Hemisphere GNSS Technical Reference Manual

Current Version: v3.0 **December 30, 2019**

Introduction

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,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.

Change History lists all the topics in this manual along with the latest update description and change date.

Troubleshooting provides troubleshooting advice for common questions.

Introduction and Quick Start

Introduction

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,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.

Change History lists all the topics in this manual along with the latest update description and change date.

Troubleshooting provides troubleshooting advice for common questions.

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 topic for commands supported by that 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 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 at an additional cost.

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,5Note 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

Multi-Function Application (MFA) Software

Your device may include MFA software that allows you to set the positioning (mode) hierarchy of your device. To verify if your device contains MFA software send the $JAPP command to the device; the response indicates whether you have MFA as follows:

- Without MFA (two specific applications listed) Example: $>JAPP,WAASRTKB,AUTODIFF,1,2
- With MFA (MFA and one specific application listed) Example: $>JAPP,MFA,SBASRTKB,1,2

The hierarchy is the path your device follows to determine what differential source to use depending on available sources. The hierarchy is as follows:

1. RTK
2. L-band (Atlas)
3. SBAS
4. Beacon
5. External RTCM
6. Autonomous

If you are running RTK and you lose your RTK radio link, the device defaults to the next highest mode, either Atlas high precision service or SBAS (if available). If the new signal becomes unusable, the next mode will be selected (for example Beacon or External RTCM). Finally, if no correction signals are available, the device defaults to Autonomous.

You can include or exclude specific sources. For example, you can exclude sources that you do not want your device to use, such as if you want to use only beacon. If you do not exclude the other sources your device may use SBAS instead. Another example is if you want to exclude Atlas (when you do not have an Atlas subscription) to conserve power. You include and exclude sources using the $JDIFFX,INCLUDE and $JDIFFX,EXCLUDE commands, respectively.

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

Hemisphere GNSS Hardware Platforms

Hemisphere GNSS offers the following hardware platforms:

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

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:

- Enclosure
- Main carrier board
- Set of three adapter boards for use with small form factor Hemisphere GNSS OEM boards
- Power cable and AC power supply
- Two serial cables- one straight serial cable and one null modem cable for RTK

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

- Enclosure
- Crescent
- Crescent Vector II
- Eclipse II
- miniEclipse
- LX-2 (L-band DGPS and high precision services)

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: v1.06 / March 10, 2015

DGNSS and RTK Solutions

SBAS

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 Automatic Tracking

SBAS Performance

WAAS DGPS

WAAS Signal Information

WAAS Reception

WAAS Coverage

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

SBAS Automatic Tracking

The SBAS demodulator featured within the receiver automatically scans and tracks multiple SBAS satellite signals, as specified by the JWAASPRN command (defaulted to WAAS PRN 135 and 138, suitable for use in North America).

If the default satellites become disabled, the receiver automatically tracks different satellites. This automatic tracking enables you to focus on other aspects of your application rather than ensuring the receiver is tracking SBAS correctly.

The SBAS demodulator features two-channel tracking that enhances the ability to maintain acquisition on an SBAS signal satellite in regions where more than one satellite is in view. This redundant tracking approach results in more consistent signal acquisition in areas where signal blockage of either satellite is possible.

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

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 both 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.

Topic Last Updated: v.1.11/November 15, 2018

WAAS DGPS

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, WAAS calculates separate errors for the following:

- Ionospheric error
- GPS satellite timing errors
- GPS satellite orbit errors

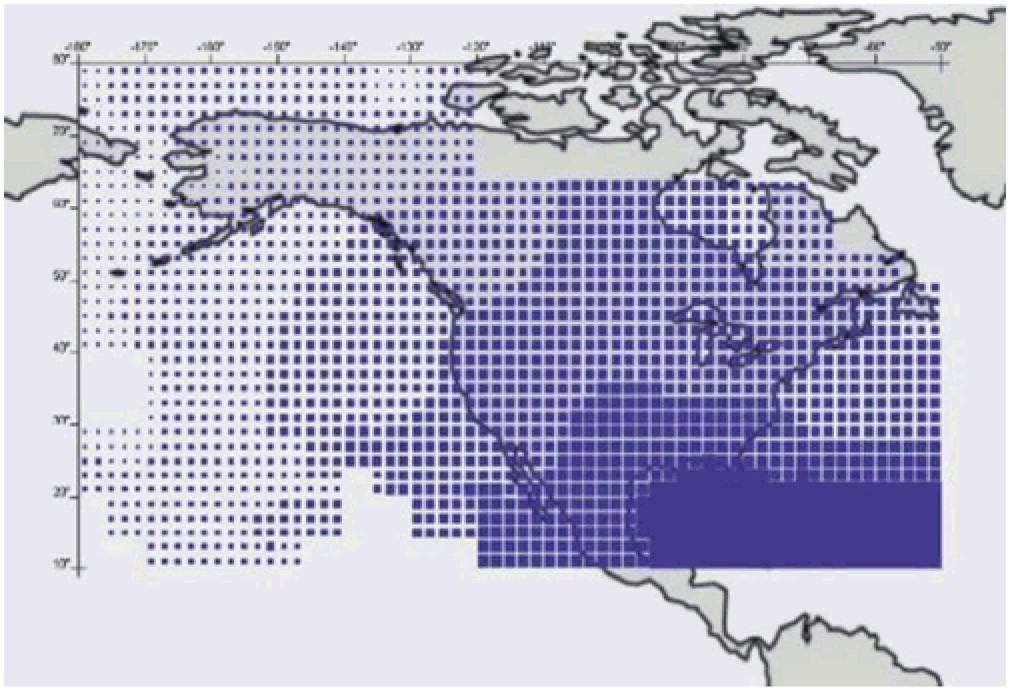
Provided that a GNSS satellite is available to the WAAS reference station network for tracking purposes, orbit and timing error corrections will be available for that satellite. Ionospheric corrections for that satellite are only available if the signal passes through the ionospheric map provided by WAAS, which covers most of North America.

To improve the ionospheric map provided by WAAS, the receiver extrapolates information from the broadcast ionospheric coverage map, extending its effective coverage. This allows the receiver to be used successfully in regions that competitive products may not. This is especially important in Canada for regions north of approximately 54° N latitude and for outer regions of the Caribbean.

The process of estimating ionospheric corrections beyond the WAAS broadcast map is not as good as having an extended WAAS map and accuracy degradation may occur.

The map links below depict the broadcast WAAS ionospheric map coverage and the Hemisphere GNSS extrapolated version, respectively. As the two maps show, the Hemisphere GNSS extrapolated version's coverage is greater in all directions, enhancing usable coverage.

- Broadcast WAAS ionospheric correction map
- Extrapolated WAAS ionospheric correction map



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

WAAS Signal Information

WAAS and other SBAS systems transmit correction data on the same frequency as GPS, allowing the use of the same receiver equipment used for GPS. Another advantage of having WAAS transmit on the same frequency as GPS is that only one antenna element is required.

