEN mass_storage:0:\filename where 'filename' is the name of the new file
Example:	If you issue the $JFLASH,FILE,OPEN command and the root level of the flash drive contains the following files: hemi_1.bin, hemi_2.bin, hemi_3.bin the response is: $>JFLASH,OPEN mass_storage:0:\HEMI_4.bin
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JFLASH,FREESPACE Command

Command Type:	General Operation and Configuration
Description:	Display the free space in kilobytes (KB) on a USB flash drive. You can use a flash drive larger than 4GB; however, this command will not display a number greater than 4GB.
Command Format:	$JFLASH,FREESPACE<CR><LF>
Receiver Response:	$>JFLASH,FREESPACE, numbytes bytes where 'numbytes' is the number of kilobytes
Example:	The following response indicates a USB flash drive with approximately 2GB of free space. $>JFLASH,FREESPACE,2001731584bytes
Additional Information:	

Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JFLASH,NOTIFY,CONNECT Command

Command Type:	General Operation and Configuration
Description:	Enable/disable the automatic response when a USB flash drive is inserted or removed (if port is not specified the response will be sent to the port that issued the command)
Command Format:	$JFLASH,NOTIFY,CONNECT,r[,PORT]<CR><LF> where: •'r' is the message status variable (0 = Off, 1 = On) •',PORT' is an optional field you use to specify the port to which the response will be sent (if you do not specify a port, the response is sent to the port from which you issued the command)
Receiver Response:	Response to issuing command to enable notification: $> Response to inserting a flash drive if notification is enabled: $>JFLASH,CONNECTED Response to removing a flash drive if notification is enabled: $>JFLASH,DISCONNECTED
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JFLASH,QUERYCONNECT Command

Command Type:	General Operation and Configuration
Description:	Manually verify if a USB flash drive is connected or disconnected
Command Format:	$JFLASH,QUERYCONNECT<CR><LF>
Receiver Response:	Response to verifying the connection status of a flash drive if the flash drive is connected: $>JFLASH,CONNECTED $> Response to verifying the connection status of a flash drive if the flash drive is

	disconnected: $>JFLASH,DISCONNECTED $>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JFREQ Command

Command Type:	Atlas
Description:	Tune the Atlas receiver (manually or automatically) or query the receiver for the current setting.
Command Format:	Tune the Atlas receiver To manually tune the receiver: $JFREQ,freq,symb<CR><LF> where: 1.'freq' is the frequency in kHz (reply is in MHz) 2.'symb' is the symbol baud rate (600) Query the current setting: $JFREQ<CR><LF> Example: Correct: $JFREQ,1545915,600 (1,545,915 Hz,) To auto-tune the receiver: $JFREQ,0<CR><LF>
Receiver Response:	Response to issuing command to tune receiver: $> Response to querying the current setting: **$>JLBEAM,Sent sfreq,Used ufreq,Baud baud,Geolon[,AUTO]** where:

<table>
<tr><td></td><td>
<table>
<tr><th>Response Component</th><th>Description</th></tr>
<tr><td>sfreq</td><td>Frequency to which the Atlas receiver is instructed to tune (in this example, 1557.8550 MHz)</td></tr>
<tr><td>ufreq</td><td>Frequency to which the Atlas receiver is tuned</td></tr>
<tr><td>baud</td><td>Baud rate of the signals being received</td></tr>
<tr><td>lon</td><td>Approximate longitude of the geostationary satellite to which the Atlas receiver is tuned</td></tr>
</table>

$>
</td></tr>
<tr><td>Example:</td><td>Manually Tune a Frequency (command and response):

$JFREQ,1545915,600

$>

Auto-Tune a Frequency based on Geographic Location (command and response):

$JFREQ,AUTO

$>

Query a Manually Tuned Receiver (response):

$>JLBEAM,Sent 1557.8350,Used 1557.8350,Baud 1200,Geo -101

Query an Auto-Tuned Receiver (response):

$>JLBEAM,Sent 1557.8550,Used 1557.8550,Baud 1200,Geo -101,AUTO</td></tr>
<tr><td>Additional Information:</td><td>The status of this command is also output when issuing the JSHOW command.

The following table provides frequency information for the Atlas satellites. This information is subject to change. Visit your Atlas service provider’s website for up-to-date satellite constellation and broadcast information.

<table>
<tr><th>Coverage Area</th><th>Frequency</th><th>Baud Rate</th><th>Satellite Name</th></tr>
<tr><td>North and South America</td><td>1545.915 MHz</td><td>600</td><td>AMERICAS</td></tr>
<tr><td>Asia-Pacific</td><td>1545.855 MHz</td><td>600</td><td>APAC</td></tr>
<tr><td>Europe, Middle East and Africa</td><td>1545.905 MHz</td><td>600</td><td>EMEA</td></tr>
<tr><td>Eastern Europe and Middle East</td><td>1545.915 MHz</td><td>600</td><td>MEAS</td></tr>
</table>
</td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v3.0 / December 30, 2019

JGEO Command

<table>
<tr><td>Command Type:</td><td>SBAS</td></tr>
<tr><td>Description:</td><td>Display information related to the current frequency of SBAS or Atlas satellite and its location in relation to the receiver’s antenna</td></tr>
<tr><td>Command Format:</td><td>$JGEO[,ALL]<CR><LF>

where ',ALL' is an optional field that displays information for all SBAS satellites (including those not being used)</td></tr>
<tr><td>Receiver Response:</td><td>$>JGEO,SENT=1575.4200,USED=1575.4200,PRN=prn,LON=lon,EL=ele,AZ=az

where:

<table>
<tr><th>Response Component</th><th>Description</th></tr>
<tr><td>JGEO</td><td>Message header</td></tr>
<tr><td>Sent=1575.4200</td><td>Frequency sent to the digital signal processor</td></tr>
<tr><td>Used=1575.4200</td><td>Frequency currently used by the digital signal processor</td></tr>
<tr><td>PRN=prn</td><td>WAAS satellite PRN number</td></tr>
<tr><td>Lon=-lon</td><td>Longitude of the satellite</td></tr>
<tr><td>El=ele</td><td>Elevation angle from the receiver antenna to the WAAS satellite, reference to the horizon</td></tr>
<tr><td>AZ=az</td><td>Azimuth from the receiver antenna to the WAAS satellite, reference to the horizon</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td>To display information related to the current frequency of SBAS issue the following command:
$JGEO[,ALL]<CR><LF>

The response is then:

$>JGEO,SENT=1575.4200,USED=1575.4200,PRN=122,LON=-54,EL=9.7,AZ=114.0

To display information for dual SBAS satellites issue the following command:

$JGEO[,ALL]<CR><LF>

The response is:

<b>$>JGEO,SENT=1575.4200,USED=1575.4200,PRN=122,LON=-54,EL=9.7,AZ=114.0

$>JGEO,SENT=1575.4200,USED=1575.4200,PRN=134,LON=178,EL=5.0,AZ=252.6

The first line of output is identical to the output from the first JGEO query above; however, the second line of output provides information on the WAAS satellite not being currently used. Both lines of output follow the same format.</td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v3.0 / December 30, 2019

JHTYPE_SHOW Command

Command Type:	
Description:	Queries the hardware type.
Command Format:	
Receiver Response:	
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

JI Command

Command Type:	General Operation and Configuration
Description:	Display receiver information, such as its serial number and firmware version
Command Format:	$JI<CR><LF>
Receiver Response:	$>JI,SN,FLT,HW,PROD,SDATE,EDATE,SW,DSP<CR><LF> where:

Response Component	Description
SN	Serial number of the GPS engine
FLT	Fleet number
HW	Hardware version
PROD	Production date code
SDATE	Subscription begin date
EDATE	Subscription expiration date
SW	Application software version number
DSP	DSP version (only valid for Atlas applications)

Example:	$>JI,19422368,20,1,04062018,01/01/1900,01/01/6455,5.9A,a08,89
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

JK Command

Command Type:	General Operation and Configuration
Description:	Subscribe the receiver to various options, such as higher update rates, Atlas or RTK. or Query for the current subscription expiration date when running Atlas application or the receiver subscription code when running all other applications
Command Format:	Subscribe the receiver to specific options: $JK,x…<CR><LF> where 'x…' is the subscription key provided by Hemisphere GNSS and is 56 characters in length Query the current setting: $JK<CR><LF>
Receiver Response:	Response to issuing command to subscribe: $> Response to querying the current setting: $>JK,DateCode,SubscriptionCode,DowngradeCode where: ·'DateCode' indicates your subscription information (compare last four digits of Date Code to determine your subscription and see the Example section below and the examples in Understanding Additive Codes) ·'SubscriptionCode' is the hex equivalent of the Date Code ·'DowngradeCode' is the output rate in Hertz indicating a downgrade from the default of 10 Hz (if 1, 2 or 5 does not appear the output rate is the default 10 Hz)
Example:	If you query the receiver for the current setting when running A t l a s applications the response will appear similar to the following: $>JK,06/30/2011,0 If you query the receiver for the current setting when running any other application, the response will appear similar to the following (Crescent Vector example response shown). Example shows no downgrade code (using default output rate of 10 Hz). $>JK,01/01/3007,7
Additional Information:	Interpreting the $JK 'Date'/Subscription Codes
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

JK SHOW Command

<table>
<tr><td>Command Type:</td><td>General Operation and Configuration</td></tr>
<tr><td>Description:</td><td>Contains authorization information</td></tr>
<tr><td>Command Format:</td><td>$>JK,SHOW,0,SUBOPT,ENDDATE,0,OPT=,SUBSCRIPTION DESCRIPTION,<CR><LF>

<table>
<tr><th>Response Component</th><th>Description</th></tr>
<tr><td>0</td><td>Unknown</td></tr>
<tr><td>SUBOPT</td><td>Subscription code (see Interpreting the $JK ‘Date’/Subscription Codes to determine the meaning of the subscription code)</td></tr>
<tr><td>END DATE</td><td>The subscription end date</td></tr>
<tr><td>0</td><td>UNKNOWN</td></tr>
<tr><td>Opt=Subscription Description</td><td>
<table>
<tr><td>X HZ</td><td>Maximum data rate</td></tr>
<tr><td>EDIF</td><td>Supports EDIF function</td></tr>
<tr><td>RTK</td><td>Supports RTK function</td></tr>
<tr><td>RAW_DATA</td><td>Supports the RAW data output</td></tr>
<tr><td>L2_L5</td><td>Supports other frequencies bedsides L1</td></tr>
<tr><td>MULTI-GNSS</td><td>Supports other satellite system besides GPS</td></tr>
<tr><td>BEIDOU_B3</td><td>Supports B3 frequencies</td></tr>
<tr><td>ATLAS_Xcm</td><td>Defines Atlas accuracies/subscription</td></tr>
</table>
</td></tr>
</table>
</td></tr>
<tr><td>Receiver Response:</td><td></td></tr>
<tr><td>Example:</td><td>$>JK,SHOW,0,157F,12/31/2016,0,OPT=,20HZ,EDIF,RTK,BASE,RAW_DATA,L2_L5,MULTI_ GNSS,BEIDOUB3,ATLAS_LBAND,ATLAS_30cm</td></tr>
<tr><td>Additional Information:</td><td>Interpreting the $JK 'Date'/Subscription Codes</td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v3.0 / February 3, 2020

JLBEAM Command

Command Type:	L-Band
Description:	Display the information of each spot beam currently in use by the Atlas receiver
Command Format:	$JLBEAM<CR><LF>
Receiver Response:	$>JLBEAM,Sent 1545.9150,Used 1545.9150,Baud 600,Geo -98,AUTO $>JLBEAM,Sent freq,Used freq,Baud xxx,Geo xxx (1) $>JLBEAM,freq1,lon1,lat1,baud1,satlon1 (2). $>JLBEAM,freqn,lonn,latn,baudn,satlonn where:

	<table><tr><th>Response Component</th><th>Description</th></tr><tr><td>"Sent" freq</td><td>Frequency sent to the digital signal processor (DSP)</td></tr><tr><td>"Used" freq</td><td>Frequency currently being used by the digital signal processor (DSP)</td></tr><tr><td>"Baud" xxxx</td><td>Currently used baud rate of the acquired signal</td></tr><tr><td>"Geo" xxx</td><td>Currently used satellites longitude (in degrees)</td></tr></table> The output second line components are described in the following table: <table><tr><th>Response Component</th><th>Description</th></tr><tr><td>freq</td><td>Frequency of the spot beam</td></tr><tr><td>lon</td><td>Longitude of the center of the spot beam (in degrees)</td></tr><tr><td>lat</td><td>Latitude of the center of the spot beam (in degrees)</td></tr><tr><td>baud</td><td>Baud rate at which this spot beam is modulated</td></tr><tr><td>satlon</td><td>Satellites longitude (in degrees)</td></tr></table>
Example:	$>JLBEAM,Sent 1551.4890,Used 1551.4890,Baud 1200,Geo -101 $>JLBEAM,1556.8250,-88,45,1200,(-101) $>JLBEAM,1554.4970,-98,45,1200,(-101) $>JLBEAM,1551.4890,-108,45,1200,(-101) $>JLBEAM,1531.2300,25,50,1200,(16) $>JLBEAM,1535.1375,-75,0,1200,(-98) $>JLBEAM,1535.1375,-165,13,1200,(-98) $>JLBEAM,1535.1525,20,6,1200,(25) $>JLBEAM,1558.5100,135,-30,1200,(160) $>JLBEAM,1535.1375,90,15,1200,(109) $>JLBEAM,1535.1375,179,15,1200,(109)
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v3.0 / December 30, 2019

JLIMIT Command

Command Type:	General Operation and Configuration
Description:	Set the threshold of estimated horizontal performance for which the DGPS position LED is illuminated or query the current setting.
Command Format:	Set the threshold of estimated horizontal performance: $JLIMIT,limit<CR><LF> where 'limit' is the new limit in meters Query the current setting:

	$JLIMIT<CR><LF>
Receiver Response:	Receiver response when setting the threshold of estimated horizontal performance $> $>JLIM,RESID,LIMIT where 'LIMIT' is the limit in meters Example: To set the threshold to 5 m issue the following command: $JLIMIT,5<CR><LF> If you then query the receiver with $JLIMIT<CR><LF> the response is: $JLIM,RESID,5.00
Example:	
Additional Information:	The default value for this parameter is a conservative 10.00 m. The status of this command is also output in the JSHOW message.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JLXBEAM Command

Command Type:	L-Band
Description:	Display spot beam debug information.
Command Format:	$JLXBEAM<CR><LF>
Receiver Response:	$>JLBEAMEX $> Beam:1,DDSfreq1,symbol1,lon1,lat1,lonrad1,latrad1,beamrot1,satlon1,* $> Beam:2,DDSfreq2,symbol2,lon2,lat2,lonrad2,latrad2,beamrot2,satlon2,* $> Beam:n,DDSfreqn,symboln,lonn,latn,lonradn,latradn,beamrotn,satlonn,* where:

Response Component	Description
DDSfreq	DDS frequency
symbol	Symbol rate used for that particular spot beam
lon	Longitude of the spot beam centroid
lat	Latitude of the spot beam centroid
lonrad	Longitude radius of the spot beam

	latrad	Latitude radius of the spot beam
	beamrot	Rotation angle of the spot beam
	satlon	Longitude of the Atlas satellite
		Reserved

Example:	$>JLBEAMEX $> Beam:22,1535125000,600,-26,40,2,41,0,9999,* $> Beam:21,1535157500,600,65,30,31,18,-21,64,* $>Beam:13,1535185000,1200,136,-25,23,28,-40,144,* $>Beam:13,1535185000,1200,172,-40,13,26,-26,144,* $>Beam:24,1557835000,1200,-100,49,6,28,0,-101,* $>Beam:24,1557835000,1200,-101,66,12,6,0,-101,* $>Beam:25,1557845000,1200,-74,52,12,30,-30,-101,* $>Beam:26,1557855000,1200,-122,45,11,30,25,-101,* $>Beam:8,1535137500,1200,-85,2,30,20,-5,-98,* $>Beam:8,1535137500,1200,-60,-25,34,36,-20,-98,* $>Beam:4,1535137500,1200,109,2,14,19,-27,109,* $>Beam:4,1535137500,1200,140,38,27,51,-56,109,* $>Beam:7,1537440000,1200,23,-2,29,49,50,25,* $>Beam:7,1537440000,1200,14,59,41,23,34,25,* $>Beam:7,1537440000,1200,11,28,17,24,0,25,*
Additional Information:	The default value for this parameter is a conservative 10.00 m. The status of this command is also output in the JSHOW message.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JMASK Command

Command Type:	GPS
Description:	Specify the elevation cutoff mask angle for the GPS engine Any satellites below this mask angle will be ignored even if available. The default angle is 5° because satellites available below this angle will have significant tropospheric refraction errors.
Command Format:	$JMASK,e<CR><LF> where the elevation mask cutoff angle 'e' may be a value from 0 to 60°
Receiver Response:	$>
Example:	To specify the elevation cutoff mask angle to 10° issue the following command: $JMASK,10<CR><LF>
Additional Information:	To query the receiver for the current setting, issue the JSHOW command.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JMODE Command

Command Type:	General Operation and Configuration
Description:	Query receiver for status of JMODE settings
Command Format:	$JMODE<CR><LF>
Receiver Response:	$>JMODES[,BASE][,FIXLOC][,FOREST][,GLOFIX][,GPSONLY][,L1ONLY][,MIXED][,NULLNMEA][,CMRPLUS]
Example:	If FOREST and TUNNEL are set to ON and all others (MIXED, NULLNMEA,SBASR, and TIMEKEEP) are set to OFF and you issue $JMODES,TUNNEL,FOREST If all features are set to OFF and you issue the JMODE command the receiver response will be: $JMODES
Additional Information:	The status of this command is also output in the JSHOW response. For example, if TUNNEL is set to ON and all other JMODE option: $>JSHOW,MODES,TUNNEL.
Related Commands and Messages:	

Topic Last Updated: v4.0 / June 30, 2020

JMODE,BASE Command

Command Type:	General Operation and Configuration, Local Differential and RTK Commands
Description:	Enable/disable base mode functionality or query the current setting: •If base mode is NO (disabled) and the receiver is receiving RTK corrections, these corrections are echoed out when RTK corrections (ROX, RTCM3, CMR) are requested •If base mode is YES (enabled), the receiver computes its own corrections, regardless of whether or not it is receiving RTK corrections from another source
Command Format:	Enable/disable base mode To enable base mode: $JMODE,BASE,YES<CR><LF> To disable base mode: $JMODE,BASE,NO<CR><LF> Query the current setting: $JMODE,BASE<CR><LF>
Receiver Response:	Response to issuing command to enable/disable base mode: $>

	Response to querying the current setting: If base mode is currently enabled the response is: $>JMODE,BASE,YES If base mode is currently disabled the response is: $>JMODE,BASE,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMODE,BDSOFF Command

Command Type:	General Operation and Configuration
Description:	Set the receiver to use BDS data in the solution
Command Format:	Close/Open BDS operation Close BDS operation: $JMODE,BDSOFF,YES<CR><LF> Open BDS operation: $JMODE,BDSOFF,NO<CR><LF>
Receiver Response:	Response to issuing command to turn enable/disable BDS operation: $> Response to querying the current setting If BDS operation is currently enabled the response is: $>JMODE,BDSOFF,YES If BDS operation is currently disabled the response is: $>JMODE,BDSOFF,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

$JMODE,CMRPLUS Command

Command Type:	General Operation and Configuration
Description:	Set the receiver to output CMR+ instead of CMR
Command Format:	Output CMR+ instead of CMR $JMODE,CMRPLUS,YES<CR><LF> Output CMR instead of CMR+: $JMODE,CMRPLUS,NO<CR><LF>
Receiver Response:	Response to issuing command to turn enable/disable CMR+ operation: $> Response to querying the current setting If CMR+ is currently enabled the response is: $>JMODE,CMRPLUS,YES If CMR is currently disabled the response is: $>JMODE,CMRPLUS,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v4.0/June 30, 2020

JMODE,FIXLOC Command

Command Type:	General Operation and Configuration
Description:	Set the receiver to not re-average (or re-average) its position or query the current setting. $JMODE,FIXLOC,YES assure that the BASE will not re-average its position. Good for permanent installations.
Command Format:	Enable/disable position re-averaging To set receiver to not re-average its position: $JMODE,FIXLOC,YES<CR><LF> To set receiver to re-average its position: $JMODE,FIXLOC,NO<CR><LF> Query the current setting:

	$JMODE,FIXLOC<CR><LF>
Receiver Response:	Response to issuing command to enable/disable position re-averaging: $> Response to querying the current setting: If setting is currently enabled (no position re-averaging) the response is: $>JMODE,FIXLOC,YES If setting is currently disabled (position re-averaging enabled) the response is: $>JMODE,FIXLOC,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMODE,MIXED Command

Command Type:	General Operation and Configuration
Description:	Include satellites that do not have DGPS or SBAS corrections in the solution or query the current setting This command is useful if you are trying to maximize the likelihood of calculating a position but are willing to sacrifice accuracy. See also JMODE,FOREST.
Command Format:	To include/exclude satellites without DGPS or SBAS corrections To include satellites without DGPS or SBAS corrections: $JMODE,MIXED,YES<CR><LF> To exclude satellites without DGPS or SBAS corrections: $JMODE,MIXED,NO<CR><LF> Query the current settin g: $JMODE,MIXED<CR><LF>
Receiver Response:	Response to issuing command to include/exclude satellites without DGPS or SBAS corrections $> Response to querying the current setting: If satellites without differential corrections are currently included the response is: $>JMODE,MIXED,YES If satellites without differential corrections are currently excluded the response is:

	$>JMODE,MIXED,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMODE,NULLNMEA Command

Command Type:	General Operation and Configuration
Description:	Enable/disable output of NULL fields in NMEA 0183 messages when no there is no fix (when position is lost) or query the current setting This only applies to position portion of the messages; it does not affect the time portion of the message. If this setting is disabled and position is lost then the positioning parameters of the message from the most recent known position are repeated (instead of being NULL if enabled).
Command Format:	Enable/disable output of NULL fields in NMEA 0183 messages To enable output: $JMODE,NULLNMEA,YES<CR><LF> To disable output: $JMODE,NULLNMEA,NO<CR><LF> Query the current setting: $JMODE,NULLNMEA<CR><LF>
Receiver Response:	Response to issuing command to enable/disable output of NULL fields in NMEA 0183 messages $> Response to querying the current setting If setting is currently enabled the response is: $>JMODE,NULLNMEA,YES If setting is currently disabled the response is: $>JMODE,NULLNMEA,NO
Example:	If the most recent GPGGA message is as follows: $GPGGA,220715.00,3333.4254353,N,11153.3506065,W,2,10,1.0,406.614,M,-26.294,M,6.0,1001*70 ...and then position is lost and JMODE,NULLNMEA is set to NO the GPGGA message repeats as follows (most recent known values do not change):

	$GPGGA,220715.00,3333.4254353,N,11153.3506065,W,2,10,1.0,406.614,M,-26.294,M,6.0,1001*70 For the same message, if position is lost and JMODE,NULLNMEA is set to YES the GPGGA message repeats as follows (position parameters are NULL): $GPGGA,220716.00,,,,,0,,,,M,,M,,*48
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.03 / January 11, 2012

JMODE,SBASNORTK Command

Command Type:	General Operation and Configuration
Description:	Disable/enable the use of SBAS ranging signals (carrier phase)in RTK
Command Format:	Disable/enable use of SBAS ranging signals in RTK To disable use of SBAS ranging signals in RTK: $JMODE,SBASNORTK,YES<CR><LF> To enable use of SBAS ranging signals in RTK: $JMODE,SBASNORTK,NO<CR><LF> Query the current setting: $JMODE,SBASNORTK<CR><LF>
Receiver Response:	Response to issuing command to disable/enable the use of SBAS ranging signals in RTK. $> Response to querying the current setting If current setting is to disable SBAS ranging the response is: $>JMODE,SBASNORTK,YES If current setting is to enable SBAS ranging the response is: $>JMODE,SBASNORTK,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMODE,SBASR Command

Command Type:	General Operation and Configuration
Description:	Enable/disable SBAS ranging or query the current setting
Command Format:	Enable/disable SBAS ranging To enable SBAS ranging: $JMODE,SBASR,YES<CR><LF> To disable SBAS ranging: $JMODE,SBASR,NO<CR><LF> Query the current setting: $JMODE,SBASR<CR><LF>
Receiver Response:	Response to issuing command to enable/disable SBAS ranging $> Response to querying the current setting: If setting is currently enabled the response is: $>JMODE,SBASR,YES If setting is currently disabled the response is: $>JMODE,SBASR,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMODE,STRICTRTK Command

Command Type:	General Operation and Configuration
Description:	Use this command to invoke stricter checks on whether RTK fix is declared. Forces float of RTK at 30 seconds of Age-of- Diff
Command Format:	Enable/disable STRICTRTK functionality To enable STRICTRTK functionality: $JMODE,STRICTRTK,YES<CR><LF> To disable STRICTRTK functionality: $JMODE,STRICTRTK,NO<CR><LF> Query the current setting:

	$JMODE,STRICTRTK<CR><LF>
Receiver Response:	$> Response to issuing command to enable/disable command Response to querying the current setting: If setting is currently enabled the response is: $>JMODE,STRICTRTK,YES If setting is currently disabled the response is: $>JMODE,STRICTRTK,NO
Example:	
Additional Information:	This mode is not saved between power cycles.
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMODE,SURETRACK Command

Command Type:	General Operation and Configuration
Description:	Enable/disable SureTrack functionality (default is enabled) or query the current setting
Command Format:	Enable/disable SureTrack functionality To enable SureTrack functionality: $JMODE,SURETRACK,YES<CR><LF> To disable SureTrack functionality: $JMODE,SURETRACK,NO<CR><LF> Query the current setting: $JMODE,SURETRACK<CR><LF>
Receiver Response:	Response to issuing command to enable/disable command: $> Response to querying the current setting: If setting is currently enabled the response is: $>JMODE,SURETRACK,YES If setting is currently disabled the response is: $>JMODE,SURETRACK,NO $>JMODE,STRICTRTK,NO
Example:	
Additional	This mode is not saved between power cycles.

Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMODE,SURVEY Command

Command Type:	General Operation and Configuration
Description:	Assure RTK fix is not declared when residual errors exceed 10 cm. Also forces use of GLONASS and prevents SureTrack operation.
Command Format:	Enable/disable continuous time updating To enable this command $JMODE,SURVEY,YES<CR><LF> To disable this command: $JMODE,SURVEY,NO<CR><LF> Query the current setting: $JMODE,SURVEY<CR><LF>
Receiver Response:	Response to issuing command to enable/disable command $> Response to querying the current setting If setting is currently enabled the response is: $>JMODE,SURVEY,YES If setting is currently disabled, the response is: $>JMODE,SURVEY,NO
Example:	
Additional Information:	This mode is not saved between power cycles.
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JMODE,TIMEKEEP Command

Command Type:	General Operation and Configuration
Description:	Enable/disable continuous time updating in NMEA 0183 messages when there is no fix (when position is lost) or query the current setting. When position is lost the time is the only parameter in the message that continues to update; all other parameters remain the same.
Command Format:	Enable/disable continuous time updating To enable continuous time updating: $JMODE,TIMEKEEP,YES<CR><LF>

	To disable continuous time updating: $JMODE,TIMEKEEP,NO<CR><LF> Query the current setting: $JMODE,TIMEKEEP<CR><LF>
Receiver Response:	Response to issuing command to enable/disable continuous time updating $> Response to querying the current setting If setting is currently enabled the response is: $>JMODE,TIMEKEEP,YES If setting is currently disabled, the response is: $>JMODE,TIMEKEEP,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JMSG99 Command

Command Type:	Vector
Description:	Change the output in the Bin99 message to be from the specified antenna
Command Format:	$JMSG99,0 where '0' is used view the primary antenna SNR (default) $JMSG99,1 where '1' is used view the secondary antenna SNR$JMODE,TIMEKEEP<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JNMEA,BDSOFFSET Command

Command Type:	BeiDou
Description:	Used to add an offset to the satellite id number for BeiDou GSV message
Command Format:	$JNMEA.BDSOFFSET,x X=’10, 100, 200
Receiver Response:	$>
Example:	The GBGSV message in its default form with no offset: $GBGSV,2,1,08,21,09,249,,26,30,313,37,29,67,072,53,30,17,040,48,1*7E $GBGSV,2,2,08,35,48,196,53,36,32,109,48,42,04,297,,45,74,032,53,1*77

	The GBGSV message with $JNMEA,BDSOFFSET,100. $GBGSV,2,1,08,121,09,249,,126,30,313,37,129,67,072,54,130,17,040,48,1*79 $GBGSV,2,2,08,135,48,196,53,136,32,109,48,142,04,297,,145,74,032,53,1*77
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v4.0/June 30, 2020

JNMEA,GGAALLGNSS Command

Command Type:	GLONASS
Description:	Configure the GGA string to include full GNSS information (the number of used GNSS satellites will be included in the GPGGA message) or query the current setting The GGA message is only supposed to report position and satellite information based on the GPS constellation. The combined constellation position and satellite data should be reported in the GNSS message, but some users with older equipment cannot utilize this message. This command allows users with older equipment that require a GGA message to be able to utilize and take advantage of the larger constellation of GNSS satellites.
Command Format:	Include/exclude full GNSS information in GGA string To include full GNSS information in GGA string: $JNMEA,GGAALLGNSS,YES<CR><LF> To exclude full GNSS information from GGA string: $JNMEA,GGAALLGNSS,NO<CR><LF> Query the current setting: $JNMEA,GGAALLGNSS<CR><LF>
Receiver Response:	Include/exclude full GNSS information in GGA string: $> Query the current setting: If set to yes, querying the current setting returns the following: >JNMEA,GGAALLGNSS,YES If set to no, querying the current setting returns the following: $>JNMEA,GGAALLGNSS,NO
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.07 February 16, 2017

$JNMEA,LIMITID,YES Command

Command Type:	General operation and configuration

Description:	Option to limit the range of the station ID field to be 0-1023 in GGA messages. Masks the base station ID by 0x3FF.
Command Format:	To enable the station ID limit: $JNMEA,LIMITID,YES<CR><LF> To disable the station ID limit: $JNMEA,LIMITID,NO<CR><LF> Query the current setting: $JNMEA,LIMITID<CR><LF>
Receiver Response:	Response to issuing command to enable/disable station ID limit: $> Response to querying the current setting: $>JNMEA,LIMITID,[YES/NO]
Example:	
Additional Information:	By default, the station ID is not limited. Atlas uses a Base ID of 4715, 4716, or 4717. This command will change the Base ID reported in the GGA message to 619, 620 or 621. This only affects the GGA message.
Related Commands and Messages:	

Topic Last Updated: v.4.0 / June 30, 2020

$JNMEA,PADHDG Command

Command Type:	General operation and configuration
Description:	Enable or disable zero-padding of digits before decimal for certain heading output messages. When enabled, this ensures that 3 digits will be printed before the decimal. Applies to the following messages: • HDT • HDG • HDM • HPR • THS • PASHR
Command Format:	To enable padding for heading: $JNMEA,PADHDG,YES<CR><LF> To disable padding for heading: $JNMEA,PADHDG,NO<CR><LF> Query the current setting:

	$JNMEA,PADHDG<CR><LF>
Receiver Response:	Response to issuing command to enable/disable padding: $> Response to querying the current setting: $>JNMEA,PADHDG,[YES/NO]
Example:	When padding is disabled, the HEHDT output message will display a heading of 22.5 degrees as: $HEHDT,22.5,T*CC When padding is enabled, the same heading would display as: $HEHDT,022.5,T*CC
Additional Information:	Padding is disabled by default
Related Commands and Messages:	

Topic Last Updated: v.3.0 / December 30, 2019

JNMEA,PRECISION Command

Command Type:	GPS, Local Differential and RTK, L-Band
Description:	Specify or query the number of decimal places to output in the GPGGA, GPGLL, and GPGNS messages or query the current setting
Command Format:	Specify the number of decimal places $JNMEA,PRECISION,x<CR><LF> where 'x' specifies the number of decimal places from 1 to 8 Query the current setting: $JNMEA,PRECISION<CR><LF>
Receiver Response:	Specify the precision: $> Query the current setting : $>JNMEA,PRECISION,x where 'x' refers to the number of decimal places to output
Example:	
Additional Information:	When using RTK or Atlas high precision services, Hemisphere GNSS recommends you set JNMEA,PRECISION to at least 7 decimal places. High accuracy positioning techniques require at least 7 decimal places to maintain millimeter (mm) accuracy. This command is the same as JNP.
Related Commands	

and Messages:	

Topic Last Updated: v1.07 / February 16, 2017

JNP Command

Command Type:	GPS, Local Differential and RTK, L-Band
Description:	Specify or query the number of decimal places to output in the GPGGA, GPGLL, and GPGNS messages or query the current setting
Command Format:	Specify the number of decimal places $JNP,x<CR><LF> where 'x' specifies the number of decimal places from 1 to 8 Query the current setting: $JNP<CR><LF>>
Receiver Response:	Specify the number of decimal places to output : $> Query the current setting: $>JNP,x where 'x' refers to the number of decimal places to output
Example:	
Additional Information:	When using RTK or Atlas high precision services, Hemisphere GNSS recommends you set JNP to at least 7 decimal places. High accuracy positioning techniques require at least 7 decimal places to maintain millimeter (mm) accuracy. This command is the same as JNMEA,PRECISION.
Related Commands and Messages:	

Topic Last Updated: v1.07 / February 16, 2017

JOFF Commands

Command Type:	GPS
Description:	Turn off all data messages being output through the current port or other port (or Port C), including any binary messages such as Bin95 and Bin96
Command Format:	$JOFF[,OTHER]<CR><LF> When you specify the ',OTHER' data field (without the brackets), this command turns off all messages on the other port. There are no variable data fields for this message. You can issue this command as follows to turn off all messages on Port C: $JOFF,PORTC<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	Turn off all data messages being output through all ports, including any binary

	messages such as Bin95 and Bin 96, see JOFF,ALL command
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JOFF,ALL Command

Command Type:	GPS
Description:	Turn off all data messages being output through all ports, including any binary messages such as Bin95 and Bin96
Command Format:	$JOFF,ALL<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	To turn off all data messages being output through a single port, including any binary messages such as Bin95 and Bin96, see the JOFF command
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JOFF Command

Command Type:	GPS
Description:	Turn off all data messages being output through the current port or other port (or Port C), including any binary messages such as Bin95 and Bin96
Command Format:	$JOFF[,OTHER]<CR><LF> When you specify the ',OTHER' data field (without the brackets), this command turns off all messages on the other port. There are no variable data fields for this message. You can issue this command as follows to turn off all messages on Port C: $JOFF,PORTC<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	To turn off all data messages being output through all ports, including any binary messages such as Bin95 and Bin96, see the JOFF,ALL command
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JPOS Command

Command Type:	General Operation and Configuration
Description:	Speed up the initial acquisition when changing continents with the receiver or query the receiver for the current position of the receiver (for example, powering

	up the receiver for the first time in Europe after it has been tested in Canada) The command enables the receiver to begin the acquisition process for the closest SBAS spot beams. This saves some time with acquisition of the SBAS service. However, use of this message is typically not required because of the quick overall startup time of the receiver module.
Command Format:	Specify the latitude and longitude $JPOS,lat,lon<CR><LF> where both 'lat' and 'lon': •Must be entered in decimal degrees •Do not need to be more accurate than half a degree Query the current setting $JPOS<CR><LF>
Receiver Response:	Receiver response when specifying the latitude and longitude $> Receiver response when querying the current setting $>JPOS,LAT,LON
Example:	
Additional Information:	The status of this command is also output in the JSHOW message.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JPPS, WIDTH Command

Command Type:	General Operation and Configuration
Description:	Specify the pps width of the receiver or query the current setting
Command Format:	Set the receiver's specific pps width (microseconds) $JPPS,WIDTH,r,SAVE<CR><LF> where: 'r' = specific pps width. The SAVE field is optional. However, if omitted this setting will not survive a power cycle. This setting is not saved with $JSAVE. It must be saved by adding the SAVE field. Query the current setting: Response to issuing command: $PPS,WIDTH<CR><LF>Query the current setting $JPOS<CR><LF>
Receiver Response:	Issue the following command to set the pps width to 2.000 on the current port: $JPPS,WIDTH,2<CR><LF> ...the response is then: $>

	If you query the current setting now, the response is: $JPPS,WIDTH,2.000<CR><LF>
Example:	
Additional Information:	This mode is not saved between power cycles
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JPPS, FREQ Command

Command Type:	General Operation and Configuration
Description:	Specify the pps frequency of the receiver or query the current setting.
Command Format:	Set the receiver's specific pps frequency (in Hz) $JPPS,FREQ,r,SAVE<CR><LF> where: 'r' = specific pps frequency The SAVE field is optional. However, if omitted this setting will not survive a power cycle. This setting is not saved with $JSAVE. It must be saved by adding the SAVE field. Query the current setting Response to issuing command: $PPS,FREQ<CR><LF>
Receiver Response:	$> Response to querying the current setting: $JPPS,FREQ,1.00<CR><LF>
Example:	Issue the following command to set the pps frequency to 2.000 on the current port: $JPPS,FREQ,2<CR><LF> ...the response is then: $> If you query the current setting now, the response is: $JPPS,FREQ,2.00<CR><LF>
Additional Information:	This mode is not saved between power cycles
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JPPS, PERIOD Command

Command Type:	General Operation and Configuration
Description:	Specify the pps frequency of the receiver or query the current setting.
Command Format:	Set the receiver's specific pps period $JPPS,PERIOD,r<CR><LF> where: 'r' = specific pps period (inverse of frequency) The SAVE field is optional. However, if omitted this setting will not survive a power cycle. This setting is not saved with $JSAVE. It must be saved by adding the SAVE field. Query the current setting Response to issuing command: $> $PPS,PERIOD<CR><LF>
Receiver Response:	Response to querying the current setting: $JPPS,PERIOD,1.0<CR><LF>
Example:	Issue the following command to set the pps period to 2 seconds (0.5 Hz) $JPPS,PERIOD,2<CR><LF> ...the response is then: $> If you query the current setting now, the response is: $JPPS,PERIOD,2.000<CR><LF>
Additional Information:	This mode is not saved between power cycles
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JPRN,EXCLUDE Command

Command Type:	General Operation and Configuration
Description:	Note: For advanced users only. Not required for typical operation. Exclude GPS and/or other GNSS satellites from being used in the positioning solution or query the current setting
Command Format:	Exclude PRNs from being used in the positioning solution Exclude GPS and/or other GNSS PRNs: $JPRN,EXCLUDE[,GPS,x,x,x…][,GLO,y,y,y…][,GAL,z,z,z…]<CR><LF>

	where: 'x,x,x...' represents the GPS PRNs you want to exclude 'y,y,y...' represents the GLONASS PRNs you want to exclude 'z,z,z…' represents the GALILEO PRNs you want to exclude Exclude no GNSS PRNs: $JPRN,EXCLUDE,NONE<CR><LF> Exclude no GPS PRNs $JPRN,EXCLUDE,GPS,NONE<CR><LF> Exclude no GLONASS PRNs: $JPRN,EXCLUDE,GLO,NONE<CR><LF> Exclude no GALILEO PRNs: $JPRN,EXCLUDE,GAL,NONE<CR><LF> Query the current setting Query all excluded PRNs (GPS and GLONASS): $JPRN,EXCLUDE<CR><LF> Query excluded GPS PRNs: $JPRN,EXCLUDE,GPS<CR><LF> Query excluded GLONASS PRNs: $JPRN,EXCLUDE,GLO<CR><LF> Query excluded GALILEO PRNs: $JPRN,EXCLUDE,GAL<CR><LF>
Receiver Response:	See Example section below
Example:	If you excluded no GPS or GLONASS PRNS and issued the **$JPRN,EXCLUDE,GPS<CR><LF>** command the response is: $>JPRN,EXCLUDE,GPS,NONE,GLO,NONE If you excluded one GPS PRN (22) and one GLONASS PRN (10) and issued the following commands, you would see the following corresponding responses: Command: $JPRN,EXCLUDE,GPS<CR><LF> Response: $>JPRN,EXCLUDE,GPS,22 Command: $JPRN,EXCLUDE,GLO<CR><LF> Response: $>JPRN,EXCLUDE,GLO,10 Command: $JPRN,EXCLUDE<CR><LF> Response: $>JPRN,EXCLUDE,GPS,22,GLO,10
Additional Information:	
Related Commands and Messages:	

$JPRN,IN/EXCLUDE,BDSPHASE3 Command

Command Type:	General Operation and Configuration
Description:	Enable or disable tracking of all Beidou Phase-3 satellites
Command Format:	To include Beidou Phase-3: $JPRN,INCLUDE,BDSPHASE3<CR><LF> To exclude Beidou Phase-3: $JPRN,EXCLUDE,BDSPHASE3<CR><LF> Query the current setting: $JPRN,BDSPHASE3<CR><LF>
Receiver Response:	Receiver response when in/excluding Beidou Phase-3: $> Receiver response when querying the current setting: $>JPRN,[INCLUDE/EXCLUDE],BDSPHASE3
Example:	
Additional Information:	This setting is automatically saved to non-volatile memory
Related Commands and Messages:	

Topic Last Updated: v.3.0 / December 30, 2019

JQUERY,GUIDE Command

Command Type:	General Operation and Configuration
Description:	Query the receiver for its determination on whether or not it is providing suitable accuracy after both the SBAS, and GPS have been acquired (up to five minutes) This feature takes into consideration the download status of the SBAS ionospheric map and also the carrier phase smoothing of the unit.
Command Format:	$JQUERY,GUIDE<CR><LF>
ceiver Response:	If the receiver is ready for use with navigation, or positioning with optimum performance, it returns: $>JQUERY,GUIDE,YES<CR><LF> Otherwise, it returns: $>JQUERY,GUIDE,NO<CR><LF>
Example:	
Additional Information:	

Related Commands and Messages:	

Topic Last Updated: v1.00 / August 11, 2010

JQUERY,RTKPROG Command

Command Type:	Local Differential and RTK
Description:	Perform a one-time query of RTK fix progress information
Command Format:	$JQUERY,RTKPROG<CR><LF> As an alternative you can log this as a message using the JASC,PSAT,RTKPROG command.
Receiver Response:	**$>JQUERY,RTKPROG,R,F,N,SS1,SS2,SS3,MASK*CC<CR><LF>** where:

Message Component	Description
R	1 = Ready to enter RTK ambiguity fix 0 = Not ready to enter RTK ambiguity fix
F	1 = Receiver running in RTK ambiguity fix mode 0 = Receiver not running in RTK ambiguity fix mode
N	Number of satellites used to fix
SS1	summer-1 SS1 must be significantly larger than SS2 and SS3 to enter R=1 mode
SS2	summer-2
SS3	summer-3
MASK	Bit mask; bits identify which GNSS observables are being received from base recently (1 = GPS, 3 = GPS + GLONASS)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Example:	$>JQUERY,RTKPROG,1,1,23,243.3,0.0,0.0,3
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JQUERY,RTKSTAT Command