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

WAAS Reception

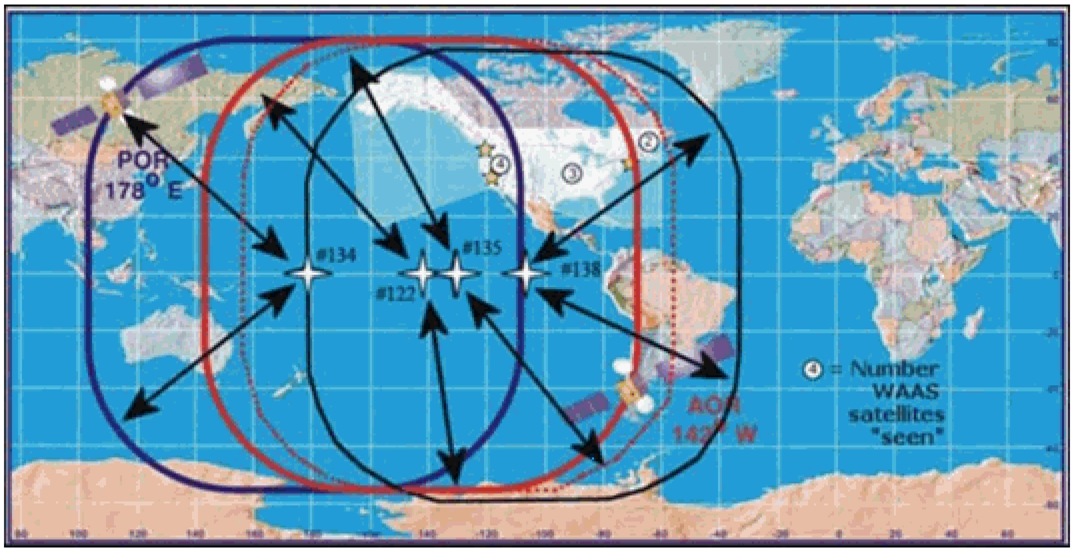
Since WAAS broadcasts on the same frequency as GPS, the signal requires a line of site in the same manner as GPS to maintain signal acquisition.

Because of their locations, SBAS satellites may appear lower on the horizon than GPS satellites—it depends on the geographic position on land. When using WAAS correction data, the receiver can provide the azimuth and elevation of all satellites to aid in determining their position with respect to the antenna.

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

WAAS Coverage

The figure below depicts the current WAAS coverage provided by the geostationary satellites.



The WAAS satellites are identified by their pseudo-range-number (PRN). In some areas, two or more satellites may be visible.

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 WAAS ionospheric coverage.

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

EGNOS

The European Geostationary Navigation Overlay Service (EGNOS) uses multiple geostationary satellites and a network of ground stations to transmit differential correction data for public use. EGNOS is currently located over the Atlantic Ocean and Africa.

Because of their location over the equator, these satellites may appear lower over the horizon as compared to GPS satellites - depending upon the geographic position on the land. In regions where the satellites appear lower on the horizon, they may be more susceptible to being masked by terrain, foliage, buildings or other objects, resulting in signal loss. Increased distance from the equator and the satellite's longitude cause the satellite to appear lower on the horizon.

The figure below shows approximate EGNOS coverage provided by the satellites. Virtually all of Europe, part of Northern Africa, and part of the Middle East is covered with at least one signal. Most of Europe is covered by three signals.

Iceland
Norway
Sweden
Finland
Russia
Estonia
Latvia
Lithuania
Denmark
Ireland
United Kingdom
Belarus
Poland
Netherlands
Belgium
Luxembourg
Germany
Czech Republic
Ukraine
Austria
Hungary
Romania
France
Switzerland
Bosnia
Bulgaria
Italy
Albania
Greece
Turkey
Portugal
Spain
Morocco
Algeria
Tunisia

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

MSAS

The MTSAT Satellite-based Augmentation System (MSAS) is currently run by the Japan Meteorological Agency (JMA). MSAS provides GPS augmentation information to aircraft through MTSAT (Multi-functional Transport Satellite) located approximately 36000 km above the equator (geostationary earth orbit).

MSAS generates GPS augmentation information by analyzing signals from GPS satellites received by monitor stations on the ground. This augmentation information consists of GPS-like ranging signal and correction information on GPS errors caused by the satellites themselves or by the ionosphere.

The MSAS signal provides accurate, stable, and reliable GPS position solutions to aircraft, resulting in a considerable improvement in the safety and reliability of GPS positioning. This enables aviation users who are under very strict safety regulations to use GPS positioning as a primary navigation system.

Visit http://www.jma.go.jp/jma/jma-eng/satellite/ for more information on MSAS and MTSAT.

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

GAGAN

The GPS Aided Geo Augmented Navigation system (**GAGAN**) was deployed by the Indian government . It operates similarly to the other SBAS regions described previously and will broadcast on one geostationary satellite (PRN 127) over the Western portion of the Indian Ocean. GAGAN is visible in India at elevation angles in excess of 50º above the horizon. This provides an excellent correction source in virtually all areas of the subcontinent.

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

Radiobeacon

Radiobeacon Range

The broadcasting range of a 300 kHz beacon depends on a number of factors, including:

- Transmission power
- Free space loss
- Ionospheric state
- Surface conductivity
- Ambient noise
- Atmospheric losses

Signal strength decreases with distance from the transmitting station, mostly due to spreading loss. This loss is a result of the signal's power being distributed over an increasing surface area as the signal radiates away from the transmitting antenna.

The expected broadcast range also depends on the conductivity of the surface over which it travels. A signal will propagate further over a surface area with high conductivity than over a surface with low conductivity.

Lower conductivity surfaces, such as dry, infertile soil, absorb the power of the transmission more than higher conductivity surfaces, such as sea water or arable land.

A radio beacon transmission has three components:

1. Direct line-of-sight wave

The line-of-sight wave is insignificant beyond visual range of the transmitting tower and does not have a substantial impact upon signal reception.

2. Ground wave

The ground wave portion of the signal propagates along the surface of the earth, losing strength due to spreading loss, atmospheric refraction and diffraction, and attenuation by the surface over which it travels (dependent upon conductivity).

3. Sky wave

Depending on its reflectance, this skyward portion of the beacon signal may bounce off the ionosphere and back to Earth, causing reception of the ground wave to fade. Fading—which may cause reception to fade in and out—occurs when the ground and sky waves interfere with each other. This problem usually occurs in the evening when the ionosphere becomes more reflective and usually on the edge of coverage areas. Fading is not usually an issue with overlapping coverage areas of beacons and their large overall range.

Atmospheric attenuation plays a minor part in signal transmission range because it absorbs and scatters the signal. This type of loss is the least significant of those described.

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

Radiobeacon Reception

Various noise sources affect beacon reception and include:

- Engine noise
- Alternator noise
- Noise from power lines
- DC to AC inverting equipment
- Electric devices such as CRTs, electric motors, and solenoids

Noise generated by these types of equipment can mask the beacon signal, reducing or impairing reception.

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

Radiobeacon Antenna Location

When using the internal beacon receiver as the correction source, antenna location will influence the performance of the internal beacon receiver.

A good location will:

- Have a clear view of the sky (important for GNSS,WAAS, and Atlas signal reception)
- Be at least three feet away from all forms of transmitting antennas,communications, and electrical equipment, to reduce the amount of noise present at the antenna
- Be the best for the application, such as the center line of the vehicle or vessel (the position calculated by the beacon receive is measured to the center of the antenna)
- Not be in areas that exceed specified environmental conditions

Topic Last Updated: v1.07 / February 1, 2017

Atlas

Atlas Signal Information

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

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

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

Base Station

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).

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

DGPS Base Station

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

e-Dif-Extended Differential Option

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.

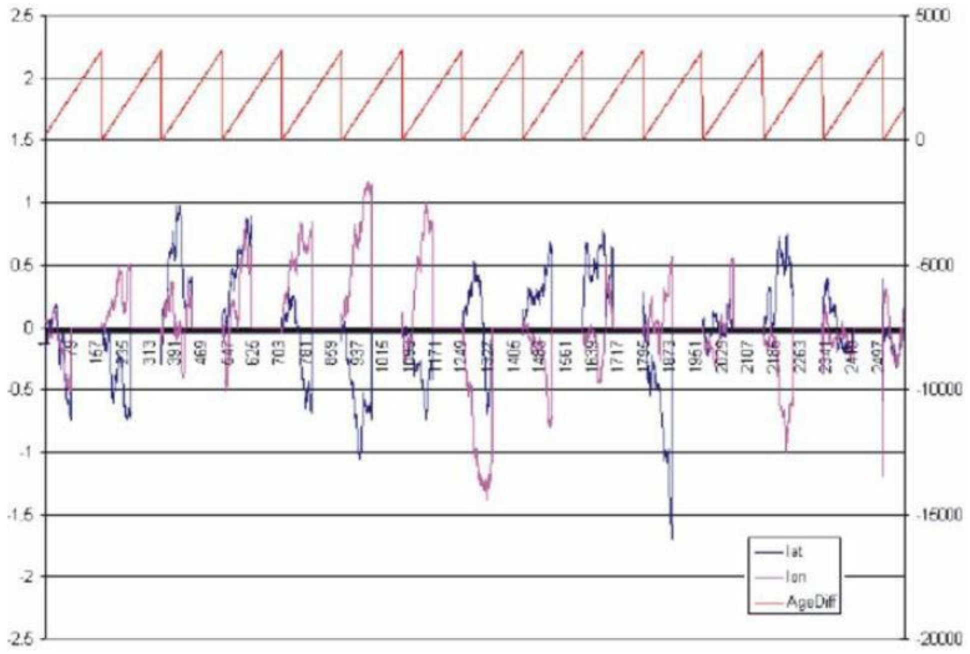
e-Dif may be used anywhere geographically and is especially useful where SBAS networks have not yet been installed, such as South America, Africa, Australia, and Asia. Two things are required to enable e-Dif. First your receiver will require standard MFA application software to be installed on it. A 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

L-Dif

L-Dif Startup

On startup, the receiver with the L-Dif running requires several commands to initialize the proprietary messages that are sent over the air.

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

L-Dif Performance

The receiver’s positioning performance in L-Dif mode is dependant upon:

- Environment of the base and rover receivers
- Distance between them and
- Accuracy of the entered coordinates of the base station

Hemisphere GNSS suggests performing your own testing at your location to determine the level of performance you would expect on average. When testing this feature, conduct tests of 12-24 hours—in different environments—and monitor performance against a known coordinate. Do this over a number of days with different states of the ionosphere.

You can monitor the energy level of the ionosphere based upon the amount of solar flare activity at http://www.spaceweather.com.

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

L-Dif Local Differential Option

Local differential (L-Dif) is a specialized message type that can be sent only between two Crescent-based receivers. One receiver is used as the base station and must remain stationary. It is extremely useful to know the coordinates of the base station position but averaging the position over several days will also suffice. The second receiver is used as a rover and the messages must be sent either through a cable or over a radio link.

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

RTK Overview

Real Time Kinematic (RTK) positioning is the highest form of navigational accuracy for GNSS receivers. Hemisphere GNSS offers RTK for both Crescent and Eclipse platforms. See RTK commands for more information.

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

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 76 (GPS), Bin 66 (GLONASS), Bin 36 (BEIDOU) Or

Bin 16 (All constellations; required for GALILEO)

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

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

(Crescent receivers must log Bin 94, 95, and 96 messages for GPS). 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: v1.11 / November 15, 2018

Receiver Operation

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

Commands and Messages

Commands and Messages Overview

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.

Commands

Beacon receiver commands and messages

DGPS base station commands

e-Dif commands

General Operation and Configuration Commands

GLONASS commands and messages

GNSS commands

Local differential and RTK commands and messages

SBAS commands

RAIM commands

Vector commands and messages

Messages

Binary messages

Data messages

NMEA 0183 messages

NMEA 2000 CAN messages

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

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

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)

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	contain 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.

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

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

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

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

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

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,GGAALLGNSS	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

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

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

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

Local Differential and RTK Commands and Messages

The following table lists the commands supported by Local Differential (L-Dif) and RTK feature 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 andthe GPGLL messages or query the current setting
JNP	Specify the number of decimal places output in the GPGGA and GPGLLmessages
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 the Local Differential (L-Dif) and 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

Beacon Receiver Commands and Messages

If integrating a Hemisphere GNSS SBX beacon module with the receiver GNSS engine, Hemisphere GNSS recommends interfacing the beacon receiver to Port D of the receiver engine. Hemisphere GNSS has implemented some command and message pass-through intelligence for such an integration. In this configuration you can issue the commands in the following table to the beacon receiver through either Port A, Port B, or Port C of the receiver. When you issue queries to the SBX primary communications port, the response messages are output interspersed with RTCM correction information. This may cause conflicts with a GNSS receiver's ability to compute differential corrected solutions. By sending these queries to the SBX secondary communications port the flow of RTCM corrections on the primary port will not be interrupted.

The following table lists the beacon commands/messages found in this Help file.

Query	NMEA 0183 Query Type	Description
GPCRQ,MSK	Standard	Query the SBX for its operational status
GPCRQ,MSS	Standard	Query the SBX for its performance status
GPMSK	Standard	Tune beacon the receiver and turn on diagnostic information
PCSI,0	Hemisphere GNSS proprietary	Query the SBX to output a list of available proprietary PCSI commands
PCSI,1	Hemisphere GNSS proprietary	Query the SBX for a selection of parameters related to the operational status of its primary channel
PCSI,1,1	Hemisphere GNSS proprietary	Obtain beacon status information from the SBX beacon engine inside the receiver
PCSI,2	Hemisphere GNSS proprietary	Query the SBX to output a selection of parameters related to the operational status of its secondary channel
PCSI,3,1	Hemisphere GNSS proprietary	Query the SBX to output the search information used for beacon selection in Automatic Beacon Search mode. The output has three frequencies per line.
PCSI,3,2	Hemisphere GNSS proprietary	Display the ten closest beacon stations
PCSI,3,3	Hemisphere GNSS proprietary	Display the contents of the beacon station database
PCSI,4	Hemisphere GNSS proprietary	Clear search history in Auto mode
PCSI,5	Hemisphere GNSS proprietary	Set the baud rate of Port0 and Port1
PCSI,6	Hemisphere GNSS proprietary	Reboot SBX receiver
PCSI,7	Hemisphere GNSS proprietary	Swap modes on the receiver

The following table lists the beacon messages found in this Help file.