<table>
<tr><td>Command Type:</td><td>Local Differential and RTK</td></tr>
<tr><td>Description:</td><td>Perform a one-time query of the most relevant parameters affecting RTK</td></tr>
<tr><td>Command Format:</td><td>$JQUERY,RTKSTAT<CR><LF>

As an alternative you can log this as a message using the JASC,PSAT,RTKSTAT command.</td></tr>
<tr><td>Receiver Response:</td><td>$>JQUERY,RTKSTAT,MODE,TYP,AGE,SUBOPT,DIST,SYS,NUM,SNR,RSF,BSF,HAG, ACCSTAT,SNT

Where:
<table>
<tr><td>Message Component</td><td>Description</td></tr>
<tr><td>MODE</td><td>Mode (FIX,FLT,DIF,AUT,NO)</td></tr>
<tr><td>TYP</td><td>Correction type (DFX,ROX,CMR,RTCM3,CMR+,...)</td></tr>
<tr><td>AGE</td><td>Age of differential corrections, in seconds</td></tr>
<tr><td>SUBOPT</td><td>Subscription code (see Interpreting the $JK 'Date'/Subscription Codes to determines the meaning of the subscription code)</td></tr>
<tr><td>DIST</td><td>Distance to base in kilometers</td></tr>
<tr><td>SYS</td><td>Systems in use:
GPS: L1, L2, L5
GLONASS: G1, G2
Galileo: E5a, E5b, E5a+b, E6</td></tr>
<tr><td>NUM</td><td>Number of satellites used by each system</td></tr>
<tr><td>SNR</td><td>Quality of each SNR path, where:
A is > 20 dB
B is > 18 dB
C is > 15 dB
D is <= 15 dB</td></tr>
<tr><td>RSF</td><td>Rover slip flag (non-zero if parity errors in last 5 minutes, good for detecting jamming and TCXO issues)</td></tr>
<tr><td>BSF</td><td>Base slip flag</td></tr>
<tr><td>HAE</td><td>Horizontal accuracy estimation</td></tr>
<tr><td>ACCSTAT</td><td>RTK accuracy status (hex), where:
0x1 = no differential or differential too old, for the application
0x2 = problems with differential message
0x4 = horizontal position estimate poor for the application
0x8 = HDOP high, poor satellite geometry
0x10 = fewer than 6 L1 sats used
0x20 = poor L1 SNRs
0x40 = not in RTK mode
0x80 = not in RTK mode or RTK only recently solved (< 10secs ago)
0x100 = RTK solution compromised, may fail
The status message can be any of the above or any combination of the above. For example, a status message of '047' indicates the following:
0x1 = no differential or differential too old, for the application
0x2 = problems with differential message</td></tr>
</table></td></tr>
</table>

		• 0x4 = horizontal position estimate poor for the application • 0x40 = not in RTK mode
	SNT	Ionospheric scintillation, values are: • 0 (little or no scintillation - does not adversely affect RTK solution) • 1-100 (scintillation detected - adversely affects RTK solution)
	<CR>	Carriage return
	<LF>	Line feed
Example:	$>JQUERY,RTKSTAT,FIX,ROX, 1,007F,0.0,(,L1,L2,G1,G2,)(,14,11,9,9,)(,A,A,A,A,),0,1,0.008,000,3	
Additional Information:		
Related Commands and Messages:	JASC,PSAT,RTKSTAT command PSAT,RTKSTAT message	

Topic Last Updated: v1.05 / January 18, 2013

JQUERY,TEMPERATURE Command

Command Type:	General Operation and Configuration
Description:	Query the receiver's temperature
Command Format:	$JQUERY,TEMPERATURE<CR><LF>
Receiver Response:	$>JQUERY,TEMPERATURE,51.88
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

RAD,1 Command

Command Type:	e-Dif, DGPS Base Station
Description:	Display the current reference position in e-Dif applications only
Command Format:	$JRAD,1<CR><LF>
Receiver Response:	$>JRAD,1,LAT,LON,HEIGHT where: Command Component: LAT — Description: Latitude of the reference point in decimal degrees

	<table><tr><td>LON</td><td>Longitude of the reference point in decimal degrees</td></tr><tr><td>HEIGHT</td><td>Ellipsoidal height of the reference point in meters</td></tr></table> Upon startup of the receiver with the e-Dif application running—as opposed to with the SBAS application— no reference position will be present in memory. If you attempt to query for the reference position, the receiver's response will be: $>JRAD,1,FAILED,PRESENT LOCATION NOT STABLE
Example:	When you issue the $JRAD,1 command the response will be similar to the following: $>JRAD,1,51.00233513,-114.08232345,1050.212
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRAD,1,LAT,LON,HEIGHT Command

Command Type:	Dif, DGPS Base Station
Description:	Use this command—a derivative of the JRAD,1,P command—when absolute positioning is required in e-Dif applications only
Command Format:	$JRAD,1,lat,lon,height<CR><LF> where: <table><tr><th>Command Component</th><th>Description</th></tr><tr><td>lat</td><td>Latitude of the reference point in decimal degrees</td></tr><tr><td>lon</td><td>Longitude of the reference point in decimal degrees</td></tr><tr><td>height</td><td>Ellipsoidal height of the reference point in meters. Ellipsoidal height can be calculated by adding the altitude and the geoidal separation, both available from the GPGGA message. **Example:** $GPGGA,173309.00,5101.04028,N,11402.38289,W,2,07,1.4, 1071.0,M,- 17.8,M,6.0, 0122*48 ellipsoidal height = 1071.0 + (-17.8) = 1053.2 meters</td></tr></table> Both latitude and longitude must be entered as decimal degrees. The receiver will not accept the command if there are no decimal places.
Receiver Response:	$>JRAD,LAT,LON,HEIGHT
Example:	
Additional Information:	

Related Commands and Messages:	

Topic Last Updated: v1.00 / August 11, 2010

JRAD,1,P Command

Command Type:	Dif, DGPS Base Station
Description:	e-Dif: Record the current position as the reference with which to compute e- Dif corrections. This would be used in relative mode as no absolute point information is specified. DGPS Base Station: Record the current position as the reference with which to compute Base Station corrections in e-Dif applications only. This would be used in relative mode as no absolute point information is specified
Command Format:	$JRAD,1,P<CR><LF>
Receiver Response:	$>JRAD,1,OK
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.00 / August 11, 2010

JRAD,2 Command

Command Type:	e-Dif
Description:	Forces the receiver to use the new reference point You normally use this command following a JRAD,1 type command.
Command Format:	$JRAD,2<CR><LF>
Receiver Response:	$>JRAD,2,OK
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.00 / August 11, 2010

JRAD,3 Command

Command Type:	e-Dif
Description:	This command has two primary purposes. To invoke the e-Dif function once the unit has started up with the e-Dif application active

	To update the e-Dif solution (calibration) using the current position as opposed to the reference position used by the JRAD,2 command
Command Format:	$JRAD,3<CR><LF>
Receiver Response:	If the receiver has tracked enough satellites for a long enough period before you issue this command, it will respond with the following. (The tracking period can be from 3 to 10 minutes and is used for modeling errors going forward.) $>JRAD,3,OK<CR><LF> If the e-Dif algorithms do not find sufficient data, the receiver responds with: $>JRAD,3,FAILED,NOT ENOUGH STABLE SATELLITE TRACKS
Example:	
Additional Information:	If you receive the failure message after a few minutes of operation, try again shortly after until you receive the “OK” acknowledgement message. The e-Dif application begins operating as soon as the $>JRAD,3,OK message has been received; however, you will still need to define a reference position for e-Dif unless relative positioning is sufficient for any needs.
Related Commands and Messages:	

Topic Last Updated: v1.00 / August 11, 2010

JRAD,7 Command

Command Type:	e-Dif
Description:	Turn auto recalibration on or off
Command Format:	$JRAD,7,n where 'n' is the auto-recalibration variable (0 = Off or 1 = On, 0 is the default)
Receiver Response:	$>JRAD,7,OK
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JRAD,9 Command

Command Type:	DGPS Base Station
Description:	Initialize the Base Station feature and use the previously entered point, either with $JRAD,1,P or $JRAD,1,LAT,LON,HEIGHT, as the reference with which to compute Base Station corrections in e-Dif applications only. Use this for both relative mode and absolute mode.
Command Format:	To initialize/turn off base station mode To initialize base station mode and use stored coordinates:

	JRAD,9,1,1<CR><LF> To turn off base station mode: $JRAD,9,0<CR><LF>
Receiver Response:	$>JRAD,9,OK (same response for turning base station mode on or off)
Example:	
Additional Information:	The $JASC,RTCM,1 command must be sent to the receiver to start outputting standard RTCM corrections.
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JRAD,10 Command

Command Type:	DGPS Base Station
Description:	Specify BDS message to be transmitted by base station
Command Format:	Specify BDS message to be transmitted by base station $JRAD,10,1 Specify BDS message to be not transmitted by base station $JRAD,10,0
Receiver Response:	$>JRAD,10,OK (same response for specify BDS to be transmitted or not)
Example:	
Additional Information:	The $JASC,RTCM,1 command must be sent to the receiver to start outputting standard RTCM corrections.
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JRESET Command

Command Type:	General Operation and Configuration
Description:	Reset the receiver to its default operating parameters by: Turning off outputs on all ports Saving the configuration Setting the configuration to its defaults (in following table)

Configuration	Setting
Elev Mask	5
Residual limit	10
Alt aiding	None

		Age of Diff	45 minutes
		Air mode	Auto
		Diff type	Default for app
		NMEA precision	5 decimals
		COG smoothing	None
		speed smoothing	None
		WAAS	UERE thresholds
Command Format:	$JRESET[,x]<CR><LF> where ',x' is an optional field: • When set to ALL does everything $JRESET does, plus it clears almanacs • When set to BOOT does everything $JRESET,ALL does, plus clears use of the real-time clock at startup,clears use of backed-up ephemeris and almanacs, and reboots the receiver when done		
Receiver Response:	$JRESET $> Saving Configuration. Please Wait... $> $> Save Complete CAUTION: $JRESET clears all parameters. For the V101 Series and the LV101 you will have to issue the $JATT, FLIPBRD,YES command to properly redefine the circuitry orientation inside the product once the receiver has reset. Failure to do so will cause radical heading behavior		
Example:			
Additional Information:			
Related Commands and Messages:			

Topic Last Updated: v1.00 / August 11, 2010

JRELAY Command

Command Type:	General Operation and Configuration
Description:	Send user-defined text out of a serial port
Command Format:	$JRELAY,PORTx,msg<CR><LF> 'x' = destination port where the message (MSG) will be sent 'msg' = message to be sent
Receiver Response:	$>
Example:	**Example 1:** Command: $JRELAY,PORTA,HELLO\nTHERE\n<CR><LF>

	Response: HELLO THERE $> **Example 2:** The following commands apply to the A101 and A325 antennas. You can configure the A101 and A325 through the serial ports using these commands. Configure the setup and output of tilt commands as follows (note that all commands are preceded with $JRELAY,PORTC, to direct them through internal Port C): $JRELAY,PORTC,$JTILT,CALIBRATE[,RESET] Output the tilt offset values for the X and Y axes. If performing a reset, ensure the A101/A325 is on a flat surface. $JRELAY,PORTC,$JTILT,TAU[,value] Output the filter constant for tilt value smoothing. $JRELAY,PORTC,$JTILT,COMPENSATION[,[ON|OFF],[heightoffset]] Turn positioning tilt compensation on/off (currently only the GPGGA data log is supported for tilt compensated position output). $JRELAY,PORTC,$JASC,GPGGA,rate[,port] Turn tilt compensated GPGGA message on. $JRELAY,PORTC,$JTILT,COGBIAS[,value] Set a COG bias to be used in the tilt compensation algorithms (for use when the A101/A325 is not mounted with the connector facing the forward direction of travel). $JRELAY,PORTC,$JASC,INTLT,rate[,port] or $JRELAY,PORTC,$JASC,PSAT,INTLT,rate[,port] Log tilt information from the A101/A325 Set/query the receiver mode—serial or NMEA2000(commands must be sent over Port A): $JRELAY,PORTC,$JQUERYMODE Query the receiver for the current mode $JRELAY,PORTC,$JSERIALMODE Set the receiver mode to serial $JRELAY,PORTC,$JN2KMODE

	Set the receiver mode to NMEA2000
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRAIM Command

Command Type:	General Operation and Configuration
Description:	Specify the parameters of the RAIM scheme that affect the output of the PSAT,GBS message or query the current setting
ommand Format:	Specify the parameters of the RAIM scheme $JRAIM,hpr,probhpr,probfalse<CR><LF> where: **Command Component** / **Description** hpr – Horizontal Protection Radius: notification in the PSAT,GBS message that the horizontal error has exceeded this amount will be received. The acceptable range for this value is 1 to 10,000 m. The default is 10 m. probhpr – Maximum allowed probability that the position computed lies outside the HPR. The acceptable range for this value is 0.001% to 50%. The default is 5%. probfalse – Maximum allowed probability that there is a false alarm (that the position error is reported outside the of the HPR, but it is really within the HPR). The acceptable range for this value is 0.001% to 50%. The default is 1%. Query the current setting $JRAIM
Receiver Response:	Response to issuing command to specify RAIM scheme parameters $> Response to querying the current setting $>JRAIM,HPR,probHPR,probFALSE
Example:	To specify the RAIM scheme parameters as HPR = 8 m, probHPR = 2%, and probFALSE= 0.5% issue the following command: $JRAIM,8,2,0.5<CR><LF> If you then query the receiver for the RAIM scheme issue the following command: $JRAIM<CR><LF>

	...and the response will be: $>JRAIM,8.00,2.0000,0.5000
Additional Information:	The purpose of the probability of false alarm is to help decide on whether to declare a fault or warning in an uncertain situation. The philosophy is to only issue a fault if the user is certain (to within the probability of a false alarm) that the protection radius has been exceeded, else issue a warning.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRTCM3,ANTNAME Command

Command Type:	Local Differential and RTK
Description:	Specify the antenna name that is transmitted in various RTCM3 messages from the base
Command Format:	Specify the antenna name $JRTCM3,ANTNAME,name where name must be an antenna name from the following list: http://www.ngs.noaa.gov/ANTCAL/LoadFile?file=ngs08.003 Query the current setting: $JRTCM3,ANTNAME<CR><LF>
Receiver Response:	Response to issuing command to specify the antenna name $> Response to querying the current setting $JRTCM3,ANTNAME,name where name is the previously specified antenna name
Example:	To specify the antenna name as a Hemisphere GNSS A42 antenna (HEMA42), issue the following command: $JRTCM3,ANTNAME,HEMA42<CR><LF> If you then issue $JRTCM3,ANTNAME<CR><LF> to query the current setting the response is: $>JRTCM3,ANTNAME,HEMA42<CR><LF>
Additional Information:	See JRTCM3,NULLANT for information on setting the antenna name to a null value (no name)
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JRTCM3,EXCLUDE

Command Type:	Local Differential and RTK
Description:	Specify the antenna name that is transmitted in various RTCM3 messages from the base
Command Format:	Specify the RTCM3 messages to not be transmitted $JRTCM3,EXCLUDE[,1004][,1005][,1006][,1007][,1008][,1012][,1033][,1104][,4011][,MSM3][,MSM4]<CR><LF> Query the current setting: $JRTCM3,EXCLUDE<CR><LF>
Receiver Response:	Response to issuing command to exclude specific RTCM3 messages from being transmitted $> Response to querying the current setting: $JRTCM3,EXCLUDE[,MSG1][,MSG2]...[,MSGn]<CR><LF> where MSG1 through MSGn represent each included message type to not be transmitted (excluded)
Example:	Assume all available RTCM3 messages are included (1004, 1005, 1006, 1007, 1008, 1012, 1033). You then issue the following command to exclude message types 1004, 1006, and 1012: $JRTCM3,EXCLUDE,1004,1006,1012<CR><LF> If you then issue $JRTCM3,EXCLUDE<CR><LF> to query the current setting the response is: $>JRTCM3,EXCLUDE,1004,1006,1012<CR><LF> Correspondingly, if you issue $JRTCM3,INCLUDE<CR><LF> to query the current setting for included messages the response is: $>JRTCM3,INCLUDE,1005,1007,1008,1033<CR><LF>
Additional Information:	See JRTCM3,INCLUDE for more information on including RTCM3 messages for transmission
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JRTCM3,INCLUDE Command

Command Type:	Local Differential and RTK
Description:	Specify RTCM3 message types to be transmitted by base station
Command Format:	Specify the RTCM3 messages to be transmitted

	$JRTCM3,INCLUDE[,1004][,1005][,1006][,1007][,1008][,1012][,1033][,1104][,4011][,MSM3][,MSM4]<CR><LF> Query the current setting $JRTCM3,INCLUDE<CR><LF>
Receiver Response:	Response to issuing command to include specific RTCM3 messages to be transmitted $> Response to querying the current setting: $JRTCM3,INCLUDE[,MSG1][,MSG2]...[,MSGn]<CR><LF> where MSG1 through MSGn represent each included message type to be transmitted
Example:	Assume none of the available RTCM3 messages are included (1004,1005, 1006, 1007, 1008, 1012, 1033). You then issue the following command to include message types 1004, 1006,and 1012 $JRTCM3,INCLUDE,1004,1006,1012<CR><LF> If you then issue $JRTCM3,INCLUDE<CR><LF> to query the current setting the response is: $>JRTCM3,INCLUDE,1004,1006,1012<CR><LF>
Additional Information:	See JRTCM3,EXCLUDE for more information on including RTCM3 messages for transmission
Related Commands and Messages:	

Topic Last Updated: v1.07 / October 13, 2016

JRTCM3,NULLANT Command

Command Type:	Local Differential and RTK
Description:	Specify the antenna name as null (no name) that is transmitted in various RTCM3 messages from the base
Command Format:	Specify the antenna name as null $JRTCM3,NULLANT<CR><LF> Response to issuing command to exclude specific RTCM3 messages from being transmitted
Receiver Response:	$>
Example:	Assume you previously specified the antenna name as a Hemisphere GNSS A42 antenna (HEMA42). If you issue $JRTCM3,ANTNAME<CR><LF> to query the current setting the response is:

	$>JRTCM3,ANTNAME,HEMA42<CR><LF> Now send the following command to specify the antenna name as null (no name): $>JRTCM3,NULLANT<CR><LF> If you then issue $JRTCM3,ANTNAME<CR><LF> to query the current setting the response is: $>JRTCM3,ANTNAME,<CR><LF>
Additional Information:	See JRTCM3,ANTNAME for information on specifying the antenna name as something other than null
Related Commands and Messages:	

Topic Last Updated: v1.06 / March 10, 2015

JRTK,1 Command

Command Type:	Local Differential and RTK
Description:	Show the receiver's reference position (can issue command to base station or rover)
Command Format:	$JRTK,1<CR><LF>
Receiver Response:	$JRTK,1,LAT,LON,HEIGHT where: **Command Component** / **Description** LAT — Latitude of the reference point in decimal degrees LON — Longitude of the reference point in decimal degrees HEIGHT — You must enter HEIGHT as ellipsoidal height in meters. Ellipsoidal height can be calculated by adding the altitude and the geoidal separation, both available from the GPGGA message. **Example:** $GPGGA,173309.00,5101.04028,N,11402.38289,W,2,07,1.4,1071.0, M,- 17.8,M,6.0, 0122*48 ellipsoidal height = 1071.0 + (-17.8) = 1053.2 meters
Example:	$>JRTK,1,33.55679117,-111.88955483,374.600
Additional Information:	See JRTCM3,ANTNAME for information on specifying the antenna name as something other than null
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRTK,1,LAT,LON,HEIGHT Command

<table>
<tr><td>Command Type:</td><td>Local Differential and RTK</td></tr>
<tr><td>Description:</td><td>Set the receiver’s reference position to the coordinates you enter (can issue command to base station or rover)</td></tr>
<tr><td>Command Format:</td><td>$JRTK,1,lat,lon,height<CR><LF>

where:

<table>
<tr><th>Command Component</th><th>Description</th></tr>
<tr><td>lat</td><td>Latitude of the reference point in decimal degrees</td></tr>
<tr><td>lon</td><td>Longitude of the reference point in decimal degrees</td></tr>
<tr><td>height</td><td>You must enter HEIGHT as ellipsoidal height in meters.

Ellipsoidal height can be calculated by adding the altitude and the geoidal separation, both available from the GPGGA message.

Example:
$GPGGA,173309.00,5101.04028,N,11402.38289,W,2,07,1.4,1071
M,- 17.8,M,6.0, 0122*48
ellipsoidal height = 1071.0 + (-17.8) = 1053.2 meters</td></tr>
</table>
</td></tr>
<tr><td>Receiver Response:</td><td>$></td></tr>
<tr><td>Example:</td><td>$>JRTK,1,33.55679117,-111.88955483,374.600</td></tr>
<tr><td>Additional Information:</td><td>See JRTCM3,ANTNAME for information on specifying the antenna name as something other than null</td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v1.02 / January 25, 2011

JRTK,1,P Command

Command Type:	Local Differential and RTK
Description:	Set the receiver’s reference coordinates to the current calculated position if you do not have known coordinates for your antenna location (can issue command to base station or rover)
Command Format:	$JRTK,1,P<CR><LF>
Receiver Response:	$>
Example:	
Additional Information:	If you have known coordinates for your antenna location, use the JRTK,1,LAT,LON,HEIGHT command to enter the latitude and longitude (in decimal degrees) and the ellipsoidal height (in meters).
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRTK,5 Command

Command Type:	Local Differential and RTK
Description:	Show the base station's transmission status for RTK applications (can issue command to base station)
Command Format:	$JRTK,5<CR><LF>
Receiver Response:	If transmission status is suspended, response is as follows: $>JRTK,6 If transmission status is not suspended, response is as follows: $>JRTK,5,1
Example:	
Additional Information:	Also see the JRTK,6 command.
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRTK,5,Transmit Command

Command Type:	Local Differential and RTK
Description:	Suspend or resume the transmission of RTK (can issue command to base station)
Command Format:	$JRTK,5,transmit<CR><LF> where "transmit" is 0 (suspend) or 1 (resume)
Receiver Response:	If the transmission status is not suspended and you issue the following command to suspend: $JRTK,5,0<CR><LF> the response is as follows: $>JRTK,5,OK Similarly, if the transmission status is suspended and you issue the following command to resume: $JRTK,5,1<CR><LF> the response is again as follows: $>JRTK,5,OK
Example:	
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRTK,6 Command

<table>
<tr><td>Command Type:</td><td>Local Differential and RTK</td></tr>
<tr><td>Description:</td><td>Display the progress of the base station (can issue command to base station)</td></tr>
<tr><td>Command Format:</td><td>$JRTK,6<CR><LF></td></tr>
<tr><td>Receiver Response:</td><td>$JRTK,6,TimeToGo,ReadyTransmit,Transmitting

where:

<table>
<tr><th>Response Component</th><th>Description</th></tr>
<tr><td>TimeToGo</td><td>Seconds left until ready to transmit RTK</td></tr>
<tr><td>ReadyTransmit</td><td>Nonzero when configured to transmit and ready to transmit RTK on at least one port. It is a bit mask of the transmitting port, with bit 0 being port A, bit 1 being port B, and bit 2 being port C. It will be equal to "Transmitting" unless transmission has been suspended with $JRTK,5,0.</td></tr>
<tr><td>Transmitting</td><td>Non-zero when actually transmitting RTK on at least one port. It is a bit mask of the transmitting port, with bit 0 being port A, bit 1 being port B, and bit 2 being port C.</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td>If the receiver is not ready to transmit:

$>JRTK,6,263,0,0

If the receiver is currently transmitting on Port B:

$>JRTK,6,0,2,2</td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td></td></tr>
</table>

Topic Last Updated: v1.02 / January 25, 2011

JRTK,12 Command

Command Type:	Local Differential and RTK
Description:	Warning! Hemisphere GNSS recommends that only advanced users employ this command. Disable or enable the receiver to go into fixed integer mode (RTK) vs. float mode (L-Dif) - can issue command to rover. Note: Requires RTK rover subscription
Command Format:	$JRTK,12,x 1 = Allow RTK (recommended, and the default) 0 = Do not allow RTK, stay in L-Dif
Receiver Response:	$>
Example:	
Additional Information:	In high multipath conditions it may be desirable to prevent the rover from obtaining a fixed position. Using $JRTK,12,0 while logging position data is useful for determining the level of multipath present.

Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRTK,17 Command

Command Type:	Local Differential and RTK
Description:	Display the transmitted latitude, longitude, and height of the base station (can issue command to base station or rover)
Command Format:	$JRTK,17<CR><LF>
Receiver Response:	$>JRTK,17,lat,lon,height
Example:	$>JRTK,17,33.55709242,-111.88916894,380.534
Additional Information:	Format is similar to JRTK,1,LAT,LON,HEIGHT
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JRTK,18 Command

Command Type:	Local Differential and RTK
Description:	Display the distance from the rover to the base station, in meters (can issue command to rover)
Command Format:	$JRTK,18<CR><LF>
Receiver Response:	$>JRTK,18,d d' is the baseline distance in meters 'm' indicates the units are meters
Example:	$>JRTK,18,13154.520
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.03 / January 11, 2012

JRTK,18,BEARING Command

Command Type:	Local Differential and RTK
Description:	Display the bearing from the base station to the rover,in degrees (can issue command to rover)
Command Format:	$JRTK,18,BEARING<CR><LF>
Receiver Response:	$>JRTK,18,b • 'b' is the bearing from base to rover in degrees • 'd' indicates the units are degrees
Example:	$>JRTK,18,20.014
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.03 / January 11, 2012

JRTK,18,NEU Command

Command Type:	Local Differential and RTK
Description:	Display the distance from the rover to the base station and the delta North, East, and Up, in meters can issue command to rover)
Command Format:	$JRTK,18,NEU<CR><LF>
Receiver Response:	$>JRTK,18,d,X,Y,Z where: 'd' is the baseline distance in meters 'm' indicates the units are meters 'X' is the North delta, in meters 'Y' is the East delta, in meters 'Z' is the Up delta, in meters
Example:	$>JRTK,18,13154.509,12360.045,4502.139,33.739
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.03 / January 11, 2012

JRTK,18,NEU Command

Command Type:	Local Differential and RTK
Description:	Set the base stationID transmitted in ROX/DFX/CMR/RTCM3 messages (can issue command to base station), where: • Default is 333 • Range is 0-4095 (except for CMR which is 0-31)
Command Format:	Set the base station ID $JRTK,28,baseid<CR><LF> where 'baseid' is the base station ID Query the current setting: $JRTK,28<CR><LF>
Receiver Response:	$>
Example:	To set the base station ID to 123 issue the following command: $JRTK,28,123<CR><LF> If the base station ID is 333 and you issue the $JRTK,28<CR><LF> query the response is: $>JRTK,28,333
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.02 / January 25, 2011

JSAVE Command

Command Type:	General Operation and Configuration
Description:	Send this command after making changes to the operating mode of the receiver
Command Format:	$JSAVE<CR><LF>
Receiver Response:	$> SAVING CONFIGURATION. PLEASE WAIT...

	then $> Save Complete
Example:	
Additional Information:	Ensure that the receiver indicates that the save process is complete before turning the receiver off or changing the configuration further. No data fields are required. The receiver indicates that the configuration is being saved and indicates when the save is complete.
Related Commands and Messages:	

Topic Last Updated: v1.00 / August 11, 2010

JSHOW Command

Command Type:	General Operation and Configuration
Description:	Query the current operating configuration of the receiver
Command Format:	$JSHOW<CR><LF>
Receiver Response:	Use the JSHOW command to provide a complete response from the receiver.
Example:	(number in parentheses corresponds to line number in table following the response): $>JSHOW,BAUD,9600 (1) $>JSHOW,BAUD,9600,OTHER (2) $>JSHOW,BAUD,9600,PORTC (3) $>JSHOW,ASC,GPGGA,1.0,OTHER (4) $>JSHOW,ASC,GPVTG,1.0,OTHER (5) $>JSHOW,ASC,GPGSV,1.0,OTHER (6) $>JSHOW,ASC,GPGST,1.0,OTHER (7) $>JSHOW,ASC,D1,1,OTHER (8) $>JSHOW,DIFF,WAAS (9) $>JSHOW,ALT,NEVER (10) $>JSHOW,LIMIT,10.0 (11) $>JSHOW,MASK,5 (12) $>JSHOW,POS,51.0,-114.0 (13) $>JSHOW,AIR,AUTO,OFF (14) $>JSHOW,FREQ,1575.4200,250 (15) $>JSHOW,AGE,1800 (16)

Description of responses:

Line	Description
1	Current port is set to a baud rate of 9600
2	Other port is set to a baud rate of 9600
3	Port C is set to a baud rate of 9600 (Port C is not usually connected externally on the finished product)
4	GPGGA is output at a rate of 1 Hz from the other port
5	GPVTG is output at a rate of 1 Hz from the other port
6	GPGSV is output at a rate of 1 Hz from the other port
7	GPGST is output at a rate of 1 Hz from the other port
8	D1 is output at a rate of 1 Hz from the other port
9	Current differential mode is WAAS
10	Status of the altitude aiding feature (see the JALT command for information how to set turn altitude aiding on or off)
11	Receiver does not support this feature
12	Elevation mask cutoff angle (in degrees)
13	Current send position used for startup, in decimal degrees
14	Current status of the AIR mode (see the JAIR command for information how to set the AIR mode)
15	Current frequency of the augmentation source in use for the receiver (depending on the configuration of the receiver), followed by the bit rate from the SBAS satellite, and optionally followed by 'AUTO' (only when the Atlas receiver is in ‘auto-tune’ mode)
16	Current maximum acceptable differential age, in seconds (see the JAGE command for information how to set the differential age)

See "Receiver Response" section above

Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.07 / February 16, 2017

JSHOW,ASC Command

Command Type:	General Operation and Configuration
Description:	Query receiver for current ASCII messages being output
Command Format:	$JSHOW,ASC[,x]<CR><LF> where x is one of the following:

<table>
<tr><td></td><td>PORTA
PORTB
PORTC
PORTD
OTHER – displays

Whatever port you are connected to you do not need to specify that port. For example, if you connected to Port A, the following two commands result in the same response:

$JSHOW,ASC<CR><LF>

$JSHOW,ASC,PORTA<CR><LF></td></tr>
<tr><td>Receiver Response:</td><td>See Example section below</td></tr>
<tr><td>Example:</td><td>The first row below shows the response to each individual command for Port A (with and without specifying Port A), Port B, and Port C.

The second row shows the response to the generic $JSHOW command with items similar to the first-row responses highlighted.

<table>
<tr><th>Command Sent to Receiver</th><th>Response</th></tr>
<tr><td>$JSHOW,ASC</td><td>$>JSHOW,ASC,RTCM,1</td></tr>
<tr><td>$JSHOW,ASC,PORTA</td><td>$>JSHOW,ASC,RTCM,1</td></tr>
<tr><td>$JSHOW,ASC,PORTB</td><td>$>JSHOW,ASC,CMR,1,OTHER</td></tr>
<tr><td>$JSHOW,ASC,PORTC</td><td>$>JSHOW,ASC,D1,1,PORTC</td></tr>
<tr><td>JSHOW</td><td>$>JSHOW,BAUD,19200
$>JSHOW,ASC,GPGNS,1.00
$>JSHOW,ASC,GPGRS,1.00
$>JSHOW,BIN,1,1.00
$>JSHOW,BIN,2,1.00
$>JSHOW,BIN,89,1
$>JSHOW,BIN,99,1
$>JSHOW,ASC,RTCM,1.0
$>JSHOW,BAUD,19200,OTHER
$>JSHOW,ASC,CMR,1,OTHER
$>JSHOW,BAUD,57600,PORTC
$>JSHOW,ASC,GPGGA,1.00,PORTC
$>JSHOW,ASC,GPGSV,1.00,PORTC
$>JSHOW,ASC,GLGSV,1.00,PORTC
$>JSHOW,BIN,69,1,PORTC
$>JSHOW,BIN,100,1,PORTC
$>JSHOW,ASC,D1,1,PORTC
$>JSHOW,DIFF,RTK
$>JSHOW,ALT,NEVER
$>JSHOW,LIMIT,10.0
$>JSHOW,MASK,5
$>JSHOW,POS,33.6,-112.2
$>JSHOW,AIR,AUTO,NORM</td></tr>
</table>
</td></tr>
</table>

		$>JSHOW,SMOOTH,LONG900 $>JSHOW,FREQ,1575.4200,250 $>JSHOW,AGE,2700 $>JSHOW,THISPORT,PORTA $>JSHOW,MODES,FOREST,BASE,GPSONLY,GLOFIX,SURETRACK
Additional Information		
Related Commands and Messages:		

Topic Last Updated: v1.04 / May 29, 2012

JSHOW,ASC Command

Command Type:	General Operation and Configuration
Description:	Query receiver for current ASCII messages being output
Command Format:	$JSHOW,ASC[,x]<CR><LF> where x is one of the following: PORTA PORTB PORTC PORTD OTHER – displays Whatever port you are connected to you do not need to specify that port. For example, if you connected to Port A, the following two commands result in the same response: $JSHOW,ASC<CR><LF> $JSHOW,ASC,PORTA<CR><LF>
Receiver Response:	See Example section below
Example:	The first row below shows the response to each individual command for Port A (with and without specifying Port A), Port B, and Port C. The second row shows the response to the generic $JSHOW command with items similar to the first row responses highlighted.

Command Sent to Receiver	Response
$JSHOW,ASC	$>JSHOW,ASC,RTCM,1
$JSHOW,ASC,PORTA	$>JSHOW,ASC,RTCM,1
$JSHOW,ASC,PORTB	$>JSHOW,ASC,CMR,1,OTHER
$JSHOW,ASC,PORT	$>JSHOW,ASC,D1,1,PORTC

		C	
		JSHOW	$>JSHOW,BAUD,19200 $>JSHOW,ASC,GPGNS,1.00 $>JSHOW,ASC,GPGRS,1.00 $>JSHOW,BIN,1,1.00 $>JSHOW,BIN,2,1.00 $>JSHOW,BIN,89,1 $>JSHOW,BIN,99,1 $>JSHOW,ASC,RTCM,1.0 $>JSHOW,BAUD,19200,OTHER $>JSHOW,ASC,CMR,1,OTHER $>JSHOW,BAUD,57600,PORTC $>JSHOW,ASC,GPGGA,1.00,PORTC $>JSHOW,ASC,GPGSV,1.00,PORTC $>JSHOW,ASC,GLGSV,1.00,PORTC $>JSHOW,BIN,69,1,PORTC $>JSHOW,BIN,100,1,PORTC $>JSHOW,ASC,D1,1,PORTC $>JSHOW,DIFF,RTK $>JSHOW,ALT,NEVER $>JSHOW,LIMIT,10.0 $>JSHOW,MASK,5 $>JSHOW,POS,33.6,-112.2 $>JSHOW,AIR,AUTO,NORM $>JSHOW,SMOOTH,LONG900 $>JSHOW,FREQ,1575.4200,250 $>JSHOW,AGE,2700 $>JSHOW,THISPORT,PORTA $>JSHOW,MODES,FOREST,BASE,GPSONLY,GLOFIX,SURETRACK
Additional Information:			
Related Commands and Messages:			

Topic Last Updated: v1.04 / May 29, 2012

JSHOW,BIN Command

Command Type:	General Operation and Configuration
Description:	Query receiver for current Bin messages being output
Command Format:	$JSHOW,BIN<CR><LF>
Receiver Response:	$>JSHOW,BIN,B1,B1R,B2,B2R...,Bn,BnR where: • B1 is the first Bin message being output B1R is the rate of B1 • B2 is the second Bin message being output B2R is the rate of B2

	• Bn is the last Bin message being output BnR is the rate of Bn
Example:	$>JSHOW,BIN,B01,1.00,B02,1.00,B69,1,B80,1,B89,1,B99,1
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JSHOW,CONF Command

Command Type:	General Operation and Configuration
Description:	Query receiver for configuration settings
Command Format:	$JSHOW,CONF<CR><LF>
Receiver Response:	$>JSHOW,CONF,AID,AIDVAL,RES,ELEV,MODE,AGE,DIFF where:

Message Component	Description	As Displayed in Example Below This Table
AID	Altitude aiding indicator as set by JALT command: A = ALWAYS N = NEVER S = SOMETIMES T = SATS	A
AIDVAL	Altitude aiding value as by JALT command: If AID = N, then AIDVAL = 0.0 If AID = A, then AIDVAL = height If AID = S, then AIDVAL = PDOP threshold If AID = T, then AIDVAL = number of sats	404.2
RES	Residual limit for the $JLIMIT command	10.0
ELEV	Elevation mask cutoff angle (in degrees) as set by JMASK command	5
MODETYPE	AIR mode type, A (AUTO) or M (MANUAL), as set by JAIR command	M
MODE	AIR mode, LOW or HIGH or NORM, as set by JAIR command	LOW
AGE	Maximum acceptable differential age (in seconds)	8100 (259200 is using e-Dif)
DIFF	Current differential mode as set by JDIFF command: T = THIS PORT P = PORTC O (letter) = OTHER PORT	A

Example:	$>JSHOW,CONF,A,404.2,10.0,5,M,LOW,259200,A
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JSHOW,GP Command

Command Type:	General Operation and Configuration
Description:	Query the receiver for each GP message currently being output through the current port and the update rate for that message To see output for other ports you must specify that port or OTHER
Command Format:	$JSHOW,GP[,PORTX][,OTHER]<CR><LF> where: • ',PORTX' = a port other than the current port, such as Port B or Port C • ',OTHER' = Port B if the current port is Port A, or Port A if the current port is Port B
Receiver Response:	$>JSHOW,M1,M1R,M2,M2R...,Mn,MnR where: • M1 is the first message being output M1R is the rate of M1 • M1 is the first message being output M1R is the rate of M1 • Mn is the last message being output MnR is the rate of Bn
Example:	$>JSHOW,GP,GGA,1.00,GST,1.00
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JSHOW,THISPORT Command

Command Type:	General Operation and Configuration
Description:	Query to determine which receiver port you are connected to
Command Format:	$JSHOW,THISPORT<CR><LF>
Receiver Response:	$>JSHOW,THISPORT,port where 'port' is the port you are connected to
Example:	Response if you are connected to Port B: $>JSHOW,THISPORT,PORTB
Additional Information:	See JSHOW for information on displaying more configuration information for a receiver
Related Commands and Messages:	

Topic Last Updated: v1.03 / January 11, 2012

JSIGNAL Command

Command Type:	General Operation and Configuration
Description:	Set the GNSS signals that the receiver will attempt to track. Specific signals

	shown here are only valid for receivers supporting the signal in question.
Command Format:	Specify the signal(s)to be used: $JSIGNAL,INCLUDE[,L1CA][,L1P][,L2P][,L2C][,G1][,G2][,E1BC][,B1][,B2][,B3][,E5B][,QZSL1CA][,QZSL2C][,ALL]<CR><LF> Specify the signal(s) NOT to be used: $JSIGNAL,EXCLUDE[,L1CA][,L1P][,L2P][,L2C][,G1][,G2][,E1BC][,B1][,B2][,B3][,E5B][,QZSL1CA][,QZSL2C][,ALL]<CR><LF> Query the current setting: $JSIGNAL,INCLUDE<CR><LF>
Receiver Response:	Response to issuing command to turn functionality on/off $> Response to querying the current setting $>JSIGNAL,INCLUDE[,L1CA][,L1P][,L2P][,L2C][,G1][,G2][,E1BC][,B1][,B2][,B3][,E5B][,QZSL1CA][,QZSL2C]<CR><LF>
Example:	Response if you are connected to Port B: $>JSHOW,THISPORT,PORTB
Additional Information:	See JSHOW for information on displaying more configuration information for a receiver
Related Commands and Messages:	

Topic Last Updated: v1.10 / February 16, 2017

JSIGNAL Command

Command Type:	GPS
Description:	Set the carrier smoothing interval (15 to 6000 seconds)or query the current setting This command provides the flexibility to tune in different environments. The default for this command is 900 seconds (15 minutes) or LONG. A slight improvement in positioning performance (depending on the multipath environment) may occur if you use either the SHORT (300 seconds) or LONG (900 seconds) smoothing interval.
Command Format:	Set the carrier smoothing interval: To set the carrier smoothing interval to a specific number of seconds issue the following command: $JSMOOTH,x<CR><LF> where 'x' is one of the following: Number of seconds: DEFAULT (equals 900 seconds) Default for e-Dif is 300 second SHORT (equals 300 seconds) LONG (equals 900 seconds) Query the current setting:

	$JSMOOTH<CR><LF>
Receiver Response:	Receiver response when setting the carrier smoothing interval $> Receiver response when querying the current carrier smoothing interval $>JSMOOTH,x where 'x' is the word 'SHORT' or 'LONG' followed by the number of seconds used: • SHORT precedes the number of seconds for any setting less than 900 seconds • LONG precedes the number of seconds for any setting greater than or equal to 900 seconds
Example:	To set the carrier smoothing interval to 750 seconds issue the following command: $JSMOOTH,750<CR><LF> ...and if you then query the receiver using $JSMOOTH the response is: $JSMOOTH,SHORT750 To set the carrier smoothing interval to 300 seconds (5 minutes) issue the following command: $JSMOOTH,SHORT<CR><LF> To set the carrier smoothing interval to 900 seconds (15 minutes) issue the following command: $JSMOOTH,LONG<CR><LF>
Additional Information:	If you are unsure of the best value for this setting, leave it at the default setting of LONG (900 seconds). The status of this command is also output in the JSHOW message.
Related Commands and Messages:	

Topic Last Updated: v1.04 / May 29, 2012

JSIGNAL Command

Command Type:	General Operation and Configuration
Description:	Returns the boot loader version from the GNSS receiver
Command Format:	$JSYSVER<CR><LF>
Receiver Response:	$>SYSVER,v where 'v' is the boot loader version
Example:	Response when the boot loader version is 162 $>SYSVER,162
Additional Information:	
Related Commands and Messages:	

Topic Last Updated: v1.05 / January 18, 2013

JT Command

<table>
<tr><td>Command Type:</td><td>General Operation and Configuration</td></tr>
<tr><td>Description:</td><td>Query the receiver for its GPS engine type</td></tr>
<tr><td>Command Format:</td><td>$JT<CR><LF></td></tr>
<tr><td>Receiver Response:</td><td>$>JT,xxxx

where xxxx indicates the GPS engine and mode:

<table>
<tr><th>JT Command Response (xxxx)</th><th>GPS Engine</th><th>Mode</th></tr>
<tr><td>DF2b</td><td>Eclipse</td><td>WAAS, RTK Base</td></tr>
<tr><td>DF2g</td><td>Eclipse</td><td>L-band</td></tr>
<tr><td>DF2r</td><td>Eclipse</td><td>RTK Rover</td></tr>
<tr><td>DF3g</td><td>Eclipse II</td><td>WAAS, RTK Base</td></tr>
<tr><td>DF3i</td><td>Eclipse II</td><td>e-Dif</td></tr>
<tr><td>DF3r</td><td>Eclipse II</td><td>RTK Rover</td></tr>
<tr><td>MF3g</td><td>miniEclipse</td><td>WAAS, RTK Base</td></tr>
<tr><td>MF3i</td><td>miniEclipse</td><td>e-Dif</td></tr>
<tr><td>MF3r</td><td>miniEclipse</td><td>RTK Rover</td></tr>
<tr><td>SX2a</td><td>Crescent Vector</td><td>WAAS RTK</td></tr>
<tr><td>SX2b</td><td>Crescent</td><td>Base</td></tr>
<tr><td>SX2g</td><td>Crescent</td><td>WAAS</td></tr>
<tr><td>SX2i</td><td>Crescent</td><td>e-Dif</td></tr>
<tr><td>SX2r</td><td>Crescent</td><td>Rover</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td>When you issue the $JT<CR><LF>command a typical response may be:

$>JT,DF2b,MX31rev=28DF2b indicates an Eclipse receiver with WAAS and RTK Base functionality.

Note:MX31rev=28 is the processor type and only appears as part of the Eclipse receiver response. You can disregard the processor type as the text that precedes it (DF2b in this example) provides the requested information (GPS engine and mode).</td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands</td><td></td></tr>
</table>

and Messages:	

Topic Last Updated: v1.03 / January 11, 2012

JTAU,COG Command

Command Type:	GPS
Description:	Note: The JATT,COGTAU command provides identical functionality but works only with Crescent Vector products. Set the course over ground (COG) time constant(0.00 to 3600.00 seconds) or query the current setting. This command allows you to adjust the level of responsiveness of the COG measurement provided in the GPVTG message. The default value is 0.00 seconds of smoothing. Increasing the COG time constant increases the level of COG smoothing.
Command Format:	Set the COG time constant $JTAU,COG,tau<CR><LF> where 'tau' is the new COG time constant that falls within the range of 0.00 to 200.0 seconds The setting of this value depends upon the expected dynamics of the Crescent. If the Crescent will be in a highly dynamic environment, this value should be set lower because the filtering window would be shorter, resulting in a more responsive measurement. However, if the receiver will be in a largely static environment, this value can be increased to reduce measurement noise. Query the current setting: $JTAU,COG<CR><LF>
Receiver Response:	Receiver response when setting the COG time constant $> Receiver response when querying the current COG time constant $>JTAU,COG,tau<CR><LF>
Example:	To set the COG time constants 2 seconds issue the following command: $JTAU,COG,2<CR><LF>
Additional Information:	You can use the following formula to determine the COG time constant: tau (in seconds) = 10 / maximum rate of change of course (in °/s) If you are unsure about the best value for this setting, it is best to be conservative and leave it at the default setting of 0.00 seconds.
Related Commands and Messages:	

Topic Last Updated: v4.2 / September 13, 2022

JTAU,SPEED Command

Command Type:	GPS
Description:	Set the speed time constant (0.00 to 3600.00 seconds) or query the current setting This command allows you to adjust the level of responsiveness of the speed measurement provided in the GPVTG message. The default value is 0.00 seconds of smoothing. Increasing the speed time constant increases the level of speed measurement smoothing.
Command Format:	Set the speed time constant: $JTAU,SPEED,tau<CR><LF> where 'tau' is the new speed time constant that falls within the range of 0.0 to 200.0 seconds The setting of this value depends upon the expected dynamics of the receiver. If the receiver will be in a highly dynamic environment, you should set this to a lower value, since the filtering window will be shorter, resulting in a more responsive measurement. However, if the receiver will be in a largely static environment, you can increase this value to reduce measurement noise. Query the current setting: $JTAU,SPEED<CR><LF>
Receiver Response:	Receiver response when setting the speed time constant $> Receiver response when querying the current speed time constants $>JTAU,SPEED,tau<CR><LF>
Example:	To set the speed time constant as 4.6 seconds issue the following command: $JTAU,SPEED,4.6<CR><LF>
Additional Information:	You can use the following formula to determine the COG time constant (Hemisphere GNSS recommends testing how the revised value works in practice): tau (in seconds) = 10 / maximum acceleration (in m/s^2) If you are unsure about the best value for this setting, it is best to be conservative and leave it at the default setting of 0.00 seconds.
Related Commands and Messages:	

Topic Last Updated: v4.2 / September 13, 2022

$JTIMING,ATLASCLOCK,YES/NO Command

Command Type:	General operation and configuration
Description:	Enable/disable receiver clock steering by the Atlas solution.
Command Format:	To enable Atlas clock steering:

	$JTIMING,ATLASCLOCK,YES<CR><LF> To disable Atlas clock steering: $JTIMING,ATLASCLOCK,NO<CR><LF> Query the current setting: $JTIMING,ATLASCLOCK<CR><LF>
Receiver Response:	Response to issuing command to enable/disable Atlas clock steering: $> Response to querying the current setting: $>JTIMING,ATLASCLOCK,[YES/NO]
Example:	
Additional Information:	Clock steering by Atlas is disabled by default.
Related Commands and Messages:	

Topic Last Updated: v.3.0 / December 30, 2019

$JTIMING,MANUALMARK[,yes/no] Command

Command Type:	General operation and configuration
Description:	The $JTIMING,MANUALMARK[,yes/no] command is used to enable or disable manual mark.
Command Format:	To enable manual mark: $JTIMING,MANUALMARK,YES<CR><LF> To disable manual mark: $JTIMING,MANUALMARK,NO<CR><LF> Query the current setting: $JTIMING,MANUALMARK<CR><LF>
Receiver Response:	Response to issuing command to enable/disable manual mark: $> Response to querying the current setting: $>JTIMING,MANUALMARK,[YES/NO]
Example:	
Additional Information:	Manual mark mode is enabled by default
Related Commands and Messages:	

. Topic Last Updated: v.3.0 / December 30, 2019

$JTIMING,MANUALMARK[,yes/no] Command

Command Type:	General operation and configuration
Description:	The $JTIMING,HALTCLOCKSTEER command is used to prevent GNSS clock steering, for use with external atomic ref clocks.
Command Format:	To disable GNSS clock steering: $JTIMING,HALTCLOCKSTEER,YES<CR><LF> To enable GNSS clock steering: $JTIMING,HALTCLOCKSTEER,NO<CR><LF> Query the current setting: $JTIMING,HALTCLOCKSTEER<CR><LF>
Receiver Response:	Response to querying the current setting: $> JTIMING,HALTCLOCKSTEER,[YES/NO]
Example:	
Additional Information:	GNSS clock steering is enabled by default. This command can be used to prevent clock steering, for example, if using an external atomic reference clock.
Related Commands and Messages:	

. Topic Last Updated: v.3.0 / December 30, 2019

NMEA 0183 Messages

GLMLA Message

Message Type:	GLONASS
Description:	GLONASS almanac data Contains complete almanac data for one GLONASS satellite. Multiple sentences may be transmitted, one for each satellite in the GLONASS constellation.
Message Format:	$JASC,GLMLA,r[,OTHER]<CR><LF> where: 'r' = 1 (on) or 0 (off) When set to on the message is sent once (one message for each tracked satellite at 1 second intervals) and then sent again whenever satellite information changes ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the

<table>
<tr><td></td><td>other port when you send the command with it (without the brackets)

$GLMLA,A.A,B.B,CC,D.D,EE,FFFF,GG,HHHH,IIII,JJJJJJ,KKKKKK,MMMMMM,NNNNNN,PPP,QQQ*hh<CR><LF>

where:
<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>A.A</td><td>Total number of sentences</td></tr>
<tr><td>B.B</td><td>Sentence number</td></tr>
<tr><td>CC</td><td>Satellite ID (satellite slot) number</td></tr>
<tr><td>D.D</td><td>Calendar day count within the four year period beginning with the previous leap year</td></tr>
<tr><td>EE</td><td>Generalized health of the satellite and carrier frequency number respectively</td></tr>
<tr><td>FFFF</td><td>Eccentricity</td></tr>
<tr><td>GG</td><td>DOT, rate of change of the draconitic circling time</td></tr>
<tr><td>HHHH</td><td>Argument of perigee</td></tr>
<tr><td>IIII</td><td>16 MSB of system time scale correction</td></tr>
<tr><td>JJJJJJ</td><td>Correction to the average value of the draconitic circling time</td></tr>
<tr><td>KKKKKK</td><td>Time of the ascension node, almanac reference time</td></tr>
<tr><td>MMMMMM</td><td>Greenwich longitude of the ascension node</td></tr>
<tr><td>NNNNNN</td><td>Correction to the average value of the inclination angle</td></tr>
<tr><td>PPP</td><td>LSB of system time scale correction</td></tr>
<tr><td>QQQ</td><td>Course value of the time scale shift</td></tr>
</table></td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GL
Similar to the GPS message GPALM</td></tr>
</table>

Topic Last Updated: v1.05 / January 18, 2013

GPALM Message

Message Type:	Data
Description:	Message number (individual and total),week number, satellite health, and the almanac data for each satellite in the GPS constellation up to a maximum of 32 messages.
Message Format:	Command Format to Request Message: $JASC,GPALM,r[,OTHER]<CR><LF>

where

'r' = 1 (on) or 0 (off)

When set to on the message is sent once (one message for each tracked satellite at 1 second intervals) and then sent again whenever satellite information changes

',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPALM,A,B,C,D,E,F,G,H,J,K,L,M,N,P,Q*CC<CR><LF>

where:

Response Component	Description	As Displayed in First Full Line of Example Below This Table
A	Total number of messages	31
B	Message number	1
C	Satellite PRN number	02
D	GPS week number (0-1023)	1617
E	Satellite health (bits 17-24 of message)	00
F	Eccentricity	50F6
G	Reference time of almanac (TOA)	0F
H	Satellite inclination angle (sigma)	FD98
J	Rate of right ascension (omega dot)	FD39
K	Square root of semi-major axis (root A)	A10CF3
L	Perigee (omega)	81389B
M	Ascending node longitude (omega O)	423632
N	Mean anomaly (mo)	BD913C
P	Clock parameter 0 (af0)	148
Q	Clock parameter 1 (af1)	001
*CC	Checksum	
<CR>	Carriage return	

		Response Component	**Description**	**As Displayed in First Full Line of Example Below This Table**
		A	Total Number of Messages	31
		B	Message number	1
		C	Satellite PRN number	02
Example:	$> $GPALM,31,1,02,1617,00,50F6,0F,FD98,FD39,A10CF3,81389B,423632,BD913C,148 ,001* $GPALM,31,2,03,1617,00,71B9,0F,F6C2,FD45,A10C96,2B833C,131DB4,BA69EE,2B1, 001* $GPALM,31,3,04,1617,00,4F01,0F,FD03,FD39,A10BFC,1C6C35,42EDB1,35B537,112, 003* $GPALM,31,4,05,1617,00,121B,0F,08C8,FD61,A10C5C,09CA99,6D7257,021B32,79F, 7FE* $GPALM,31,5,06,1617,00,337F,0F,FB6B,FD49,A10CC2,DBE103,161127,10CD11,18C, 7FE*. $GPALM,31,29,30,1617,00,6A85,0F,0ADD,FD5C,A11A83,3F6243,EBCC46,E8548D,145, 001 $GPALM,31,30,31,1617,00,4037,0F,1778,FD3E,A10C28,D62817,C32ADF,781125,01B, 001 $GPALM,31,31,32,1617,00,65B5,0F,0956,FD65,A10DD0,DD74BA,71125D,985AE3,751, 7FE		
Additional Information:				
Related Commands and Messages:	Similar to the GLONASS message GLMLAJASC,GP			

Topic Last Updated: v1.05 / January 18, 2013

GPDTM Message

Message Type:	Data
Description:	Datum reference
Message Format:	Command Format to Request Message: $JASC,GPDTM,r[,OTHER]<CR><LF> where: 'r' = message rate (in Hz) of (1 or 0) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPDTM,CCC,A,X.X,K,X.X,L,X.X,CCC*CC<CR><LF>

where:
$GPALM,A,B,C,D,E,F,G,H,J,K,L,M,N,P,Q*CC<CR><LF>

where:

Message Component	Description
CCC	Local datum (normally W84, but could be NAD83 when using beacon in North America)
A	Local datum subdivision code
X.X	Latitude offset, in minutes
K	Latitude indicator; value is N (North latitude) or S (South latitude)
X.X	Longitude offset, in minutes
L	Longitude indicator; value is E (East longitude) or W (West longitude)
X.X	Altitude offset, in meters
CCC	Reference datum (always W84)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Example:	$GPDTM,W84,,0.0,N,0.0,E,0.0,W84*CC<CR><LF>
Additional Information:	
Related Commands and Messages:	JASC,GP

Topic Last Updated: v1.05 / January 18, 2013

GPGGA Message

Message Type:	Data
Description:	Detailed GNSS position information (most frequently used NMEA 0183 data message)
Message Format:	$GPGGA,HHMMSS.SS,DDMM.MMMMM,K,DDDMM.MMMMM,L,N,QQ,PP.P,AAA A.AA,M,±XX.XX,M, SSS,RRRR*CC<CR><LF>

where:

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the position
DDMM.MMMMM	Latitude in degrees, minutes, and decimal minutes (you ca set the number of decimal places using the JNP commanc
K	Latitude indicator; value is N (North latitude) or S (South latitude)
DDDMM.MMMMM	Longitude in degrees, minutes, and decimal minutes (you can set the number of decimal places using the JNP command)
L	Longitude indicator; value is E (East longitude) or W (West longitude)
N	Quality indicator; value is: 0 = no position 1 = no differential corrections(autonomous) 2 = differentially corrected position (SBAS, DGPS,Atlas DGPSservice, L- Dif and e-Dif) 4 = RTK fixed or, Atlas high precision services converged 5 = RTK float,Atlas high precision services converging
QQ	Number of satellites used in position solution
P.P	Horizontal dilution of precision (HDOP)
A.A	Antenna altitude, in meters, re: mean-sea-level (geoid)
M	Units of antenna altitude (M = meters)
G.G	Geoidal separation (in meters)
M	Units of geoidal separation (M = meters)
SSS	Age of differential corrections, in seconds
RRRR	Differential reference station ID
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Example:	$$GPGGA,001038.00,3334.2313457,N,11211.0576940,W,2,04,5.4,354.682,M,-26.574,M,7.0,0138*79
Additional Information:	This message provides information specific to the satellite system identified by the first two characters of the message. GPGGA - GPS information. The JNMEA,GGAALLGNSS command significantly affects the output of the GGA message. If you are tracking more than GNSS signals, Hemisphere GNSS highly recommends that you review this command.
Related Commands and Messages:	JASC,GP, JASC,GN, JASC,GL, JNMEA,GGAALLGNSS

Topic Last Updated: v1.07 / February 16, 2017

GPGLL Message

Message Type:	Data
Description:	Latitude and longitude data
Message Format:	Command Format to Request Message: $JASC,GPGLL,r[,OTHER]<CR><LF>

<table>
<tr><td></td><td>where:

'r' = message rate in Hz of 20, 10, 2, 1, 0, or .2 (0 turns off the message)
',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPGLL,DDMM.MMMMM,S,DDDMM.MMMMM,S,HHMMSS.SS,S*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>DDMM.MMMMM</td><td>Latitude in degrees, minutes, and decimal minutes</td></tr>
<tr><td>S</td><td>S = N (North latitude) or S (South latitude)</td></tr>
<tr><td>DDDMM.MMMMM</td><td>Longitude in degrees, minutes, and decimal minutes</td></tr>
<tr><td>S</td><td>S = E (East longitude) or W (West longitude)</td></tr>
<tr><td>HHMMSS.SS</td><td>UTC time in hours, minutes, and seconds of GNSS position</td></tr>
<tr><td>S</td><td>Status, S = A (valid) or V (invalid)</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td>This message provides information specific to the satellite system identified by the first two characters of the message. GPGLL - GPS information GNGLL - GNSS information GLGLL - GLONASS information.

The JNMEA,GGAALLGNSS command significantly affects the output of the GLL message. If you are tracking more than GNSS signals, Hemisphere GNSS highly recommends that you review this command.</td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP, JASC,GN, JASC,GL, JNMEA,GGAALLGNSS</td></tr>
</table>

Topic Last Updated: v1.07 / February 16, 2017

GPGNS Message

Message Type:	Data
Description:	Fixes data for single or combined (GPS, GLONASS, possible future satellite systems, and systems combining these) satellite navigation systems
Message Format:	Command Format to Request Message:

$JASC,GPGNS,r[,OTHER]<CR><LF>

where:

'r' = message rate (in Hz) of 20, 10, 2, 1, 0, or .2 (0 turns off the message)
',OTHER' = optional field, enacts a change on the current port when you send the command without it (and the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPGNS,HHMMSS.SS,DDMM.MMMMM,K,DDDMM.MMMMM,L,MM,QQ,H.H,A.A,G.G,D.D,R.R,NS*CC<C

where:

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the position
DDMM.MMMMM	Latitude in degrees, minutes, and decimal minutes (you can set the number of decimal places using the JNP command)
K	Latitude indicator; value is N (North latitude) or S (South latitude)
DDDMM.MMMMM	Longitude in degrees, minutes, and decimal minutes (you can set the number of decimal places using the JNP command)
L	Longitude indicator; value is E (East longitude) or W (West longitude)
MM	Mode indicator Variable length valid character field type with the first two characters currently defined. First character indicates the use of GPS satellites Second character indicates the use of GLONASS satellites. If another satellite system is added to the standard, the mode indicator will be extended to three characters. New satellite systems shall always be added on the right, so the order of characters in the Mode Indicator is: GPS, GLONASS, other satellite systems in the future. The characters shall take one of the following values: N = No fix. Satellite system not used in position fix, or fix not valid A = Autonomous. Satellite system used in non-differential mode in position fix D = Differential. Satellite system used in differential mode in position fix

		P = Precise. Satellite system used in precision mode. Precision mode is defined as no deliberate degradation (such as Selective Availability) and higher resolution code (P-code) is used to compute position fix. R = Real Time Kinematic. Satellite system used in RTK mode with fixed integers F = Float RTK. Satellite system used in real time kinematic mode with floating integers E = Estimated (dead reckoning) mode M = Manual input mode S = Simulator mode The mode indicator shall not be a null field.
	QQ	Number of satellites used in position solution
	P.P	Horizontal dilution of precision (HDOP)
	A.A	Antenna altitude, in meters, re: mean-sea-level (geoid)
	G.G	Geoidal separation (in meters)
	SSS	Age of differential corrections, in seconds
	RRRR	Differential reference station ID
	NS	Navigational status; options are: S = Safe C = Caution U = Unsafe V = Not valid for navigation
	*CC	Checksum
	<CR>	Carriage return
	<LF>	Line feed
Example:	$GPGNS,224749.00,3333.4268304,N,11153.3538273,W,D,19,0.6,406.110,-26.294,6.0,0138,S,*6A	
Additional Information:	This message provides information specific to the satellite system identified by the first two characters of the message. GPGNS - GPS information GNGNS - GNSS information GLGNS - GLONASS information GAGNS – GALILEO information The JNMEA,GGAALLGNSS command significantly affects the output of the GNS message. If you are tracking more than GNSS sign Hemisphere GNSS highly recommends that you review this command. The JNMEA,GGAALLGNSS command significantly affects the output of the GLL message. If you are tracking more than GNSS signals, Hemisphere GNSS highly recommends that you review this command.	
Related Commands and Messages:	JASC,GP, JASC,GN, JASC,GL, JNMEA,GGAALLGNSS	

GPGRS Message

<table>
<tr><td>Message Type:</td><td>Data</td></tr>
<tr><td>Description:</td><td>Supports Receiver Autonomous Integrity Monitoring (RAIM)</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message

$JASC,GPGRS,r[,OTHER]<CR><LF>

where:

'r' = message rate in Hz of 1, 0, or .2 (0 turns off the message)
□□ ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPGRS,HHMMSS.SS,M,X.X ... X.X,GSID,SID*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>HHMMSS.SS</td><td>UTC time</td></tr>
<tr><td>M</td><td>Mode:

0 = residuals used to calculate the position given in the GPGGA or GPGNS message

1 = residuals were recomputed after the GPGGA or GPGNS message position was computed</td></tr>
<tr><td>X.X ... X.X</td><td>Range residuals, in meters, for satellites used in the navigation solution. Order must match order of satellite ID numbers in GPGSA message. When GPGRS message is used, the GPGSA and GPGSV messages are generally required with this message.</td></tr>
<tr><td>GSID</td><td>GNSS system ID, value is 1 (GPS)</td></tr>
<tr><td>SID</td><td>Signal ID, value is 1 (L1 C/A)</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.04 / May 29, 2012

GNGSA Message

<table>
<tr><td>Message Type:</td><td>Data</td></tr>
<tr><td>Description:</td><td>DOP and active satellite information
Only satellites used in the position computation are present in this message. Null fields are present when data is unavailable due to the number of satellites tracked.</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JASC,GNGSA,r[,OTHER]<CR><LF>

where:

'r' = message rate in Hz of 1 or 0 (0 turns off the message)
',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GNGSA,A,B,CC ... NN,P.P,Q.Q,R.R,GSID*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>A</td><td>Satellite acquisition mode (M = manually forced to 2D or 3D, A = automatic swap between 2D and 3D)</td></tr>
<tr><td>B</td><td>Position mode (1 = fix not available, 2 = 2D fix, 3 = 3D fix)</td></tr>
<tr><td>CC to NN</td><td>Satellites used in the position solution; a null field occurs if a channel is unused</td></tr>
<tr><td>P.P</td><td>Position Dilution of Precision (PDOP) = 1.0 to 9.9</td></tr>
<tr><td>Q.Q</td><td>Horizontal Dilution of Precision (HDOP) 1.0 to 9.9</td></tr>
<tr><td>R.R</td><td>Vertical Dilution of Precision (VDOP) = 1.0 to 9.9</td></tr>
<tr><td>GSID</td><td>GNSS system ID, value is 1 (GPS), 2 (GLONASS), 3 (GALILEO), 5 (BEIDOU)</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td>This message provides information specific to the satellite system(s) identified by the first two characters of the message.

GNGSA - GNSS
information (all constellations) GPGSA - GPS
information GLGSA - GLONASS information</td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP, JASC,GN, JASC,GL</td></tr>
</table>

Topic Last Updated: v1.07 / February 16, 2017

GPGST Message

<table>
<tr><td>Message Type:</td><td>Data</td></tr>
<tr><td>Description:</td><td>GNSS pseudorange error statistics and position accuracy</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

Message Format:
$JASC,GPGST,r[,OTHER]<CR><LF>

where:
'r' = message rate in Hz of 1 or 0 (0 turns off the message)
',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

$GPGST,HHMMSS.SS,A.A,B.B,C.C,D.D,E.E,F.F,G.G*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>HHMMSS.SS</td><td>UTC time in hours, minutes, and seconds of the GPS position</td></tr>
<tr><td>A.A</td><td>Root mean square (rms) value of the standard deviation of the range inputs to the navigation process. Range inputs include pseudoranges and differential GNSS (DGNSS) corrections.</td></tr>
<tr><td>B.B</td><td>Standard deviation of semi-major axis of error ellipse, in meters</td></tr>
<tr><td>C.C</td><td>Standard deviation of semi-minor axis of error ellipse, in meters</td></tr>
<tr><td>D.D</td><td>Error in Eclipse’s semi major axis origination, in decimal degrees, true north</td></tr>
<tr><td>E.E</td><td>Standard deviation of latitude error, in meters</td></tr>
<tr><td>F.F</td><td>Standard deviation of longitude error, in meters</td></tr>
<tr><td>G.G</td><td>Standard deviation of altitude error, in meters</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.01 / September 23, 2010

GPGSV Message

<table>
<tr><td>Message Type:</td><td>Data</td></tr>
<tr><td>Description:</td><td>GPS satellites in view

Null fields occur where data is unavailable due to the number of satellites tracked.</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JASC,GPGSV,r[,OTHER]<CR><LF>

where:

'r' = message rate in Hz of 05, .1, .5, .2, 1, 2, 5, 10, 20.
',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPGSV,T,M,N,II,EE,AAA,SS,…II,EE,AAA,SS,SID*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>T</td><td>Total number of messages</td></tr>
<tr><td>M</td><td>Message number (1 to 3)</td></tr>
<tr><td>N</td><td>Total number of satellites in view</td></tr>
<tr><td>II</td><td>Satellite number</td></tr>
<tr><td>EE</td><td>Elevation, in degrees (0 to 90)</td></tr>
<tr><td>AAA</td><td>Azimuth (true), in degrees (0 to 359)</td></tr>
<tr><td>SS</td><td>Signal strength, in dB-Hz (0 - 99)

To compare with SNR values found in Bin messages (such as Bin96) subtract 30 from this signal strength value for an approximate SNR value

SS - 30 = SNR (from Bin message)</td></tr>
<tr><td>SID</td><td>Signal ID, value is 1 (L1 C/A)</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td>This message provides information specific to the satellite system identified by the first two characters of the message.
GPGSV – GPS information
GLGSV – GLONASS information</td></tr>
</table>

	GBGSV-BeiDou information GAGSV – GALILEO information GQGSV – QZSS information If you request GNGSV the receiver will respond with GPGSV messages only.
Related Commands and Messages:	JASC,GP, JASC,GL, BEIDOU

Topic Last Updated: v1.11 / November 15, 2018

GAGSV Message

Message Type:	Data
Description:	Galileo satellites in view Null fields occur where data is unavailable due to the number of satellites tracked.
Message Format:	Command Format to Request Message: $JASC,GAGSV,r[,OTHER]<CR><LF> where: 'r' = message rate in Hz of ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets)and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $GAGSV,T,M,N,II,EE,AAA,SS,…II,EE,AAA,SS,SID*CC<CR><LF>

Message Component	Description
T	Total number of messages
M	Message number (1 to 3)
N	Total number of satellites in view
II	Satellite number
EE	Elevation, in degrees (0 to 90)
AAA	Azimuth (true), in degrees (0 to 359)
SS	Signal strength, in dB-Hz (0 - 99) To compare with SNR values found in Bin messages (such as Bin96) subtract 30 from this signal strength value for an approximate SNR value SS - 30 = SNR (from Bin message)
SID	Signal ID, value is 1 (L1 C/A)
*CC	Checksum

<CR>	Carriage return
<LF>	Line feed

Example:	
Additional Information:	This message provides information specific to the satellite system identified by the first two characters of the message. GPGSV – GPS information GLGSV – GLONASS information GBGSV- BeiDou information GAGSV – GALILEO information GQGSV – QZSS information If you request GNGSV the receiver will respond with GPGSV messages only.
Related Commands and Messages:	JASC,GP, JASC,GL, BEIDOU

Topic Last Updated: v1.11 / November 15, 2018

GBGSV Message

Message Type:	Data
Description:	BeiDou satellites in view Null fields occur where data is unavailable due to the number of satellites tracked.
Message Format:	Command Format to Request Message: $JASC,GBGSV,r[,OTHER]<CR><LF> where: 'r' = message rate in Hz of 05, .1, .5, .2, 1, 2, 5, 10, 20• ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $GBGSV,T,M,N,II,EE,AAA,SS,…II,EE,AAA,SS,SID*CC<CR><LF> where:

Message Component	Description
T	Total number of messages
M	Message number (1 to 3)
N	Total number of satellites in view

	<table><tr><td>II</td><td>Satellite number</td></tr><tr><td>EE</td><td>Elevation, in degrees (0 to 90)</td></tr><tr><td>AAA</td><td>Azimuth (true), in degrees (0 to 359)</td></tr><tr><td>SS</td><td>Signal strength, in dB-Hz (0 - 99) To compare with SNR values found in Bin messages (such as Bin96) subtract 30 from this signal strength value for an approximate SNR value SS - 30 = SNR (from Bin message)</td></tr><tr><td>SID</td><td>Signal ID, value is 1 (L1 C/A)</td></tr><tr><td>*CC</td><td>Checksum</td></tr><tr><td><CR></td><td>Carriage return</td></tr><tr><td><LF></td><td>Line feed</td></tr></table>
Example:	
Additional Information:	This message provides information specific to the satellite system identified by the first two characters of the message. GPGSV – GPS information GLGSV – GLONASS information GBGSV- BeiDou information GAGSV – GALILEO information GQGSV – QZSS information If you request GNGSV the receiver will respond with GPGSV messages only.
Related Commands and Messages:	JASC,GP, JASC,GL, BEIDOU

Topic Last Updated: v1.11 / November 15, 2018

GPHDG/HEHDG Message

Message Type:	Data
Description:	Magnetic deviation and variation for calculating magnetic or true heading. The message simulates data from a magnetic sensor although it does not actually contain one. The purpose of this message is to support older systems that may not be able to accept the HDT message that is recommended for use.
Message Format:	Command Format to Request Message: $JASC,GPHDG,r[,OTHER]<CR><LF> where:

<table>
<tr><td></td><td>'r' = message rate in Hz of 20, 10, 2, 1, 0 or .2 (0 turns off the message)

',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPHDG,s.s,d.d,D,v.v,V*CC<CR><LF>

or

$HEHDG,s.s,d.d,D,v.v,V*CC<CR><LF>

where:

<table><tr><th>Message Component</th><th>Description</th></tr><tr><td>s.s</td><td>Magnetic sensor reading, in degrees</td></tr><tr><td>d.d</td><td>Magnetic deviation, in degrees</td></tr><tr><td>D</td><td>E = Easterly deviation, W = Westerly deviation</td></tr><tr><td>v.v</td><td>Magnetic variation, in degrees</td></tr><tr><td>V</td><td>E = Easterly deviation, W = Westerly deviation</td></tr><tr><td>*CC</td><td>Checksum</td></tr><tr><td><CR></td><td>Carriage return</td></tr><tr><td><LF></td><td>Line feed</td></tr></table></td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td>You can change the HDG message header to either GP or HE using the JATT,NMEAHE command.</td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.00 / August 11, 2010

GPHEB Message

Message Type:	Data
Description:	Heave value in meters
Message Format:	Command Format to Request Message: $JASC,GPHEV,1<CR><LF> Message Format: $GPHEV,H,*CC<CR><LF> where:

	<table><tr><th>Message Component</th><th>Description</th></tr><tr><td>H</td><td>Heave value, in meters</td></tr><tr><td>*CC</td><td>Checksum</td></tr><tr><td><CR></td><td>Carriage return</td></tr><tr><td><LF></td><td>Line feed</td></tr></table>
Example:	
Additional Information:	You can change the HDG message header to either GP or HE using the JATT,NMEAHE command.
Related Commands and Messages:	JASC,GP

Topic Last Updated: v1.00 / August 11, 2010

GPRMC Message

Message Type:	Data
Description:	Contains recommended minimum specific GNSS data
Message Format:	Command Format to Request Message: $JASC,GPRMC,r[,OTHER]<CR><LF> where: 'r' = message rate in Hz of 10, 2, 1, 0, or .2 (0 turns off the message) ',OTHER' = optional field,enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $GPRMC,HHMMSS.SS,A,DDMM.MMM,N,DDDMM.MMM,W,Z.Z,Y.Y,DDMMYY,D.D,V,M,NS*CC<CR><LF where: <table><tr><th>Message Component</th><th>Description</th></tr><tr><td>HHMMSS.SS</td><td>UTC time in hours, minutes, and seconds of the GPS position</td></tr><tr><td>A</td><td>Status (A = valid, V = invalid)</td></tr><tr><td>DDMM.MMM</td><td>Latitude in degrees, minutes, and decimal minutes</td></tr><tr><td>N</td><td>Latitude location (N = North latitude, S = South latitude)</td></tr><tr><td>DDDMM.MMM</td><td>Longitude in degrees, minutes, and decimal minutes</td></tr><tr><td>W</td><td>Longitude location (E = East longitude, W = West longitude)</td></tr><tr><td>Z.Z</td><td>Ground speed, in knots</td></tr></table>

Y.Y	Track made good, reference to true north
DDMMYY	UTC date of position fix in day, month, and year
D.D	Magnetic Variation, in degrees
V	Variation sense (E = East, W = West)
M	Mode indicator Variable length valid character field type with the first two characters currently defined. First character indicates the use of GPS satellites If another satellite system is added to the standard, the mode indicator will be extended to three characters. New satellite systems shall always be added on the right, so the order of characters in the Mode Indicator is: GPS, GLONASS, other satellite systems in the future. The characters shall take one of the following values: N = No fix. Satellite system not used in position fix, or fix not valid A = Autonomous. Satellite system used in non-differential mode in positionfix D = Differential. Satellite system used in differential mode in position fix P = Precise. Satellite system used in precision mode. Precision mode is defined as no deliberate degradation (such as Selective Availability) and higher resolution code (P-code) is used to compute position fix. R = Real Time Kinematic. Satellite system used in RTK mode with fixed integers F = Float RTK. Satellite system used in real time kinematic mode with floating integers E = Estimated (dead reckoning) mode M = Manual input mode S = Simulator mode The mode indicator shall not be a null field.
NS	Navigational status; options are: S = Safe C = Caution U = Unsafe V = Not valid for navigation
*CC	Checksum

	<CR>	Carriage return
	<LF>	Line feed
Example:		
Additional Information:		
Related Commands and Messages:	JASC,GP	

Topic Last Updated: v1.04 / May 29, 2012

GPHDM/HEHDM Message

Message Type:	Data
Description:	Magnetic heading of the vessel derived from the true heading calculated
Message Format:	Command Format to Request Message: $JASC,GPHDM,r[,OTHER]<CR><LF> where: 'r' = message rate in Hz of 20, 10, 2, 1, 0 or .2 (0 turns off the message) ',OTHER' = optional field, enacts a change on the currentport when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $GPHDM,X.X,M*CC<CR><LF> or $HCHDM,X.X,M*CC<CR><LF> where:

Message Component	Description
X.X	Current heading, in degrees
T	Indicates true heading
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Example:	
Additional Information:	You can change the HDM message header to either GP or HE using the JATT,NMEAHE command.
Related Commands and Messages:	JASC,GP

Topic Last Updated: v1.02 / January 25, 2011

GPROT/HEROT Message

<table>
<tr><td>Message Type:</td><td>Data</td></tr>
<tr><td>Description:</td><td>Vessel’s rate of turn (ROT) information</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JASC,GPROT,r[,OTHER]<CR><LF>

where:

'r' = messagerate in Hz of 20, 10, 2, 1, 0 or .2 (0 turns off the message)

',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPROT,X.X,A*CC<CR><LF>

or

$HEROT,X.X,A*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>X.X</td><td>Rate of turn in °/min (negative when the vessel bow turns to port)</td></tr>
<tr><td>A</td><td>Flag indicating the data is valid</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td>You can change the ROT message header to either GP or HE using the JATT,NMEAHE command.</td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.00 / August 11, 2010

GPRRE Message

<table>
<tr><td>Message Type:</td><td>Data</td></tr>
<tr><td>Description:</td><td>Satellite range residuals and estimated position error</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JASC,GPRRE,r[,OTHER]<CR><LF></td></tr>
</table>

<table>
<tr><td></td><td>where:

'r' = message rate in Hz of 1 or 0 (0 turns off the message)

',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPRRE,N,II,RR ... II,RR,HHH.H,VVV.V*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>N</td><td>Number of satellites used in position computation</td></tr>
<tr><td>II</td><td>Satellite number</td></tr>
<tr><td>RR</td><td>Range residual, in meters</td></tr>
<tr><td>HHH.H</td><td>Horizontal position error estimate, in meters</td></tr>
<tr><td>VVV.V</td><td>Vertical position error estimate, in meters</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td>You can change the ROT message header to either GP or HE using the JATT,NMEAHE command.</td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.00 / August 11, 2010

GPVTG Message

Message Type:	Data
Description:	Course over ground and ground speed
Message Format:	Command Format to Request Message: $JASC,GPVTG,r[,OTHER]<CR><LF> where: 'r' = message rate in Hz of 20, 10, 2, 1, 0, or .2 (0 turns off the message) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

<table>
<tr><td></td><td>Message Format:

$GPVTG,TTT,T,MMM,M,NNN.NN,N,KKK.KK,K,X*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>TTT</td><td>True course over ground (COG) in degrees (000 to 359)</td></tr>
<tr><td>T</td><td>True course over ground indicator (always 'T')</td></tr>
<tr><td>MMM</td><td>Magnetic course over ground in degrees (000 to 359)</td></tr>
<tr><td>M</td><td>Magnetic course over ground indicator (always 'M')</td></tr>
<tr><td>NNN.NN</td><td>Speed over ground in knots</td></tr>
<tr><td>N</td><td>Speed over ground in knots indicator (always 'N')</td></tr>
<tr><td>KKK.KK</td><td>Speed over ground in km/h</td></tr>
<tr><td>K</td><td>Speed over ground in km/h indicator (always 'K')</td></tr>
<tr><td>X</td><td>Mode

A = Autonomous mode
D = Differential mode
E = Estimated (dead reckoning) mode
M = Manual input mode
S = Simulator mode
N = Data not valid</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td>Sample message output:

$GPVTG,103.85,T,92.79,M,0.14,N,0.25,K,D*1E</td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.00 / August 11, 2010

GPZDA Message

Message Type:	Data
Description:	UTC time and date information
Message Format:	Command Format to Request Message: $JASC,GPZDA,r[,OTHER]<CR><LF> where:

<table>
<tr><td></td><td>'r' = message rate in Hz of 20, 10, 2, 1, 0, or .2 (0 turns off the message)

',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GPZDA,HHMMSS.SS,DD,MM,YYYY,XX,YY*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>HHMMSS.SS</td><td>UTC time in hours, minutes, and seconds of the GPS unit</td></tr>
<tr><td>DD</td><td>Day (0 to 31)</td></tr>
<tr><td>MM</td><td>Month (1 to 12)</td></tr>
<tr><td>YYYY</td><td>Year</td></tr>
<tr><td>XX</td><td>Local zone description in hours (-13 to 13)</td></tr>
<tr><td>YY</td><td>Local zone description in minutes (0 to 59)</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
<tr><td>HHMMSS.SS</td><td>UTC time in hours, minutes, and seconds of the GPS unit</td></tr>
<tr><td>DD</td><td>Day (0 to 31)</td></tr>
<tr><td>MM</td><td>Month (1 to 12)</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.00 / August 11, 2010

PASHR Message

Message Type:	Vector, Data
Description:	Time, true heading, roll, pitch, and heave data in one message
Message Format:	Command Format to Request Message: $JASC,PASHR,r[,OTHER]<CR><LF> where: 'r' = message rate (in Hz) of 20, 10, 5, 4, 2, 1, 0, or .2 (0 turns off the message)

<table>
<tr><td></td><td>',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.