Message	Description
CRMSK	Operational status message of SBX
CRMSS	Performance status message of SBX

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

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

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 pseudorange error statistics
GPGSV	20 Hz	GPS satellites in view
GLGSV	20 Hz	GLONASS satelllites in view
GAGSV	20 Hz	Galileo sattelites 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

Binary Messages

Message Structure

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	Values
Header	Synchronization String	4 byte string	4	$BIN
	Block ID - type of binary message	Unsigned short	2	1, 2, 80, 93, 94, 95, 96, 97, 98, or 99
	DataLength - the length of the binary messages	Unsigned short	2	52, 16, 40, 56, 96, 128, 300, 28, 68, or 304
Data	Binary Data - varying fields of data with a total length of DataLength bytes	Mixed fields	52, 16, 40, 56, 96, 128, 300, 28, 68, or 304	Varies - see message tables
Epilogue	Checksum - sum of all bytes of the data (all DataLength bytes); the sum is placed in a 2-byte integer	Unsigned short	2	Sum of data bytes
	CR- Carriage return	Byte	1	0D hex
	LF - Line feed	Byte	1	0A hex

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

Beacon Receiver Commands

CRMSK Message

Message Type:

Beacon Receiver

Description:

Operational status message of SBX

Command Format to Request Message:

$GPCRQ,MSK<CR><LF>

Command Format:

$CRMSK,FFF.F,X,DDD,Y,N*CC<CR><LF>

Where:

Message Component	Description
FFF.F	Frequency, in kHz (283.5 to 325)
X	Tune mode (M = manual, A = automatic)
DDD	MSK bit rate, in bps (100 or 200)
Y	MSK rate selection mode (M = manual, A = automatic)
N	Period of output of performance status message, in seconds (0 to 100); see CRMSS
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Receiver Response:

Additional Information:

Related Commands

GPCRQ,MSK

Topic Last Updated: v. 1.00 / August 11, 2010

CRMSS Message

Message Type:

Beacon Receiver

Description:

Performance status message of SBX

Command Format to Request Message:

$GPCRQ,MSS<CR><LF>

Message Format:

$CRMSS,XX,YY,FFF.F,DDD*CC<CR><LF>

where:

Message Component	Description
XX	Signal strength, in dB µV/m
YY	Signal-to-noise ratio, in dB
FFF.F	Frequency, in kHz (283.5 to 325)
DDD	MSK bit rate in bps (100 or 200)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Commands:

GPCRQ,MSS

Topic Last Updated: v.1.00 / August 11, 2020

GPCRQ,MSS Command

Command Type:

Beacon Receiver

Description:

Standard query to prompt the SBX for its performance status (response is the CRMSS message)

You can issue this command through the secondary serial port with a standard response issued to the same port. This will not affect the output of RTCM data from the main serial port when the receiver has acquired a lock on a beacon station.

Command Format:

$GPCRQ,MSS<CR><LF>

Receiver Response:

$CRMSS,xx,yy,fff.f,ddd*CC<CR><LF>

where:

Response Component	Description
xx	Signal strength in dBμV/m
yy	Signal-to-noise ratio (SNR) in dB
fff.f	Frequency in kHz (283.5 to 325)
ddd	MSK bit rate in bps (100 or 200)

Response example:

$CRMSS,65,36,322.0,100*CC

The signal strength is 65 dBμV/m, SNR is 36 dB, frequency is 322.0 kHz, and MSK bit rate is 100 bps.

Additional Information:

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

GPMSK Command

Command Type:

Beacon Receiver
Description:

Beacon Tune command

Instruct the SBX to tune to a specified frequency and automatically select the correct MSK rate. When you send this command through Port A, Port B, or Port C, it is automatically routed to Port D. The resulting confirmation of this message is returned to the same port from which you sent the command.

Command Format:

$GPMSK,fff.f,F,mmm,M[,n]<CR><LF>

where:

Command/Response Component	Description
fff.f	Beacon frequency in kHz (283.5 to 325) This may be left blank if the following field 'F' is set to 'A' (automatic) or 'D' (database)
F	Frequency selection mode (M = manual, A = automatic, D = database)
mmm	MSK bit rate This may be left blank if the following field 'M' is set to 'A' (automatic) or 'D' (database)
M	MSK rate selection mode (M = manual, A = automatic, D = database)
n	Period of output of CRMSS performance status message (0 to 100 seconds), where leaving the field blank will output the message once **Note:** This field is optional when using database tuning mode or automatic tuning mode.

Receiver Response:

$CRMSS,xx,yy,fff.f,ddd*CC<CR><LF>

where:

Response Component	Description
xx	Signal strength in dBµV/m

yy	Signal-to-noise ratio (SNR) in dB
fff.f	Frequency in kHz (283.5 to 325)
ddd	MSK bit rate in bps (100 or 200)

Example:

To instruct the SBX to tune to 310.5 kHz with a bit rate of 100 and output the CRMSS message every 20 seconds issue the following command:

$GPMSK,310.5,M,100,M,20<CR><LF>

...and the receiver response is:

$CRMSS,65,36,310.5,100*CC

(repeating every n=20 seconds)

If using database tuning mode issue the following command:

$GPMSK,,D,,D<CR><LF>

If using automatic tuning mode issue the following command:

$GPMSK,,A,,A<CR><LF>

Additional Information:

When the SBX acknowledges this message, it immediately tunes to the specified frequency and demodulates at the specified rate.

When you set 'n' to a non-zero value, the SBX outputs the CRMSS message at that period through the serial port from which the SBX was tuned. When you issue this command with a non-zero'n' value through Port B, the periodic output of the CRMSS performance status message does not impact the output of RTCM on Port A. However, when tuning the SBX with a non-zero 'n' value through Port A, the CRMSS message is interspersed with the RTCM data. Most GPS engines will not be able to filter the CRMSS message,causing the overall data to fail parity checking. When power to the SBX is removed and reapplied, the status output interval resets to zero (no output).

When tuning the SBX engine, if the 'n' field in this message is non-zero, the CRMSS message output by the SBX may interrupt the flow of RTCM data to the GPS receiver. Repower the SBX to stop the output of the CRMSS message or retune the Beacon receiver with 'n' set to zero.

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

GPS 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?

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

General Operation and Configuration

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.

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

JALT Command

Command Type:

General Operation and Configuration

Description:

Turn altitude aiding for the receiver on or off

When set to something other than NEVER, altitude aiding uses a fixed altitude instead of using one satellite's observations to calculate the altitude. The advantage of this feature, when operating in an application where a fixed altitude is acceptable, is that the extra satellite's observations can be used to the betterment of the latitude, longitude, and time offset calculations, resulting in improved accuracy and integrity. Marine markets, for example, may be well suited for use of this feature.

Command Format:

$JALT,c[,h[,GEOID]]<CR><LF>

where 'c' (feature status variable) and 'h' (threshold variable) may be one of the following:

c Value	Corresponding h Value	Description	Format
NEVER	N/A	Default mode of operation where altitude aiding is not used.	$JALT,NEVER<CR><LF>
SOMETIMES	PDOP	Sets the receiver to use altitude aiding depending upon the PDOP threshold.	$JALT,SOMETIMES,PDOP<CR><LF>
SATS	NUMSATS	Sets the receiver to use altitude aiding depending upon the number of visible satellites. If there are fewer visible satellites than specified by NUMSATS, altitude aiding is used.	$JALT,SATS,NUMSATS<CR><LF>
ALWAYS	HEIGHT	Sets the receiver to use altitude aiding regardless	$JALT,ALWAYS,HEIGHT<CR><LF> $JALT,ALWAYS,HEIGHT,GEOID<CR><LF>

To obtain a HEIGHT value to use with ALWAYS (using DGPS positions), average the HEIGHT over a period of time (the longer the time period, the more accurate this HEIGHT value). This is the ellipsoidal height.