Message Format:

$PASHR,hhmmss.ss,HHH.HH,T,RRR.RR,PPP.PP,heave,rr.rrr,pp.ppp,hh.hhh,QF*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>hhmmss.ss</td><td>UTC time</td></tr>
<tr><td>HHH.HH</td><td>Heading value in decimal degrees</td></tr>
<tr><td>T</td><td>True heading (T displayed if heading is relative to true north)</td></tr>
<tr><td>RRR.RR</td><td>Roll in decimal degrees (- sign will be displayed when applicable)</td></tr>
<tr><td>PPP.PP</td><td>Pitch in decimal degrees (- sign will be displayed when applicable)</td></tr>
<tr><td>heave</td><td>Heave, in meters</td></tr>
<tr><td>rr.rrr</td><td>Roll standard deviation in decimal degrees</td></tr>
<tr><td>pp.ppp</td><td>Pitch standard deviation in decimal degrees</td></tr>
<tr><td>hh.hhh</td><td>Heading standard deviation in decimal degrees</td></tr>
<tr><td>QF</td><td>Quality Flag

0 = No position
1 = All non-RTK fixed integer positions
2 = RTK fixed integer position</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,PASHR</td></tr>
</table>

Topic Last Updated: v1.05 / January 18, 2013

HGNSS Proprietary Messages

PSAT, ATTSTAT Message

<table>
<tr><td>Message Type:</td><td>Data</td></tr>
<tr><td>Description:</td><td>Heading diagnostic information</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JASC,PSAT,ATTSTAT,r[,OTHER]<CR><LF>

where:

•'r' = message rate in Hz of 1 or 0 (0 turns off the message)
•',OTHER' = optional field, enacts a change on the current port when you send the command without it (and brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$PSAT,ATTSTAT,S,MSEP,CSEP,Heading,TYPE,Pitch,Roll,Q,N,SYS,NUMTRACKED,SNR,NUMUSED,*CC

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>S</td><td>ID of the secondary antenna</td></tr>
<tr><td>MSEP</td><td>custom separation between antennas manually entered (when the value is MOV, it means MOVEBASE is on)</td></tr>
<tr><td>CSEP</td><td>auto GPS antenna separation</td></tr>
<tr><td>Heading</td><td>Heading</td></tr>
<tr><td>TYPE</td><td>Heading indicator, value is: N= Heading used GNSS G=Heading used gyroscope</td></tr>
<tr><td>Pitch</td><td>pitch</td></tr>
<tr><td>Roll</td><td>roll</td></tr>
<tr><td>Q</td><td>The current setting of antenna directivity, value is P= antennas placed front and back, output pitch R= antennas placed left and right, output roll</td></tr>
<tr><td>N</td><td>The number of satellites used by the secondary antenna</td></tr>
<tr><td>SYS</td><td>Systems in use:

GPS: L1, L2, L5

GLONASS: G1, G2

BDS: B1,B2 B3
Galileo: E5a, E5b, E5a+b, E6</td></tr>
</table>
</td></tr>
</table>

		NUMTRACKED	Number of satellites tracked for each system
		SNR	Quality of each SNR path, where: A is > 20 dB B is > 18 dB C is > 15 db D is <= 15 dB
		NUMUSED	Number of satellites used by each system
		*CC	Checksum
		<CR>	Carriage return
		<LF>	Line feed
Example:	$PSAT,ATTSTAT,1,MOV,0.504,334.75,N,1.71,8.0,P,30,(,L1,L2,G1,G2,B1,B2,B3,)(,12,10,9,9,10,10,0,)(, A,A,C,B,B,B,D,)(,12,10,8,8,9,9,0)*22		
Additional Information:	Issuing the JSAVE command after setting JASC,PSAT,ATTSTAT to 1 (message on at 1Hz) does not save this setting. You must JASC,PSAT,ATTSTAT (set it to 1) each time you power on the receiver.		
Related Commands and Messages:	JASC,PSAT,ATTSTAT		

Topic Last Updated: v5.0/June 30, 2020

PSAT, GBS Message

Message Type:	Data
Description:	Used to support Receiver Autonomous Integrity Monitoring (RAIM)
Message Format:	Command Format to Request Message: $JASC,GPGBS,r[,OTHER]<CR><LF> where: 'r' = message rate in Hz of 1 or 0 (0 turns off the message) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $PSAT,GBS,HHMMSS.SS,KK.K,LL.L,AA.A,ID,P.PPPPP,B.B,S.S,FLAG,GSID,SID*CC<CR><LF where:

	<table><tr><th>Message Component</th><th>Description</th></tr><tr><td>HHMMSS.SS</td><td>UTC time in hours, minutes, and seconds of the GGA or GNS fix associated with this sentence</td></tr><tr><td>KK.K</td><td>Expected error in latitude</td></tr><tr><td>LL.L</td><td>Expected error in longitude</td></tr><tr><td>AA.A</td><td>Expected error in altitude</td></tr><tr><td>ID</td><td>ID number of most likely failed satellite</td></tr><tr><td>P.PPPPP</td><td>Probability of HPR fault</td></tr><tr><td>B.B</td><td>Estimate of range bias, in meters, on most likely failed satellite</td></tr><tr><td>S.S</td><td>Standard deviation of range bias estimate</td></tr><tr><td>FLAG</td><td>Based on horizontal radius: 0 = Good 1 = Warning 2 = Bad or Fault</td></tr><tr><td>GSID</td><td>GNSS system ID, value is 1 (GPS)</td></tr><tr><td>SID</td><td>Signal ID, value is 1 (L1 C/A)</td></tr><tr><td>*CC</td><td> Checksum</td></tr><tr><td><CR></td><td>Carriage return</td></tr><tr><td><LF></td><td>Line feed</td></tr></table>
Example:	
Additional Information:	
Related Commands and Messages:	JASC,GP

Topic Last Updated: v1.04 / May 29, 2012

PSAT, HPR Message

Message Type:	Data
Description:	Proprietary NMEA message that provides the true heading, pitch, roll, and time in a single message This message provides heading, pitch and roll. Heading is derived from GNSS. If $JATT,ROLL,YES is set roll will be derived from GNSS and pitch will come from the inertial sensor. If $JATT,ROLL,NO is set, pitch will be derived from GNSS and roll will come from the inertial sensor. While coasting heading is based on gyro and pitch/roll are from the inertial sensor. To know when the receiver is coasting, see the TYPE field below.
Message Format:	Command Format to Request Message:

<table>
<tr><td></td><td>$JASC,GPHPR,r[,OTHER]<CR><LF>

where:

'r' = message rate in Hz of 20, 10, 2, 1, 0 or .2 (0 turns off the message)',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$PSAT,HPR,TIME,HEADING,PITCH,ROLL,TYPE*CC<CR><LF>
where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>TIME</td><td>UTC time (HHMMSS.SS)</td></tr>
<tr><td>HEADING</td><td>Heading (degrees)</td></tr>
<tr><td>PITCH</td><td>Pitch (degrees)</td></tr>
<tr><td>ROLL</td><td>Roll (degrees)</td></tr>
<tr><td>TYPE</td><td>N = GNSSS derived heading G = gyro heading</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP
$JATT,ROLL</td></tr>
</table>

Topic Last Updated: v4.0 / June 30, 2020

PSAT, INTLT Message

Message Type:	Data
Description:	Proprietary NMEA message that provides the tilt measurements from the internal inclinometers in degrees. It delivers an output of crude accelerometer measurements of pitch and roll with no temperature compensation or calibration for GPS heading/pitch/roll. Pitch and roll are factory calibrated over temperature to be accurate to ±3°C. CAUTION: User calibration will clear out precise factory calibration.
Message Format:	Command Format to Request Message: $JASC,INTLT,r[,OTHER]<CR><LF> where:

<table>
<tr><td></td><td>•'r' = message rate in Hz of 1 or 0 (0 turns off the message)
• ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$PSAT,INTLT,PITCH,ROLL*CC<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>PITCH</td><td>Pitch (degrees)</td></tr>
<tr><td>ROLL</td><td>Roll (degrees)</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,GP</td></tr>
</table>

Topic Last Updated: v1.00 / August 11, 2010

PSAT, BLV Message

Message Type:	Data, Local Differential and RTK
Description:	Contains RTK fix progress information
Message Format:	Command Format to Request Message: $JASC,PSAT,BLV,r[,OTHER]<CR><LF> where: •'r' = message rate in Hz of 1 or 0 (0 turns off the message) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $PSAT,BLV,HHMMSS.SS,DATE,A.A,B.B,C.C,ID,STATE,number,pdop*CC<CR><L F> where:

	<table><tr><th>Message Component</th><th>Description</th></tr><tr><td>HHMMSS.SS</td><td>UTC time (HHMMSS.SS)</td></tr><tr><td>DATE</td><td>Date (day-month-year)</td></tr><tr><td>A.A</td><td>North component of base to rover vector (m)</td></tr><tr><td>B.B</td><td>East component of base to rover vector (m)</td></tr><tr><td>C.C</td><td>Up component of base to rover vector (m)</td></tr><tr><td>ID</td><td>Base station ID</td></tr><tr><td>STATE</td><td>Quality indicator; value is: 0 = no position 1 = not differentially corrected position (autonomous) 2 = differentially corrected position (SBAS, DGPS, Atlas DGPS service, L-Dif and e-Dif) 4 = RTK fixed integer ,Atlas high precision services converged 5 = RTK float, Atlas high precision services converging</td></tr><tr><td>NUMBER</td><td>Number of used satellite</td></tr><tr><td>PDOP</td><td>PDOP</td></tr><tr><td>*CC</td><td>Checksum</td></tr><tr><td><CR></td><td>Carriage return</td></tr><tr><td><LF></td><td>Line feed</td></tr></table>
Example:	$PSAT,BLV,000151.00,051115,-0.001,0.002,-0.003,0333,4,20,1.2*52
Additional Information:	
Related Commands and Messages:	JASC, PSAT, BLV

Topic Last Updated: v4.0/ June 30, 2020

PSAT, FVI Message

Message Type:	Data, Local Differential and RTK
Description:	Position, attitude and standard deviations.
Message Format:	Command Format to Request Message: $JASC,PSAT,FVI,r[,OTHER]<CR><LF> where: •'r' = message rate in Hz of 0,1,2,5,10,20 (0 turns off the message) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port

when you send the command with it (without the brackets)

Message Format:

$PSAT,FVI,HHMMSS.SS, DD.ddddddddd, DDD.ddddddddd, AA.AAA,

E.E,F.F,G.G,HHH.HHH,hh.hhh,PP.PP,pp.ppp,RR.RRR,rr.rrr,ve.eee,v
n.nnn,vu.uuu,vv.vvv,LE.EEE,LN.NNN,LU.UUU,ZONE,UEEE.EEEE,UNNN.N
NNN,PN,SN,p,h,L,sss*CC<CR><LF>

where:

Message Component	**Description**
HHMMSS.SS	UTC time
DD.ddddddddd	Latitude in degrees and decimal minutes (+ is North)
DDD.ddddddddd	Longitude in degrees, and decimal minutes (+ is East)
AA.AAA	altitude
E.E	Standard deviation of latitude error, in meters
F.F	Standard deviation of longitude error, in meters
G.G	Standard deviation of altitude error, in meters
HHH.HHH	Heading (degrees)
hh.hhh.	Standard deviation of heading error, in degrees
PP.PP	Pitch (degrees)
pp.ppp	Standard deviation of pitch error, in degrees
RR.RRR	Roll (degrees)
rr.rrr	Standard deviation of roll error, in degrees
Ve.eee	East to speed（m/s）
Vn.nnn	North to speed (m/s)
Vu.uuu	Vertical speed (m/s)
Vv.vvv	Speed over ground (m/s)
LE.EEE	East component of master to slave vector (m)
LN.NNN	North component of master to slave vector (m)
Vv.vvv	Speed over ground (m/s)
LE.EEE	East component of master to slave vector (m)
LN.NNN	North component of master to slave vector (m)
LU.UUU	Up component of master to slave vector (m)
ZONE	projection area
UEEE.EEEE	East to position of projection area
UNNN.NNNN	North to position of projection area

PN	Number of satellites used by the primary antenna
SN	Number of satellites used by the secondary antenna
P	Position indicator; value is: 0 = no position 1 = not differentially corrected position (autonomous) 2 = differentially corrected position (SBAS, DGPS ,Atlas DGPS service, L-Dif and e -Dif) 4 = RTK fixed integer Atlas high precision services converged 5 = RTK float, Atlas high precision services converging
H	Heading indicator; value is: 0 = no heading or heading is invalid 1 = heading is valid
L	Distance between base and rover in meter
SSS	Age of differential corrections, in seconds
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Example:	$PSAT,FVI,011657.00,40.071345258,116.326680384,51.2922,0.001,0.003,0.003,28.358,0.106,-5.306,0.087,,,0.030,- 0.001,-0.062,0.030,-0.001,0.001,-0.002,117.0,442562.296,4437668.138,25,26,4,1,4.759,1*6B
Additional Information:	
Related Commands and Messages:	JASC,PSAT,FVI Bin3

Topic Last Updated: v4.2/ September 13, 2022

PSAT, RPTKPROG Message

Message Type:	Data, Local Differential and RTK
Description:	Contains RTK fix progress information.
Message Format:	Command Format to Request Message: $JASC,PSAT,RTKPROG,r[,OTHER]<CR><LF> where: 'r' = message rate in Hz of 1 or 0 (0 turns off the message) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $PSAT,RTKPROG,,R,F,N,SS1,SS2,SS3,MASK*CC<CR><LF>

<table>
<tr><td></td><td>where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>R</td><td>1 = Ready to enter RTK ambiguity fix

0 = Not ready to enter RTK ambiguity fix</td></tr>
<tr><td>F</td><td>1 = Receiver running in RTK ambiguity fix mode

0 = Receiver not running in RTK ambiguity fix mode</td></tr>
<tr><td>N</td><td>Number of satellites used to fix</td></tr>
<tr><td>SS1</td><td>summer-1

SS1 must be significantly larger than SS2 and SS3 to enter R=1 mode</td></tr>
<tr><td>SS2</td><td>summer-2</td></tr>
<tr><td>SS3</td><td>summer-3</td></tr>
<tr><td>MASK</td><td>Bit mask; bits identify which GNSS observables are being received from base recently (1 = GPS, 3 = GPS + GLONASS)</td></tr>
<tr><td>*CC</td><td>Checksum</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table></td></tr>
<tr><td>Example:</td><td></td></tr>
<tr><td>Additional Information:</td><td>Issuing the JSAVE command after setting JASC,PSAT,RTKPROG to 1 (message on at 1Hz) does not save this setting. You must enable JASC,PSAT,RTKPROG (set it to 1) each time you power on the receiver.</td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,PSAT,RTKPROG</td></tr>
</table>

Topic Last Updated: v1.04 / May 29, 2012

PSAT, RTKSTAT Message

Message Type:	Data, Local Differential and RTK
Description:	Contains the most relevant parameters affecting RTK.
Message Format:	Command Format to Request Message: $JASC,PSAT,RTKSTAT,r[,OTHER]<CR><LF> where: •'r' = message rate in Hz of 1 or 0 (0 turns off the message) •',OTHER' = optional field, enacts a change on the current port when you send the command without it (and brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format:

$PSAT,RTKSTAT,MODE,TYP,AGE,SUBOPT,DIST,SYS,NUM,SNR,RSF,BSF,HAG,ACCSTAT,SNT*CC

where:

Message Component	Description
MODE	Mode (FIX,FLT,DIF,AUT,NO)
TYP	Correction type (DFX,ROX,CMR,RTCM3,CMR+,...)
AGE	Age of differential corrections, in seconds
SUBOPT	Subscription code (see Interpreting the $JK 'Date'/Subscription Codes to determine the meaning of the subscription code)
DIST	Distance to base in kilometers
SYS	Systems in use: GPS: L1, L2, L2C, L5 GLONASS: G1, G2 BDS: B1,B2, B3, B1C, B2A,B2B,B2AB Galileo: E1B, E5A, E5B, E5AB E6 QZSS: QL1, QL2, QL5
NUM	Number of satellites used by each system
SNR	Quality of each SNR path, where: • A is > 20 dB • B is > 18 dB • C is > 15 db • D is <= 15 dB
RSF	Rover slip flag (non zero if parity errors in last 5 minutes, good for detecting jamming and TCXO issues)
BSF	Base slip flag
HAE	Horizontal accuracy estimation
ACCSTAT	RTK accuracy status (hex), where: • 0x1 = no differential or differential too old, for the application • 0x2 = problems with differential message • 0x4 = horizontal position estimate poor for the application • 0x8 = HDOP high, poor satellite geometry • 0x10 = fewer than 6 L1 sats used • 0x20 = poor L1 SNRs • 0x40 = not in RTK mode

			• 0x80 = not in RTK mode or RTK only recently solved (< 10 secs ago) • 0x100 = RTK solution compromised, may fail • The status message can be any of the above or any combination of the above. For example, a status message of '047' indicates the following: • 0x1 = no differential or differential too old, for the application • 0x2 = problems with differential message • 0x4 = horizontal position estimate poor for the application • 0x40 = not in RTK mode
		SNT	Ionospheric scintillation, values are: • 0 (little or no scintillation - does not adversely affect RTK solution) • 1-100 (scintillation detected - adversely affects RTK solution)
		*CC	Checksum
		<CR>	Carriage return
		<LF>	Line feed
Example:	$PSAT,RTKSTAT,FIX,ROX,1,007F,9.5,(,L1,L2,G1,G2,)(,14,11,9,9,)(,A,A,A,A,),0,1,0.0 11,000 Fixed mode • ROX corrections • Diff age = 1 second • Subscribed options = 7F (see Understanding Additive Codes for information on subscriptions) • Distance to base = 9.5 km • L1,L2,G1,G2 are the systems in use • Satellites used: L1 = 14, L2 = 11, G1 = 9, G2 = 9 • SNR quality is (> 20 dB), (> 20 dB), (> 20 dB), (> 20dB) • Rover slip flag = 0 • Base slip flag = 1		

	• Horizontal accuracy estimation = 0.011 • RTK accuracy status = 000 (no issues or errors) • Little or no ionospheric scintillation
Additional Information:	Issuing the JSAVE command after setting JASC,PSAT,RTKSTAT to 1 (message on at 1Hz) does not save this setting. You must e JASC,PSAT,RTKSTAT (set it to 1) each time you power on the receiver.
Related Commands and Messages:	JASC,PSAT,RTKSTAT JQUERY,RTKSTAT

Topic Last Updated: v4.0 / June 30,2020

PSAT, VCT Message

Message Type:	Data, Local Differential and RTK
Description:	Provides the vector from the primary to the secondary antenna
Message Format:	Command Format to Request Message: $JASC,PSAT,VCT,r[,OTHER]<CR><LF> where: 'r' =0,1,2,5,10,20HZ (0 turns off the message) ',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets) Message Format: $PSAT,VCT,ID,HHMMSS.SS,A.A,B.B,C.C,D,E.E,F.F,G.G,H.H*CC<CR><LF> where:

Message Component	Description
ID	antenna pair ID (always 1 for now)
HHMMSS.SS	UTC time in hours, minutes, and seconds of the position
A.A	Heading in degree
B.B	Pitch in degree
C.C	Roll in degree
N	Normal, not coasting
E.E	distance between antennas (m)
F.F	North component of master to slave vector (m)
G.G	East component of master to slave vector (m)
H.H	Up component of master to slave vector (m)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Example:	$PSAT,VCT,1,011657.00,28.358,-5.306,,N,4.7591,4.1530,2.2823,- 0.4401*1F
Additional Information:	Issuing the JSAVE command after setting JASC,PSAT,RTKSTAT to 1 (message on at 1Hz) does not save this setting. You must e JASC,PSAT,RTKSTAT (set it to 1) each time you power on the receiver.
Related Commands and Messages:	JASC,PSAT,VCT

Topic Last Updated: v4.0 / June 30,2020

RD1 Message

Message Type:	Data
Description:	SBAS diagnostic information
Message Format:	Command Format to Request Message: $JASC,D1,r[,OTHER]<CR><LF> where: •'r' = message rate (0 = Off, 1 = On at 1Hz) •',OTHER' = optional field, enacts a change in the RD1 message on the current port when you send the command without it (and without the brackets) and enacts a change in the RD1 message on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology. Message Format $RD1,SEC,WEEK,FREQ,DSPLOCK,BER2,AGC,DDS,DOPPLER,DSPSTAT,ARMSTAT, DIFFSTAT,NAVCON<CR><LF> where:

Message Component	Description
SEC	Second of GPS week (may be a couple of seconds old)
WEEK	GPS week number
FREQ	L-band frequency in MHz (1575.4200 is used for SBAS)
DSPLOCK	N/A
BER2	BER - given for both SBAS satellites being tracked
AGC	L-band signal strength
DDS	0.0 for SBAS
DOPPLER	0 for SBAS
DSPSTAT	Status bit mask for the DSP tracking of SBAS • Bit 0 = Carrier lock • Bit 1 = BER OK (Viterbi lock) (yellow LED 2) • Bit 2 =Atlas: DSP got lock and has stable freq; WAAS: Frame sync2

<table>
<tr><td></td><td></td><td></td><td>

Bit 3 = Frame sync1
Bit 4 = Track mode (same as carrier lock)
Bits 5 - 15 Unused

</td></tr>
<tr><td></td><td></td><td>ARMSTAT</td><td>Status bit mask for the ARM GPS solution (ARM status values shown below)

Bit 0 = GPS lock (yellow LED 1)
Bit 1 = DGPS valid data
Bit 2 = ARM has lock
Bit 3 = Diff and GPS (flashing green LED 3)
Bit 4 = GPS solution is good (solid green LED 3)
Bit 5 = ARM controls yellow LED 2
Bit 6 = ARM command for yellow LED 2
Bits 7 - 15 Unused

</td></tr>
<tr><td></td><td></td><td>DIFFSTAT</td><td>SBAS PRN of the satellite in use</td></tr>
<tr><td></td><td></td><td>NAVCON</td><td>Series of hex character fields with each field representing the number of GPS satellites satisfying a certain condition, all of which conditions are required if the satellite is to be used in the solution

Example of NAVCON for the value 179889A shown below (read right to left)
<table>
<tr><td>Hex Field</td><td>Description</td></tr>
<tr><td>1</td><td>Right Hexadecimal count of satellites with valid tracks</td></tr>
<tr><td>2</td><td>Hexadecimal count of satellites for which a 9-ephemeris message has been received</td></tr>
<tr><td>3</td><td>Hexadecimal count of satellites which are healthy 8</td></tr>
<tr><td>4</td><td>Hexadecimal count of satellites which passed the criteria of hex fields 1,2,3 and 5 (satellites that er tracked, have an ephemeris, are healthy, and are above the elevation mask) 8</td></tr>
<tr><td>5</td><td>Hexadecimal count of satellites above the elevation mask 9</td></tr>
<tr><td>6</td><td>Hexadecimal count of satellites for which a differential correction is available 7</td></tr>
<tr><td>7</td><td>Hexadecimal count of satellites for which a differential 1correction is NOT available 1</td></tr>
</table>
</td></tr>
<tr><td></td><td></td><td><CR></td><td>Carriage return</td></tr>
<tr><td>Example:</td><td colspan="3"></td></tr>
<tr><td>Additional Information:</td><td colspan="3"></td></tr>
<tr><td>Related Commands and Messages:</td><td colspan="3">JASC,D1 (RD1)</td></tr>
</table>

TSS1 Message

<table>
<tr><td>Message Type:</td><td>Vector, Data</td></tr>
<tr><td>Description:</td><td>Heave, pitch, and roll message in the commonly used TSS1 message format</td></tr>
<tr><td>Message Format:</td><td>
Command Format to Request Message:

$JASC,PTSS1,r[,OTHER]<CR><LF>

where:
'r' = message rate (in Hz) of 0 (off), 0.25,0.5, 1, 2, 4, 5, 10, or 20 (if subscribed)
',OTHER' = optional field, enacts a change on the current port when you send the command without it (and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets). See Configuring the Data Message Output for detailed information on 'THIS' and 'OTHER' port terminology.

Message Format:

:XXAAAASMHHHHQMRRRRSMPPPP<CR><LF>

where:

<table>
<tr><th>Message Component</th><th>Description</th></tr>
<tr><td>XX</td><td>Horizontal acceleration (hex value), in 3.83 cm/s², with a range of zero to 9.81 m/s²</td></tr>
<tr><td>AAAA</td><td>Vertical acceleration (hex value - 2's complement), in 0.0625 cm/s², with a range of –20.48 to +20.48 m/s²</td></tr>
<tr><td>S</td><td>Space character</td></tr>
<tr><td>M</td><td>Space if positive; minus if negative</td></tr>
<tr><td>HHHH</td><td>Heave, in centimeters, with a range of –99.99 to +99.99 meters</td></tr>
<tr><td>Q</td><td>Status flag

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>h</td><td>Heading aided mode (settling) - The System is receiving heading aiding signals from a gyrocompass but is still awaiting the end of the three minutes settling period after power-on or a change of mode or heave bandwidth.

The gyrocompass takes approximately five minutes to settle after it has been powered on. During this time, gyrocompass aiding of the System will not be perfect. The status flag does NOT indicate this condition.</td></tr>
<tr><td>F</td><td>Full aided mode (settled condition) - The System is receiving and using aiding signals from a gyrocompass and from a GNSS receiver or a Doppler log.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>M</td><td>Space if positive; minus if negative</td></tr>
<tr><td>RRRR</td><td>Roll, in units of 0.01 degrees (ex: 1000 = 10°), with a range of –99.99° to +99.99°</td></tr>
<tr><td>S</td><td>Space character</td></tr>
<tr><td>M</td><td>Space if positive; minus if negative</td></tr>
<tr><td>PPPP</td><td>Pitch, in units of 0.01 degrees (ex: 1000 = 10°), with a range of –99.99° to +99.99°</td></tr>
<tr><td><CR></td><td>Carriage return</td></tr>
<tr><td><LF></td><td>Line feed</td></tr>
</table>
</td></tr>
<tr><td>Example:</td><td>:020010 -0001F 0023 -0169

where:

• XX = 02, horizontal acceleration, which is 7.66 cm/s²
• (XX = 02 (hex) = decimal 2, multiplied by 3.83 cm/s² yields 7.66 cm/s²)
• AAAA = 0010, vertical acceleration, which is 1 cm/s²
• (AAAA = 0010 (hex), which = decimal 16, multiplied by 0.0625 cm/s² yields 1 cm/s²)
• S = (space)
• M = (minus), meaning following heave value is negative
• HHHH = 0001, heave, which is 1 cm (-1 cm based on the M value)
• Q = F, status flag, which is full aided mode
• M = (space),meaning following roll value is positive
• RRRR = 0023, roll, which is 0.23°
• S = (space)
• M = (minus), meaning following pitch value is negative
• PPPP = 0169, pitch, which is 1.69°</td></tr>
<tr><td>Additional Information:</td><td></td></tr>
<tr><td>Related Commands and Messages:</td><td>JASC,PTSS1</td></tr>
</table>

Topic Last Updated: v1.07 / February 16, 2017

Binary Messages Code

Binary Message Header File with Binary Codes

For an electronic copy of the Binary Message Header File, see the HGNSS website.

```
// BinaryMsgH.h

#ifndef __BinaryMsg_H__
#define __BinaryMsg_H__
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Copyright (c) 2006  Hemisphere GPS and CSI Wireless Inc.,
 * All Rights Reserved.
 *
 * Use and copying of this software and preparation of derivative works
```

```
 * based upon this software are permitted. Any copy of this software or
 * of any derivative work must include the above copyright notice, this
 * paragraph and the one after it.  Any distribution of this software or
 * derivative works must comply with all aplicable laws.
 *
 * This software is made available AS IS, and COPYRIGHT OWNERS DISCLAIMS
 * ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE, AND NOTWITHSTANDING ANY OTHER PROVISION CONTAINED HEREIN, ANY
 * LIABILITY FOR DAMAGES RESULTING FROM THE SOFTWARE OR ITS USE IS
 * EXPRESSLY DISCLAIMED, WHETHER ARISING IN CONTRACT, TORT (INCLUDING
 * NEGLIGENCE) OR STRICT LIABILITY, EVEN IF COPYRIGHT OWNERS ARE ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGES.
 */

#include <stdint.h>

//xx #if defined(WIN32) || (__ARMCC_VERSION>=300441)  // all compilers that we use today
    #pragma pack(push)
    #pragma pack(4)
//#endif


/*************************************************/
/*  SBinaryMsgHeader                             */
/*************************************************/
typedef struct
{
    char           m_strSOH[4];      /* start of header ($BIN)      */
    unsigned short m_byBlockID;      /* ID of message (1,2,99,98,97,96,95,94,93 or 80 ) */
    unsigned short m_wDataLength;    /* 52 16,304,68,28,300,128,96,56, or 40 */
} SBinaryMsgHeader;

typedef struct
{
    unsigned long     ulDwordPreamble;    /* 0x4E494224  = $BIN  */
    unsigned long     ulDwordInfo;        /*    0x00340001 or 0x00100002 or 0x01300063 */
} SBinaryMsgHeaderDW;                     /* or 0x00440062 or 0x001C0061 or 0x012C0060 */
                                          /* or 0x0080005F or 0x0060005E or 0x0038005D */
                                          /* or 0x00280050 */


#define BIN_MSG_PREAMBLE    0x4E494224  /* $BIN = 0x4E494224 */
#define BIN_MSG_HEAD_TYPE1  0x00340001  /* 52 = 0x34 */
#define BIN_MSG_HEAD_TYPE2  0x00100002  /* 16 = 0x10 */
#define BIN_MSG_HEAD_TYPE3  0x00740003  /* 116 = 0x74 */
#define BIN_MSG_HEAD_TYPE4  0x00280004  /* 40 = 0x28 */
#define BIN_MSG_HEAD_TYPE5  0x00480005  /* 72 = 0x48 */
#define BIN_MSG_HEAD_TYPE6  0x000C0006  /* 12 = 0x0C */
#define BIN_MSG_HEAD_TYPE99 0x01300063  /* 99 = 0x63, 304 = 0x130 */ //GPS
#define BIN_MSG_HEAD_TYPE100 0x01040064 /* 100 = 0x64, 260 = 0x104 */
#define BIN_MSG_HEAD_TYPE98 0x00440062  /* 98 = 0x62, 68  = 0x44  */   //GPS Almanac
#define BIN_MSG_HEAD_TYPE97 0x001C0061  /* 97 = 0x61, 28  = 0x1C  */
#define BIN_MSG_HEAD_TYPE96 0x012C0060  /* 96 = 0x60, 300 = 0x12C */  //GPS L1CA phase observables  //deprecated, use BIN16
#define BIN_MSG_HEAD_TYPE95 0x0080005F  /* 95 = 0x5F, 128 = 0x80  */  //GPS L1CA ephemeris data
#define BIN_MSG_HEAD_TYPE94 0x0060005E  /* 94 = 0x5E, 96  = 0x60  */
#define BIN_MSG_HEAD_TYPE93 0x0038005D  /* 93 = 0x5D, 56  = 0x38  */
#define BIN_MSG_HEAD_TYPE92 0x0034005C  /* 92 = 0x5C, 52 =  0x34  */
#define BIN_MSG_HEAD_TYPE89 0x00500059  /* 89 = 0x59, 80  = 0x50  */
#define BIN_MSG_HEAD_TYPE80 0x00280050  /* 80 = 0x50, 40  = 0x28  */
#define BIN_MSG_HEAD_TYPE81 0x00280051  /* 81 = 0x51, 40  = 0x28  = total size in bytes -8 -2 -2*/
#define BIN_MSG_HEAD_TYPE76 0x01C0004C  /* 76 = 0x4C, 448 = 0x1C0 = total size in bytes -8 -2 -2*/ //deprecated, use BIN16
#define BIN_MSG_HEAD_TYPE62 0x0028003E  /* 62 = 0x3E,  40 =  0x28 */
#define BIN_MSG_HEAD_TYPE65 0x00440041  /* 65 = 0x41,  68 =  0x44 */
#define BIN_MSG_HEAD_TYPE66 0x01600042  /* 66 = 0x42, 352 = 0x160 */  //deprecated, use BIN16
#define BIN_MSG_HEAD_TYPE69 0x012C0045  /* 69 = 0x45, 300 = 0x12C */ //Glonass
#define BIN_MSG_HEAD_TYPE52 0x00340034  /* 52 = 0x34,  52 =  0x34 */ //IRNSS
#define BIN_MSG_HEAD_TYPE55 0x00800037  /* 55 = 0x37, 128 = 0x80  */ //IRNSS subframe words --- similar to GPS
//////////////////////#define BIN_MSG_HEAD_TYPE59 0x0100003B  /* 59 = 0x3B, 256 = 0x100 */ //GPS L2C
#define BIN_MSG_HEAD_TYPE49 0x012C0031  /* 49 = 0x31, 300 = 0x12C */  //Galileo Channel Data for SLXMON //deprecated, use BIN19
#define BIN_MSG_HEAD_TYPE45 0x0080002D  /* 45 = 0x2D, 128 = 0x80  */  //Galileo subframe words --- similar to GPS
#define BIN_MSG_HEAD_TYPE44 0x0038002C  /* 44 = 0x2C,  56 = 0x38  */  //Galileo time offsets
#define BIN_MSG_HEAD_TYPE42 0x0034002A  /* 42 = 0x2A,  52 =  0x34 */
#define BIN_MSG_HEAD_TYPE32 0x00340020  /* 32 = 0x20,  52 =  0x34 */
#define BIN_MSG_HEAD_TYPE34 0x00200022  /* 34 = 0x22, 32 = 0x20   */  //BeiDou time offsets
#define BIN_MSG_HEAD_TYPE35 0x00800023  /* 35 = 0x23, 128 = 0x80  */  //BeiDou subframe words --- similar to GPS
#define BIN_MSG_HEAD_TYPE36 0x01500024  /* 36 = 0x24, 336 = 0x150 */  //BeiDou phase observables //deprecated, use BIN16
```

```
#define BIN_MSG_HEAD_TYPE39 0x019C0027  /* 39 = 0x27, 412 = 0x19C */  //BeiDou Channel Data for SLXMON //deprecated, use BIN19
#define BIN_MSG_HEAD_TYPE22 0x00340016  /* 22 = 0x16,  52 =  0x34 */
#define BIN_MSG_HEAD_TYPE25 0x00800019  /* 25 = 0x19, 128 = 0x80  */  //QZSS L1CA ephemeris data
#define BIN_MSG_HEAD_TYPE16 0x01380010  /* 16 = 0x10, 312 = 0x138 */  //GNSS phase observables
#define BIN_MSG_HEAD_TYPE19 0x01780013  /* 19 = 0x13, 376 = 0x178 */  //Generic Channel Data for SLXMON
#define BIN_MSG_HEAD_TYPE135 0x00800087  /* 135 = 0x87, 128 = 0x80  */  //BeiDou phase3 subframe words
#define BIN_MSG_HEAD_TYPE125 0x0080007D  /* 125 = 0x7D, 128 = 0x80  */  // QZSS L5 ephemeris data
#define BIN_MSG_HEAD_TYPE195 0x008000C3  /* 195 = 0xC3, 128 = 0x80  */  // GPS L5 ephemeris data
#define BIN_MSG_HEAD_TYPE209 0x014C00D1 // 209 = 0xD1,  332 = 0x14C
#define BIN_MSG_HEAD_TYPE309 0x01EC0135 // 309 = 0x135, 492 = 0x1EC
#define BIN_MSG_CRLF        0x0A0D      /* CR LF = 0x0D, 0x0A */

#define CHANNELS_12   12
#define CHANNELS_20   20
#define CHANNELS_gen  16   // CHANNELS FOR 16 and 19 general messages


typedef union
{
    SBinaryMsgHeader   sBytes;
    SBinaryMsgHeaderDW sDWord;
}  SUnionMsgHeader;




/************************************************/
/*  SBinaryMsg1                                 */
/************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char  m_byAgeOfDiff;       /* age of differential, seconds (255 max)*/
    unsigned char  m_byNumOfSats;       /* number of satellites used (12 max)    */
    unsigned short m_wGPSWeek;          /* GPS week */
    double         m_dGPSTimeOfWeek;    /* GPS tow  */
    double         m_dLatitude;         /* Latitude degrees, -90..90 */
    double         m_dLongitude;        /* Longitude degrees, -180..180 */
    float          m_fHeight;           /* (m), Altitude ellipsoid */
    float          m_fVNorth;           /* Velocity north       m/s */
    float          m_fVEast;            /* Velocity east        m/s */
    float          m_fVUp;              /* Velocity up  m/s */
    float          m_fStdDevResid;      /* (m), Standard Deviation of   Residuals */
    unsigned short m_wNavMode;
    unsigned short m_wAgeOfDiff;        /* age of diff using 16 bits  */
    unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg1;                          /* length = 8 + 52 + 2 + 2 = 64 */


/************************************************/
/*  SBinaryMsg2                                 */
/************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned long  m_ulMaskSatsTracked;  /* SATS Tracked, bit mapped 0..31 */
    unsigned long  m_ulMaskSatsUsed;     /* SATS Used, bit mapped 0..31 */
    unsigned short m_wGpsUtcDiff;        /* GPS/UTC time difference (GPS minus UTC) */
    unsigned short m_wHDOPTimes10;       /* HDOP        (0.1 units) */
    unsigned short m_wVDOPTimes10;       /* VDOP        (0.1 units) */
    unsigned short m_wWAASMask;          /* Bits 0-1: tracked sats, Bits 2-3:
                                            used sats, Bits 5-9 WAAS PRN 1 minus
                                            120, Bits 10-14 WAAS PRN 1 minus 120 */
    unsigned short m_wCheckSum;          /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;              /* Carriage Return Line Feed */
} SBinaryMsg2;                           /* length = 8 + 16 + 2 + 2 = 28 */


//-************************************************
//-*  SBinaryMsg3
//-*  Lat/Lon/Hgt, Covariances, RMS, DOPs and COG, Speed, Heading
//-************************************************
typedef struct
{
    SUnionMsgHeader  m_sHead;              //                                    [8]
    double           m_dGPSTimeOfWeek;     // GPS tow                            [8 bytes]
```

```
    unsigned short   m_wGPSWeek;           // GPS week                              [2 bytes]
    unsigned short   m_wNumSatsTracked;    // SATS Tracked                          [2 bytes]
    unsigned short   m_wNumSatsUsed;       // SATS Used                             [2 bytes]
    unsigned char    m_byNavMode;          // Nav Mode (same as message 1)          [1 byte ]
    unsigned char    m_bySpare00;          // Spare                                 [1 byte ]
    double           m_dLatitude;          // Latitude degrees, -90..90             [8 bytes]
    double           m_dLongitude;         // Longitude degrees, -180..180          [8 bytes]
    float            m_fHeight;            // (m), Altitude ellipsoid               [4 bytes]
    float            m_fSpeed;             // Horizontal Speed   m/s                [4 bytes]
    float            m_fVUp;               // Vertical Velocity +up  m/s            [4 bytes]
    float            m_fCOG;               // Course over Ground, degrees           [4 bytes]
    float            m_fHeading;           // Heading (degrees), Zero unless vector[4 bytes]
    float            m_fPitch;             // Pitch (degrees), Zero unless vector   [4 bytes]
    float            m_fRoll;              // Roll (degrees), Zero unless vector    [4 bytes]
    unsigned short   m_wAgeOfDiff;         // age of differential, seconds          [2 bytes]
    // m_wAttitudeStatus: bit {0-3}  = sStatus.eYaw
    //                    bit {4-7}  = sStatus.ePitch
    //                    bit {8-11} = sStatus.eRoll
    // where sStatus can be 0 = INVALID, 1 = GNSS, 2 = Inertial, 3= Magnetic
    unsigned short   m_wAttitudeStatus;    // Attitude Status, Zero unless vector   [2 bytes]
    float            m_fStdevHeading;      // Yaw stdev, degrees, 0 unless vector   [4 bytes]
    float            m_fStdevPitch;        // Pitch stdev, degrees, 0 unless vector[4 bytes]
    float            m_fHRMS;              // Horizontal RMS                        [4 bytes]
    float            m_fVRMS;              // Vertical   RMS                        [4 bytes]
    float            m_fHDOP;              // Horizontal DOP                        [4 bytes]
    float            m_fVDOP;              // Vertical DOP                          [4 bytes]
    float            m_fTDOP;              // Time DOP                              [4 bytes]
    float            m_fCovNN;             // Covaraince North-North                [4 bytes]
    float            m_fCovNE;             // Covaraince North-East                 [4 bytes]
    float            m_fCovNU;             // Covaraince North-Up                   [4 bytes]
    float            m_fCovEE;             // Covaraince East-East                  [4 bytes]
    float            m_fCovEU;             // Covaraince East-Up                    [4 bytes]
    float            m_fCovUU;             // Covaraince Up-Up                      [4 bytes]
    unsigned short   m_wCheckSum;          // sum of all bytes of the header and data
    unsigned short   m_wCRLF;              // Carriage Return Line Feed
} SBinaryMsg3;                             // length = 8 + 116 + 2 + 2 = 128  (108 = 74 hex)



//-*************************************************
//-*  SBinaryMsg5
//-*  Base Location and Base ID
//-*************************************************
typedef struct
{
    SUnionMsgHeader  m_sHead;              //                                       [8]
    double           m_dLatitude;          // Base Latitude degrees, -90..90        [8 bytes]
    double           m_dLongitude;         // Base Longitude  degrees, -180..180    [8 bytes]
    float            m_fHeight;            // Base Altitude ellipsoid, (m)          [4 bytes]
    unsigned short   m_wBaseID;            // BaseID                                [2 bytes]
    unsigned short   m_wSpare;             // Spare                                 [2 bytes]
    char             m_szDiffFormat[16];   // String giving format of Differential  [16 bytes]
    unsigned short   m_awSpare[16];        // 32 bytes of spare                     [32 bytes]
    unsigned short   m_wCheckSum;          // sum of all bytes of the header and data
    unsigned short   m_wCRLF;              // Carriage Return Line Feed
} SBinaryMsg5;                             // length = 8 + 72 + 2 + 2 = 84   (72 = 48 hex)


/*************************************************/
/*  SBinaryMsg6 Manual Mark Tag Preceding MM messages*/
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    double           m_dTow;                   /* Time in seconds */
    unsigned short   m_wWeek;                  /* GPS Week Number */
    unsigned short   m_wSpare1;                /* 16 bit spare word */
    unsigned short   m_wCheckSum;              /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                  /* Carriage Return Line Feed */
} SBinaryMsg6;                                 /* length = 8 + (12) + 2 + 2 = 24 */


/*************************************************/
/*  SChannelData                                 */
/*************************************************/
```

```
typedef struct
{
    unsigned char m_byChannel;      /* channel number  */
    unsigned char m_bySV;           /* satellite being tracked, 0 == not tracked  */
    unsigned char m_byStatus;       /* Status bits (code carrier bit frame...)  */
    unsigned char m_byLastSubFrame; /* last subframe processed */

    unsigned char m_byEphmVFlag;    /* ephemeris valid flag */
    unsigned char m_byEphmHealth;   /* ephemeris health */
    unsigned char m_byAlmVFlag;     /* almanac valid flag */
    unsigned char m_byAlmHealth;    /* almanac health */

    char          m_chElev;         /* elevation angle */
    unsigned char m_byAzimuth;      /* 1/2 the Azimuth angle */
    unsigned char m_byURA;          /* User Range Error */
    unsigned char m_byDum;          /* Place Holder */

    unsigned short m_wCliForSNR;    /* code lock indicator for SNR divided by 32 */
    short          m_nDiffCorr;     /* Differential correction * 100 */

    short          m_nPosResid;     /* position residual * 10 */
    short          m_nVelResid;     /* velocity residual * 10 */

    short          m_nDoppHz;       /* expected doppler in HZ */
    short          m_nNCOHz;        /* track from NCO in HZ */

} SChannelData;  /* 24 bytes */



/*************************************************/
/*  SChannelL2Data                               */
/*************************************************/
//#if defined(_DUAL_FREQ_)
typedef struct
{
    unsigned char m_byChannel;      /* channel number  */
    unsigned char m_bySV;           /* satellite being tracked, 0 == not tracked  */
    unsigned char m_byL2CX;         /* Status bits for L2P (code carrier bit frame...)  */
    unsigned char m_byL1CX;         /* Status bits for L1P (code carrier bit frame...)  */

    unsigned short m_wCliForSNRL2P; /* code lock indicator for SNR divided by 32 */
    unsigned short m_wCliForSNRL1P; /* code lock indicator for L1P SNR divided by 32 */

    short          m_nC1_L1;        /* C1-L1 in meters * 100 */
    short          m_nP2_C1;        /* P2-C1 in meters * 100 */

    short          m_nP2_L1;        /* P2-L1 in meters * 100 */
    short          m_nL2_L1;        /* L2-L1 in meters * 100 */

    short          m_nP2_P1;        /* P2-P1 in meters * 100 */
    short          m_nNCOHz;        /* track from NCO in HZ */

} SChannelL2Data;  /* 20 bytes */
//#endif

/*************************************************/
/*  SChannelL2CData     for USING_GPSL2CL        */
/*************************************************/
typedef struct
{
    unsigned char m_byChannel;      // channel number
    unsigned char m_bySV;           // satellite being tracked, 0 == not tracked
    unsigned char m_byL2CX;         // Status bits for L2P (code carrier bit frame...)
    unsigned char spare1;

    unsigned short m_wCliForSNRL2C; // code lock indicator for SNR divided by 32
    unsigned short spare2;

    short          m_nL2C_L1Ca;     //L2CL - CA code error  meters * 100
    short          m_nL2C_L2P;      //L2CL - L2P code error meters * 100

    short          m_nL2_L1;        //L2CL - L1CA phase error meters * 100
    short          m_nL2_L2P;       //L2CL - L2P phase error meters * 100

    short          spare3;
    short          m_nNCOHz;        // track from NCO in HZ
```

```
} SChannelL2CData;  // 20 bytes

/*************************************************/
/*  SBinaryMsg99                                 */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char  m_byNavMode;         /* Nav Mode FIX_NO, FIX_2D, FIX_3D (high bit =has_diff) */
    char           m_cUTCTimeDiff;      /* whole Seconds between UTC and GPS   */
    unsigned short m_wGPSWeek;          /* GPS week */
    double         m_dGPSTimeOfWeek;    /* GPS tow  */
    SChannelData   m_asChannelData[CHANNELS_12]; /* channel data */
    short          m_nClockErrAtL1;     /* clock error at L1, Hz */
    unsigned short m_wSpare;            /* spare */
    unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg99;                         /* length = 8 + 304 + 2 + 2 = 316 */


//some legacy binary messages are limited to only 3-SBAS
//   SBinaryMsg89
//   SBinaryMsg76
//   SBinaryMsg118
// do not change these or we will invalid those messages
#define CHANNELS_SBAS_E    3
#define CHANNELS_12_PLUS (CHANNELS_12+2)              /* up to two SBAS satellites */
#define CHANNELS_L1_E    (CHANNELS_12+CHANNELS_SBAS_E) /* All L1 (including SBAS satellites) */

/*************************************************/
/*  SBinaryMsg89  * Supports 3 SBAS Satellites      */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    long           m_lGPSSecOfWeek;     /* GPS tow integer sec */
    unsigned char  m_byMaskSBASTracked; /* SBAS Sats Tracked, bit mapped 0..3 */
    unsigned char  m_byMaskSBASUSED;    /* SBAS Sats Used, bit mapped 0..3 */
    unsigned short m_wSpare;            /* spare */
    SChannelData   m_asChannelData[CHANNELS_SBAS_E];  /* SBAS channel data */
    unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg89;                         /* length = 8 + 80 + 2 + 2 = 92 */

/*************************************************/
/*  SBinaryMsg100                                 */
/*************************************************/
//#if defined(_DUAL_FREQ_)
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char  m_byNavMode;         /* Nav Mode FIX_NO, FIX_2D, FIX_3D (high bit =has_diff) */
    char           m_cUTCTimeDiff;      /* whole Seconds between UTC and GPS   */
    unsigned short m_wGPSWeek;          /* GPS week */
    unsigned long  m_ulMaskSatsUsedL2P; /* L2P SATS Used, bit mapped 0..31 */
    double         m_dGPSTimeOfWeek;    /* GPS tow  */
    unsigned long  m_ulMaskSatsUsedL1P; /* L1P SATS Used, bit mapped 0..31 */
    SChannelL2Data m_asChannelData[CHANNELS_12]; /* channel data */
    unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg100;                        /* length = 8 + 260 + 2 + 2 = 272 */
//#endif



//-*************************************************
//-*  SSVSNRData
//-*************************************************
typedef struct
{
    unsigned short m_wStatus_SYS_PRNID; // status, GNSS system, PRN ID
                                        //    Bit 0-5  PRNID  (for SBAS , PRNID = PRN-120)
                                        //    Bit 6-8  SYS: 0 = GPS, 1 = GLONASS, 2 = GALILEO, 3 = BEIDOU, 4=QZSS, 5=IRNSS, 7 =
SBAS
                                        //    Bit 9  = code and Carrier Lock on L1,G1,B1
```

```
                                    //    Bit 10 = code and Carrier Lock on L2,G2,B2
                                    //    Bit 11 = code and Carrier Lock on L5,E5,B3
                                    //    Bit 12 = Bit Lock  and Frame lock (decoding data)
                                    //    Bit 13 = Ephemeris Available
                                    //    Bit 14 = Health OK
                                    //    Bit 15 = Satellite used in Navigation Solution
                                    //    m_wStatus_SYS_PRNID = 0 ==> unfilled data
    char          m_chElev;         // Elevation angle, LSB = 1 deg
    unsigned char m_byAzimuth;      // 1/2 the Azimuth angle, LSB = 2 deg
    unsigned long m_ulSNR3_SNR2_SNR1;  // 3 SNRs, 2 @ 11 bit & 1 @ 10 bits, each SNR = 10.0*log10( 0.8192*SNR_value)
                                    //    Bits 0-10  SNR1  (L1,G1,B1, etc)  11 bits => Max SNR = 32.2 dB
                                    //    Bits 11-21 SNR2  (L2,G2,B2, etc)  11 bits => Max SNR = 32.2 dB
                                    //    Bits 22-31 SNR3  (L5,E5,B3, etc)  10 bits => Max SNR = 29.2 dB
} SSVSNRData;  // 8 bytes

//-*****************************************************
//-*  SBinaryMsg209
//-*  SNR and status for all GNSS tracks
//-*****************************************************
typedef struct
{
    SUnionMsgHeader  m_sHead;             //                                                  [8]
    double           m_dGPSTimeOfWeek;    // GPS tow                                          [8 bytes]
    unsigned short   m_wGPSWeek;          // GPS week                                         [2 bytes]
    char             m_cUTCTimeDiff;      // Whole Seconds between UTC and GPS                [1 byte]
    unsigned char    m_byPage;            // Bits 0-1 = Antenna: 0 = Master, 1 = Slave, 2 = Slave2 [1 byte]
                                          // Bits 2-4 = Page ID: 0 = page 1, 1 = page 2, etc
                                          // Bits 5-7 = Max page ID: 0 = only 1 page, 1 = 2 pages
    SSVSNRData       m_asSVData[40];      // SNR data                                         [320 bytes]
    unsigned short   m_wCheckSum;         // sum of all bytes of the header and data
    unsigned short   m_wCRLF;             // Carriage Return Line Feed
} SBinaryMsg209;                          // length = 8 + 332 + 2 + 2 = 344

//-*****************************************************
//-*  SSVSNRData309
//-*****************************************************
typedef struct
{
    unsigned short m_wSYS_PRNID;        // GNSS system, PRN ID
                                        //    Bit 0-6 PRNID  (For SBAS , PRNID = PRN-120. For QZSS,  PRNID = PRN-192)
                                        //    Bit 7-10 SYS: 0 = GPS, 1 = GLONASS, 2 = GALILEO, 3 = BEIDOU, 4=QZSS, 5 = IRNSS, 7
= SBAS
                                        //    Bit 11-15   Spare
                                        //    m_wSYS_PRNID = 0 ==> unfilled data
    unsigned short m_wStatus;           // status
                                        //    Bit 0 = code and Carrier Lock on Signal 0
                                        //    Bit 1 = code and Carrier Lock on Signal 1
                                        //    Bit 2 = code and Carrier Lock on Signal 2
                                        //    Bit 3 = code and Carrier Lock on Signal 3
                                        //    Bit 4 = code and Carrier Lock on Signal 4
                                        //    Bit 5 = code and Carrier Lock on Signal 5
                                        //    Bit 6 = code and Carrier Lock on Signal 6
                                        //    Bit 7 = code and Carrier Lock on Signal 7
                                        //       GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3, L1C=4
                                        //       GLO Signal ID: G1C/G1P=0, G2C/G2P=1, G10C=4, G20C=5, G30C=6
                                        //       GAL Signal ID: E1BC=0, E5A=1, E5B=2, E6=3, ALTBOC=4
                                        //       BDS Signal ID: B1I=0, B2I=1, B3I=2,B1BOC=3,B2A=4,B2B=5,B3C=6,ACEBOC=7
                                        //       QZS Signal ID: L1CA=0, L2C=2, L5=3, L1C=4, LEX=5
                                        //       IRN Signal ID: L5=0
                                        //    Bit 8  = Bit Lock and Frame lock (decoding data) on Signal 0
                                        //    Bit 9  = Bit Lock and Frame lock (decoding data) on Signal 1
                                        //    Bit 10 = Bit Lock and Frame lock (decoding data) on Signal 2
                                        //    Bit 11 = spare
                                        //    Bit 12 = spare
                                        //    Bit 13 = Ephemeris Available
                                        //    Bit 14 = Health OK
                                        //    Bit 15 = Satellite used in Navigation Solution

    char           m_chElev;             // Elevation angle, LSB = 1 deg
    unsigned char  m_byAzimuth;          // 1/2 the Azimuth angle, LSB = 2 deg
    unsigned short m_wLower2BitsSNR7_6_5_4_3_2_1_0;   // Lower 2 bits of 10 bit SNR for channel 7-0
                                        // Bit 0-1,   lower two bits of SNR on Signal 0
                                        // Bit 2-3,   lower two bits of SNR on Signal 1
                                        // Bit 4-5,   lower two bits of SNR on Signal 2
                                        // Bit 6-7,   lower two bits of SNR on Signal 3
                                        // Bit 8-9,   lower two bits of SNR on Signal 4
                                        // Bit 10-11, lower two bits of SNR on Signal 5
```

```
                                        // Bit 12-13, lower two bits of SNR on Signal 6
                                        // Bit 14-15, lower two bits of SNR on Signal 7
    unsigned char  m_abySNR8Bits[8];    // 8 SNRs, Upper 8 bits of 10 bit SNR, SNR = 10.0*log10( 0.8192*SNR_value),
                                        //                                     Max SNR = 29.2 dB
                                        // SNR_value for i'th SNR = ((unsigned long)m_abySNR8Bits[i] << 2) + Lower2Bits
                                        // Lower2Bits = (m_wLower2BitsSNR7_6_5_4_3_2_1_0 >> (2*i)) & 0x3;
                                        // m_abySNR8Bits[0]  8 bits of SNR on signal 0
                                        // m_abySNR8Bits[1]  8 bits of SNR on signal 1
                                        // m_abySNR8Bits[2]  8 bits of SNR on signal 2
                                        // m_abySNR8Bits[3]  8 bits of SNR on signal 3
                                        // m_abySNR8Bits[4]  8 bits of SNR on signal 4
                                        // m_abySNR8Bits[5]  8 bits of SNR on signal 5
                                        // m_abySNR8Bits[6]  8 bits of SNR on signal 6
                                        // m_abySNR8Bits[7]  8 bits of SNR on signal 7
} SSVSNRData309;  // 16 bytes


//-*************************************************
//-*  SBinaryMsg309
//-*  SNR and status for all GNSS tracks
//-*************************************************
typedef struct
{
    SUnionMsgHeader  m_sHead;              //                                                    [8]
    double           m_dGPSTimeOfWeek;     // GPS tow                                            [8 bytes]
    unsigned short   m_wGPSWeek;           // GPS week                                           [2 bytes]
    char             m_cUTCTimeDiff;       // Whole Seconds between UTC and GPS                  [1 byte]
    unsigned char    m_byPage;             // Bits 0-1 = Antenna: 0 = Master, 1 = Slave, 2 = Slave2 [1 byte]
                                           // Bits 2-4 = Page ID: 0 = page 1, 1 = page 2, etc
                                           // Bits 5-7 = Max page ID: 0 = only 1 page, 1 = 2 pages
    SSVSNRData309    m_asSVData309[30];    // SNR data                                           [480 bytes]
    unsigned short   m_wCheckSum;          // sum of all bytes of the header and data
    unsigned short   m_wCRLF;              // Carriage Return Line Feed
} SBinaryMsg309;                           // length = 8 + 492 + 2 + 2 = 504


/*************************************************/
/*  SSVAlmanData                                 */
/*************************************************/
typedef struct
{
    short         m_nDoppHz;        /* doppler in HZ for stationary receiver */
    unsigned char m_byCountUpdate;  /* count of almanac updates */
    unsigned char m_bySVindex;      /* 0 through 31 (groups of 8)*/
    unsigned char m_byAlmVFlag;     /* almanac valid flag */
    unsigned char m_byAlmHealth;    /* almanac health */
    char          m_chElev;         /* elevation angle */
    unsigned char m_byAzimuth;      /* 1/2 the Azimuth angle */
} SSVAlmanData;  /* 8 bytes */



/*************************************************/
/*  SBinaryMsg98                                 */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    SSVAlmanData   m_asAlmanData[8];    /* SV data, 8 at a time */
    unsigned char  m_byLastAlman;       /* last almanac processed */
    unsigned char  m_byIonoUTCVFlag;    /* iono UTC flag */
    unsigned short m_wSpare;            /* spare */
    unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg98;                         /* length = 8 + (64+1+1+2) + 2 + 2 = 80 */


/*************************************************/
/*  SBinaryMsg97                                 */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned long  m_ulCPUFactor;       /* CPU utilization Factor (%=multby 450e-6) */
    unsigned short m_wMissedSubFrame;   /* missed subframes */
    unsigned short m_wMaxSubFramePend;  /* max subframe pending */
    unsigned short m_wMissedAccum;      /* missed accumulations */
```

```
    unsigned short m_wMissedMeas;       /* missed measurements */
    unsigned long  m_ulSpare1;          /* spare 1 (zero)*/
    unsigned long  m_ulSpare2;          /* spare 2 (zero)*/
    unsigned long  m_ulSpare3;          /* spare 3 (zero)*/
    unsigned short m_wSpare4;           /* spare 4 (zero)*/
    unsigned short m_wSpare5;           /* spare 5 (zero)*/
    unsigned short m_wCheckSum;         /* sum of all bytes of the headerand data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg97;                         /* length = 8 + (28) + 2 + 2 = 40 */



/************************************************/
/*  SObservations                               */
/************************************************/
typedef struct
{
    unsigned long    m_ulCS_TT_SNR_PRN; /* Bits 0-7 PRN (PRN is 0 if no data) */
                                        /* Bits 8-15 SNR_value
                                           SNR = 10.0*log10( 0.8192*SNR_value) */
                                        /* Bits 16-23 Phase Track Time in units
                                           of 1/10 second (range = 0 to 25.5
                                           seconds  (see next word) */
                                        /* Bits 24-31 Cycle Slip Counter
                                           Increments by 1 every cycle slip
                                           with natural roll over after 255 */
    unsigned long    m_ulDoppler_FL;    /* Bit  0: 1 if Valid Phase, 0 otherwise
                                           Bit  1: 1 if Track Time > 25.5 sec,
                                                   0 otherwise
                                           Bits 2-3: unused
                                           Bits 4-32: Signed (two's compliment)
                                           doppler in units of m/sec x 4096.
                                           (i.e.,  LSB = 1/4096). Range =
                                           +/- 32768 m/sec. Computed as
                                           phase change over 1/10 sec. */
    double           m_dPseudoRange;    /* pseudo ranges (m) */
    double           m_dPhase;          /* phase (m) L1 wave len = 0.190293672798365*/
} SObservations;  /* 24 bytes */



/************************************************/
/*  SBinaryMsg96                                */
/************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned short   m_wSpare1;            /* spare 1 (zero)*/
    unsigned short   m_wWeek;              /* GPS Week Number */
    double           m_dTow;               /* Predicted GPS Time in seconds */
    SObservations    m_asObvs[CHANNELS_12];/* 12 sets of observations */
    unsigned short   m_wCheckSum;          /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;              /* Carriage Return Line Feed */
} SBinaryMsg96;                            /* length = 8 + (300) + 2 + 2 = 312 */



/************************************************/
/*  SBinaryMsg95                                */
/************************************************/
/* sent only upon command or when values change */
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned short   m_wSV;               /* The satellite to which this data belongs. */
    unsigned short   m_wSpare1;           /* spare 1 (chan number (as zero 9/1/2004)*/
    unsigned long    m_TOW6SecOfWeek;     /* time at which this arrived (LSB = 6sec) */
    unsigned long    m_SF1words[10];      /* Unparsed SF 1 message words. */
    unsigned long    m_SF2words[10];      /* Unparsed SF 2 message words. */
    unsigned long    m_SF3words[10];      /* Unparsed SF 3 message words. */
                                          /* Each of the subframe words contains
                                             one 30-bit GPS word in the lower
                                             30 bits, The upper two bits are ignored
                                             Bits are placed in the words from left to
                                             right as they are received */
    unsigned short   m_wCheckSum;         /* sum of all bytes of the header and data */
```

```
    unsigned short   m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg95;                           /* length = 8 + (128) + 2 + 2 = 140 */


//-------------------------------------------------------------------------
//  SBinaryMsg94
//  I think we will need similar binary messages for Galileo and BeiDou
//  Or maybe not, it seems that much of this is optional for RINEX
//-------------------------------------------------------------------------
// sent only upon command or when values change
typedef struct
{
    SUnionMsgHeader  m_sHead;

    /* Iono parameters. */

    double         m_a0,m_a1,m_a2,m_a3;             /* AFCRL alpha parameters. */
    double         m_b0,m_b1,m_b2,m_b3;             /* AFCRL beta parameters.  */

    /* UTC conversion parameters. */

    double         m_A0,m_A1;              /* Coeffs for determining UTC time. */
    unsigned long  m_tot;      /* Reference time for A0 & A1, sec of GPS week. */
    unsigned short m_wnt;                /* Current UTC reference week number. */
    unsigned short m_wnlsf;       /* Week number when dtlsf becomes effective. */
    unsigned short m_dn;    /* Day of week (1-7) when dtlsf becomes effective. */
    short          m_dtls;                  /* Cumulative past leap seconds. */
    short          m_dtlsf;                /* Scheduled future leap seconds. */
    unsigned short m_wSpare1;                                /* spare 4 (zero)*/
    unsigned short m_wCheckSum;     /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;                      /* Carriage Return Line Feed */
} SBinaryMsg94;                             /* length = 8 + (96) + 2 + 2 =  108 */



/*************************************************/
/*  SBinaryMsg93                                 */
/*************************************************/
/* sent only upon command or when values change */
/* WAAS ephemeris */
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned short   m_wSV;             /* The satellite to which this data belongs. */
    unsigned short   m_wWeek;           /* Week corresponding to m_lTOW*/
    unsigned long    m_lSecOfWeekArrived;/* time at which this arrived (LSB = 1sec) */
    unsigned short   m_wIODE;
    unsigned short   m_wURA;        /* See 2.5.3 of Global Pos Sys Std Pos Service Spec */
    long m_lTOW;                        /* Sec of WEEK Bit 0 = 1 sec */
    long m_lXG;                         /* Bit 0 = 0.08 m */
    long m_lYG;                         /* Bit 0 = 0.08 m */
    long m_lZG;                         /* Bit 0 = 0.4 m */
    long m_lXGDot;                      /* Bit 0 = 0.000625 m/sec */
    long m_lYGDot;                      /* Bit 0 = 0.000625 m/sec */
    long m_lZGDot;                      /* Bit 0 = 0.004 m/sec */
    long m_lXGDotDot;                   /* Bit 0 = 0.0000125 m/sec/sec */
    long m_lYGDotDot;                   /* Bit 0 = 0.0000125 m/sec/sec */
    long m_lZGDotDot;                   /* Bit 0 = 0.0000625 m/sec/sec */
    short  m_nGf0;                      /* Bit 0 = 2**-31 sec */
    short  m_nGf0Dot;                   /* Bit 0 = 2**-40 sec/sec */
    unsigned short   m_wCheckSum;       /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;           /* Carriage Return Line Feed */
} SBinaryMsg93;                         /* length = 8 + (56) + 2 + 2 = 68 */


/*************************************************/
/*  SBinaryMsg80                                 */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned short m_wPRN;              /* Broadcast PRN */
    unsigned short m_wSpare;            /* spare (zero) */
    unsigned long  m_ulMsgSecOfWeek;    /* Seconds of Week For Message */
    unsigned long  m_aulWaasMsg[8];     /* Actual 250 bit waas message*/
    unsigned short m_wCheckSum;         /* sum of all bytes of the headerand data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
```

```
} SBinaryMsg80;                          /* length = 8 + (40) + 2 + 2 = 52 */


/***************************************************/
/*  SBinaryMsg81                                   */
/***************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned short m_wPRN;              // Broadcast PRN
    unsigned short m_wFreqAndType;      // Freq (Bit0: 0-L1, 1-L5)  Type (Bit5/Bit4): 00: L1CA_WAAS, 01: L5_WAAS, 10: L5_DFMC
    unsigned long  m_ulMsgSecOfWeek;    // Seconds of Week For Message
    unsigned long  m_aulWaasMsg[8];     // Actual 250 bit waas message
    unsigned short m_wCheckSum;         // sum of all bytes of the headerand data
    unsigned short m_wCRLF;             // Carriage Return Line Feed
} SBinaryMsg81;                         // length = 8 + (40) + 2 + 2 = 52



/***************************************************/
/*  SMsg91Data                                     */
/***************************************************/
typedef struct
{
    unsigned char  bySV;             /* satellite being tracked, 0 == not tracked  */
    unsigned char  byStatus;         /* Status bits (code carrier bit frame...)  */
    unsigned char  byStatusSlave;    /* Status bits (code carrier bit frame...)  */
    unsigned char  byChannel;        /* Not used */

    unsigned short wEpochSlew;             /* 20*_20MS_EPOCH_SLEW + _1MS_EPOCH_SLEW */
    unsigned short wEpochCount;            /* epoch_count */

    unsigned long  codeph_SNR;                /* 0-20 = code phase (21 bits), 28-32 = SNR/4096, upper 4 bits */
    unsigned long  ulCarrierCycles_SNR;       /* 0-23 = carrier cycles, 24-32 = SNR/4096 lower 8 bits */
    unsigned short wDCOPhaseB10_HalfWarns;    /* 0-11 = DCO phase, 12-14 = Half Cycle Warn
                                                 15 = half Cycle added */
    unsigned short m_wPotentialSlipCount;     /* potential slip count */

    /* SLAVE DATA */
    unsigned long  codeph_SNR_Slave;             /* 0-20 = code phase (21 bits), 28-32 = SNR/4096, upper 4 bits */
    unsigned long  ulCarrierCycles_SNR_Slave;    /* 0-23 = carrier cycles, 24-32 = SNR/4096 lower 8 bits */
    unsigned short wDCOPhaseB10_HalfWarns_Slave; /* 0-11 = DCO phase, 12-14 = Half Cycle Warn
                                                    15 = half Cycle added */
    unsigned short m_wPotentialSlipCount_Slave;  /* potential slip count */
} SMsg91Data; /* 32 bytes */


/***************************************************/
  /*  SBinaryMsg91                                   */
  /*  Comment: Transmits data from Takemeas.c        */
  /*           debugging structure.                  */
  /*           Added by bbadke 7/07/2003             */
  /***************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;               /* 8 */
    double           m_sec;                 /* 8 bytes */
    int              m_iWeek;               /* 4 bytes */
    unsigned long    m_Tic;                 /* 4 bytes */
    long             lTicOfWeek;            /* 4 bytes */
    long             lProgTic;              /* 4 bytes */
    SMsg91Data       s91Data[CHANNELS_12];  /* 12*32= 384 bytes */
    unsigned short   m_wCheckSum;           /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;               /* Carriage Return Line Feed */
} SBinaryMsg91;                             /* length = 8 + (408) + 2 + 2 = 420 */


/***************************************************/
/*  SObsPacket                                     */
/***************************************************/
typedef struct
{

    unsigned long    m_ulCS_TT_W3_SNR;   /* Bits 0-11 (12 bits) =SNR_value
                                            For All signals except GPS L2, SNR = 10.0*log10( 0.1024*SNR_value)
                                            For GPS L2, SNR = 10.0*log10( 0.1164*SNR_value) */
```

```
                                        /* Bits 12-14 (3 bits) = 3 bits of warning
                                           for potential 1/2 cycle slips.  A warning
                                           exists if any of these bits are set. */
                                        /* bit 15: (1 bit) 1 if Track Time > 25.5 sec,
                                                           0 otherwise */
                                        /* Bits 16-23 (8 bits): Track Time in units
                                           of 1/10 second (range = 0 to 25.5 seconds) */
                                        /* Bits 24-31 (8 bits) = Cycle Slip Counter
                                           Increments by 1 every cycle slip
                                           with natural roll-over after 255 */
    unsigned long    m_ulP7_Doppler_FL; /* Bit 0: (1 bit) 1 if Valid Phase, 0 otherwise
                                           Bit 1-23: (23 bits) =Magnitude of doppler
                                               LSB = 1/512 cycle/sec
                                               Range = 0 to 16384 cycle/sec
                                           Bit 24: sign of doppler, 1=negative, 0=pos
                                           Bits 25-31 (7 bits) = upper 7 bits of the
                                               23 bit carrier phase.
                                               LSB = 64 cycles,  MSB = 4096 cycles */

    unsigned long    m_ulCodeAndPhase;  /* Bit 0-15 (16 bits) lower 16 bits of code
                                               pseudorange
                                               LSB = 1/256 meters
                                               MSB = 128 meters
                                               Note, the upper 19 bits are given in
                                               m_aulCACodeMSBsPRN[] for CA code
                                           Bit 16-31 lower 16 bits of the carrier phase,
                                                     7 more bits are in m_ulP7_Doppler_FL
                                               LSB = 1/1024 cycles
                                               MSB = 32 cycles */


} SObsPacket;  /* 12 bytes , note: all zero if data not available */

/* A NOTE ON DECODING MESSAGE 76
 * Notation: "code" -- is taken to mean the PseudoRange derived from code phase.
 *           "phase" -- is taken to mean range derived from carrier phase.
 *                      This will contain cycle ambiguities.
 *
 * Only the lower 16 bits of L1P code, L2P code and the lower 23 bits of
 * carrier phase are provided. The upper 19 bits of the L1CA code are found
 * in m_aulCACodeMSBsPRN[].  The upper 19 bits of L1P or L2P must be derived
 * using the fact that L1P and L2P are within 128 meters of L1CA.  To
 * determine L1P or L2P, use the lower 16 bits provided in the message and
 * set the upper bits to that of L1CA.  Then add or subtract one LSB of the
 * upper bits (256 meters) so that L1P or L2P are within 1/2 LSB (128 meters)
 * of the L1CA code.
 *     The carrier phase is in units of cycles, rather than meters,
 * and is held to within 1023 cycles of the respective code range.  Only
 * the lower 16+7=23 bits of carrier phase are transmitted in Msg 76.
 * In order to determine the remaining bits, first convert the respective
 * code range (determined above) into cycles by dividing by the carrier
 * wavelength.  Call this the "nominal reference phase". Next extract the 16
 * and 7 bit blocks of carrier phase from Msg 76 and arrange to form the lower
 * 23 bits of carrier phase.  Set the upper bits (bit 23 and above) equal to
 * those of the nominal reference phase.  Then, similar to what was done for
 * L1P and L2P, add or subtract the least significant upper bit (8192 cycles)
 * so that carrier phase most closely agrees with the nominal reference phase
 * (to within 4096 cycles).
 */


/*************************************************/
/*  SBinaryMsg76                                 */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    double           m_dTow;                  /* GPS Time in seconds */
    unsigned short   m_wWeek;                 /* GPS Week Number */
    unsigned short   m_wSpare1;               /* spare 1 (zero)*/
    unsigned long    m_ulSpare2;              /* spare 2 (zero)*/
    SObsPacket       m_asL2PObs[CHANNELS_12];   /* 12 sets of L2(P) observations */
    SObsPacket       m_asL1CAObs[CHANNELS_L1_E];  /* 15 sets of L1(CA) observations */
    unsigned long    m_aulCACodeMSBsPRN[CHANNELS_L1_E]; /* array of 15 words.
                                               bit 7:0 (8 bits) = satellite PRN, 0
                                               if no satellite
                                               bit 12:8 (5 bits) = spare
```

```
                                                bit 31:13 (19 bits) = upper 19 bits
                                                of L1CA  LSB = 256 meters
                                                         MSB = 67108864 meters */
    unsigned long    m_auL1Pword[CHANNELS_12];  /* array of 12 words relating to L1(P) code.
                                                   Bit 0-15 (16 bits) lower 16 bits of the
                                                   L1P code pseudo range.
                                                   LSB = 1/256 meters
                                                   MSB = 128 meters
                                                   Bits 16-27 (12 bits) = L1P SNR_value
                                                   SNR = 10.0*log10( 0.1164*SNR_value)
                                                   If Bits 16-27 all zero, no L1P track
                                                   Bits 28-31 (4 bits) spare */
    unsigned short   m_wCheckSum;              /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                  /* Carriage Return Line Feed */
} SBinaryMsg76;                                /* length = 8 + (448) + 2 + 2 = 460 */




/*****************************************************/
/*  SGLONASSChanData                                 */
/*****************************************************/
typedef struct
{
    unsigned char m_bySV;            /* Bit (0-6) = SV slot, 0 == not tracked
                                      * Bit 7 = Knum flag
                                      * = KNum+8 if bit 7 set
                                      */
    unsigned char m_byAlm_Ephm_Flags;/* ephemeris and almanac status flags */
                                     /* bit 0: Ephemeris available but timed out
                                      * bit 1: Ephemeris valid
                                      * bit 2: Ephemeris health OK
                                      * bit 3: unused
                                      * bit 4: Almanac available
                                      * bit 5: Almanac health OK
                                      * bit 6: unused
                                      * bit 7: Satellite doesn't exist
                                      */

    unsigned char m_byStatus_L1;    /* Status bits (code carrier bit frame...)  */
    unsigned char m_byStatus_L2;    /* Status bits (code carrier bit frame...)  */

    char          m_chElev;         /* elevation angle */
    unsigned char m_byAzimuth;      /* 1/2 the Azimuth angle */
    unsigned char m_byLastMessage;  /* last message processed */
    unsigned char m_bySlip01;       /* cycle slip on chan 1 */

    unsigned short m_wCliForSNR_L1; /* code lock indicator for SNR divided by 32 */
    unsigned short m_wCliForSNR_L2; /* code lock indicator for SNR divided by 32 */

    short         m_nDiffCorr_L1;    /* Differential correction * 100 */
    short         m_nDoppHz;         /* expected doppler in HZ at glonass L1 */

    short         m_nNCOHz_L1;       /* track from NCO in HZ */
    short         m_nNCOHz_L2;       /* track from NCO in HZ */

    short         m_nPosResid_1;    /* position residual 1 * 1000 */
    short         m_nPosResid_2;    /* position residual 2 * 1000 */

} SGLONASSChanData;  /* 24 bytes */

/*****************************************************/
/*  SBinaryMsg69                                     */
/*****************************************************/
typedef struct
{
    SUnionMsgHeader    m_sHead;
    long               m_lSecOfWeek;      /* tow  */
    unsigned short     m_wL1usedNavMask;  /* mask of L1 channels used in nav solution */
    unsigned short     m_wL2usedNavMask;  /* mask of L2 channels used in nav solution */
    SGLONASSChanData   m_asChannelData[CHANNELS_12]; /* channel data 12X24 = 288 */
    unsigned short     m_wWeek;          /* week */
    unsigned char      m_bySpare01;      /* spare 1 */
```

```
    unsigned char      m_bySpare02;      /* spare 2 */
    unsigned short     m_wCheckSum;      /* sum of all bytes of the header and data */
    unsigned short     m_wCRLF;          /* Carriage Return Line Feed */
} SBinaryMsg69;                          /* length = 8 + 300 + 2 + 2 = 312 */


//Need to add to this for E1B and E1C.
/***************************************************/
/*  SGALILEOChanData                               */
/***************************************************/
typedef struct
{
    unsigned char m_bySV;            /* Bit (0-6) = SV slot, 0 == not tracked
                                      * Bit 7 = Knum flag
                                      * = KNum+8 if bit 7 set
                                      */
    unsigned char m_byAlm_Ephm_Flags;/* ephemeris and almanac status flags */
                                     /* bit 0: Ephemeris available but timed out
                                      * bit 1: Ephemeris valid
                                      * bit 2: Ephemeris health OK
                                      * bit 3: unused
                                      * bit 4: Almanac available
                                      * bit 5: Almanac health OK
                                      * bit 6: unused
                                      * bit 7: Satellite doesn't exist
                                      */

    unsigned char m_byStatus_L1;    /* Status bits (code carrier bit frame...)  */
    unsigned char m_byStatus_L2;    /* Status bits (code carrier bit frame...)  */

    char          m_chElev;         /* elevation angle */
    unsigned char m_byAzimuth;      /* 1/2 the Azimuth angle */
    unsigned char m_byLastMessage;  /* last message processed */
    unsigned char m_bySlip01;       /* cycle slip on chan 1 */

    unsigned short m_wCliForSNR_L1; /* code lock indicator for SNR divided by 32 */
    unsigned short m_wCliForSNR_L2; /* code lock indicator for SNR divided by 32 */

    short         m_nDiffCorr_L1;    /* Differential correction * 100 */
    short         m_nDoppHz;         /* expected doppler in HZ at glonass L1 */

    short         m_nNCOHz_L1;       /* track from NCO in HZ */
    short         m_nNCOHz_L2;       /* track from NCO in HZ */

    short         m_nPosResid_1;    /* position residual 1 * 1000 */
    short         m_nPosResid_2;    /* position residual 2 * 1000 */

} SGALILEOChanData;  /* 24 bytes */

/***************************************************/
/*  SBinaryMsg49    (Galileo E5A, E1B, E1C)        */
/***************************************************/
typedef struct
{
    SUnionMsgHeader    m_sHead;
    long               m_lSecOfWeek;      /* tow  */
    unsigned short     m_wL1usedNavMask;  /* mask of L1 channels used in nav solution */
    unsigned short     m_wL2usedNavMask;  /* mask of L2 channels used in nav solution */
    SGALILEOChanData   m_asChannelData[CHANNELS_12]; /* channel data 12X24 = 288 */
    unsigned short     m_wWeek;          /* week */
    unsigned char      m_bySpare01;      /* spare 1 */
    unsigned char      m_bySpare02;      /* spare 2 */
    unsigned short     m_wCheckSum;      /* sum of all bytes of the header and data */
    unsigned short     m_wCRLF;          /* Carriage Return Line Feed */
} SBinaryMsg49;                          /* length = 8 + 300 + 2 + 2 = 312 */

//-------------------------------------------------------------------------------
// SBinaryMsg36 BeiDou observations (see notes on mesage 76)
// Indivdual pages for B1I, B2I, B3I, etc ... to allow for BeiDou Phase III
// Allows for a maximum of 20 channels
//-------------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader  m_sHead;                //                (8 bytes)
    double           m_dTow;                 // Time in seconds (8 bytes)
    unsigned short   m_wWeek;                // GPS Week Number (2 bytes)
    unsigned short   m_wSpare1;              // spare 1 (zero)  (2 bytes)
```

```
    unsigned long    m_uFreqPage; //[0-19] Spare bits
                                  //[20,21,22,23] Number of Pages
                                  //[24,25,26,27] Page Number
                                  //[28,29,30,31] Signal ID (B1I, B2I, B3I, etc)

    SObsPacket       m_asObs[CHANNELS_20];    // 20 sets of BeiDou observations (20*12=240 bytes)
    unsigned long    m_aulCodeMSBsPRN[CHANNELS_20]; // array of 20 words   (20*4=80 bytes)
                                                    // bit 7:0 (8 bits) = satellite PRN, 0
                                                    // if no satellite
                                                    // bit 12:8 (5 bits) = spare
                                                    // bit 31:13 (19 bits) = upper 19 bits
                                                    // of B1/B2/B3 LSB = 256 meters
                                                    //             MSB = 67108864 meters

    unsigned short   m_wCheckSum;               /// sum of all bytes of the header and data (2 bytes)
    unsigned short   m_wCRLF;                   // Carriage Return Line Feed                (2 bytes)
} SBinaryMsg36;                                 // length = 8 + (8+2+2+4+240+80=336) + 2 + 2 = 348

//--------------------------------------------------------------------------------
// SBinaryMsg16 Generic GNSS observations (see notes on mesage 76)
// 12 observations per message, multiple pages of messages,  See BIN_MSG_HEAD_TYPE16
//--------------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader  m_sHead;                //                (8 bytes)
    double           m_dTow;                 // Time in seconds (8 bytes)
    unsigned short   m_wWeek;                // GPS Week Number (2 bytes)
    unsigned short   m_wSpare1;              // spare 1 (zero)  (2 bytes)

    unsigned long    m_uPageCount; //[0-15] Spare bits
                                   //[16,17,18,19,20,21] Number of Pages = N
                                   //[22,23,24,25,26,27] Page Number [0...N-1]
                                   //[28,29,30,31] Spare bits
                                     // Bit mask of all signals included in the set of pages
    unsigned long    m_uAllSignalsIncluded_01;  // bit 0  = GPS:L1CA included
                                                // bit 1  = GPS:L2P included
                                                // bit 2  = GPS:L2C included
                                                // bit 3  = GPS:L5 included
                                                // bit 4  = GPS:L1C included
                                                // bit 7:5 = spare
                                                // bit 8  = GLO:G1C or GLO:G1P included
                                                // bit 9  = GLO:G2C or GLO:G1P included
                                                // bit 10 = Spare
                                                // bit 11 = Spare
                                                // bit 12 = GLO:G10C included
                                                // bit 13 = GLO:G20C included
                                                // bit 14 = GLO:G30C included
                                                // bit 15 = spare
                                                // bit 16 = GAL:E1BC included
                                                // bit 17 = GAL:E5A included
                                                // bit 18 = GAL:E5B included
                                                // bit 19 = GAL:E6 included
                                                // bit 20 = GAL:ALTBOC included
                                                // bit 23:21 = spare
                                                // bit 24 = BDS:B1I included
                                                // bit 25 = BDS:B2I included
                                                // bit 26 = BDS:B3I included
                                                // bit 27 = BDS:B1BOC included
                                                // bit 28 = BDS:B2A included
                                                // bit 29 = BDS:B2B included
                                                // bit 30 = BDS:B3C included
                                                // bit 31 = BDS:ACEBOC included
    unsigned long    m_uAllSignalsIncluded_02;  // bit 0  = QZS:L1CA included
                                                // bit 1  = spare
                                                // bit 2  = QZS:L2C included
                                                // bit 3  = QZS:L5 included
                                                // bit 4  = QZS:L1C included
                                                // bit 5  = QZS:LEX included
                                                // bit 7:6 = spare
                                                // bit 8  = IRNSS:L5
                                                // bit 31:9 = spare
    SObsPacket       m_asObs[CHANNELS_gen];     // 16 sets of observations (16*12=192 bytes)
    unsigned long    m_aulCodeMSBsPRN[CHANNELS_gen]; // array of 16, 32 bit words   (16*4=64 bytes)
                                                     // bit 7:0 (8 bits) = satellite PRN,
                                                     //                  = 0 if no satellite
                                                     // bit 12:8 (5 bits) = Log_Base_2(X+1)
```

```
                                                //        where X = Time, in units of 1/100th sec,
                                                //        since carrier phase tracking was last stressed
                                                //        or cycle slipped
                                                // bit 31:13 (19 bits) = upper 19 bits
                                                // of code pseudorange LSB = 256 meters
                                                //                     MSB = 67108864 meters
    unsigned short  m_awChanSignalSYS[CHANNELS_gen]; // Array of 16, 16 bit words (32 bytes)
                                               //[15,14]  spare bits
                                               //[13] = 1 if GLONASS P-Code
                                               //[12,11,10,9,8] = Channel (0 is the first channel)
                                               //[7,6,5,4] = Signal ID (L1CA, L5, G1, B1I, B2I, B3I, etc)
                                               // GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3, L1C=4
                                               // GLO Signal ID: G1C/G1P=0, G2C/G2P=1, G10C=4, G20C=5, G30C=6
                                               // GAL Signal ID: E1BC=0, E5A=1, E5B=2, E6=3, ALTBOC=4
                                               // BDS Signal ID: B1I=0, B2I=1, B3I=2,B1BOC=3,B2A=4,B2B=5,B3C=6,ACEBOC=7
                                               // QZS Signal ID: L1CA=0, L2C=2, L5=3, L1C=4
                                               // IRN Signal ID: L5=0
                                               //[3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS,5=INRSS
    unsigned short   m_wCheckSum;              /// sum of all bytes of the header and data (2 bytes)
    unsigned short   m_wCRLF;                  // Carriage Return Line Feed                (2 bytes)
} SBinaryMsg16;                                // length = 8 + (8+2+2+4+4+4+192+64+32=312) + 2 + 2 = 324


//================================================================
//  SGENERICchanData  (was called SBEIDOUChanData)
//
//  Note: Currently we have some redundant stuff in all 3 pages
//        perhaps we should eliminate the redundant stuff
//        and only put in page 1 and not 2 & 3???
//================================================================
typedef struct
{
    unsigned char m_bySV;             // Bit (0-6) = SV slot, 0 == not tracked
    unsigned char m_byAlm_Ephm_Flags; // ephemeris and almanac status flags
                                      // bit 0: Ephemeris available but timed out
                                      // bit 1: Ephemeris valid
                                      // bit 2: Ephemeris health OK
                                      // bit 3: unused
                                      // bit 4: Almanac available
                                      // bit 5: Almanac health OK
                                      // bit 6: unused
                                      // bit 7: Satellite doesn't exist
    unsigned char m_byStatus;         // Status bits (code carrier bit frame...)
    char          m_chElev;           // elevation angle

    unsigned char m_byAzimuth;        // 1/2 the Azimuth angle
    unsigned char m_byLastMessage;    // last message processed
    unsigned char m_bySlip;           // cycle slip on chan 1
    char          m_cFlags;           // RFR_150501 was m_cSpare1;
                                      // [0] bChanEnabled
                                      // [1] bUsedInSolution

    unsigned short  m_wCliForSNR;     // code lock indicator for SNR divided by 32
    short           m_nDiffCorr;      // Differential correction * 100

    short           m_nDoppHz;        // expected doppler in HZ at B1 frequency
    short           m_nNCOHz;         // track from NCO in HZ

    short           m_nPosResid;      // position residual * 1000
    unsigned short  m_wAllocType;     //RFR_150501 was m_nSpare2

} SGENERICchanData; //Changed to generic B1/B2/B3 message 3/18/2013 (20 bytes)

//-------------------------------------------------------------------------------
//  SBinaryMsg39
//  Populates SLXMON window
//  Indivdual pages for B1I, B2I, B3I, etc ... to allow for BeiDou Phase III
//  Allows for a maximum of 20 channels
//-------------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader   m_sHead;          //8 bytes
    long              m_lSecOfWeek;     //tow (4 bytes)

    unsigned long     m_uMaskFreqPage; //[0-19] Mask of channels used in nav solution
                                       //[20,21,22,23] Number of Pages
                                       //[24,25,26,27] Page Number
```

```
                                    //[28,29,30,31] Signal ID (B1I, B2I, B3I, etc)

    SGENERICchanData   m_asChannelData[CHANNELS_20]; //channel data 20x20 = 400
    unsigned short    m_wWeek;          // week
    unsigned short    m_wSpare;         // spare
    unsigned short    m_wCheckSum;      // sum of all bytes of the header and data
    unsigned short    m_wCRLF;          // Carriage Return Line Feed
} SBinaryMsg39;                         // length = 8+(4+4+400+2+2)+2+2 = 8 + (412) + 2 + 2 = 424



//---------------------------------------------------------------------------
//  SBinaryMsg19
//  Generic GNSS message for populating SLXmon windows
//---------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader   m_sHead;        // 8 bytes
    long              m_lSecOfWeek;   // tow (4 bytes)
    unsigned short    m_wGPSWeek;     // GPS Week Number (2 bytes)
    unsigned char     m_byNavMode;    // Nav Mode FIX_NO, FIX_2D, FIX_3D (high bit =has_diff)
    char              m_cUTCTimeDiff; // whole Seconds between UTC and GPS
    unsigned long     m_uPageCount;   // [0-15] Spare bits   (4 bytes)
                                      // [16,17,18,19,20,21] Number of Pages = N
                                      // [22,23,24,25,26,27] Page Number [0...N-1]
                                      // [28,29,30,31] Spare bits
                                            // Bit mask of all signals included in the set of pages
    unsigned long    m_uAllSignalsIncluded_01;  // bit 0  = GPS:L1CA included
                                            // bit 1  = GPS:L2P included
                                            // bit 2  = GPS:L2C included
                                            // bit 3  = GPS:L5 included
                                            // bit 4  = GPS:L1C included
                                            // bit 7:5 = spare
                                            // bit 8  = GLO:G1C or GLO:G1P included
                                            // bit 9  = GLO:G2C or GLO:G1P included
                                            // bit 10 = Spare
                                            // bit 11 = Spare
                                            // bit 12 = GLO:G10C included
                                            // bit 13 = GLO:G20C included
                                            // bit 14 = GLO:G30C included
                                            // bit 15 = spare
                                            // bit 16 = GAL:E1BC included
                                            // bit 17 = GAL:E5A included
                                            // bit 18 = GAL:E5B included
                                            // bit 19 = GAL:E6 included
                                            // bit 20 = GAL:ALTBOC included
                                            // bit 23:21 = spare
                                            // bit 24 = BDS:B1I included
                                            // bit 25 = BDS:B2I included
                                            // bit 26 = BDS:B3I included
                                            // bit 27 = BDS:B1BOC included
                                            // bit 28 = BDS:B2A included
                                            // bit 29 = BDS:B2B included
                                            // bit 30 = BDS:B3C included
                                            // bit 31 = BDS:ACEBOC included
    unsigned long    m_uAllSignalsIncluded_02;  // bit 0  = QZS:L1CA included
                                            // bit 1  = spare
                                            // bit 2  = QZS:L2C included
                                            // bit 3  = QZS:L5 included
                                            // bit 4  = QZS:L1C included
                                            // bit 5  = QZS:LEX included
                                            // bit 7:6 = spare
                                            // bit 8  = IRNSS:L5
                                            // bit 31:9 = spare

    short             m_nClockErrAtL1;// clock error at L1, Hz  (2 bytes)
    unsigned short    m_wSpare1;      // spare (2 bytes)
    SGENERICchanData  m_asChannelData[CHANNELS_gen]; // channel data 16x20 = 320
    unsigned short  m_awChanSignalSYS[CHANNELS_gen]; // Array of 16, 16 bit words (32 bytes)
                                            //[15,14]  spare bits
                                            //[13] = 1 if GLONASS P-Code
                                            //[12,11,10,9,8] = Channel (0 is the first channel)
                                            //[7,6,5,4] = Signal ID (L1CA, L5, G1, B1I, B2I, B3I, etc)
                                            // GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3, L1C=4
                                            // GLO Signal ID = 0:G1C/G1P, 1:G2C/G2P, 4:G10C, 5:G20C, 6:G30C
                                            // GAL Signal ID = 0:E1BC, 1:E5A, 2:E5B, 3:E6,   4:ALTBOC
                                            // BDS Signal ID = 0:B1I,  1:B2I, 2:B3I, 3:B1BOC,4:B2A, 5:B2B, 6:B3C, 7:ACEBOC
```

```
                                        // QZS Signal ID = 0:L1CA, 1:xxx, 2:L2C, 3: L5,  4: L1C
                                        // IRN Signal ID = 0:L5
                                        //[3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS,5=INRSS

    unsigned short    m_wCheckSum;     // sum of all bytes of the header and data
    unsigned short    m_wCRLF;         // Carriage Return Line Feed
} SBinaryMsg19;                        // length = 8+(4+2+1+1+4+4+4+2+2+320+32)+2+2 = 8 + (376) + 2 + 2 = 388


#if defined(_USING_BEIDOU_TIME_OFFSETS_)
//---------------------------------------------------------------------------
// SBinaryMsg34  --- BeiDou -> GPS, ->GLO, -> GAL, ->UTC time offset parameters
// Information is in both D1 and D2, but we are only going to use D1 because
// it is the same data as in D2
//---------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader   m_sHead;       //8 bytes
    int               m_A0UTC;       //BDT clock bias relative to UTC
    int               m_A1UTC;       //BDT clock rate relative to UTC
    short             m_A0GPS;       //BDT clock bias relative to GPS time
    short             m_A1GPS;       //BDT clock rate relative to GPS time
    short             m_A0GAL;       //BDT clock bias relative to Galileo system time
    short             m_A1GAL;       //BDT clock rate relative to Galileo system time
    short             m_A0GLO;       //BDT clock bias relative to GLONASS time
    short             m_A1GLO;       //BDT clock rate relative to GLONASS time
    unsigned char     m_toa;         //Almanac reference time (assuming this is also correct for the time offsets)
    unsigned char     m_Wna;         //almanac week number (assuming this is also correct for the time offsets)
    char              m_dtls;        //Delta time due to leap seconds before the new leap second effective
    unsigned char     m_wnlsf;       //Week number of the new leap second
    unsigned char     m_dn;          //Day number of week of the new leap second
    char              m_dtlsf;       //Delta time due to leap seconds after the new leap second effective
    short             m_spare1;
    short             m_spare2;
    short             m_spare3;
    unsigned short    m_wCheckSum;   //sum of all bytes of the header and data
    unsigned short    m_wCRLF;       // Carriage Return Line Feed
} SBinaryMsg34;                      // length = 8+(4+4+2+2+2+2+2+2+1+1+1+1+1+1+2+2+2=32)+2+2 = 8 + (32) + 2 + 2 = 44
#endif



//---------------------------------------------------------------------------
//  SBinaryMsg44
//    Galileo Time Conversion Parameters
//---------------------------------------------------------------------------
typedef struct
{
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    SUnionMsgHeader m_sHead;          // Header of message.
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  (8 bytes)
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    // Galileo Time to UTC conversion parameters (32 bytes).
    double          m_A0;             // Constant term of polynomial to
                                      //  determine UTC from Galileo Time.
    double          m_A1;             // 1st order term of polynomial to
                                      //  determine UTC from Galileo Time.
    unsigned long   m_tot;            // Reference time for A0 & A1, sec of
                                      //  Galileo week.
    unsigned short  m_wnt;            // Current Galileo reference week.
    unsigned short  m_wnlsf;          // GST Week number when m_dtlsf
                                      //  becomes effective.
    unsigned short  m_dn;             // Day of the week 1 (= Sunday) to
                                      //  7 (= Saturday) when m_dtlsf
                                      //  becomes effective.
    short           m_dtls;           // Cumulative past leap seconds.
    short           m_dtlsf;          // Scheduled future (past) leap
                                      //  seconds.
    unsigned short  m_wSpare1;        // Spare (zero).
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - (32 bytes)
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    // GPS Time to Galileo Time conversion parameters (GGTO Parameters).
    //
    //   dTsys = Tgal - Tgps = m_A0G + m_A1G [TOW - m_t0G + 604800*(WN - m_WN0G)]
    //
    //     where,
    //        dTsys = The time difference between systems
```

```
    //        Tgal  = Galileo Time
    //        Tgps  = GPS Time
    //        TOW   = Galileo Time of Week
    //        WN    = Galileo Week Number
    //        remaining parameters follow.
    double          m_A0G;             // Constant term of GGTO polynomial.
    double          m_A1G;             // 1st order term of GGTO polynomial.
    unsigned long   m_t0G;             // Reference time of week for GGTO.
    unsigned short  m_WN0G;            // Reference week for GGTo.
    unsigned short  m_wGGTOisValid;    // Coded: 0 == GGTO Invalid,
                                       //        1 == GGTO Valid.
                                       //  The Galileo OS-SIS-ICD indicates
                                       //  that when satellite broadcasts
                                       //  all 1 bit values for A0G, A1G,
                                       //  t0G, and WN0G then "the GGTO is
                                       //  considered as not valid."
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - (24 bytes)
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    // Message Tail
    unsigned short  m_wCheckSum;       // Sum of all bytes of the header and
                                       //  data.
    unsigned short  m_wCRLF;           // Carriage Return Line Feed.
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  (4 bytes)
} SBinaryMsg44;                        // length = 8 + (32+24) + 2 + 2 = 68.


/*************************************************/
/*  SBinaryMsg35                                 */
/*************************************************/
typedef SBinaryMsg95 SBinaryMsg35;  //BeiDou ephemeris

/*************************************************/
/*  SBinaryMsg135                                */
/*************************************************/
typedef SBinaryMsg95 SBinaryMsg135;  //BeiDou phase3 ephemeris


/*************************************************/
/*  SBinaryMsg45                                 */
/*************************************************/
typedef SBinaryMsg95 SBinaryMsg45;  //Galileo ephemeris

/*************************************************/
/*  SBinaryMsg55                                 */
/*************************************************/
typedef SBinaryMsg95 SBinaryMsg55;  //IRNSS ephemeris

/*************************************************/
/*  SBinaryMsg125                                */
/*************************************************/
typedef SBinaryMsg95 SBinaryMsg125;  // QZS L5 ephemeris

/*************************************************/
/*  SBinaryMsg195                                */
/*************************************************/
typedef SBinaryMsg95 SBinaryMsg195;  // GPS L5 ephemeris

/*************************************************/
/*  SMsg61Data                                   */
/*************************************************/
typedef struct
{
    unsigned char  bySV;               /* satellite slot  0 == not tracked  */
    unsigned char  byStatusL1;         /* Status bits (code carrier bit frame...)  */
    unsigned char  byStatusL2;         /* Status bits (code carrier bit frame...)  */
    unsigned char  byL1_L2_DCO;        /* 0-3 = upper 4 bits of L1 carrier DCO Phase
                                        * 4-7 = upper 4 bits of L2 carrier DCO Phase
                                        */
    unsigned short wEpochSlewL1;       /* 0-9 = slew, 0 to 1000 count for ms of sec
                                        * 10-15 = 6 bits of L1 slip count */
    unsigned short wEpochCountL1;      /* 0-9 = epoch_count, 0 to 1000 count for ms of sec
                                        * 10-15 = 6 bits of L2 slip count */

    unsigned long  codeph_SNR_L1;      /* 0-20 =  L1 code phase (21 bits = 9+12),
                                        * 21-32 = L1 SNR/4096 (upper 11 of 12 bits) */
    unsigned long  ulCarrierCycles_L1; /* 0-23 = L1 carrier cycles,
                                        * 24-32 = L1 Carrier DCO lower 8 bits */
```

```
    unsigned long  codeph_SNR_L2;      /* 0-20 =  L2 code phase (21 bits = 9+12),
                                        * 21-32 = L2 SNR/4096 (upper 11 of 12 bits) */
    unsigned long  ulCarrierCycles_L2; /* 0-23 = L2 carrier cycles,
                                        * 24-32 = L2 Carrier DCO lower 8 bits */

} SMsg61Data; /* 24 bytes */

/*************************************************/
  /*  SBinaryMsg61                                 */
  /*  Comment: Transmits data from TakemeasGLONASS.c  */
  /*           debugging structure for Dual Freq.     */
  /*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;                 /* 8 */
    unsigned long    m_Tic;                   /* 4 bytes */
    unsigned long    ulSpare;                 /* 4 bytes */
    unsigned short   awHalfWarns[CHANNELS_12]; /* 12*2 = 24 bytes */
                                              /* each word is
                                               * bit 0-2  L1 Half Cycle Warn
                                               * bit 3 = L1 half cycle added
                                               * bit 4-6  L2 Half Cycle Warn
                                               * bit 7 = L2 half cycle added
                                               * 8 =  LSB of 12 bit L1 SNR/4096
                                               * 9 =  LSB of 12 bit L2 SNR/4096
                                               * bit 10-15 Ktag of the SV */

    SMsg61Data       as61Data[CHANNELS_12];   /* 12*24 = 288 bytes */
    unsigned short   m_wCheckSum;             /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                 /* Carriage Return Line Feed */
} SBinaryMsg61;                               /* length = 8 + (320) + 2 + 2 = 332 */



/*************************************************/
/*  SBinaryMsg66  GLONASS OBS (see notes on mesage 76) */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    double           m_dTow;                   /* Time in seconds */
    unsigned short   m_wWeek;                  /* GPS Week Number */
    unsigned short   m_wSpare1;                /* 16 bit spare word */
    unsigned long    m_ulP_Code;               /* Bit [31,24] spare bits */
                                               /* Bit [23,12] Pcode On for m_asL2Obs Obs 0-11, Bit 12 = channel 0*/
                                               /* Bit [11,0]  Pcode On for m_asL1Obs Obs 0-11  Bit 0  = channel 0*/
    SObsPacket       m_asL1Obs[CHANNELS_12];   /* 12 sets of L1(Glonass) observations */
    SObsPacket       m_asL2Obs[CHANNELS_12];   /* 12 sets of L2(Glonass) observations */
    unsigned long    m_aulL1CodeMSBsSlot[CHANNELS_12]; /* array of 12 words.
                                                   bit 7:0 (8 bits) = satellite Slot, 0
                                                   if no satellite
                                                   bit 12:8 (5 bits) = spare
                                                   bit 31:13 (19 bits) = upper 19 bits
                                                   of L1  LSB = 256 meters
                                                          MSB = 67108864 meters */
    unsigned short   m_wCheckSum;              /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                  /* Carriage Return Line Feed */
} SBinaryMsg66;                                /* length = 8 + (352) + 2 + 2 = 364 */


/*************************************************/
/*  SGLONASS_String, added for glonass strings      */
/*************************************************/
typedef struct
{
   unsigned long m_aul85Bits[3];  /* holds bits 9-85 of the GLONASS string  */
                                  /*
                                   * bit order in message 65
                                   *                MSB             LSB
                                   * m_aul85Bits[0]: 85 84...........54
                                   * m_aul85Bits[1]: 53 52...........22
                                   * m_aul85Bits[2]: 21 20......9
                                   */
} SGLONASS_String;                /* 12 bytes (max of 96 bits) */
```

```
/*************************************************/
/*  SBinaryMsg65, added by JL for glonass subframe immediate data + string_5  */
/*************************************************/
/* sent only upon command or when values change (not including changes in tk) */
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char    m_bySV;                        /* The satellite to which this data belongs. */
    unsigned char    m_byKtag;                      /* The satellite K Number + 8. */
    unsigned short   m_wSpare1;                     /* Spare, keeps alignment to 4 bytes */
    unsigned long    m_ulTimeReceivedInSeconds;     /* time at which this arrived */

    SGLONASS_String  m_asStrings[5];                /* first 5 Strings of Glonass Frame (60 bytes) */
    unsigned short   m_wCheckSum;                   /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                       /* Carriage Return Line Feed */
} SBinaryMsg65;                                     /* length = 8 + (68) + 2 + 2 = 80 */


/*****************************************************************/
/*  SBinaryMsg62, Glonass almanac data. Containing string
 *   5 and the two string pair for each satellite after string 5.
 *   String 5 contains the time reference for the glonass almanac
 *   and gps-glonass time differences.
 *
 *****************************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char    m_bySV;                        /* The satellite to which this data belongs. */
    unsigned char    m_byKtag_ch;                   /* Proprietary data */
    unsigned short   m_wSpare1;                     /* Spare, keeps alignment to 4 bytes */
    SGLONASS_String  m_asStrings[3];                /* glonass almanac data  (36 bytes)
                                                       String 0 & 1 = Two almanac Strings, String 2 = ICD String 5*/
    unsigned short   m_wCheckSum;                   /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                       /* Carriage Return Line Feed */
} SBinaryMsg62;                                     /* length = 8 + (40) + 2 + 2 = 52 */



/*************************************************/
/*  SBinaryMsg42 Galileo(42), BeiDou(32), GPS(92) or QZSS(22) Almanac */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char    m_bySV;                        // The satellite to which this data belongs.
    unsigned char    m_bySpare;                     // Spare, keeps alignment to 4 bytes
    unsigned short   m_wSpare1;                     // Spare, keeps alignment to 4 bytes
    long alAlmwords[12];                            // Almanac words (different for different GNSS)
    unsigned short   m_wCheckSum;                   // sum of all bytes of the header and data
    unsigned short   m_wCRLF;                       // Carriage Return Line Feed
} SBinaryMsg42;                                     // length = 8 + (4+48=52) + 2 + 2 = 64

typedef SBinaryMsg42 SBinaryMsg22;  //QZSS Almanac
typedef SBinaryMsg42 SBinaryMsg32;  //BeiDou Almanac
typedef SBinaryMsg42 SBinaryMsg92;  //GPS Almanac
typedef SBinaryMsg42 SBinaryMsg52;  //IRNSS Almanac



//=====================================================================
//  SBinaryMsg109, Log 3-Axis Gyro/Acc
//
//=====================================================================
typedef struct
{
    SUnionMsgHeader    m_sHead;                 //  [8 bytes]
    unsigned short     m_wIRQ;                  //  [2 bytes] Number of IRQs since previous integrated and dump of raw sensor
measurements
    unsigned short     m_wSpare;                //  [2 bytes] spare
    unsigned long      m_ulTIC;                 //  [4 bytes] Current TIC
    unsigned short     m_wGyroCount;            //  [2 bytes] Gyro count for this epoch
    unsigned short     m_wAcclCount;            //  [2 bytes] Acc count for this epoch
    unsigned short     m_wMagnCount;            //  [2 bytes] Mag count for this epoch
    unsigned short     m_wTemperatureCount;     //  [2 bytes] temperature count for this epoch
```

```
    long                m_lGyroSum_X;            //  [4 bytes] accumulated gyro X value for this epoch
    long                m_lGyroSum_Y;            //  [4 bytes] accumulated gyro Y value for this epoch, first 32 bytes
    long                m_lGyroSum_Z;            //  [4 bytes] accumulated gyro Z value for this epoch
    long                m_lAcclSum_X;            //  [4 bytes] accumulated acc X value for this epoch
    long                m_lAcclSum_Y;            //  [4 bytes] accumulated acc Y value for this epoch
    long                m_lAcclSum_Z;            //  [4 bytes] accumulated acc Z value for this epoch
    long                m_lMagnSum_X;            //  [4 bytes] Magnetic X for this epoch
    long                m_lMagnSum_Y;            //  [4 bytes] Magnetic Y for this epoch
    long                m_lMagnSum_Z;            //  [4 bytes] Magnetic Z for this epoch, next 32 bytes
    long                m_lTemperatureSum;  //  [4 bytes] gyro temperature

    unsigned short      m_wAcclError;            //  [2 bytes] Are accl measurements valid or not
    unsigned short      m_wGyroError;            //  [2 bytes] Are gyro measurements valid or not
    unsigned short      m_wCheckSum;             //  [2 bytes] sum of all bytes of the header and data
    unsigned short      m_wCRLF;                 //  [2 bytes] Carriage Return Line Feed
} SBinaryMsg109;  //


//xx #if defined(WIN32) || (__ARMCC_VERSION>=300441)  // all compilers that we use today
    #pragma pack(pop)
//xx #endif

#ifdef __cplusplus
}
#endif
#endif // __BinaryMsg_H_
```