$JALT,ALWAYS,HEIGHT<CR><LF>

If you use the height reported from the GPGGA message (this is actually geoidal and not ellipsoidal), use the following command:

$JALT,ALWAYS,HEIGHT,GEOID<CR><LF>

Receiver Response:

$>

Example:

To turn altitude aiding on to SOMETIMES with a PDOP of 5 issue the following command:

$JALT,SOMETIMES,5<CR><LF> 7

To turn altitude aiding on to ALWAYS using the height of 401.6 m as reported in the GPGGA message (geoidal height) issue the following command:

$JALT,ALWAYS,401.6,GEOID<CR><LF>

Additional Information:

To query the receiver for the current setting, issue the JSHOW command. For example, if you issue the following command:

$JALT,ALWAYS,404.2<CR><LF>

...then issuing the JSHOW command displays the following as part of its output:

$>JSHOW,ALT,ALWAYS,404.2

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

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

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

JASC Commands

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)

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

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.

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

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.

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>

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
GLMLA	1 (on) or 0 (off) When set to on the message is sent once (one message for each tracked satellite) and then sent again whenever satellite information changes	GLONASS almanac data
GLGSV	1 or 0	GLONASS satellite in view

Receiver Response:

$>

Example:

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>

Additional Information:

The status of this command is also output in the JSHOW message. What does <CR><LF> mean?

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	R (rate in Hz)	Description
GNGNS	20, 10, 2, 1, 0 or .2	All GNSS fix data (GAGNS output is GALILEO)
GAGSV	1 or 0	GALILEO satellites in view

Receiver Response:

$>

Additional Information:

The status of this command is also output in the JSHOW message. What does <CR><LF> mean?

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	R (rate in Hz)	Description
GQGSV	1 or 0	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 Informatio:

The status of this command is also output in the JSHOW message. What does <CR><LF> mean?

Topic Last Updated: v2.0/ 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 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
GNGGA	20, 10, 2, 1, 0 or .2	GNSS fix data
GNGLL	20, 10, 2, 1, 0 or .2	Geographic position - latitude/longitude
GNGNS	20, 10, 2, 1, 0 or .2	GNSS fix data
GNGSA	1 or 0	GNSS DOP and active satellites

Receiver Response:

$>

Example:

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>

Additional Informatio:

The status of this command is also output in the JSHOW message. What does <CR><LF> mean?

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 range residuals
GNGSA	1 or 0	GPS DOP and active satellite information
GPGST	1 or 0	GNSS pseudorange error statistics
GPGSV	20, 10, 2, 1, 0 or .2	GPS satellites in view

GPHDG or HEHDG	20, 10, 2, 1, 0 or .2	Magnetic deviation and variation for calculating magnetic or true heading
GPHDM or HEHDM	20, 10, 2, 1, 0 or .2	Magnetic heading of the vessel derived from the true heading calculated
GPHDT or HEHDT	20, 10, 2, 1, 0 or .2	True heading of the vessel
GPHEV	20, 10, 2, 1, 0 or .2	Heave value in meters
GPHPR	20, 10, 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?

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

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

Additional Information:

PSAT,INTLT message

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 th 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:

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>

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>

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.

See also: PSAT,RTKPROG message.

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

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 and Messages:

JQUERY,RTKSTAT command PSAT,RTKSTAT message

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

JASC,PSAT,VCT Command

Command Type:

Local Differential and RTK

Description:

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 Value Description 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

Additional Information:

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.

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.

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 Information:

To query the receiver for the current setting, issue the JSHOW command. To change the broadcast station ID, use JRTK,28.

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 :

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

JATLAS Commands

JATLAS,LIMIT Command

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:

Response to issuing command to tune
receiver :

$>

$JATLAS,POS,PRESENT [,OTHER]

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.

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>
```

Query the current setting:

Receiver Response:

```
$>JATLAS,POS,lat,lon,hgt,(LatStDev,LonStDev,HgtStDev)
```

where:

Command Component	Description
lat	Latitude in decimal degrees
lon	Longitude in decimal degrees
hgt	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
LatStDev	Standard deviation of latitude in meters
LonStDev	Standard deviation of longitude in meters
HgtStDev	Standard deviation of height in meters

Example:

A response to querying the saved position would look like:

```
$>JATLAS,POS,33.64334383,-111.89596094,455.244,(0.062,0.086,0.156)
```

Additional Information:

See $JATLAS,MODE,AUTOSEED and $JATLAS,SEED for additional information about position seeding.

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

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

Command Type:

ATLAS

Description:

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.

Command Format:

```
$JATLAS,POS,lat,lon,hgt,[LatStDev,LonStDev,HgtStDev][,OTHER]<CR><LF>
```

where:

Command Component	Description
lat	Latitude in decimal degrees
lon	Longitude in decimal degrees
hgt	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
LatStDev	Standard deviation of latitude in meters [optional]
LonStDev	Standard deviation of longitude in meters [optional]
HgtStDev	Standard deviation of height in meters [optional]

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.

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.

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

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

Command Type:

ATLAS

Description:

Manually seed the Atlas solution with the input position and optionally standard deviations.

Command Format:

```
 $JATLAS,SEED,lat,lon,hgt,[LatStDev,LonStDev,HgtStDev]<CR><LF>
```

where:

Command Component	Description
lat	Latitude in decimal degrees
lon	Longitude in decimal degrees
hgt	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
LatStDev	Standard deviation of latitude in meters [optional]
LonStDev	Standard deviation of longitude in meters [optional]
HgtStDev	Standard deviation of height in meters [optional]

Query the current setting:

Receiver Response:

$>

If the input coordinates are not close enough to the current location, the response is:

```
$>JATLAS,SEED,Current Position Too Far From Seed
```

Example:

```
$JATLAS,SEED,33.64334383,-111.89596094,455.244,0.062,0.086,0.156<CR><LF>
```

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.

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.

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

$JATLAS,RESET,ENGINE

Description:

The **$JATLAS,RESET,ENGINE** command resets the Atlas engine

Command Type:

ATLAS

Description:

Reset the Atlas engine, forcing the solution to re-converge.

Command Format:

```
$JATLAS,RESET,ENGINE
```

Query the current setting:

Receiver Response:

```
$>
```

Example:

Additional Information:

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.

Description:

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.

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

JATT Commands

JATT

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

JATT,COGTAU Command

Note: The JTAU,COG command provides identical functionality but works with positioning and heading products.

Command Type:

Vector

Description:

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.

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:

$>

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.

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

Additional Information:

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:

$>

Additional Information:

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:

```
 $>
```

Additional Information:

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:

$>

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.

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:

$>

Additional Information:

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
```

Additional Information:

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:

```
$>
```

Additional Information:

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:

$>

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.

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:

```
$>
```

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.