Bin1 Message

Message Type:	Binary
Description:	GNSS position message (position and velocity data)
Message Format:	Command Format to Request Message: $JBIN,1,r<CR><LF> where: '1' = Bin1 message 'r' = message rate in Hz (20, 10, 2, 1, 0, or .2) Message Format:

Message Component	Description	Type	Bytes
AgeOfDiff	Age of differential, seconds. Use Extended AgeOfDiff first. If both = 0, then no differential. Values range from 0 to 255.	unsigned char	1
NumOfSats	Number of satellites used in the GPS solution. Values range from 0 to 255	unsigned char	1
GPSWeek	GPS week associated with this message. Values ranges from 0 to 65535.	unsigned short	2
GPSTimeOfWeek	GPS tow (sec) associated with this message. Values range from 0.0 to	double	8

		604800.0		
	Latitude	Latitude in degrees north. Values range from -90.0 to 90.0	double	8
	Longitude	Longitude in degrees East. Values range from -180.0 to 180.0	double	8
	Height	Altitude above the ellipsoid in meters	float	
	VNorth	Velocity north in m/s	float	
	VEast	Velocity east in m/s	float	
	Vup	Velocity up in m/s	float	
	StdDevResid	Standard deviation of residuals in meters. When value is positive.	float	4
	NavMode	Navigation mode: 0 = No fix 1 = Fix 2d no diff 2 =Fix 3d no diff 3 = Fix 2D with diff 4 = Fix 3D with diff 5 =RTK float 6 = RTK integer fixed When: $JDISNAVMODE,PHOENIX is enabled 7 = RTK float (SureFix enabled) 8 = RTK integer fixed (SureFix enabled) 9 = RTK SureFixed 10 = aRTK integer fixed 11 = aRTK float 12 = aRTK Atlas converged 13 = aRTK Atlas un- converged 14 = Atlas converged 15 = Atlas un-converged Bits 0 through 6 =Navmode Bit 7 = Manual mark If bit 7 is set (left-most bit), then this is a manual position	unsigned short	2
	Extended AgeOfDiff	Extended age of differential, seconds. If 0, use 1 byte AgeOfDiff listed above	unsigned short	2

Structure:

```
/**************************************************/
/*  SBinaryMsg1                                   */
/**************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char  m_byAgeOfDiff;       /* age of differential, seconds (255 max)*/
    unsigned char  m_byNumOfSats;       /* number of satellites used (12 max)    */
```

	`unsigned short m_wGPSWeek; /* GPS week */` `double m_dGPSTimeOfWeek; /* GPS tow */` `double m_dLatitude; /* Latitude degrees, -90..90 */` `double m_dLongitude; /* Longitude degrees, -180..180 */` `float m_fHeight; /* (m), Altitude ellipsoid */` `float m_fVNorth; /* Velocity north m/s */` `float m_fVEast; /* Velocity east m/s */` `float m_fVUp; /* Velocity up m/s */` `float m_fStdDevResid; /* (m), Standard Deviation of Residuals */` `unsigned short m_wNavMode;` `unsigned short m_wAgeOfDiff; /* age of diff using 16 bits */` `unsigned short m_wCheckSum; /* sum of all bytes of the header and data */` `unsigned short m_wCRLF; /* Carriage Return Line Feed */` `} SBinaryMsg1; /* length = 8 + 52 + 2 + 2 = 64 */`
Additional Information:	Message has a BlockID of 1 and is 52 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.0 / June 30, 2020

Bin2 Message

Message Type:	Binary
Description:	GPS DOPs (Dilution of Precision) This message contains various quantities that are related to the GNSS solution, such as satellites tracked, satellites used, and DOPs.
Message Format:	Command Format to Request Message: $JBIN,2,r<CR><LF> where: •'2' = Bin2 message •'r' = message rate in Hz (1 or 0) Message Format:

Message Component	Description	Type	Bytes
MaskSatsTracked	Mask of satellites tracked by the GPS. Bit 0 corresponds to the GPS satellite with PRN 1. Individual bits represent satellites	unsigned long	4
MaskSatsUsed	Mask of satellites used in the GPS solution. Bit 0 corresponds to the GPS satellite with PRN 1. Individual bits represent satellites	unsigned long	4
GpsUtcDiff	Whole seconds between UTC and GPS time (GPS minus UTC). Where the value is positive.	unsigned short	2
HDOPTimes10	Horizontal dilution of precision scaled by10 (0.1 units). Where the value is positive.	unsigned short	2

		VDOPTimes10	Vertical dilution of precision scaled by 10 (0.1 units). Where the value is positive.	unsigned short	2
		WAASMask	PRN and tracked or used status masks where: . Bit 00 - Mask of satellites tracked by first WAAS satellite Bit 01 - Mask of satellites tracked by second WAAS satellite Bit 02 - Mask of satellites used by first WAAS satellite Bit 03 - Mask of satellites used by second WAAS satellite Bit 04 – Unused Bit 05 – Bit 09-first WAAS satellite PRN minus 120 Bit 10- Bit 14-second WAAS satellite PRN minus 120	unsigned short	2

Structure:

```
/**************************************************/
/*  SBinaryMsg2                                   */
/**************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned long  m_ulMaskSatsTracked;  /* SATS Tracked, bit mapped 0..31 */
    unsigned long  m_ulMaskSatsUsed;     /* SATS Used, bit mapped 0..31 */
    unsigned short m_wGpsUtcDiff;        /* GPS/UTC time difference (GPS minus UTC) */
    unsigned short m_wHDOPTimes10;       /* HDOP        (0.1 units) */
    unsigned short m_wVDOPTimes10;       /* VDOP        (0.1 units) */
    unsigned short m_wWAASMask;          /* Bits 0-1: tracked sats, Bits 2-3:
                                            used sats, Bits 5-9 WAAS PRN 1 minus
                                            120, Bits 10-14 WAAS PRN 1 minus 120 */
    unsigned short m_wCheckSum;          /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;              /* Carriage Return Line Feed */
} SBinaryMsg2;                           /* length = 8 + 16 + 2 + 2 = 28 */
```

Additional Information:	Message has a BlockID of 2 and is 16 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.0 / June 30, 2020

Bin3 Message

Message Type:	Binary
Description:	Lat/Lon/Hgt, Covariances, RMS, DOPs and COG, Speed,Heading.

Message Format:

Command Format to Request Message:

$JBIN,3,r<CR><LF>

where:

'3' = Bin3 message

'r' = message rate in Hz

Message Format:

Message Component	Description	Type	Bytes
GPSTimeOfWeek	GPS tow (sec) associated with this message	double	GPSTimeOfWeek
GPSWeek	GPS week associated with this message	unsigned short	2
SATS Tracked	Number of satellites tracked in the GPS solution	unsigned short	2
NumOfSats	Number of satellites used in the GPS solution	unsigned short	2
NAV Mode	Navigation mode: 0 = No fix 1 = Fix 2d no diff 2 = Fix 3d no diff 3 = Fix 2D with diff 4 = Fix 3D with diff 5 = RTK float 6 = RTK integer fixed When $JDISNAVMODE,PHOENIX enabled 7 = RTK float (SureFix enabled) 8 = RTK integer fixed (SureFix enabled) 9 = RTK SureFixed 10 = aRTK integer fixed 11 = aRTK float 12 = aRTK Atlas converged 13 = aRTK Atlas un-converged 14 = Atlas converged 15 = Atlas un-converged If bit 7 is set (left-most bit), then this is a manual position	unsigned char	1
Spare		unsigned char	1
Latitude	Latitude in degrees north	double	8
Longitude	Longitude in degrees east	double	8
Height	Altitude above the ellipsoid in meters	float	4

Horizontal Speed	Velocity horizontal in m/s	float	4
Vup	Velocity up in m/s	float	4
COG	Course over Ground, degrees	float	4
Heading	Heading(degrees), Zero unless vector	float	4
Pitch	Pitch (degrees), Zero unless vector	float	4
Roll	Roll (degrees), Zero unless vector	float	4
AgeOfDiff	Age of differential, seconds. Use Extended AgeOfDiff first. If both = 0, then no differential	unsigned short	2
Attitude Status	Attitude Status, zero unless vector Bits 0 – 3 = status.eYaw Bits 4 – 7 = status.ePitch Bits 8 – 11 = status.eRoll Where status can be 0=INVALID, 1=GNSS, 2=Inertial, 3=Magnetic	unsigned short	4
StdDevHeading	Yaw stddev (degrees), zero unless vector	float	4
StdDevPitch	Pitch stddev (degrees), zero unless vector	float	4
HRMS	Horizontal RMS	float	4
VRMS	Vertical RMS	float	4
HDOP	Horizontal DOP	float	4
VDOP	Vertical DOP	float	4
TDOP	Time DOP	float	4
CovNN	Covaraince North-North	float	4
CovNE	Covaraince North-East	float	4
CovNU	Covaraince North-Up	float	4
CovEE	Covaraince East-East	float	4
CovEU	Covaraince East-Up	float	4
CovUU	Covaraince Up-Up	float	4

Structure:

```
//-*****************************************************
//-*  SBinaryMsg3
//-*  Lat/Lon/Hgt, Covariances, RMS, DOPs and COG, Speed, Heading
//-*****************************************************
typedef struct
{
    SUnionMsgHeader  m_sHead;              //                              [8]
    double           m_dGPSTimeOfWeek;     // GPS tow                      [8 bytes]
    unsigned short   m_wGPSWeek;           // GPS week                     [2 bytes]
    unsigned short   m_wNumSatsTracked;    // SATS Tracked                 [2 bytes]
    unsigned short   m_wNumSatsUsed;       // SATS Used                    [2 bytes]
```

```
    unsigned char    m_byNavMode;          // Nav Mode (same as message 1)         [1 byte ]
    unsigned char    m_bySpare00;          // Spare                                [1 byte ]
    double           m_dLatitude;          // Latitude degrees, -90..90            [8 bytes]
    double           m_dLongitude;         // Longitude degrees, -180..180         [8 bytes]
    float            m_fHeight;            // (m), Altitude ellipsoid              [4 bytes]
    float            m_fSpeed;             // Horizontal Speed   m/s               [4 bytes]
    float            m_fVUp;               // Vertical Velocity +up  m/s           [4 bytes]
    float            m_fCOG;               // Course over Ground, degrees          [4 bytes]
    float            m_fHeading;           // Heading (degrees), Zero unless vector[4 bytes]
    float            m_fPitch;             // Pitch (degrees), Zero unless vector  [4 bytes]
    float            m_fRoll;              // Roll (degrees), Zero unless vector   [4 bytes]
    unsigned short   m_wAgeOfDiff;         // age of differential, seconds         [2 bytes]
    // m_wAttitudeStatus: bit {0-3}  = sStatus.eYaw
    //                    bit {4-7}  = sStatus.ePitch
    //                    bit {8-11} = sStatus.eRoll
    // where sStatus can be 0 = INVALID, 1 = GNSS, 2 = Inertial, 3= Magnetic
    unsigned short   m_wAttitudeStatus;    // Attitude Status, Zero unless vector  [2 bytes]
    float            m_fStdevHeading;      // Yaw stdev, degrees, 0 unless vector  [4 bytes]
    float            m_fStdevPitch;        // Pitch stdev, degrees, 0 unless vector[4 bytes]
    float            m_fHRMS;              // Horizontal RMS                       [4 bytes]
    float            m_fVRMS;              // Vertical   RMS                       [4 bytes]
    float            m_fHDOP;              // Horizontal DOP                       [4 bytes]
    float            m_fVDOP;              // Vertical DOP                         [4 bytes]
    float            m_fTDOP;              // Time DOP                             [4 bytes]
    float            m_fCovNN;             // Covaraince North-North               [4 bytes]
    float            m_fCovNE;             // Covaraince North-East                [4 bytes]
    float            m_fCovNU;             // Covaraince North-Up                  [4 bytes]
    float            m_fCovEE;             // Covaraince East-East                 [4 bytes]
    float            m_fCovEU;             // Covaraince East-Up                   [4 bytes]
    float            m_fCovUU;             // Covaraince Up-Up                     [4 bytes]
    unsigned short   m_wCheckSum;          // sum of all bytes of the header and data
    unsigned short   m_wCRLF;              // Carriage Return Line Feed
} SBinaryMsg3;                             // length = 8 + 116 + 2 + 2 = 128  (108 = 74 hex)
```

Additional Information:	Message has a BlockID of 3 and is 116 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.11 / November 15, 2018

Bin5 Message

Message Type:	Binary
Description:	Base station information
Message Format:	Command Format to Request Message: $JBIN,5,r<CR><LF> where: •'5' = Bin5 message •'r' = message rate in Hz Message Format:

Message Component	Description	Type	Bytes
Latitude	Latitude of base station in degrees north. Where values range from -90.0 to 90.0	double	8
Longitude	Longitude of base station in degrees east. Where values	double	8

		range from -180.0 to 180.0		
	Height	Base station altitude in meters	float	4
	BaseID	Base station ID Where values range from 0 to 65535	unsigned short	2
	Spare		unsigned short	2
	DiffFormat	String giving the format of the differential (i.e. RTCM3)	char array	16*1 = 16
	Spare		unsigned short array	16*2 = 32

Structure:

```
//-*****************************************************
//-*  SBinaryMsg5
//-*  Base Location and Base ID
//-*****************************************************
typedef struct
{
    SUnionMsgHeader  m_sHead;              //                                    [8]
    double           m_dLatitude;          // Base Latitude degrees, -90..90     [8 bytes]
    double           m_dLongitude;         // Base Longitude  degrees, -180..180 [8 bytes]
    float            m_fHeight;            // Base Altitude ellipsoid, (m)       [4 bytes]
    unsigned short   m_wBaseID;            // BaseID                             [2 bytes]
    unsigned short   m_wSpare;             // Spare                              [2 bytes]
    char             m_szDiffFormat[16];   // String giving format of Differential [16 bytes]
    unsigned short   m_awSpare[16];        // 32 bytes of spare                  [32 bytes]
    unsigned short   m_wCheckSum;          // sum of all bytes of the header and data
    unsigned short   m_wCRLF;              // Carriage Return Line Feed
} SBinaryMsg5;                             // length = 8 + 72 + 2 + 2 = 84   (72 = 48 hex)
```

Additional Information: Message has a BlockID of 5 and is 72 bytes, excluding the header and epilogue

Related Commands and Messages: JBIN

Topic Last Updated: v1.09 / January 8, 2018

Bin6 Message

Message Type: Binary

Description: Manual Mark Tag

Message Format: Command Format to Request Message:

Message Format:
The message is output when the manual mark is triggered.

Message Component	Description	Type	Bytes
Time of Week	GPS tow (sec) associated with this message. Where values range from 0.0 to 604800.0	double	8
GPS Week	GPS week associated with this message. Where values range from 0 to 65535	unsigned short	2
Spare	Spare		

Structure:

```
/*****************************************************/
```

	/* SBinaryMsg6 Manual Mark Tag Preceding MM messages*/ /***/ typedef struct { SUnionMsgHeader m_sHead; double m_dTow; /* Time in seconds */ unsigned short m_wWeek; /* GPS Week Number */ unsigned short m_wSpare1; /* 16 bit spare word */ unsigned short m_wCheckSum; /* sum of all bytes of the header and data */ unsigned short m_wCRLF; /* Carriage Return Line Feed */ } SBinaryMsg6; /* length = 8 + (12) + 2 + 2 = 24 */
Additional Information:	Message has a BlockID of 6 and is 12 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: June 1, 2018

Bin16 Message

Message Type:	Binary
Description:	Generic GNSS observations (see notes on message 76)
Message Format:	Command Format to Request Message: Message Format: $JBIN,16,r<CR><LF> where: •'16' = Bin16 message •'r' = message rate in Hz (1 or 0) Message Format:

Message Component	Description	Type	Bytes
Tow	Time in seconds	double	8
Week	GPS week number. When values range from 0-65535	unsigned short	2
Spare1	Future Use	unsigned short	2
PageCount [0-15] Spare bits [16,17,18,19,20,21] Number of Pages =N [22,23,24,25,26,27] Page Number [0...N-1] [28,29,30,31] Spare bits	Page information	unsigned long	4
AllSignalsIncluded_01	Bit mask of all signals included in the set of pages	unsigned long	4

		bit 0 = GPS:L1CA included bit 1 = GPS:L2P included bit 2 = GPS:L2C included bit 3 = GPS:L5 included bit 7:4 = spare bit 8 = GLO:G1C or GLO:G1P included bit 9 = GLO:G2C or GLO:G1P included it 15:10 = spare bit 16 = GAL:E1BC included bit 17 = GAL:E5A included bit 18 = GAL:E5B included bit 23:19 = spare bit 24 = BDS:B1I included bit 25 = BDS:B2I included bit 26 = BDS:B3I included // bit 27 = BDS:B1BOC included // bit 28 = BDS:B2A included // bit 29 = BDS:B2B included // bit 30 = BDS:B3C included // bit 31 = BDS:ACEBOC included			
		AllSignalsInclude d_02 bit 0 = QZS:L1CA included bit 1 = spare bit 2 = QZS:L2C included bit 3 = QZS:L5 included bit 4 = QZS:L1C included bit 5 = QZS:LEX included bit 6 = spare bit 7 = spare bit 8 = IRNSS:L5 included bit 31:9 = spare	Bit mask of all signals included in the set of pages	unsigned long	4
		Obs[16]	16 sets of observations	structure array	**16*12 =**192
		CodeMSBsPRN bit 7:0 (8 bits) = satellite PRN,= 0 if no satellite bit 12:8 (5 bits) = Log_Base_2(X+1) where X = Time, in units of 1/100th sec, since carrier phase tracking was last stressed or	Array of 16 32-bit words	arrayof unsigned longs	16*4=64

	cycle slipped bit 31:13 (19 bits) = upper 19 bits of code pseudorange LSB = 256 meters MSB = 67108864 meter[15,14] spare bits [13] = 1 if GLONASS P-Code [12,11,10,9,8] = Channel (0 is the first channel) [7,6,5,4] = Signal ID (L1CA, L5, G1, B1I, B2I, B3I, etc) GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3 GLO Signal ID: G1C/G1P=0, G2C/G2P=1, G10C=4, G20C=5, G30C=6 GAL Signal ID: E1BC=0, E5A=1, E5B=2, E6=3, ALTBOC=4 BDS Signal ID: B1I=0, B2I=1, B3I=2,B1BOC=3,B2A=4,B2B=5,B3C=6,ACEBOC=7 QZS Signal ID: L1CA=0, L2C=2, L5=3, L1C=4 [3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS, 4=QZS IRNSS NavIC signal ID: L5=0 [3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS, 4=QZS, 5=IRNSS NavIC			
	CheckSum	Sum of all bytes of header and data	unsigned short	2
	CRLF	Carriage return line feed	unsigned short	2

Structure:

```
//------------------------------------------------------------------------------
// SBinaryMsg16 Generic GNSS observations (see notes on mesage 76)
// 12 observations per message, multiple pages of messages,  See BIN_MSG_HEAD_TYPE16
//------------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader  m_sHead;                //                (8 bytes)
    double           m_dTow;                 // Time in seconds (8 bytes)
    unsigned short   m_wWeek;                // GPS Week Number (2 bytes)
    unsigned short   m_wSpare1;              // spare 1 (zero)  (2 bytes)

    unsigned long    m_uPageCount; //[0-15] Spare bits
                                   //[16,17,18,19,20,21] Number of Pages = N
                                   //[22,23,24,25,26,27] Page Number [0...N-1]
                                   //[28,29,30,31] Spare bits
                                     // Bit mask of all signals included in the set of pages
    unsigned long    m_uAllSignalsIncluded_01;  // bit 0  = GPS:L1CA included
                                     // bit 1  = GPS:L2P included
                                     // bit 2  = GPS:L2C included
                                     // bit 3  = GPS:L5 included
                                     // bit 4  = GPS:L1C included
                                     // bit 7:5 = spare
                                     // bit 8  = GLO:G1C or GLO:G1P included
                                     // bit 9  = GLO:G2C or GLO:G1P included
                                     // bit 10 = Spare
                                     // bit 11 = Spare
                                     // bit 12 = GLO:G10C included
                                     // bit 13 = GLO:G20C included
                                     // bit 14 = GLO:G30C included
                                     // bit 15 = spare
                                     // bit 16 = GAL:E1BC included
                                     // bit 17 = GAL:E5A included
```

```
                                        // bit 18 = GAL:E5B included
                                        // bit 19 = GAL:E6 included
                                        // bit 20 = GAL:ALTBOC included
                                        // bit 23:21 = spare
                                        // bit 24 = BDS:B1I included
                                        // bit 25 = BDS:B2I included
                                        // bit 26 = BDS:B3I included
                                        // bit 27 = BDS:B1BOC included
                                        // bit 28 = BDS:B2A included
                                        // bit 29 = BDS:B2B included
                                        // bit 30 = BDS:B3C included
                                        // bit 31 = BDS:ACEBOC included
    unsigned long    m_uAllSignalsIncluded_02; // bit 0  = QZS:L1CA included
                                        // bit 1  = spare
                                        // bit 2  = QZS:L2C included
                                        // bit 3  = QZS:L5 included
                                        // bit 4  = QZS:L1C included
                                        // bit 5  = QZS:LEX included
                                        // bit 7:6 = spare
                                        // bit 8  = IRNSS:L5
                                        // bit 31:9 = spare
    SObsPacket       m_asObs[CHANNELS_gen];     // 16 sets of observations (16*12=192 bytes)
    unsigned long    m_aulCodeMSBsPRN[CHANNELS_gen]; // array of 16, 32 bit words   (16*4=64 bytes)
                                          // bit 7:0 (8 bits) = satellite PRN,
                                          //                  = 0 if no satellite
                                          // bit 12:8 (5 bits) = Log_Base_2(X+1)
                                          //        where X = Time, in units of 1/100th sec,
                                          //        since carrier phase tracking was last stressed
                                          //        or cycle slipped
                                          // bit 31:13 (19 bits) = upper 19 bits
                                          // of code pseudorange LSB = 256 meters
                                          //                      MSB = 67108864 meters
    unsigned short  m_awChanSignalSYS[CHANNELS_gen]; // Array of 16, 16 bit words (32 bytes)
                                        //[15,14]  spare bits
                                        //[13] = 1 if GLONASS P-Code
                                        //[12,11,10,9,8] = Channel (0 is the first channel)
                                        //[7,6,5,4] = Signal ID (L1CA, L5, G1, B1I, B2I, B3I, etc)
                                        // GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3, L1C=4
                                        // GLO Signal ID: G1C/G1P=0, G2C/G2P=1, G10C=4, G20C=5, G30C=6
                                        // GAL Signal ID: E1BC=0, E5A=1, E5B=2, E6=3, ALTBOC=4
                                        // BDS Signal ID: B1I=0, B2I=1,
B3I=2,B1BOC=3,B2A=4,B2B=5,B3C=6,ACEBOC=7
                                        // QZS Signal ID: L1CA=0, L2C=2, L5=3, L1C=4
                                        // IRN Signal ID: L5=0
                                        //[3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS,5=INRSS
    unsigned short   m_wCheckSum;       /// sum of all bytes of the header and data (2 bytes)
    unsigned short   m_wCRLF;           // Carriage Return Line Feed                (2 bytes)
} SBinaryMsg16;                         // length = 8 + (8+2+2+4+4+4+192+64+32=312) + 2 + 2 = 324


//================================================================
//  SGENERICchanData  (was called SBEIDOUChanData)
//
//  Note: Currently we have some redundant stuff in all 3 pages
//        perhaps we should eliminate the redundant stuff
//        and only put in page 1 and not 2 & 3???
//================================================================
typedef struct
{
    unsigned char m_bySV;             // Bit (0-6) = SV slot, 0 == not tracked
    unsigned char m_byAlm_Ephm_Flags; // ephemeris and almanac status flags
                                      // bit 0: Ephemeris available but timed out
                                      // bit 1: Ephemeris valid
                                      // bit 2: Ephemeris health OK
                                      // bit 3: unused
                                      // bit 4: Almanac available
                                      // bit 5: Almanac health OK
                                      // bit 6: unused
                                      // bit 7: Satellite doesn't exist
    unsigned char m_byStatus;         // Status bits (code carrier bit frame...)
    char          m_chElev;           // elevation angle

    unsigned char m_byAzimuth;        // 1/2 the Azimuth angle
    unsigned char m_byLastMessage;    // last message processed
    unsigned char m_bySlip;           // cycle slip on chan 1
    char          m_cFlags;           // RFR_150501 was m_cSpare1;
                                      // [0] bChanEnabled
                                      // [1] bUsedInSolution

    unsigned short  m_wCliForSNR;     // code lock indicator for SNR divided by 32
    short           m_nDiffCorr;      // Differential correction * 100

    short           m_nDoppHz;        // expected doppler in HZ at B1 frequency
    short           m_nNCOHz;         // track from NCO in HZ

    short           m_nPosResid;      // position residual * 1000
    unsigned short  m_wAllocType;     //RFR_150501 was m_nSpare2

} SGENERICchanData; //Changed to generic B1/B2/B3 message 3/18/2013 (20 bytes)
```

Additional Information:	Message has a BlockID of 16 and is 312 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.2 / September 13, 2022

Bin19 Message

Message Type:	Binary
Description:	GNSS diagnostic information
Message Format:	Command Format to Request Message: $JBIN,19,r<CR><LF> where: '19' = Bin19 message 'r' = message rate in Hz (1 or 0)

Message Component	Description	Type	Bytes
SecOfWeek	Time of Week	long	4
GPSWeek	GPS Week Number	unsigned short	2
NavMode	Nav Mode. Where values range from 0-255	unsigned char	1
UTCTimeDiff	Whole seconds between UTC and GPS time	char	1
PageCount	Information about the paging of the BIN19 message. Bits [16,17,18,19,20,21] Number of Pages = N Bits [22,23,24,25,26,27] Page Number [0...N-1]	unsigned long	4
AllSignalsIncludes01	Bitmask of all signals includes in this set of pages bit 0 = GPS:L1CA included bit 1 = GPS:L2P included bit 2 = GPS:L2C included bit 3 = GPS:L5 included bit 7:4 = spare bit 8 = GLO:G1C or GLO:G1P included bit 9 = GLO:G2C or GLO:G1P included bit 10 = Spare bit 11 = Spare	unsigned long	4

			bit 12 = GLO:G10C included bit 13 = GLO:G10C included bit 14 = GLO:G10C included bit 15 = spare bit 16 = GAL:E1BC included bit 17 = GAL:E5A included bit 18 = GAL:E5B included bit 19 = GAL:E6 included bit 20 = GAL:ALTBOC included bit 23:21 = spare bit 24 = BDS:B1I included bit 25 = BDS:B2I included bit 26 = BDS:B3I included bit 27 = BDS:B1BOC included bit 28 = BDS:B2A included bit 29 = BDS:B2B included bit 30 = BDS:B3C included bit 31 = BDS:ACEBOC included bit 1 = spare bit 2 = QZS:L2C included bit 3 = QZS:L5 included bit 4 = QZS:L1C included bit 5 = QZS:LEX included bit 6 = spare bit 7 = spare bit 8 = IRNSS:L5 included bit 31:9 = spare		
		AllSignalsIncluded02	Continued bitmask of all signals included in this set of pages.	unsigned long	4
		Spare		unsigned short	
		ChannelData[16]	Detailed data for each signal included.	SGENERIC chanData[]	
		ChanSignalSYS	Information about the type of signal represented by each entry in ChannelData [15,14] spare bits [13] = 1 if GLONASS P-Code [12,11,10,9,8] = Channel (0 is the first channel) [7,6,5,4] = Signal ID (L1CA, L5, G1, B1I, B2I, B3I, etc) GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3 GLO Signal ID: G1C/G1P=0, G2C/G2P=1, G10C=4, G20C=5, G30C=6 GAL Signal ID: E1BC=0, E5A=1, E5B=2, E6=3, ALTBOC=4 BDS Signal ID: B1I=0, B2I=1, B3I=2,B1BOC=3,B2A=4,B2B=5, B3C=6,ACEBOC=7	unsigned short[]	32

		QZS Signal ID: L1CA=0, L2C=2, L5=3, L1C=4 IRNSS NavIC Signal ID: L5=0 [3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS, 4=QZS, 5=IRNSS NavIC		
	CheckSum	Sum of all bytes of header and data	unsigned short	2
	CRLF	Carriage return line feed	unsigned short	2

Structure:

```
//------------------------------------------------------------------------
//  SBinaryMsg19
//  Generic GNSS message for populating SLXmon windows
//------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader   m_sHead;        // 8 bytes
    long              m_lSecOfWeek;   // tow (4 bytes)
    unsigned short    m_wGPSWeek;     // GPS Week Number (2 bytes)
    unsigned char     m_byNavMode;    // Nav Mode FIX_NO, FIX_2D, FIX_3D (high bit =has_diff)
    char              m_cUTCTimeDiff; // whole Seconds between UTC and GPS
    unsigned long     m_uPageCount;   // [0-15] Spare bits   (4 bytes)
                                      // [16,17,18,19,20,21] Number of Pages = N
                                      // [22,23,24,25,26,27] Page Number [0...N-1]
                                      // [28,29,30,31] Spare bits
                                            // Bit mask of all signals included in the set of pages
    unsigned long    m_uAllSignalsIncluded_01;  // bit 0  = GPS:L1CA included
                                            // bit 1  = GPS:L2P included
                                            // bit 2  = GPS:L2C included
                                            // bit 3  = GPS:L5 included
                                            // bit 4  = GPS:L1C included
                                            // bit 7:5 = spare
                                            // bit 8  = GLO:G1C or GLO:G1P included
                                            // bit 9  = GLO:G2C or GLO:G1P included
                                            // bit 10 = Spare
                                            // bit 11 = Spare
                                            // bit 12 = GLO:G10C included
                                            // bit 13 = GLO:G20C included
                                            // bit 14 = GLO:G30C included
                                            // bit 15 = spare
                                            // bit 16 = GAL:E1BC included
                                            // bit 17 = GAL:E5A included
                                            // bit 18 = GAL:E5B included
                                            // bit 19 = GAL:E6 included
                                            // bit 20 = GAL:ALTBOC included
                                            // bit 23:21 = spare
                                            // bit 24 = BDS:B1I included
                                            // bit 25 = BDS:B2I included
                                            // bit 26 = BDS:B3I included
                                            // bit 27 = BDS:B1BOC included
                                            // bit 28 = BDS:B2A included
                                            // bit 29 = BDS:B2B included
                                            // bit 30 = BDS:B3C included
                                            // bit 31 = BDS:ACEBOC included
    unsigned long    m_uAllSignalsIncluded_02;  // bit 0  = QZS:L1CA included
                                            // bit 1  = spare
                                            // bit 2  = QZS:L2C included
                                            // bit 3  = QZS:L5 included
                                            // bit 4  = QZS:L1C included
                                            // bit 5  = QZS:LEX included
                                            // bit 7:6 = spare
                                            // bit 8  = IRNSS:L5
                                            // bit 31:9 = spare

    short             m_nClockErrAtL1;// clock error at L1, Hz  (2 bytes)
    unsigned short    m_wSpare1;      // spare (2 bytes)
    SGENERICchanData  m_asChannelData[CHANNELS_gen]; // channel data 16x20 = 320
    unsigned short  m_awChanSignalSYS[CHANNELS_gen]; // Array of 16, 16 bit words (32 bytes)
                                            //[15,14]  spare bits
                                            //[13] = 1 if GLONASS P-Code
                                            //[12,11,10,9,8] = Channel (0 is the first channel)
                                            //[7,6,5,4] = Signal ID (L1CA, L5, G1, B1I, B2I, B3I, etc)
                                            // GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3, L1C=4
                                            // GLO Signal ID = 0:G1C/G1P, 1:G2C/G2P, 4:G10C, 5:G20C, 6:G30C
                                            // GAL Signal ID = 0:E1BC, 1:E5A, 2:E5B, 3:E6,   4:ALTBOC
                                            // BDS Signal ID = 0:B1I,  1:B2I, 2:B3I, 3:B1BOC,4:B2A, 5:B2B,
6:B3C, 7:ACEBOC
                                            // QZS Signal ID = 0:L1CA, 1:xxx, 2:L2C, 3: L5,  4: L1C
                                            // IRN Signal ID = 0:L5
                                            //[3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS,5=INRSS

    unsigned short    m_wCheckSum;     // sum of all bytes of the header and data
    unsigned short    m_wCRLF;         // Carriage Return Line Feed
```

	`} SBinaryMsg19;  // length = 8+(4+2+1+1+4+4+4+2+2+320+32)+2+2 = 8 + (376) + 2 + 2 = 388` `#if defined(_USING_BEIDOU_TIME_OFFSETS_)`
Additional Information:	Message has a BlockID of 35 and is 376 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.2 / September 13, 2022

Bin22 Message

Message Type:	Binary
Description:	QZSS Almanac
Message Format:	Command Format to Request Message: $JBIN,22,r<CR><LF> where: '22' = Bin22 message 'r' = 1 to turn on the message, 0 to turn off the message Message Format:

Message Component	Description	Type	Bytes
PRN Satellite	PRN	unsigned char	1
Spare1	Spare	unsigned char	1
Spare2	Spare	unsigned char	2
Almwords	Almanac Words	long[12]	12*4 = 48

alAlmwords[0]	toa
alAlmwords[1]	$\sqrt{A}$
alAlmwords[2]	e
alAlmwords[3]	ω
alAlmwords[4]	M0
alAlmwords[5]	Ω_0
alAlmwords[6]	$\dot{\Omega}$
alAlmwords[7]	δ_i
alAlmwords[8]	SV Health
alAlmwords[9]	WNa

All of the scalefactors, number of bits, and units can be found in the BeiDou ICD on page 37

http://en.beidou.gov.cn/SYSTEMS/I

			CD/201806/P020180608523308843290.pdf		
Structure:					
Additional Information:					
Related Commands and Messages:					

Topic Last Updated: v2.0 / April 30, 2019

Bin32 Message

Message Type:	Binary
Description:	BeiDou Almanac
Message Format:	Command Format to Request Message: $JBIN,32,r<CR><LF> where: '32' = Bin32 message 'r' = 1 to turn on the message, 0 to turn off the message Message Format:

Message Component	Description	Type	Bytes
PRN	Satellite PRN	unsigned char	1
Spare1	Spare	unsigned char	1
Spare2	Spare	unsigned char	2
Almwords	Almanac Words	long[12]	12*4 = 48

alAlmwords[	toa
alAlmwords[1]	$\sqrt{A}$
alAlmwords[2]	e
alAlmwords[3]	ω
alAlmwords[4]	M0
alAlmwords[5]	Ω_0
alAlmwords[6]	$\dot{\Omega}$
alAlmwords[7]	δ_i
alAlmwords[8]	a0 |a1 |Health] Bits [30:20|19:9|8:0]
alAlmwords[9]	WNa

			All of the scalefactors, number of bits, and units can be found in the BeiDou ICD on page 37: http://en.beidou.gov.cn/SYSTEMS/ICD/201806/P020180608523308843290.pdf		

Structure:

```
/***************************************************/
/*  SBinaryMsg42 Galileo(42), BeiDou(32), GPS(92) or QZSS(22) Almanac */
/***************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char    m_bySV;                    // The satellite to which this data belongs.
    unsigned char    m_bySpare;                 // Spare, keeps alignment to 4 bytes
    unsigned short   m_wSpare1;                 // Spare, keeps alignment to 4 bytes
    long alAlmwords[12];                        // Almanac words (different for different GNSS)
    unsigned short   m_wCheckSum;               // sum of all bytes of the header and data
    unsigned short   m_wCRLF;                   // Carriage Return Line Feed
} SBinaryMsg42;                                 // length = 8 + (4+48=52) + 2 + 2 = 64

typedef SBinaryMsg42 SBinaryMsg22;  //QZSS Almanac
typedef SBinaryMsg42 SBinaryMsg32;  //BeiDou Almanac
typedef SBinaryMsg42 SBinaryMsg92;  //GPS Almanac
typedef SBinaryMsg42 SBinaryMsg52;  //IRNSS Almanac
```

Additional Information:	Message has a BlockID of 32 and is 52 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN, Bin42

Topic Last Updated: v2.0 / April 30, 2019

Bin 34 Message

Message Type:	Binary
Description:	BeiDou -> GPS, ->GLO, -> GAL, ->UTC time offset parameters
Message Format:	Command Format to Request Message: $JBIN,34,r<CR><LF> where: '34' = Bin34 message 'r' = message rate in Hz (1 or 0) Message Format:

Message Component	Description	Type	Bytes
A0UTC, A1UTC	BDT clock bias relative to UTC	Int	4 x 2=8
A0GPS, A1GPS	BDT click bias relative to GPS time	short	2 x 2=4
A0GAL, A1GAL	BDT clock bias relative to Galileo system time	short	2 x 2=4
A0GLO, A1GLO	BDT clock bias relative to GLONASS time	short	2 x 2=4

Toa	Almanac reference time	unsigned char	1
Wna	Almanac week number	unsigned char	1
Dtls	Delta time due to leap seconds before the new leap second effective	char	1
Wnlsf	Week number of the new leap second	unsigned char	1
Dn	Day number of week of the new leap second	unsigned char	1
Dtslf	Delta time due to leap seconds after the new leap second effective	char	1
Spare1	Future use	short	2
Spare2	Future use	short	2
Spare3	Future use	short	2

Structure:

```
#if defined(_USING_BEIDOU_TIME_OFFSETS_)
//---------------------------------------------------------------------------
// SBinaryMsg34  --- BeiDou -> GPS, ->GLO, -> GAL, ->UTC time offset parameters
// Information is in both D1 and D2, but we are only going to use D1 because
// it is the same data as in D2
//---------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader   m_sHead;       //8 bytes
    int               m_A0UTC;       //BDT clock bias relative to UTC
    int               m_A1UTC;       //BDT clock rate relative to UTC
    short             m_A0GPS;       //BDT clock bias relative to GPS time
    short             m_A1GPS;       //BDT clock rate relative to GPS time
    short             m_A0GAL;       //BDT clock bias relative to Galileo system time
    short             m_A1GAL;       //BDT clock rate relative to Galileo system time
    short             m_A0GLO;       //BDT clock bias relative to GLONASS time
    short             m_A1GLO;       //BDT clock rate relative to GLONASS time
    unsigned char     m_toa;         //Almanac reference time (assuming this is also correct for the time offsets)
    unsigned char     m_Wna;         //almanac week number (assuming this is also correct for the time offsets)
    char              m_dtls;        //Delta time due to leap seconds before the new leap second effective
    unsigned char     m_wnlsf;       //Week number of the new leap second
    unsigned char     m_dn;          //Day number of week of the new leap second
    char              m_dtlsf;       //Delta time due to leap seconds after the new leap second effective
    short             m_spare1;
    short             m_spare2;
    short             m_spare3;
    unsigned short    m_wCheckSum;   //sum of all bytes of the header and data
    unsigned short    m_wCRLF;       // Carriage Return Line Feed
} SBinaryMsg34;                      // length = 8+(4+4+2+2+2+2+2+2+1+1+1+1+1+1+2+2+2=32)+2+2 = 8 + (32) + 2 + 2 =
44
#endif
```

Additional Information:	Message has a BlockID of 34 and is 32 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.10 June 1, 2018

Bin35 Message

Message Type:	Binary
Description:	BeiDou ephemeris information
Message Format:	Command Format to Request Message: $JBIN,35,r<CR><LF>

where:

'35' = Bin35 message ''r' = 1 (on) or 0 (off),
When set to on the message is sent once (one message for each tracked satellite at 1 second intervals) and then sent again whenever satellite information changes

Message Format:

Message Component	Description	Type	Bytes
SV	Satellite to which this data belongs	unsigned short	2
Spare1	Not used at this time	unsigned short	2
SecOfWeek	Time at which this arrived (LSB=6)	unsigned long	4
BeiDouNav[30]	Unparsed BeiDou Navigation message	see following	4 x 30 = 120

Elements correspond to the ephemeris values as defined in the BeiDou ICD:

1. Element 00, BDS_tow, Unsigned (4 bytes)
2. Element 01, BDS_toc, Unsigned (4 bytes)
3. Element 02, BDS_a0, Signed (4 bytes)
4. Element 03, BDS_a1,Signed (4 bytes)
5. Element 04, BDS_a2, Signed (4 bytes)
6. Element 05, BDS_toe, Unsigned (4 bytes)
7. Element 06, BDS_Root_A, Unsigned (4 bytes)
8. Element 07, BDS_Eccentricity, Unsigned (4 bytes)
9. Element 08, BDS_omega_perigee, Signed (4 bytes)
10. Element 09, BDS_DeltaN_MeanMotionDiff, Signed (4 bytes)
11. Element 10, BDS_M_MeanAnomaly, Signed (4 bytes)
12. Element 11, BDS_OMEGA0_Lon_Ascending, Signed (4 bytes)
13. Element 12, BDS_OMEGA_DOT, Signed (4 bytes)
14. Element 13, BDS_io_InclinationAngle, Signed (4 bytes)
15. Element 14, BDS_IDOT_RateInclination, Signed (4 bytes)
16. Element 15, BDS_Cuc_AmpCosHarmonicLat, Signed (4 bytes)
17. Element 16, BDS_Cus_AmpSinHarmonicLat, Signed (4 bytes)

	18. Element 17, BDS_Crc_AmpCosHarmonicRadius, Signed (4bytes) 19. Element 18, BDS_Crs_AmpSinHarmonicRadius, Signed (4bytes) 20. Element 19, BDS_Cic_AmpCosHarmonicInclination, Signed (4 bytes) 21. Element 20, BDS_Cir_AmpSinHarmonicInclination, Signed (4 bytes) 22. Element 21, BDS_TGD1_TGD2, Unsigned (4 bytes) TGD1 in lower 10 bits (bits 0-9) TGD2 in next 10 bits (10-19) 23. Element 22, BDS_WN, Unsigned (4 bytes) 24. Element 23, BDS_alpha_0_1_2_3, Unsigned (4 bytes) Packed with 4, 8-bit words, exactly as defined in the BeiDou ICD Alpha3 in lower 8 bits (bits 0-7) Alpha2 in next 8 bits (bits 8-15) Alpha1 in next 8 bits (bits 16-23) Alpha0 in upper 8 bits (bits 24-31) 25. Element 24, BDS_beta_0_1_2_3, Unsigned (4 bytes) Packed with 4, 8-bit words, exactly as defined in the BeiDou ICD Beta3 in lower 8 bits (bits 0-7) Beta2 in next 8 bits (bits 8-15) Beta1 in next 8 bits (bits 16-23) Beta0 in upper 8 bits (bits 24-31) 26. Element 25, BDS_SatH1_IODC_URA1_IODE, Unsigned (4 bytes) IODE in lower 5 bits (bits 0-4) URA1 in next 4 bits (bits 5-8) IODC in next 5 bits (bits 9-13) SatH1in next 1 bit (bit 14) 27. Element 26, spare (4 bytes) 28. Element 27, spare (4 bytes) 29. Element 28, spare (4 bytes) 30. Element 29, spare (4 bytes)
Structure:	`/*****************************************************/` `/*  SBinaryMsg35                                     */` `/*****************************************************/` `typedef SBinaryMsg95 SBinaryMsg35;  //BeiDou ephemeris`
Additional Information:	Message has a BlockID of 35 and is 128 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin36 Message

<table>
<tr><td>Message Type:</td><td>Binary</td></tr>
<tr><td>Description:</td><td>BeiDou code and carrier phase information (all frequencies)</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JBIN,36,r<CR><LF>

where:

•'36' = Bin36 message
•'r' = message rate in Hz (20, 10, 2, 1, or 0)

Message Format:

<table>
<tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr>
<tr><td>Tow</td><td>Time in seconds</td><td>double</td><td>2</td></tr>
<tr><td>Week</td><td>GPS week number</td><td>unsigned short</td><td>2</td></tr>
<tr><td>Spare1</td><td>Spare 1 (zero)</td><td>unsigned short</td><td>2</td></tr>
<tr><td>FreqPage</td><td>31.Bits 0-19 (20 bits) Spare bits

32.Bits 20-23 (4 bits) Number of pages

33.Bits 24-27 (4 bits) Page number

34.Bits 28-31 (4 bits)

Signal ID (0 = B1I, 1 = B2I, 2 = B3I)</td><td>unsigned long</td><td>4</td></tr>
<tr><td>Obs[CHANNELS_20]</td><td>20 sets of BeiDou observations</td><td>SObsPacket</td><td>20 x 12 = 240</td></tr>
<tr><td>1CodeMSBsPRN[CHANNELS_20]</td><td>Bits 0-7 (8 bits) Satellite PRN, 0 if no satellite
Bits 8-12 (5 bits) Spare bits
Bits 13- 31 (19 bits)
Upper 19 bits of B1/B2/B3, LSB = 256 meters,
MSB = 67108864 meters</td><td>unsigned long</td><td>20 x 4 = 80</td></tr>
</table>
</td></tr>
<tr><td>Structure:</td><td>

```
//------------------------------------------------------------------------
// SBinaryMsg36 BeiDou observations (see notes on mesage 76)
// Indivdual pages for B1I, B2I, B3I, etc ... to allow for BeiDou Phase III
// Allows for a maximum of 20 channels
//------------------------------------------------------------------------
typedef struct
{
    SUnionMsgHeader  m_sHead;               //               (8 bytes)
    double           m_dTow;                // Time in seconds (8 bytes)
    unsigned short   m_wWeek;               // GPS Week Number (2 bytes)
    unsigned short   m_wSpare1;             // spare 1 (zero)  (2 bytes)
```

</td></tr>
</table>

	`unsigned long m_uFreqPage; //[0-19] Spare bits` `//[20,21,22,23] Number of Pages` `//[24,25,26,27] Page Number` `//[28,29,30,31] Signal ID (B1I, B2I, B3I, etc)` `SObsPacket m_asObs[CHANNELS_20]; // 20 sets of BeiDou observations (20*12=240 bytes)` `unsigned long m_aulCodeMSBsPRN[CHANNELS_20]; // array of 20 words (20*4=80 bytes)` `// bit 7:0 (8 bits) = satellite PRN, 0` `// if no satellite` `// bit 12:8 (5 bits) = spare` `// bit 31:13 (19 bits) = upper 19 bits` `// of B1/B2/B3 LSB = 256 meters` `// MSB = 67108864 meters` `unsigned short m_wCheckSum; /// sum of all bytes of the header and data (2 bytes)` `unsigned short m_wCRLF; // Carriage Return Line Feed (2 bytes)` `} SBinaryMsg36; // length = 8 + (8+2+2+4+240+80=336) + 2 + 2 = 348`
Additional Information:	Message has a BlockID of 36 and is 332 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.11 / November 15, 2018

Bin42 Message

Message Type:	Binary
Description:	Galileo Almanac
Message Format:	Command Format to Request Message: $JBIN,42r<CR><LF> where: '42' = Bin42 message 'r' = 1 to turn on the message, 0 to turn off the message Message Format:

Message Component	Description	Type	Bytes
PRN	Satellite PRN	unsigned char	1
Spare1	Spare	unsigned char	1
Spare2	Spare	unsigned char	2
Almwords	Almanac Words	long[12]	12*4 = 48

alAlmwords[0]	WNa
alAlmwords[1]	toa
alAlmwords[2]	$\Delta\sqrt{A}$
alAlmwords[3]	e
alAlmwords[4]	δ_i

	alAlmwords[5] — Ω_0 alAlmwords[6] — $\dot{\Omega}$ alAlmwords[7] — ω alAlmwords[8] — M0 alAlmwords[9] — af0 alAlmwords[10] — af1 alAlmwords[11] — [E5aHS | E5bHS | E1BHS] Bits [2|2|2] All of the scalefactors, number of bits, and units can be found in the Galileo ICD on page 53: https://www.gsc-europa.eu/system/files/galileo_documents/Galileo-OS-SIS-ICD.pdf
Structure:	

```
/*************************************************/
/*  SBinaryMsg42 Galileo(42), BeiDou(32), GPS(92) or QZSS(22) Almanac */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char    m_bySV;                    // The satellite to which this data belongs.
    unsigned char    m_bySpare;                 // Spare, keeps alignment to 4 bytes
    unsigned short   m_wSpare1;                 // Spare, keeps alignment to 4 bytes
    long alAlmwords[12];                        // Almanac words (different for different GNSS)
    unsigned short   m_wCheckSum;               // sum of all bytes of the header and data
    unsigned short   m_wCRLF;                   // Carriage Return Line Feed
} SBinaryMsg42;                                 // length = 8 + (4+48=52) + 2 + 2 = 64

typedef SBinaryMsg42 SBinaryMsg22;  //QZSS Almanac
typedef SBinaryMsg42 SBinaryMsg32;  //BeiDou Almanac
typedef SBinaryMsg42 SBinaryMsg92;  //GPS Almanac
typedef SBinaryMsg42 SBinaryMsg52;  //IRNSS Almanac
```

Additional Information:	Message has a BlockID of 42 and is 52 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v2.0 / April 30, 2019

Bin44 Message

Message Type:	Binary
Description:	Galileo time conversion parameters
Message Format:	Command Format to Request Message: $JBIN,44,r<CR><LF>

where:
'44' = Bin44 message 'r' = 1 (on) or 0 (off)
When set to on the message is sent once and then sent again whenever satellite information changes

Message Format:

Message Component	Description	Type	Bytes
A0, A1	Coefficients for determining UTC time	double	8 x 2 = 16
tot	Reference time for A0 and A1, second of Galileo week	unsigned long	4
wnt	Current Galileo reference week	unsigned short	2
wnlsf	Week number when dtlsf becomes effective	unsigned short	2
dn	Day of week (1-7) when dtlsf becomes effective	unsigned short	2
dtls	Cumulative past leap seconds	short	2
dtlsf	Scheduled future leap seconds	short	2
Spare	Not used at this time	short	2
A0G, A1G	Coefficients of GGTO polynomial	double	8 x 2 = 16
T0G	Reference time of week for GGTO	unsigned long	4
WN0G	Reference week for GGTO	unsigned short	2
GGTOisValid	Indicates if GGTO is valied If values range from 0 = GGTO Invalid To 1 = GGTO Valid.	unsigned short	2
CheckSum	Sum of all bytes of header and data	unsigned short	2
CRLF	Carriage return line feed	unsigned short	2

Structure:

```
//------------------------------------------------------------------------
//  SBinaryMsg44
//    Galileo Time Conversion Parameters
//------------------------------------------------------------------------
typedef struct
{
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    SUnionMsgHeader m_sHead;           // Header of message.
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  (8 bytes)
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    // Galileo Time to UTC conversion parameters (32 bytes).
    double          m_A0;              // Constant term of polynomial to
                                       //  determine UTC from Galileo Time.
    double          m_A1;              // 1st order term of polynomial to
                                       //  determine UTC from Galileo Time.
    unsigned long   m_tot;             // Reference time for A0 & A1, sec of
                                       //  Galileo week.
    unsigned short  m_wnt;             // Current Galileo reference week.
    unsigned short  m_wnlsf;           // GST Week number when m_dtlsf
                                       //  becomes effective.
```

```
    unsigned short  m_dn;              // Day of the week 1 (= Sunday) to
                                       //  7 (= Saturday) when m_dtlsf
                                       //  becomes effective.
    short           m_dtls;            // Cumulative past leap seconds.
    short           m_dtlsf;           // Scheduled future (past) leap
                                       //  seconds.
    unsigned short  m_wSpare1;         // Spare (zero).
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - (32 bytes)
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    // GPS Time to Galileo Time conversion parameters (GGTO Parameters).
    //
    //   dTsys = Tgal - Tgps = m_A0G + m_A1G [TOW - m_t0G + 604800*(WN - m_WN0G)]
    //
    //     where,
    //        dTsys = The time difference between systems
    //        Tgal  = Galileo Time
    //        Tgps  = GPS Time
    //        TOW   = Galileo Time of Week
    //        WN    = Galileo Week Number
    //        remaining parameters follow.
    double          m_A0G;             // Constant term of GGTO polynomial.
    double          m_A1G;             // 1st order term of GGTO polynomial.
    unsigned long   m_t0G;             // Reference time of week for GGTO.
    unsigned short  m_WN0G;            // Reference week for GGTo.
    unsigned short  m_wGGTOisValid;    // Coded: 0 == GGTO Invalid,
                                       //        1 == GGTO Valid.
                                       //  The Galileo OS-SIS-ICD indicates
                                       //  that when satellite broadcasts
                                       //  all 1 bit values for A0G, A1G,
                                       //  t0G, and WN0G then "the GGTO is
                                       //  considered as not valid."
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - (24 bytes)
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    // Message Tail
    unsigned short  m_wCheckSum;       // Sum of all bytes of the header and
                                       //  data.
    unsigned short  m_wCRLF;           // Carriage Return Line Feed.
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  (4 bytes)
} SBinaryMsg44;                        // length = 8 + (32+24) + 2 + 2 = 68.
```

Additional Information:	Message has a BlockID of 44 and is 56 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.07 / February 16, 2017

Bin45 Message

Message Type:	Binary
Description:	Galileo ephemeris information
Message Format:	Command Format to Request Message: $JBIN,45,r<CR><LF> where: '45' = Bin45 message 'r' = 1 (on) or 0 (off), When set to on the message is sent once (one message for each tracked satellite at 1 second intervals) and then sent again whenever satellite information changes

Message Format:

Message Component	Description	Type	Bytes
SV	Satellite to which this data belongs	unsigned short	2
Spare1	Not used at this time	unsigned short	2
SecOfWeek	Time at which this arrived (LSB = 6)	unsigned long	4
SF1words[10]	Unparsed SF 1 message	unsigned long	4 x 10 = 40
SF2words[10]	Unparsed SF 2 message	unsigned long	4 x 10 = 40
SF3words[10]	Unparsed SF 3 message	unsigned long	4 x 10 = 40

Structure:

```
/*************************************************/
/*  SBinaryMsg45                                 */
/*************************************************/
typedef SBinaryMsg95 SBinaryMsg45;  //Galileo ephemeris
```

Additional Information: Message has a BlockID of 45 and is 128 bytes, excluding the header and epilogue

Related Commands and Messages: JBIN

Topic Last Updated: v1.07 / February 16, 2017

Bin62 Message

Message Type: Binary, GLONASS

Description: GLONASS almanac information

Message Format: Command Format to Request Message:

$JBIN,62,r<CR><LF>

where:

'62' = Bin62 message

'r' = message rate in Hz (1 or 0)

Message Format:

Message Component	Description	Type	Bytes
SV	Satellite to which this data belongs	unsigned char	1
Ktag_ch	Proprietary data	unsigned char	1
Spare1	Spare, keeps alignment to 4 bytes	unsigned short	2
Strings[3]	GLONASS almanac data (36 bytes)	SGLONASS string	36

		0 & 1 = Two almanac SFs 2= ICD String 5		

Structure:

```
/*****************************************************************/
/*  SBinaryMsg62, Glonass almanac data. Containing string
 *   5 and the two string pair for each satellite after string 5.
 *   String 5 contains the time reference for the glonass almanac
 *   and gps-glonass time differences.
 *
 *****************************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char    m_bySV;                  /* The satellite to which this data belongs. */
    unsigned char    m_byKtag_ch;             /* Proprietary data */
    unsigned short   m_wSpare1;               /* Spare, keeps alignment to 4 bytes */
    SGLONASS_String  m_asStrings[3];          /* glonass almanac data  (36 bytes)
                                                 String 0 & 1 = Two almanac Strings, String 2 = ICD String 5*/
    unsigned short   m_wCheckSum;             /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                 /* Carriage Return Line Feed */
} SBinaryMsg62;                               /* length = 8 + (40) + 2 + 2 = 52 */
```

Additional Information:	Message has a BlockID of 62 and is 40 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.2 / September 13, 2022

Bin65 Message

Message Type:	Binary, GLONASS
Description:	GLONASS ephemeris information
Message Format:	Command Format to Request Message: $JBIN,65,r<CR><LF> where: '65' = Bin65 message "r' = 1 (on) or 0 (off), When set to on the message is sent once (one message for each tracked satellite at 1 second intervals) and then sent again whenever satellite information changes Message Format:

Message Component	Description	Type	Bytes
SV	Satellite to which this data belongs	unsigned char	1
Ktag	Satellite K Number + 8	unsigned char	1
Spare1	Spare, keeps alignment to 4 bytes	unsigned short	2
TimeReceivedInSeconds	Time at which this arrived	unsigned long	4
Strings[5]	First five strings of GLONASS frame (60 bytes)	SGLONASS string	60

Structure:

```
/*************************************************/
```

	`` /* SBinaryMsg65, added by JL for glonass subframe immediate data + string_5 */ /***/ /* sent only upon command or when values change (not including changes in tk) */ typedef struct { SUnionMsgHeader m_sHead; unsigned char m_bySV; /* The satellite to which this data belongs. */ unsigned char m_byKtag; /* The satellite K Number + 8. */ unsigned short m_wSpare1; /* Spare, keeps alignment to 4 bytes */ unsigned long m_ulTimeReceivedInSeconds; /* time at which this arrived */ SGLONASS_String m_asStrings[5]; /* first 5 Strings of Glonass Frame (60 bytes) */ unsigned short m_wCheckSum; /* sum of all bytes of the header and data */ unsigned short m_wCRLF; /* Carriage Return Line Feed */ } SBinaryMsg65; /* length = 8 + (68) + 2 + 2 = 80 */ ```
Additional Information:	Message has a BlockID of 65 and is 68 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin66 Message

Message Type:	Binary, GLONASS
Description:	GLONASS L1/L2 code and carrier phase information
Message Format:	Command Format to Request Message: $JBIN,66,r<CR><LF> where: '66' = Bin66 message 'r' = message rate in Hz (20, 10, 2, 1, or 0) Message Format:

Message Component	Description	Type	Bytes
Tow	Time in seconds	double	2
Week	GPS week number	unsigned short	1
Spare1	Spare 1 (zero)	unsigned short	1
Spare2	Spare 2 (zero)	unsigned long	4
L1Obs[CHANNELS_12]	12 sets of L1 (GLONASS) observations	SObsPacket	12 x 12 = 144
L2Obs[CHANNELS_12]	12 sets of L2 (GLONASS) observations	SObsPacket	12 x 12 = 144
L1CodeMSBsSlot[CHANNELS_12]	*See following* •Bits 0-7 (8 bits) Satellite slot, 0 if no satellite •Bits 8-12 (5 bits) Spare bit •Bits 13- 31 (19 bits) Upper 19 bits of L1, LSB = 256 meters, MSB = 67108864 meters	unsigned long	4

Structure:	

```
/***********************************************/
/*  SBinaryMsg66  GLONASS OBS (see notes on mesage 76) */
/***********************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    double           m_dTow;                  /* Time in seconds */
    unsigned short   m_wWeek;                 /* GPS Week Number */
    unsigned short   m_wSpare1;               /* 16 bit spare word */
    unsigned long    m_ulP_Code;              /* Bit [31,24] spare bits */
                                              /* Bit [23,12] Pcode On for m_asL2Obs Obs 0-11, Bit 12 = channel 0*/
                                              /* Bit [11,0]  Pcode On for m_asL1Obs Obs 0-11  Bit 0  = channel 0*/
    SObsPacket       m_asL1Obs[CHANNELS_12];  /* 12 sets of L1(Glonass) observations */
    SObsPacket       m_asL2Obs[CHANNELS_12];  /* 12 sets of L2(Glonass) observations */
    unsigned long    m_aulL1CodeMSBsSlot[CHANNELS_12]; /* array of 12 words.
                                                  bit 7:0 (8 bits) = satellite Slot, 0
                                                  if no satellite
                                                  bit 12:8 (5 bits) = spare
                                                  bit 31:13 (19 bits) = upper 19 bits
                                                  of L1  LSB = 256 meters
                                                         MSB = 67108864 meters */
    unsigned short   m_wCheckSum;             /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                 /* Carriage Return Line Feed */
} SBinaryMsg66;                               /* length = 8 + (352) + 2 + 2 = 364 */
```

Additional Information:	Message has a BlockID of 66 and is 352 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin69 Message

Message Type:	Binary, GLONASS
Description:	GLONASS L1/L2 diagnostic information
Message Format:	Command Format to Request Message: $JBIN,69,r<CR><LF> where: '69' = Bin69 message 'r' = message rate in Hz (1 or 0) Message Format:

Message Component	Description	Type	Bytes
SecOfWeek	Tow	long	
L1usedNavMask	Mask of L1 channels used in nav solution	unsigned short	
L2usedNavMask	Mask of L2 channels used in nav solution	unsigned short	
ChannelData[CHANNELS_12]	Channel data 12X24 = 288	SGLONASSChanData	
Week	Week	unsigned short	
Spare01	Spare 1	unsigned char	

	Spare02 | Spare 2 | unsigned char |
Structure:	`/*************************************************/` `/*  SBinaryMsg69                                 */` `/*************************************************/` `typedef struct` `{` `  SUnionMsgHeader  m_sHead;` `  long  m_lSecOfWeek;  /* tow  */` `  unsigned short  m_wL1usedNavMask;  /* mask of L1 channels used in nav solution */` `  unsigned short  m_wL2usedNavMask;  /* mask of L2 channels used in nav solution */` `  SGLONASSChanData  m_asChannelData[CHANNELS_12]; /* channel data 12X24 = 288 */` `  unsigned short  m_wWeek;  /* week */` `  unsigned char  m_bySpare01;  /* spare 1 */` `  unsigned char  m_bySpare02;  /* spare 2 */` `  unsigned short  m_wCheckSum;  /* sum of all bytes of the header and data */` `  unsigned short  m_wCRLF;  /* Carriage Return Line Feed */` `} SBinaryMsg69;  /* length = 8 + 300 + 2 + 2 = 312 */`
Additional Information:	Message has a BlockID of 69 and is 300 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin76 Message

Message Type:	Binary
Description:	GPS L1/L2 code and carrier phase information Note: "Code" means pseudo range derived from code phase. "Phase" means range derived from carrier phase. This will contain cycle ambiguities. Only the lower 16 bits of L1P code, L2P code and the lower 23 bits of carrier phase are provided. The upper 19 bits of the L1CA code are found in m_aulCACodeMSBsPRN[]. The upper 19 bits of L1P or L2P must be derived using the fact L1P and L2P are within 128 m (419.9 ft) of L1CA. To determine L1P or L2P: 1.Use the lower 16 bits provided in the message. 2.Set the upper bits to that of L1CA. 3.Add or subtract on LSB of the upper bits (256 meters (839.9 feet)) so that L1P or L2P are with in 1/2 LSB (128 m (419.9ft)) The carrier phase is in units of cycles, rather than meters, and is held to within 1023 cycles of the respective code range. Only the lower 16+7 = 23 bits of carrier phase are transmitted in Bin 76. To determine the remaining bits:

<table>
<tr><td></td><td>1.Convert the respective code range (determined above) into cycles by dividing by the carrier wave length. This is the nominal reference phase.

2.Extract the 16 and 7 bit blocks of carrier phase from bin 76 and arrange it to form the lower 23 bits of carrier phase.

3.Set the upper bits (bit 23 and above) equal to those of the nominal reference phase Add or subtract the least significant upper bit (8192 cycles) so that carrier phase most closely agrees with the nominal reference phase (to within 4096 cycles</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JBIN,76,r<CR><LF>

where:

'76' = Bin76message

'r' = message rate in Hz (20, 10, 2, 1, 0, or .2)

Message Format:

<table>
<tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr>
<tr><td>TOW</td><td>Predicted GPS time in seconds</td><td>double</td><td>8</td></tr>
<tr><td>Week</td><td>GPS week number</td><td>unsigned short</td><td>2</td></tr>
<tr><td>Spare1</td><td></td><td>unsigned long</td><td>2</td></tr>
<tr><td>Spare2</td><td></td><td>unsigned long</td><td>4</td></tr>
<tr><td>L2PSatObs[12] (array for next 3 fields)</td><td>L2 satellite observation data</td><td>structure array</td><td>12 x 12 =144</td></tr>
<tr><td>CS_TT_W3_SNR</td><td>See following
•Bits 0-11 (12 bits)
SNR; 10.0 X log10 (0.1164xSNR_value)
•Bits 12-14 (3 bits)
Cycle Slip Warn (warning for potential 1/2 cycle slips); a warning exists if any of these bits are set
•Bit 15 (1 bit)
Long Track Time;1 if Track Time > 25.5 sec (0 otherwise)
•Bits 16-23 (8 bits)
Track Time (signal tracking time in seconds); LSB = 0.1 seconds; Range = 0 to 25.5 seconds
•Bits 24-31 (8 bits)
Cycle Slips; increments by 1 every cycle slip with natural roll-over after 255</td><td>unsigned long</td><td>4</td></tr>
<tr><td>P7_Doppler_FL</td><td>See following
•Bit 0 (1 bit)</td><td>unsigned long</td><td>4</td></tr>
</table>
</td></tr>
</table>

			Phase Valid (Boolean);1 if valid phase (0 otherwise) •Bits 1-23 (23 bits) Doppler (magnitude of Doppler);LSB = 1/512 cycle/sec; Range = 0 to 16384 cycle/sec •Bit 24 (1 bit) Doppler Sign (sigh of Doppler);1 = negative, 0 = positive •Bits 25-31 (7 bits) Carrier Phase (High part) (Upper 7 bits of the 23 bit carrier phase): LSB = 64 cycles, MSB = 4096 cycles		
		CodeAndPhase	See following •Bits 0-15 (16 bits) Pseudorange (lower 16 bits of code pseudorange);LSB = 1/256 meters, MSB = 128 meters Note: For CA code, the upper 19 bits are given in L1CACodeMSBsPRN[] below •Bits 16-31 (16 bits) Carrier Phase (lower 16 bits of the carrier phase); LSB = 1/1024 cycles, MSB = 32 cycles Note: The 7 MSBs are given in P7_Doppler_FL (see preceding row in this table)	unsigned long	4
		L1CASatObs[15] (array for next 3 fields)	L1 satellite code observation data	structure array	15 x 12 =180
		CS_TT_W3_SNR	See following •Bits 0-11 (12 bits) SNR; 10.0 X log10(0.1024xSNR_value) •Bits 12-14 (3 bits) Cycle Slip Warn (warning for potential 1/2 cycle slips); a warning exists if any of these bits are set •Bit 15 (1 bit) Long Track Time;1 if Track Time > 25.5 sec (0 otherwise) •Bits 16-23 (8 bits) Track Time (signal tracking time in seconds); LSB = 0.1 seconds; Range = 0 to 25.5 seconds	unsigned long	4

		•Bits 24-31 (8 bits) Cycle Slips; increments by 1 every cycle slip with natural roll-over after 255		
	P7_Doppler_FL	See following: •Bit 0 (1 bit) Phase Valid (Boolean);1 if valid phase (0 otherwise) •Bits 1-23 (23 bits) Doppler (magnitude of Doppler);LSB = 1/512 cycle/sec; Range = 0 to 16384 cycle/sec •Bit 24 (1 bit) Doppler Sign (sigh of Doppler);1 = negative, 0 = positive •Bits 25-31 (7 bits) Carrier Phase (High part) (Upper 7 bits of the 23 bit carrier phase): LSB = 64 cycles, MSB = 4096 cycles Bits 25-31 (7 bits) Carrier Phase (High part) (Upper 7 bits of the 23 bit carrier phase): LSB = 64 cycles, MSB = 4096 cycles	unsigned long	4
	CodeAndPhase	See following •Bits 0-15 (16 bits) Pseudorange (lower 16 bits of code pseudorange);LSB = 1/256 meters, MSB = 128 meters Note: For CA code, the upper 19 bits are given in L1CACodeMSBsPRN[] below •Bits 16-31 (16 bits) Carrier Phase (lower 16 bits of the carrier phase); LSB = 1/1024 cycles, MSB = 32 cycles Note: The 7 MSBs are given in P7_Doppler_FL (see preceding row in this table)	unsigned long	4
	L1CACodeMSBsPRN[15]	L1CA code observation Bits 0-7 (8 bits) PRN (space vehicle ID);PRN = 0 if no data Bits 8-12 (5 bits) Unused Bits 13-31 (19 bits) L1CA Range (upper 19 bits of L1CA); LSB = 256 meters, MSB = 67,108,864 meters	Array of 15 unsigned long	15 x 4 =60
	L1PCode[12]	L1(P) code observation data •Bits 0-15 (16 bits)	Array of 12 unsigned	12 x 4 = 48

		L1P Range (lower 16 bits of the L1P code pseudorange);LSB = 1/256 meters, MSB = 128 meters •Bits 16-27 (12 bits) L1P SNR (L1P signal-to-noise ratio); SNR = 10.0 x log(0.1164 x SNR_value), if 0, then L1P channel not tracked Bits 28-31 (4 bits) Unused	long	
	wCeckSum	Sum of all bytes of header and data	unsigned short	2
	wCRLF	Carriage return line feed	unsigned short	2

Structure:

```
/*************************************************/
/*  SBinaryMsg76                                 */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    double           m_dTow;                  /* GPS Time in seconds */
    unsigned short   m_wWeek;                 /* GPS Week Number */
    unsigned short   m_wSpare1;               /* spare 1 (zero)*/
    unsigned long    m_ulSpare2;              /* spare 2 (zero)*/
    SObsPacket       m_asL2PObs[CHANNELS_12];   /* 12 sets of L2(P) observations */
    SObsPacket       m_asL1CAObs[CHANNELS_L1_E];  /* 15 sets of L1(CA) observations */
    unsigned long    m_aulCACodeMSBsPRN[CHANNELS_L1_E]; /* array of 15 words.
                                                   bit 7:0 (8 bits) = satellite PRN, 0
                                                   if no satellite
                                                   bit 12:8 (5 bits) = spare
                                                   bit 31:13 (19 bits) = upper 19 bits
                                                   of L1CA  LSB = 256 meters
                                                            MSB = 67108864 meters */
    unsigned long    m_auL1Pword[CHANNELS_12];  /* array of 12 words relating to L1(P) code.
                                                  Bit 0-15 (16 bits) lower 16 bits of the
                                                  L1P code pseudo range.
                                                  LSB = 1/256 meters
                                                  MSB = 128 meters
                                                  Bits 16-27 (12 bits) = L1P SNR_value
                                                  SNR = 10.0*log10( 0.1164*SNR_value)
                                                  If Bits 16-27 all zero, no L1P track
                                                  Bits 28-31 (4 bits) spare */
    unsigned short   m_wCheckSum;             /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;                 /* Carriage Return Line Feed */
} SBinaryMsg76;                               /* length = 8 + (448) + 2 + 2 = 460 */
```

Additional Information:	Message has a BlockID of 76 and is 448 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin80 Message

Message Type:	Binary
Description:	SBAS data frame information
Message Format:	Command Format to Request Message: Message Format:

	<table><tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr><tr><td>PRN</td><td>Broadcast PRN</td><td>unsigned short</td><td>2</td></tr><tr><td>Spare</td><td>Not used at this time</td><td>unsigned short</td><td>2</td></tr><tr><td>MsgSecOfWeek</td><td>Seconds of week for message</td><td>unsigned long</td><td>4</td></tr><tr><td>WaasMsg[8]</td><td>250-bit WAAS message (RTCA DO0229). 8 unsigned longs, with most significant bit received first.</td><td>unsigned long</td><td>4 x 8 =32</td></tr></table>
Structure:	`/*************************************************/` `/*  SBinaryMsg80                                 */` `/*************************************************/` `typedef struct` `{` `  SUnionMsgHeader  m_sHead;` `  unsigned short m_wPRN;  /* Broadcast PRN */` `  unsigned short m_wSpare;  /* spare (zero) */` `  unsigned long  m_ulMsgSecOfWeek;  /* Seconds of Week For Message */` `  unsigned long  m_aulWaasMsg[8];  /* Actual 250 bit waas message*/` `  unsigned short m_wCheckSum;  /* sum of all bytes of the headerand data */` `  unsigned short m_wCRLF;  /* Carriage Return Line Feed */` `} SBinaryMsg80;  /* length = 8 + (40) + 2 + 2 = 52 */`
Additional Information:	Message has a BlockID of 80 and is 40 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin89 Message

Message Type:	Binary
Description:	SBAS satellite tracking information (supports three SBAS satellites)
Message Format:	Command Format to Request Message: $JBIN,89,r<CR><LF> where: '89' = Bin89 message 'r' = message rate in Hz (1 or 0) Message Format: <table><tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr><tr><td>GPSSecOfWeek</td><td>GPS tow integer sec</td><td>long</td><td></td></tr><tr><td>MaskSBASTracked</td><td>SBAS satellites tracked, bit mapped 0..3</td><td>unsigned char</td><td></td></tr></table>

	<table><tr><td>MaskSBASUSED</td><td>SBAS satellites used, bit mapped 0..3</td><td>unsigned char</td><td></td></tr><tr><td>Spare</td><td>Spare</td><td>unsigned short</td><td></td></tr><tr><td>ChannelData[CHANNELS_SBAS_E]</td><td>SBAS channel data</td><td>SChannel Data</td><td></td></tr></table>
Structure:	<pre>/***/ /* SBinaryMsg89 * Supports 3 SBAS Satellites */ /***/ typedef struct { SUnionMsgHeader m_sHead; long m_lGPSSecOfWeek; /* GPS tow integer sec */ unsigned char m_byMaskSBASTracked; /* SBAS Sats Tracked, bit mapped 0..3 */ unsigned char m_byMaskSBASUSED; /* SBAS Sats Used, bit mapped 0..3 */ unsigned short m_wSpare; /* spare */ SChannelData m_asChannelData[CHANNELS_SBAS_E]; /* SBAS channel data */ unsigned short m_wCheckSum; /* sum of all bytes of the header and data */ unsigned short m_wCRLF; /* Carriage Return Line Feed */ } SBinaryMsg89; /* length = 8 + 80 + 2 + 2 = 92 */</pre>
Additional Information:	Message has a BlockID of 89 and is 80 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin92 Message

Message Type:	Binary
Description:	GPS Almanac
Message Format:	Command Format to Request Message: $JBIN,92,r<CR><LF> where: '92' = Bin92 message 'r' = 1 to turn on the message, 0 to turn off the message Message Format: <table><tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr><tr><td>PRN</td><td>Satellite PRN</td><td>unsigned char</td><td>1</td></tr><tr><td>Spare1</td><td>Spare</td><td>unsigned char</td><td>1</td></tr><tr><td>Spare2</td><td>Spare</td><td>unsigned char</td><td>2</td></tr><tr><td>Almwords</td><td>Almanac Words</td><td>long[12]</td><td>12*4 =</td></tr></table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>Refer to GPS ICD Almanac page 36.