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:

$>

Additional Information:

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: $>

Additional Information:

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:

$>

Additional Information:

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

JATT,NEGTILT Command

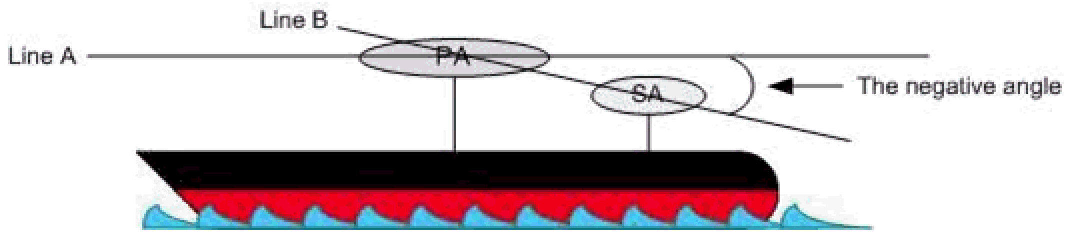
Command Type:

Vector

Description:

Turn the negative tilt feature on or off or query the current setting.

When the secondary GPS antenna (SA) is below the primary GPS antenna (PA), there is an angle formed between a horizontal line through the center of the primary antenna (Line A in the diagram below) and an intersecting line through the center of the primary and secondary antennas (Line B). This angle is considered to be negative.



Depending on the convention for positive and negative pitch/roll, you want to change the sign (either positive or negative) of the pitch/roll.

Command Format:

Turn negative tilt feature on/off

To change the sign of the pitch/roll measurement:

$JATT,NEGTILT,YES<CR><LF>

To return the sign of the pitch/roll measurement to its original value:

$JATT,NEGTILT,NO<CR><LF>

Query the current setting:

$JATT,NEGTILT<CR><LF>

Receiver Response:

$>

Additional Information:

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

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

Additional Information:

The HDM message is for a magnetic compass. The message will be HCHDM when requesting with $JATT,NMEAHE,1specified.

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

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:

$>

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).

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:

$>

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)

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:

```
$>
```

Additional Information:

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

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:

$>

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.

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

JATT,SPDTAU Command

Note: The JTAU,SPEED command provides identical functionality but works with Crescent and Eclipse products in addition to Crescent Vector products.

Command Type:

Vector

Description:

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:

$>

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/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.

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:

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' Value Value 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

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:

Topic Last Updated: v.1. 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 issuing command to turn tilt aiding on/off:

$>

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

Additional Information:

Tilt aiding is required to increase the antenna separation of the Vector OEM beyond the default 0.5 m length.

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

JATT,TILTCAL Command

Command Type:

Vector

Description:

Calibrate the internal tilt sensors of the Vector Calibration takes approximately two seconds and is automatically saved to memory for subsequent power cycles.

You can calibrate the tilt sensor of the Vector in the field but the Vector enclosure must be horizontal when you calibrate.

$JATT,TILTCAL<CR><LF>

Command Format:

Receiver Response:

```
$>
```

Additional Information:

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

JBAUD Command

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.

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

JBIN Command

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
Bin122	122	20, 10, 5, 2, 1, 0, .5, .2, .1	Alternate position solution data
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.

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

JBOOT Commands

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:

```
$>
```

Additional Information:

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

JCONN Command

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

Examples:

$JCONN,A,B<CR><LF>

$JCONN,PortA,PortB<CR><LF>

To disconnect virtual connection:

$JCONN,X<CR><LF>

Receiver Response:

$>

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.

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

JDIFF Commands

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.
BEACON	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.
WAAS	Instruct the receiver to use SBAS. This is also the response when running the local dif application as the base.
RTK	Response when running the local dif or rover RTK application for the rover.
LBAND	Instruct the receiver to turn on theAtlas module and useAtlas. Setting diff to anything other thanAtlas turns off theAtlas module.

X	Instruct the receiver to use e-Dif mode
NONE	Instruct the receiver to operate in autonomous mode. This turns off the use of SBAS,Atlas, and RTCM2 (DGPS); however, RTK is still allowed.

,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 specifying the differential source:

$>

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:
The status of this command is also output in the JSHOW message.

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

JDIFF,AVAILABLE Command

Command Type:

General Operation and Configuration

Description:

Query the receiver for the differential types currently being received

Command Format:

$JDIFF,AVAILABLE<CR><LF>

Receiver Response:

$>JDIFFX,AVAILABLE,x[,x][,x]...[,x]

where 'x' is the differential type(s)

Example:

Additional Information:

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

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 issuing command to exclude differential sources

$>

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:

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:

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.

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

JDIFFX,SOURCE Command

Command Type:

General Operation and Configuration

Description:

Query the receiver for the differential source

Command Format:

$JDIFFX,SOURCE<CR><LF>

Receiver Response:

$>JDIFFX,source

where 'source' is the differential source

Example:

Response if Atlas is the differential source:

$>JDIFFX,SOURCE,LBAND

Response if RTK is the differential source through Port B:

$>JDIFFX,SOURCE,PORTB

Additional Information:

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

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:

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]

Additional Information:

This setting is automatically saved and can be reset to default by sending $JRESET

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

JEPHOUT, PERIODSEC Command

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 ionoutc 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

Additional Information:

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

JETHERNET Commands

JETHERNET Command Overview

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
JETHERNET, PORTI	Enable/Disable PORTI virtual serial port
JETHERNET, NTRIPCLIENT	Receive RTK correction over IP from NTRIP caster

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

JETHERNET,MODE

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:

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

JETHERNET,PORTI

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:

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.

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,PASS
WORD<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.

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

JFLASH Commands

JFLASH Command Overview

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

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:

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:

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:

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:

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:

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

Additional Information:

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

$>

Additional Information:

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

JFREQ Command

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)

Note:

Examples:

Correct: $JFREQ,1545915,600 (1,545,915 Hz,)

To auto-tune the receiver:

$JFREQ,0<CR><LF>

Note:

Query the current
setting:

$JFREQ<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:

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

Example:

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

Additional Information:

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.

Coverage Area	Frequency	Baud Rate	Satellite Name
North and South America	1545.915 MHz	600	AMERICAS
Asia-Pacific	1545.855 MHz	600	APAC
Europe, Middle East and Africa	1545.905 MHz	600	EMEA
Eastern Europe and Middle East	1545.915 MHz	600	MEAS

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

JGEO Command

JGEO Command

Command Type:

SBAS

Description:

Display information related to the current frequency of SBAS or Atlas satellite and its location in relation to the receiver's antenna

Command Format:

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

where ',ALL' is an optional field that displays information for all SBAS satellites (including those not being used)

Receiver Response:

$>JGEO,SENT=1575.4200,USED=1575.4200,PRN=prn,LON=lon,EL=ele,AZ=az

where:

Response Component	Description
JGEO	Message header
Sent=1575.4200	Frequency sent to the digital signal processor
Used=1575.4200	Frequency currently used by the digital signal processor
PRN=prn	WAAS satellite PRN number
Lon=-lon	Longitude of the satellite
El=ele	Elevation angle from the receiver antenna to the WAAS satellite, reference to the horizon
AZ=az	Azimuth from the receiver antenna to the WAAS satellite, reference to the horizon

Example:

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:

$>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.

Additional Information:

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

JHTYPE_SHOW Command

JHTYPE_SHOW Command

Queries the hardware type.