<table>
<tr><td>alAlmwords[0]</td><td>toa</td></tr>
<tr><td>alAlmwords[1]</td><td>$\sqrt{A}$</td></tr>
<tr><td>alAlmwords[2]</td><td>e</td></tr>
<tr><td>alAlmwords[3]</td><td>ω</td></tr>
<tr><td>alAlmwords[4]</td><td>M0</td></tr>
<tr><td>alAlmwords[5]</td><td>Ω_0</td></tr>
<tr><td>alAlmwords[6]</td><td>$\dot{\Omega}$</td></tr>
<tr><td>alAlmwords[7]</td><td>δ_i</td></tr>
<tr><td>alAlmwords[8]</td><td>SV Health</td></tr>
<tr><td>alAlmwords[9]</td><td>WNa</td></tr>
</table></td><td></td><td>48</td></tr>
</table>
</td></tr>
<tr><td>Structure:</td><td>

```
/***************************************************/
/*  SBinaryMsg42 Galileo(42), BeiDou(32), GPS(92) or QZSS(22) Almanac */
/***************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char    m_bySV;                    // The satellite to which this data belongs.
    unsigned char    m_bySpare;                 // Spare, keeps alignment to 4 bytes
    unsigned short   m_wSpare1;                 // Spare, keeps alignment to 4 bytes
    long alAlmwords[12];                        // Almanac words (different for different GNSS)
    unsigned short   m_wCheckSum;               // sum of all bytes of the header and data
    unsigned short   m_wCRLF;                   // Carriage Return Line Feed
} SBinaryMsg42;                                 // length = 8 + (4+48=52) + 2 + 2 = 64

typedef SBinaryMsg42 SBinaryMsg22;  //QZSS Almanac
typedef SBinaryMsg42 SBinaryMsg32;  //BeiDou Almanac
typedef SBinaryMsg42 SBinaryMsg92;  //GPS Almanac
typedef SBinaryMsg42 SBinaryMsg52;  //IRNSS Almanac
```

</td></tr>
<tr><td>Additional Information:</td><td>Message has a BlockID of 92 and is 52 bytes, excluding the header and epilogue</td></tr>
<tr><td>Related Commands and Messages:</td><td>JBIN, Bin42</td></tr>
</table>

Topic Last Updated: v4.2 / September 13, 2022

Bin93 Message

Message Type:	Binary
Description:	SBAS ephemeris information
Message Format:	Command Format to Request Message: $JBIN,93,r<CR><LF> where: '93' = Bin93 message 'r' = message rate in Hz (1 or 0) Message Format:

Message Component	Description	Type	Bytes
SV	Satellite to which this data belongs	unsigned short	2
Spare	Not used at this time	unsigned short	2
TOWSecOfWeek	Time at which this arrived (LSB = 1 sec)	unsigned long	4
IODE		unsigned short	2
URA	Consult the ICD-GPS-200 for definition in Appendix A	unsigned short	2
TO	Bit 0 = 1 sec	long	4
XG	Bit 0 = 0.08 m	long	4
YG	Bit 0 = 0.08 m	long	4
ZG	Bit 0 = 0.4 m	long	4
XGDot	Bit 0 = 0.000625 m/sec	long	4
YXDot	Bit 0 = 0.000625 m/sec	long	4
ZGDot	Bit 0 = 0.004 m/sec	long	4
XGDotDot	Bit 0 = 0.0000125 m/sec/sec	long	4
YGDotDot	Bit 0 = 0.0000125 m/sec/sec	long	4
ZGDotDot	Bit 0 = 0.0000625 m/sec/sec	long	4
Gf0	Bit 0 = 2**-31 sec	unsigned short	2
Gf0Dot	Bit 0 = 2**-40sec/sec	unsigned short	2

Structure:

```
/*****************************************************/
/*  SBinaryMsg93                                     */
/*****************************************************/
/* sent only upon command or when values change */
/* WAAS ephemeris */
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned short   m_wSV;             /* The satellite to which this data belongs. */
    unsigned short   m_wWeek;           /* Week corresponding to m_lTOW*/
    unsigned long    m_lSecOfWeekArrived;/* time at which this arrived (LSB = 1sec) */
    unsigned short   m_wIODE;
    unsigned short   m_wURA;        /* See 2.5.3 of Global Pos Sys Std Pos Service Spec */
    long m_lTOW;                        /* Sec of WEEK Bit 0 = 1 sec */
    long m_lXG;                         /* Bit 0 = 0.08 m */
    long m_lYG;                         /* Bit 0 = 0.08 m */
    long m_lZG;                         /* Bit 0 = 0.4 m */
    long m_lXGDot;                      /* Bit 0 = 0.000625 m/sec */
    long m_lYGDot;                      /* Bit 0 = 0.000625 m/sec */
    long m_lZGDot;                      /* Bit 0 = 0.004 m/sec */
    long m_lXGDotDot;                   /* Bit 0 = 0.0000125 m/sec/sec */
    long m_lYGDotDot;                   /* Bit 0 = 0.0000125 m/sec/sec */
    long m_lZGDotDot;                   /* Bit 0 = 0.0000625 m/sec/sec */
    short  m_nGf0;                      /* Bit 0 = 2**-31 sec */
    short  m_nGf0Dot;                   /* Bit 0 = 2**-40 sec/sec */
    unsigned short   m_wCheckSum;       /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;           /* Carriage Return Line Feed */
} SBinaryMsg93;                         /* length = 8 + (56) + 2 + 2 = 68 */
```

Additional Information:	Message has a BlockID of 93 and is 56 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.0 / June 30, 2020

Bin94 Message

Message Type:	Binary
Description:	Ionospheric and UTC conversion parameters
Message Format:	Command Format to Request Message: $JBIN,94,r<CR><LF> where: '94' = Bin94 message 'r' = 1 (on) or 0 (off) Message Format:

Message Component	Description	Type	Bytes
a0, a1, a2, a3	AFCRL alpha parameters	double	8 x 4 = 32
b0, b1, b2, b3	AFCRL beta parameters	double	8 x 4 = 32
A0, A1	Coefficients for determining UTC time	double	8 x 2 = 16
tot	Reference time for A0 and A1, seconds of GPS week	unsigned long	4
wnt	Current UTC reference week	unsigned short	2
wnlsf	Week number when dtlsf becomes effective	unsigned short	2
dn	Day of week (1-7) when dtlsf becomes effective	unsigned short	2
dtls	Cumulative past leap	short	2
dtlsf	Scheduled future leap	short	2
Spare1	Not used at this time	unsigned short	2

When set to on the message is sent once and then sent again whenever satellite information changes

Structure:

```
//------------------------------------------------------------------------------
//  SBinaryMsg94
//  I think we will need similar binary messages for Galileo and BeiDou
//  Or maybe not, it seems that much of this is optional for RINEX
//------------------------------------------------------------------------------
// sent only upon command or when values change
typedef struct
{
    SUnionMsgHeader  m_sHead;
```

```
    /* Iono parameters. */

    double         m_a0,m_a1,m_a2,m_a3;             /* AFCRL alpha parameters. */
    double         m_b0,m_b1,m_b2,m_b3;             /* AFCRL beta parameters.  */

    /* UTC conversion parameters. */

    double         m_A0,m_A1;              /* Coeffs for determining UTC time. */
    unsigned long  m_tot;      /* Reference time for A0 & A1, sec of GPS week. */
    unsigned short m_wnt;                /* Current UTC reference week number. */
    unsigned short m_wnlsf;       /* Week number when dtlsf becomes effective. */
    unsigned short m_dn;    /* Day of week (1-7) when dtlsf becomes effective. */
    short          m_dtls;                    /* Cumulative past leap seconds. */
    short          m_dtlsf;                  /* Scheduled future leap seconds. */
    unsigned short m_wSpare1;                                 /* spare 4 (zero)*/
    unsigned short m_wCheckSum;     /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;                       /* Carriage Return Line Feed */
} SBinaryMsg94;                              /* length = 8 + (96) + 2 + 2 =  108 */
```

Additional Information:	Message has a BlockID of 94 and is 96 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin95 Message

Message Type:	Binary
Description:	GPS ephemeris information
Message Format:	Command Format to Request Message: $JBIN,95,r<CR><LF> where: '95' = Bin95 message 'r' = 1 (on) or 0 (off) Message Format:

Message Component	Description	Type	Bytes
SV	Satellite to which this data belongs	unsigned short	2
Spare1	Not used at this time	unsigned short	2
SecOfWeek	Time at which this arrived (LSB = 6)	unsigned long	4
SF1words[10]	Unparsed SF 1 message	unsigned long	4 x 10 = 40
SF2words[10]	Unparsed SF 2 message	unsigned long	4 x 10 = 40
SF3words[10]	Unparsed SF 3 message	unsigned long	4 x 10 = 40

	When set to on the message is sent once (one message for each tracked satellite at 1 second intervals) and then sent again whenever satellite information changes
Structure:	<pre>/**/ /* SBinaryMsg95 */ /**/ /* sent only upon command or when values change */ typedef struct { SUnionMsgHeader m_sHead; unsigned short m_wSV; /* The satellite to which this data belongs. */ unsigned short m_wSpare1; /* spare 1 (chan number (as zero 9/1/2004)*/ unsigned long m_TOW6SecOfWeek; /* time at which this arrived (LSB = 6sec) */ unsigned long m_SF1words[10]; /* Unparsed SF 1 message words. */ unsigned long m_SF2words[10]; /* Unparsed SF 2 message words. */ unsigned long m_SF3words[10]; /* Unparsed SF 3 message words. */ /* Each of the subframe words contains one 30-bit GPS word in the lower 30 bits, The upper two bits are ignored Bits are placed in the words from left to right as they are received */ unsigned short m_wCheckSum; /* sum of all bytes of the header and data */ unsigned short m_wCRLF; /* Carriage Return Line Feed */ } SBinaryMsg95; /* length = 8 + (128) + 2 + 2 = 140 */</pre>
Additional Information:	Message has a BlockID of 95 and is 128 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin96 Message

Message Type:	Binary
Description:	GPS L1 code and carrier phase information
Message Format:	Command Format to Request Message: $JBIN,96,r<CR><LF> where: '96' = Bin96 message 'r' = message rate in Hz (20, 10, 2, 1, or 0) •Bit 0 (1 bit) Phase; Location 0; 1 if valid (0 otherwise) •Bit 1 (1 bit) TrackTime; 1 if track time > 25.5 seconds (0 otherwise) •Bits 2-3 (2 bits) Unused •Bits 4-31 (28 bits) Doppler; Signed (two's compliment) Doppler in units of m/sec x 4096. (i.e., LSB=1/4096), range= +/- 32768 m/sec. Computed as phase change over 1/10 sec.

Message Format:

Message Component	Description	Type	Bytes
PseudoRange[12]	Pseudorange	double	8
Phase[12]	Phase (m) L1 wave = 0.190293672798365	double	8

Structure:

```
/************************************************/
/*  SBinaryMsg96                                */
/************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned short   m_wSpare1;            /* spare 1 (zero)*/
    unsigned short   m_wWeek;              /* GPS Week Number */
    double           m_dTow;               /* Predicted GPS Time in seconds */
    SObservations    m_asObvs[CHANNELS_12];/* 12 sets of observations */
    unsigned short   m_wCheckSum;          /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;              /* Carriage Return Line Feed */
} SBinaryMsg96;                            /* length = 8 + (300) + 2 + 2 = 312 */
```

Additional Information: Message has a BlockID of 96 and is 300 bytes, excluding the header and epilogue

Related Commands and Messages: JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin97 Message

Message Type: Binary

Description: Processor statistics

Message Format: Command Format to Request Message:

$JBIN,97,r<CR><LF>

where:

'97' = Bin97 message
'r' = message rate in Hz (20, 10, 2, 1, 0, or .2)

Message Format:

Message Component	Description	Type	Bytes
CPUFactor	CPU utilization factor Multiply by 450e- 06 to get percentage of spare CPU that is available **Note:** This field is only relevant on the old SLX platforms and Eclipse platform. It is not relevant for the Crescent receivers.	unsigned long	4

MissedSubFrame	Total number of missed sub frames in the navigation message since power on	unsigned short	2
MaxSubFramePnd	Max sub frames queued for processing at any one time	unsigned short	2
MissedAccum	Total number of missed code accumulation measurements in the channel tracking loop	unsigned short	2
MissedMeas	Total number missed pseudorange measurements	unsigned short	2
Spare 1	Not used at this time	unsigned long	4
Spare 2	Not used at this time	unsigned long	4
Spare 3	Not used at this time	unsigned long	4
Spare 4	Not used at this time	unsigned short	2
Spare 5	Not used at this time	unsigned short	2

Structure:

```
/**************************************************/
/*  SBinaryMsg97                                  */
/**************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned long  m_ulCPUFactor;       /* CPU utilization Factor (%=multby 450e-6) */
    unsigned short m_wMissedSubFrame;   /* missed subframes */
    unsigned short m_wMaxSubFramePend;  /* max subframe pending */
    unsigned short m_wMissedAccum;      /* missed accumulations */
    unsigned short m_wMissedMeas;       /* missed measurements */
    unsigned long  m_ulSpare1;          /* spare 1 (zero)*/
    unsigned long  m_ulSpare2;          /* spare 2 (zero)*/
    unsigned long  m_ulSpare3;          /* spare 3 (zero)*/
    unsigned short m_wSpare4;           /* spare 4 (zero)*/
    unsigned short m_wSpare5;           /* spare 5 (zero)*/
    unsigned short m_wCheckSum;         /* sum of all bytes of the headerand data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg97;                         /* length = 8 + (28) + 2 + 2 = 40 */
```

Additional Information:	Message has a BlockID of 97 and is 28 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin98 Message

Message Type:	Binary
Description:	GPS satellite and almanac information
Message Format:	Command Format to Request Message: $JBIN,98,r<CR><LF>

where:

'98' = Bin98 message

'r' = message rate in Hz (1 or 0)

Message Format:

Message Component	Description	Type	Bytes
AlmanData[8]	SV data, 8 at a time	SSVAlman Data	8 x 8 = 64
LastAlman	Last almanac processed	unsigned char	1
IonoUTCVFlag	Flag that is set when ionosphere modeling data is extracted from the GPS sub frame 4 0 = not logged 2 = valid	unsigned char	1
Spare	Not used at this time	unsigned short	2

Structure:

```
/*************************************************/
/*  SBinaryMsg98                                 */
/*************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    SSVAlmanData    m_asAlmanData[8];    /* SV data, 8 at a time */
    unsigned char  m_byLastAlman;        /* last almanac processed */
    unsigned char  m_byIonoUTCVFlag;     /* iono UTC flag */
    unsigned short m_wSpare;             /* spare */
    unsigned short m_wCheckSum;          /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;              /* Carriage Return Line Feed */
} SBinaryMsg98;                          /* length = 8 + (64+1+1+2) + 2 + 2 = 80 */
```

Additional Information: Message has a BlockID of 98 and is 68 bytes, excluding the header and epilogue

Related Commands and Messages: JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin99 Message

Message Type:	Binary
Description:	GPS L1 diagnostic information
Message Format:	Command Format to Request Message: $JBIN,99,r<CR><LF> where: '99' = Bin99 message Message Format:

Message Component	Description	Type	Bytes
NavMode	Navigation mode data. Lower 3 bits hold the GPS mode, upper bit set if differential is available. 0 = time not valid 1 = No fix 2 = 2D fix 3 = 3D fix Upper bit 1 = differential available	unsigned char	1
UTCTimeDiff	Whole seconds between UTC and GPS time (GPS minus UTC)	char	1
GPSWeek	GPS week associated with this message	unsigned short	2
GPSTimeofWeek	GPS tow (sec) associated with this message	double	8
sChannelData[CHANNELS_12]	Channel data	SChannelData	12 x 24 = 288
ClockErrAtL1	Clock error of the GPS clock oscillator at L1 frequency in Hz	short	2
Spare	Not used at this time	unsigned short	2

Structure:

```
/***************************************************/
/*  SBinaryMsg99                                   */
/***************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char  m_byNavMode;         /* Nav Mode FIX_NO, FIX_2D, FIX_3D (high bit =has_diff) */
    char           m_cUTCTimeDiff;      /* whole Seconds between UTC and GPS   */
    unsigned short m_wGPSWeek;          /* GPS week */
    double         m_dGPSTimeOfWeek;    /* GPS tow  */
    SChannelData   m_asChannelData[CHANNELS_12]; /* channel data */
    short          m_nClockErrAtL1;     /* clock error at L1, Hz */
    unsigned short m_wSpare;            /* spare */
    unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg99;                         /* length = 8 + 304 + 2 + 2 = 316 */
```

Additional Information:	Message has a BlockID of 99 and is 304 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v1.06 / March 10, 2015

Bin100 Message

Message Type:	Binary
Description:	GPS L2 diagnostic information
Message Format:	Command Format to Request Message: $JBIN,100,r<CR><LF> where:

'100' = Bin100 message

'r' = message rate in Hz (1 or 0)

Message Format:

Message Component	Description	Type	Bytes
NavMode	Navigation mode data (lower 3 bits hold the GPS mode, upper bit set if differential is available) Values from Lower 3 bits take on the values: 0 = time not valid 1 = No fix 2 = 2D fix 3 = 3D fix Upper bit (bit 7) is 1 if differential is available	unsigned char	1
UTCTimeDiff	Whole seconds between UTC and GPS time (GPS minus UTC) Values are Positive	char	1
GPSWeek	GPS week associated with this message	unsigned short	2
MaskSatsUsedL2P	L2P satellites used, bit mapped 0..31	unsigned long	
GPSTimeofWeek	GPS tow (sec) associated with this message	double	8
MaskSatsUsedL1P	L1P satellites used, bit mapped 0..31	unsigned long	
sChannelData[CHANNELS_12]	L2 channel data	SChannelData	12 x 24 = 288

Structure:

```
/*************************************************/
/*  SBinaryMsg100                                */
/*************************************************/
//#if defined(_DUAL_FREQ_)
typedef struct
{
    SUnionMsgHeader  m_sHead;
    unsigned char  m_byNavMode;         /* Nav Mode FIX_NO, FIX_2D, FIX_3D (high bit =has_diff) */
    char           m_cUTCTimeDiff;      /* whole Seconds between UTC and GPS   */
    unsigned short m_wGPSWeek;          /* GPS week */
    unsigned long  m_ulMaskSatsUsedL2P; /* L2P SATS Used, bit mapped 0..31 */
    double         m_dGPSTimeOfWeek;    /* GPS tow  */
    unsigned long  m_ulMaskSatsUsedL1P; /* L1P SATS Used, bit mapped 0..31 */
    SChannelL2Data m_asChannelData[CHANNELS_12]; /* channel data */
    unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
    unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg100;                        /* length = 8 + 260 + 2 + 2 = 272 */
//#endif
```

Additional Information: Message has a BlockID of 100 and is 260 bytes, excluding the header and epilogue

Related Commands and Messages: JBIN

Topic Last Updated: v1.08 / June 9, 2017

Bin209 Message

<table>
<tr><td>Message Type:</td><td>Binary</td></tr>
<tr><td>Description:</td><td>SNR and status for all GNSS tracks</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JBIN,209,r<CR><LF>

where:
'209' = Bin209 message
'r'=message rate in Hz

Message Format:

<table>
<tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr>
<tr><td>GPSTimeofWeek</td><td>GPS tow (sec) associated with this message. Where values range from 0.0 to 604800.0</td><td>double</td><td>8</td></tr>
<tr><td>GPSWeek</td><td>GPS week associated with this message. Where values range from 0 to 65535</td><td>unsigned short</td><td>2</td></tr>
<tr><td>UTCTimeDiff</td><td>Whole Seconds between UTC and GPS</td><td>char</td><td>1</td></tr>
<tr><td>Page</td><td>Bits 0-1 = Antenna:
0 = Master, 1 = Slave, 2 = Slave2
Bits 2-4 = Page ID:
0 = page 1, 1 =page 2, etc.
Bits 5-7 = Max page ID:
0 = only 1 page, 1 = 2 pages</td><td>unsigned char</td><td>1</td></tr>
<tr><td>sSVSNRData</td><td>SNR data</td><td>SSVSNRData</td><td>40 * 8 = 320</td></tr>
</table>
</td></tr>
<tr><td>Structure:</td><td>

```
//-***************************************************
//-*  SBinaryMsg209
//-*  SNR and status for all GNSS tracks
//-***************************************************
typedef struct
{
    SUnionMsgHeader  m_sHead;             //                                                    [8]
    double           m_dGPSTimeOfWeek;    // GPS tow                                           [8 bytes]
    unsigned short   m_wGPSWeek;          // GPS week                                          [2 bytes]
    char             m_cUTCTimeDiff;      // Whole Seconds between UTC and GPS                 [1 byte]
    unsigned char    m_byPage;            // Bits 0-1 = Antenna: 0 = Master, 1 = Slave, 2 = Slave2 [1 byte]
                                          // Bits 2-4 = Page ID: 0 = page 1, 1 = page 2, etc
                                          // Bits 5-7 = Max page ID: 0 = only 1 page, 1 = 2 pages
    SSVSNRData       m_asSVData[40];      // SNR data                                          [320 bytes]
    unsigned short   m_wCheckSum;         // sum of all bytes of the header and data
    unsigned short   m_wCRLF;             // Carriage Return Line Feed
} SBinaryMsg209;                          // length = 8 + 332 + 2 + 2 = 344
```

</td></tr>
<tr><td>Additional Information:</td><td>Message has a BlockID of 209 and is 332 bytes, excluding the header and epilogue</td></tr>
<tr><td>Related Commands and Messages:</td><td>JBIN</td></tr>
</table>

Topic Last Updated: v4.2 / September 13, 2022

SSVSNRData

<table>
<tr><td>Message Type:</td><td>Binary</td></tr>
<tr><td>Description:</td><td>Satellite Data for Bin 209</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

N/A

Message Format:

<table>
<tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr>
<tr><td>Status_SYS_PRNID</td><td>status, GNSS system, PRN ID
0-5 PRNID (for SBAS , PRNID = PRN-120)
Bit 6-8 SYS:
0 = GPS,
1 = GLONASS,
2 = GALILEO,
3 = BEIDOU,
4 = QZSS,
5 = IRNSS NavIC
7 = SBAS

Bit 9 = code and Carrier Lock on L1,G1,B1
Bit 10 = code and Carrier Lock on L2,G2,B2
Bit 11 = code and Carrier Lock on L5,E5,B3
Bit 12 = Bit Lock and Frame lock (decoding data)
Bit 13 = Ephemeris Available
Bit 14 = Health OK
Bit 15 = Satellite used in Navigation Solution
m_wStatus_SYS_PRNID = 0 ==> unfilled data</td><td>unsigned short</td><td>2</td></tr>
<tr><td>m_chElev</td><td>Elevation angle, LSB = 1 deg</td><td>char</td><td>1</td></tr>
<tr><td>m_byAzimuth</td><td>1/2 the Azimuth angle, LSB = 2 deg</td><td>unsigned char</td><td>1</td></tr>
<tr><td>m_ulSNR3_SNR2_SNR1</td><td>Bits 0-10 SNR1 (L1,G1,B1, etc) 11 bits => Max SNR = 32.2 dB

Bits 11-21 SNR2 (L2,G2,B2, etc) 11 bits => Max SNR =32.2 dB

Bits 22-31 SNR3 (L5,E5,B3, etc) 10 bits => Max SNR =29.2 dB</td><td>unsigned long</td><td>4</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Structure:</td><td>

```
//-*************************************************
//-*  SSVSNRData
//-*************************************************
typedef struct
{
    unsigned short m_wStatus_SYS_PRNID; // status, GNSS system, PRN ID
                                        //    Bit 0-5  PRNID  (for SBAS , PRNID = PRN-120)
                                        //    Bit 6-8  SYS: 0 = GPS, 1 = GLONASS, 2 = GALILEO, 3 = BEIDOU, 4=QZSS,
5=IRNSS, 7 = SBAS
                                        //    Bit 9  = code and Carrier Lock on L1,G1,B1
                                        //    Bit 10 = code and Carrier Lock on L2,G2,B2
                                        //    Bit 11 = code and Carrier Lock on L5,E5,B3
                                        //    Bit 12 = Bit Lock  and Frame lock (decoding data)
                                        //    Bit 13 = Ephemeris Available
                                        //    Bit 14 = Health OK
                                        //    Bit 15 = Satellite used in Navigation Solution
                                        //    m_wStatus_SYS_PRNID = 0 ==> unfilled data
    char          m_chElev;             // Elevation angle, LSB = 1 deg
    unsigned char m_byAzimuth;          // 1/2 the Azimuth angle, LSB = 2 deg
    unsigned long m_ulSNR3_SNR2_SNR1;   // 3 SNRs, 2 @ 11 bit & 1 @ 10 bits, each SNR = 10.0*log10(
0.8192*SNR_value)
                                        //    Bits 0-10  SNR1  (L1,G1,B1, etc)  11 bits => Max SNR = 32.2 dB
                                        //    Bits 11-21 SNR2  (L2,G2,B2, etc)  11 bits => Max SNR = 32.2 dB
                                        //    Bits 22-31 SNR3  (L5,E5,B3, etc)  10 bits => Max SNR = 29.2 dB
} SSVSNRData;  // 8 bytes
```

</td></tr>
<tr><td>Additional Information:</td><td>Message is 8 bytes, excluding the header and epilogue</td></tr>
<tr><td>Related Commands and Messages:</td><td>JBIN</td></tr>
</table>

Topic Last Updated: v4.2 / September 13, 2022

Bin309 Message

<table>
<tr><td>Message Type:</td><td>Binary</td></tr>
<tr><td>Description:</td><td>SNR and status for all GNSS tracks</td></tr>
<tr><td>Message Format:</td><td>Command Format to Request Message:

$JBIN,309,r<CR><LF>

where:
'309' = Bin309 message
'r'=message rate in Hz

Message Format:

<table>
<tr><th>Message Component</th><th>Description</th><th>Type</th><th>Bytes</th></tr>
<tr><td>GPSTimeofWeek</td><td>GPS tow (sec) associated with this message. Where values range from 0.0 to 604800.0</td><td>double</td><td>8</td></tr>
<tr><td>GPSWeek</td><td>GPS week associated with this message. Where values range from 0 to 65535</td><td>unsigned short</td><td>2</td></tr>
<tr><td>UTCTimeDiff</td><td>Whole Seconds between UTC and GPS</td><td>char</td><td>1</td></tr>
<tr><td>Page</td><td>Bits 0-1 = Antenna:
0 = Master, 1 = Slave, 2 = Slave2
Bits 2-4 = Page ID:
0 = page 1, 1 =page 2, etc.
Bits 5-7 = Max page ID:
0 = only 1 page, 1 = 2 pages</td><td>unsigned char</td><td>1</td></tr>
</table>
</td></tr>
</table>

	sSVData309 | SNR data | SSVSNRData309 | 30 * 16 = 480
Structure:	`SBinaryMsg309` — SNR and status for all GNSS tracks `typedef struct` `{` `SUnionMsgHeader m_sHead; // [8]` `double m_dGPSTimeOfWeek; // GPS tow [8 bytes]` `unsigned short m_wGPSWeek; // GPS week [2 bytes]` `char m_cUTCTimeDiff; // Whole Seconds between UTC and GPS [1 byte]` `unsigned char m_byPage; // Bits 0-1 = Antenna: 0 = Master, 1 = Slave, 2 = Slave2 [1 byte]` `// Bits 2-4 = Page ID: 0 = page 1, 1 = page 2, etc` `// Bits 5-7 = Max page ID: 0 = only 1 page, 1 = 2 pages` `SSVSNRData309 m_asSVData309[30]; // SNR data [480 bytes]` `unsigned short m_wCheckSum; // sum of all bytes of the header and data` `unsigned short m_wCRLF; // Carriage Return Line Feed` `} SBinaryMsg309; // length = 8 + 492 + 2 + 2 = 504`
Additional Information:	Message has a BlockID of 309 and is 492 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.2 / September 13, 2022

SSVSNRData309

Message Type:	Binary
Description:	Satellite Data for Bin309
Message Format:	Command Format to Request Message: N/A Message Format:

Message Component	Description	Type	Bytes
m_wSYS_PRNID	status, GNSS system, PRN ID 0-6 PRNID (for SBAS , PRNID = PRN-120) Bit 7-10 SYS: 0 = GPS, 1 = GLONASS, 2 = GALILEO, 3 = BEIDOU, 4 = QZSS, 5 = IRNSS NavIC 7 = SBAS, Bit 11 – 15 Spare M_wSYS_PRNID = 0 ==> unfilled data	unsigned short	2
m_wStatus	Bit 0 = code and Carrier Lock on Signal 0 Bit 1 = code and Carrier Lock on Signal 1 Bit 2 = code and Carrier	unsigned short	2

			Lock on Signal 2 Bit 3 = code and Carrier Lock on Signal 3 Bit 4 = code and Carrier Lock on Signal 4 Bit 5 = code and Carrier Lock on Signal 5 Bit 6 = code and Carrier Lock on Signal 6 Bit 7 = code and Carrier Lock on Signal 7 GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3GLO Signal ID: G1C/G1P=0, G2C/G2P=1, G10C=4, G20C=5, G30C=6 GAL Signal ID: E1BC=0, E5A=1, E5B=2, E6=3, ALTBOC=4BDS Signal ID: B1I=0, B2I=1,B3I=2,B1BOC=3,B2A=4,B2B =5,B3C=6,ACEBOC=7 QZS Signal ID: L1CA=0, L2C=2, L5=3, L1C=4, LEX=5 Bit 8 = Bit Lock and Frame lock (decoding data) on Signal 0 Bit 9 = Bit Lock and Frame lock (decoding data) on Signal 1 Bit 10 = Bit Lock and Frame lock (decoding data) on Signal 2 Bit 11 = spare Bit 12 = spare Bit 13 = Ephemeris Available Bit 14 = Health OK Bit 15 = Satellite used in Navigation Solution		
		m_chElev	Elevation angle, LSB = 1 deg	char	1
		m_byAzimuth	1/2 the Azimuth angle, LSB = 2 deg	unsigned char	1
		m_wLower2BitsSNR 7_6_5_ 4_3_2_1_0	Lower 2 bits of 10 bit SNR for channel 7-0 Bit 0-1, lower two bits of SNR on Signal 0 Bit 2-3, lower two bits of SNR on Signal 1 Bit 4-5, lower two bits of SNR on Signal 2 Bit 6-7, lower two bits of SNR on Signal 3 Bit 8-9, lower two bits of SNR on Signal 4 Bit 10-11, lower two bits of SNR on Signal 5 Bit 12-13, lower two bits of SNR on Signal 6 Bit 14-15, lower two bits of SNR on Signal 7	unsigned short	2
		m_abySNR8Bits[8]	8 SNRs, Upper 8 bits of 10 bit SNR, SNR = 10.0*log10(0.8192*SNR_value), Max SNR = 29.2 dB	unsigned char	8

		SNR_value for i'th SNR = ((unsigned long)m_abySNR8Bits[i] << 2) + Lower2Bits Lower2Bits = (m_wLower2BitsSNR7_6_5_4_3_2_1_0 >> (2*i)) & 0x3; m_abySNR8Bits[0] 8 bits of SNR on signal 0 m_abySNR8Bits[1] 8 bits of SNR on signal 1 // m_abySNR8Bits[2] 8 bits of SNR on signal 2 // m_abySNR8Bits[3] 8 bits of SNR on signal 3 // m_abySNR8Bits[4] 8 bits of SNR on signal 4 // m_abySNR8Bits[5] 8 bits of SNR on signal 5 // m_abySNR8Bits[6] 8 bits of SNR on signal 6 // m_abySNR8Bits[7] 8 bits of SNR on signal 7		

Structure:

```
//-*************************************************
//-*  SSVSNRData309
//-*************************************************
typedef struct
{
    unsigned short m_wSYS_PRNID;        // GNSS system, PRN ID
                                        //    Bit 0-6 PRNID  (For SBAS , PRNID = PRN-120. For QZSS,  PRNID = PRN-
192)
                                        //    Bit 7-10 SYS: 0 = GPS, 1 = GLONASS, 2 = GALILEO, 3 = BEIDOU, 4=QZSS, 5
= IRNSS, 7 = SBAS
                                        //    Bit 11-15   Spare
                                        //    m_wSYS_PRNID = 0 ==> unfilled data
    unsigned short m_wStatus;           // status
                                        //    Bit 0 = code and Carrier Lock on Signal 0
                                        //    Bit 1 = code and Carrier Lock on Signal 1
                                        //    Bit 2 = code and Carrier Lock on Signal 2
                                        //    Bit 3 = code and Carrier Lock on Signal 3
                                        //    Bit 4 = code and Carrier Lock on Signal 4
                                        //    Bit 5 = code and Carrier Lock on Signal 5
                                        //    Bit 6 = code and Carrier Lock on Signal 6
                                        //    Bit 7 = code and Carrier Lock on Signal 7
                                        //       GPS Signal ID: L1CA=0, L2P=1, L2C=2, L5=3, L1C=4
                                        //       GLO Signal ID: G1C/G1P=0, G2C/G2P=1, G10C=4, G20C=5, G30C=6
                                        //       GAL Signal ID: E1BC=0, E5A=1, E5B=2, E6=3, ALTBOC=4
                                        //       BDS Signal ID: B1I=0, B2I=1,
B3I=2,B1BOC=3,B2A=4,B2B=5,B3C=6,ACEBOC=7
                                        //       QZS Signal ID: L1CA=0, L2C=2, L5=3, L1C=4, LEX=5
                                        //       IRN Signal ID: L5=0
                                        //    Bit 8  = Bit Lock and Frame lock (decoding data) on Signal 0
                                        //    Bit 9  = Bit Lock and Frame lock (decoding data) on Signal 1
                                        //    Bit 10 = Bit Lock and Frame lock (decoding data) on Signal 2
                                        //    Bit 11 = spare
                                        //    Bit 12 = spare
                                        //    Bit 13 = Ephemeris Available
                                        //    Bit 14 = Health OK
                                        //    Bit 15 = Satellite used in Navigation Solution

    char           m_chElev;             // Elevation angle, LSB = 1 deg
    unsigned char  m_byAzimuth;          // 1/2 the Azimuth angle, LSB = 2 deg
    unsigned short m_wLower2BitsSNR7_6_5_4_3_2_1_0;   // Lower 2 bits of 10 bit SNR for channel 7-0
                                        // Bit 0-1,   lower two bits of SNR on Signal 0
                                        // Bit 2-3,   lower two bits of SNR on Signal 1
                                        // Bit 4-5,   lower two bits of SNR on Signal 2
                                        // Bit 6-7,   lower two bits of SNR on Signal 3
                                        // Bit 8-9,   lower two bits of SNR on Signal 4
                                        // Bit 10-11, lower two bits of SNR on Signal 5
                                        // Bit 12-13, lower two bits of SNR on Signal 6
                                        // Bit 14-15, lower two bits of SNR on Signal 7
    unsigned char  m_abySNR8Bits[8];    // 8 SNRs, Upper 8 bits of 10 bit SNR, SNR = 10.0*log10( 0.8192*SNR_value),
                                        //                                     Max SNR = 29.2 dB
                                        // SNR_value for i'th SNR = ((unsigned long)m_abySNR8Bits[i] << 2) +
```

	Lower2Bits // Lower2Bits = (m_wLower2BitsSNR7_6_5_4_3_2_1_0 >> (2*i)) & 0x3; // m_abySNR8Bits[0] 8 bits of SNR on signal 0 // m_abySNR8Bits[1] 8 bits of SNR on signal 1 // m_abySNR8Bits[2] 8 bits of SNR on signal 2 // m_abySNR8Bits[3] 8 bits of SNR on signal 3 // m_abySNR8Bits[4] 8 bits of SNR on signal 4 // m_abySNR8Bits[5] 8 bits of SNR on signal 5 // m_abySNR8Bits[6] 8 bits of SNR on signal 6 // m_abySNR8Bits[7] 8 bits of SNR on signal 7 } SSVSNRData309; // 16 bytes
Additional Information:	Message is 16 bytes, excluding the header and epilogue
Related Commands and Messages:	JBIN

Topic Last Updated: v4.2 / September 13, 2022

Resources

Reference Documents

National Marine Electronics Association, National Marine Electronics Association (NMEA) Standard for Interfacing Marine Electronic Devices Version 2.1, October 15, NMEA 1995

National Marine Electronics Association
7 Riggs Avenue Severna Park
MD 21146

Tel:+1-410-975-9425
Tel Toll Free: +1-800-808-6632

http://www.nmea.org/

Radio Technical Commission for Maritime Services, RTCM Recommended Standards for Differential NAVSTAR GPS Service Version 2.2 Developed by Special Committee No. 104, RTCM 1998

Radio Technical Commission for Maritime Services
800 N Kent St, Suite 1060
Arlington, VA 22209 USA

Tel: +1-703-527-2000
http://www.rtcm.org/

Radio Technical Commission for Aeronautics, Minimum Operational Performance Standards (MOPS) for Global Positioning System/Wide Area Augmentation System Airborne Equipment Document RTCA D0-229A, Special Committee No. 159, RTCA 1998

Radio Technical Commission for Aeronautics
71828 L Street, NW, Suite 805
Washington, D.C. 20036

Tel:+1-202-833-9339
http://www.rtca.org/

ARIC Research Corporation, Interface Control Document, Navstar GPS Space Segment/Navigation User Interfaces ICD-GPS-200, April 12, 2000

ARIC Research Corporation
2250 E. Imperial Highway, Suite
450 El Segundo, CA 90245-
3509

http://www.navcen.uscg.gov/

Topic Last Updated: v1.02 / January 25, 2011

Websites

Hemisphere GNSS
http://www.hemispheregnss.com

FAA WAAS

This site offers general information on the WAAS service provided by the U.S. FAAS.
http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/waas/

ESA EGNOS System Test Bed

This site contains information relating to past performance, real-time performance, and broadcast schedule of EGNOS.
http://www.esa.int/esaNA/egnos.html

Solar and Ionospheric Activity

The following sites are useful in providing details regarding solar and ionospheric activity.
http://iono.jpl.nasa.gov
http://www.spaceweather.com
https://www.swpc.noaa.gov

Topic Last Updated: v4.2 / September 13, 2022