JI Command

JI Command

Command Type:

General Operation and Configuration

Description:

Display receiver information, such as its serial umber 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,19422368,20,1,04062018,01/01/1900,01/01/6455,5.9Aa08,89

Additional Information:

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

JK Command

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 similarto 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

Last Updated: v.3.0/ December 30, 2019

JK SHOW Command

Command Type:

General Operation and Configuration

Description:

Contain authorization information

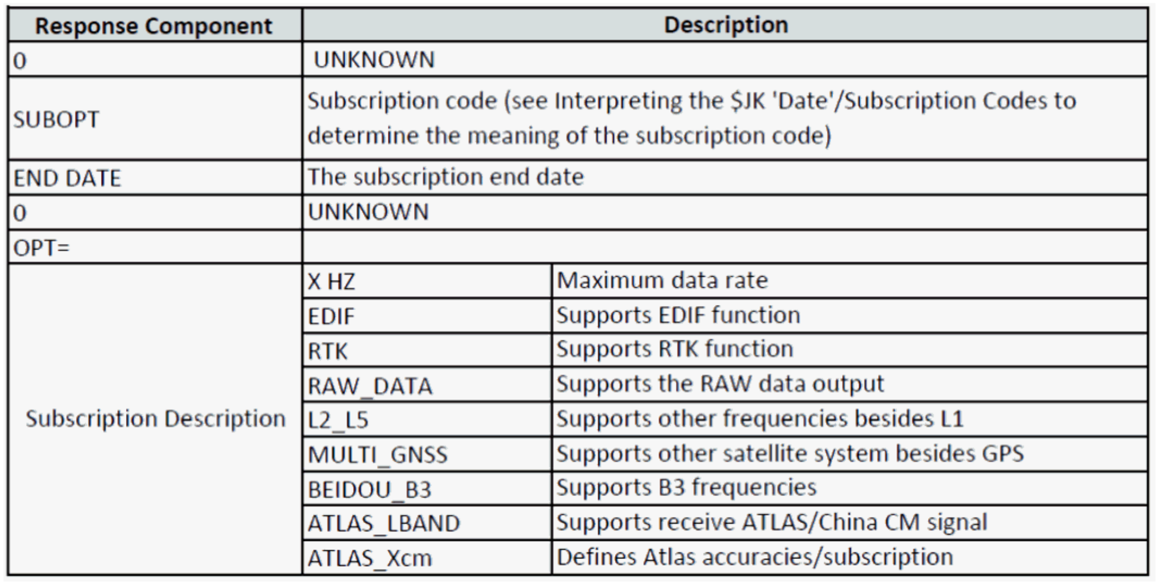
Command Format:

$JK,SHOW<CR><LF>

Receiver Response:

$>JK,SHOW,0,SUBOPT,ENDDATE,0,OPT=,SUBSCRIPTION DESCRIPTION,<CR><LF>

where:



Response Component	Description	
0	UNKNOWN	
SUBOPT	Subscription code (see Interpreting the $JK 'Date'/Subscription Codes to determine the meaning of the subscription code)	
END DATE	The subscription end date	
0	UNKNOWN	
OPT=		
Subscription Description	X HZ	Maximum data rate
	EDIF	Supports EDIF function
	RTK	Supports RTK function
	RAW_DATA	Supports the RAW data output
	L2_L5	Supports other frequencies besides L1
	MULTI_GNSS	Supports other satellite system besides GPS
	BEIDOU_B3	Supports B3 frequencies
	ATLAS_LBAND	Supports receive ATLAS/China CM signal
	ATLAS_Xcm	Defines Atlas accuracies/subscription

Example:

$>JK,SHOW,0,157F,12/31/2016,0,OPT=,20HZ,EDIF,RTK,BASE,RAW_DATA,L2_L5,MULTI_ GNSS,BEIDOUB3,ATLAS_LBAND,ATLAS_30cm

Additional Information:

Interpreting the $JK 'Date'/Subscription Codes

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

JLBEAM Command

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:

Response Component	Description
"Sent" freq	Frequency sent to the digital signal processor (DSP)
"Used" freq	Frequency currently being used by the digital signal processor (DSP)
"Baud" xxxx	Currently used baud rate of the acquired signal
"Geo" xxx	Currently used satellites longitude (in degrees)

The output second line components are described in the following table:

Response Component	Description
freq	Frequency of the spot beam
lon	Longitude of the center of the spot beam (in degrees)
lat	Latitude of the center of the spot beam (in degrees)
baud	Baud rate at which this spot beam is modulated
satlon	Satellites longitude (in degrees)

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 Command

$>JLBEAM,1535.1375,179,15,1200,(109)

Additional Information:

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

JLIMIT Command

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

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.

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

JLXBEAM Command

JLXBEAM Command

Command Type:

L-B
and

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,*
```

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

JMASK Command

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.

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

JMODE Commands

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] [,NULLNM

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

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

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 receiveris 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:

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

Additional Information:

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

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:

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

JMODE,FOREST Command

Command Type:

General Operation and Configuration

Description:

Enable/disable high gain functionality (for tracking under canopy) 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,MIXED.

Command Format:

Enable/disable high gain functionality

To enable high gain functionality:

$JMODE,FOREST,YES<CR><LF>

To disable high gain functionality:

$JMODE,FOREST,NO<CR><LF>

Query the current setting:

$JMODE,FOREST<CR><LF>

Receiver Response:

Response to issuing command to turn functionality on/off:

$>

Response to querying the current setting:

If high gain functionality is currently enabled the response is:

$>JMODE,FOREST,YES

If high gain functionality is currently disabled the response is:

$>JMODE,FOREST,NO

Additional Information:

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

JMODE,GLOFIX

Command Type:

General Operation and Configuration

Description:

Enable/disable use of RTCM v3 (RTK) GLONASS correctors.

GLOFIX does not affect CMR or ROX (CMR does not have GLONASS, and ROX correctors are always used regardless of the GLOFIX setting) and SureTrack is automatically used for any satellite that does not have GLONASS correctors.

Command Format:

Enable/disable use of RTCM v3 GLONASS correctors

To enable use of RTCM v3 GLONASS correctors:

$JMODE,GLOFIX,YES<CR><LF>

To disable use of RTCM v3 GLONASS correctors:

$JMODE,GLOFIX,NO<CR><LF>

Query the current setting:

$JMODE,GLOFIX<CR><LF>

Receiver Response:

Response to issuing command to turn functionality on/off:

$>

Response to querying the current setting

If use of RTCM v3 GLONASS correctors is currently enabled the response is:

$>JMODE,GLOFIX,YES

If use of RTCM v3 GLONASS correctors is currently disabled the response is:

$>JMODE,GLOFIX,NO

Additional Information:

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

JMODE,GLOOFF Command

Command Type:

General Operation and Configuration

Description:

Set the receiver to use GLONASS data in the solution

Command Format:

Close/Open GLONASS operation

Close GLONASS operation:

$JMODE,GLOOFF,YES<CR><LF>

Open GLONASS operation:

$JMODE,GLOOFF,NO<CR><LF>

Receiver Response:

Response to issuing command to turn enable/disable GLONASS operation

$>

Response to querying the current setting

If GLONASS operation is currently enabled the response is:

$>JMODE,GLOOFF,NO

If GLONASS operation is currently disabled the response is:

$>JMODE,GLOOFF,YES

Additional Information:

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

JMODE,GPSOFF Command

Command Type:

General Operation and Configuration

Description:

Set the receiver to use GPS data in the solution or query the current setting

Command Format:

Close/Open GPS Operation

Close GPS operation:

$JMODE,GPSOFF,YES<CR><LF>

Open GPS operation:

$JMODE,GPSOFF,NO<CR><LF>

Receiver Response:

Response to issuing command to turn enable/disable GPS-only operation

$>

Response to querying the current setting

If GPS-only operation is currently enabledthe response is:

$>JMODE,GPSONLY,YES

If GPS-only operation is currently disabled the response is:

$>JMODE,GPSONLY,NO

Additional Information:

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

JMODE,GPSONLY Command

Command Type:

General Operation and Configuration

Description:

Set the receiver to use GPS data in the solution or query the current setting (if GLONASS is available, setting to YES will cause the receiver to only use GPS data)

Command Format:

Enable/disable GPS-only operation

Enable GPS-only operation:

$JMODE,GPSONLY,YES<CR><LF>

Disable GPS-only operation (use GLONASS as well if available):

$JMODE,GPSONLY,NO<CR><LF>

Query the current setting:

$JMODE,GPSONLY<CR><LF>

Receiver Response:

Response to issuing command to turn enable/disable GPS-only operation

$>

Response to querying the current setting

If GPS-only operation is currently enabled the response is:

$>JMODE,GPSONLY,YES

If GPS-only operation is currently disabled the response is:

$>JMODE,GPSONLY,NO

Additional Information:

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

JMODE,L1ONLY Command

Command Type:

General Operation and Configuration

Description:

Set the receiver to use L1 data even if L2 data is available or query the current setting:

- When set to YES receiver will use Atlas DGPS service or L1 RTK
- When set to NO receiver will use Atlas high precision services or L1/L2 RTK

Command Format:

Set receiver to use/not use L1 data even if L2 data is available

To use L1 data (even if L2 data is available):

$JMODE,L1ONLY,YES<CR><LF>

To use L2 data if it is available:

$JMODE,L1ONLY,NO<CR><LF>

Query the current setting:

$JMODE,L1ONLY<CR><LF>

Receiver Response:

Response to issuing command to turn functionality on/off

$>

Response to querying the current setting

If the receiver is currently using L1 data only even if L2 data is available the response is:

$>JMODE,L1ONLY,YES

If the receiver is currently using L2 data if it is available the response is:

$>JMODE,L1ONLY,NO

Additional Information:

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

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

Additional Information:

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:

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 signalsin 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 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:

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

Additional Information:

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,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,STRICTRTK,YES

If setting is currently disabled the response is:

$>JMODE,STRICTRTK,NO

Additional Information:

This mode is not saved between power cycles.

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/disableSureTrack 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

Additional Information:

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>

Response to issuing command to enable/disable command

Receiver Response:

$>

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

Additional Information:

This mode is not saved between power cycles (for now)..

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

Additional Information:

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

JMODE,TUNNEL Command

Command Type:

General Operation and Configuration

Description:

Enable/disable faster reacquisition after coming out of a tunnel or query the current setting

Command Format:

Enable/disable faster reacquisition after coming out of a tunnel

To enable faster reacquisition:

$JMODE,TUNNEL,YES<CR><LF>

To disable faster reacquisition:

$JMODE,TUNNEL,NO<CR><LF>

Query the current setting:

$JMODE,TUNNEL<CR><LF>

Receiver Response:

Response to issuing command to turn functionality on/off

$>

Response to querying the current setting

If setting is currently enabled the response is:

$>JMODE,TUNNEL,YES

If setting is currently disabled the response is:

$>JMODE,TUNNEL,NO

Additional Information:

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

JMSG99 Command

JMSG99 Command

Command Type:

Vector

Description:

Change the output in the Bin99 message to be from the specified antenna

Format:

$JMSG99,0

where '0' is used view the primary antenna SNR (default)

$JMSG99,1

where '1' is used view the secondary antenna SNR

Receiver Response:

$>

Other:

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

JNMEA

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 followin:

>JNMEA,GGAALLGNSS,YES

If set to no, querying the current setting returns the following:

$>JNMEA,GGAALLGNSS,NO

Additional Information:

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.

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.

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

$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:

3. HDT
4. HDG
5. HDM
6. HPR
7. THS
8. 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.

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

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.

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

JNP Command

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

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.

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

JOFF Commands

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:

$>

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

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

$JOFF,ALL<CR><LF>

Command Format:

Receiver Response:

$>

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

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

JOFF Command

Command Type:

GPS

Description:

Turn off all data messages being output through the curren tport 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:

$>

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

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

JPOS Command

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

Additional Information:

The status of this command is also output in the JSHOW message.

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

JPPS Command

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>

Receiver Response:

Response to querying the current setting:

$JPPS,WIDTH,999.996<CR><LF>

Example:

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>

Additional Information:

This mode is not saved between power cycles

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.

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

JPPS, PERIOD Command

Command Type:

General Operation and Configuration

Description:

Specify the pps period (in seconds) 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

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

JPRN,EXCLUDE Command

JPRN,EXCLUDE Command

Note: For advanced users only. Not required for typical operation.

Command Type:

General Operation and Configuration Commands

Description:

For advanced users only.

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:

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

$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.

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

JQUERY Commands

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>

Receiver 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>

Additional Information:

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:

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

JQUERY,RTKSTAT Command

Command Type:

Local Differential and RTK

Description:

Perform a one-time query of the most relevant parameters affecting RTK

Command Format:

$JQUERY,RTKSTAT<CR><LF>

As an alternative you can log this as a message using the JASC,PSAT,RTKSTAT command.

Receiver Response:

$>JQUERY,RTKSTAT,MODE,TYP,AGE,SUBOPT,DIST,SYS,NUM,SNR,RSF,BSF,HAG, ACCSTAT,SNT

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, L5 • GLONASS: G1, G2 • Galileo: E5a, E5b, E5a+b, E6
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 (< 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 • 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

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
```

Additional Information:

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

JRAD Commands

JRAD Command Overview

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
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
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

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

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

JRAD,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	Description
LAT	Latitude of the reference point in decimal degrees
LON	Longitude of the reference point in decimal degrees
HEIGHT	Ellipsoidal height of the reference point in meters

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:

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:

Command Component	Description
lat	Latitude of the reference point in decimal degrees
lon	Longitude of the reference point in decimal degrees
height	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

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

Additional Information:

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

Additional Information:

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

Additional Information:

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

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, a you will still need to define a reference position for e-Dif unless relative positioning is sufficient for any needs.

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

Additional Information:

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)

Additional Information:

The $JASC,RTCM,1 command must be sent to the receiver to start outputting standard RTCM corrections.

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)

Additional Information:

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.

Additional Information:

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 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:

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

JRAIM Command

Command Type:

RAIM

Description:

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

Command 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 make a decision 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.

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

JRTCM3 Commands

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)

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

JRTCM3,EXCLUDE

Command Type:

Local Differential and RTK

Description:

Specify RTCM3 message types to not be transmitted (excluded) by base station

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 excludes pecific 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

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 message sto 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

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 antennan ame 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

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

JRTK Commands

JRTK Command Overview

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 issuecommand to base station or rover)
JRTK,1,P	Set the receiver's reference coordinates to the current calculated position if you donot 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 issuecommand 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

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:

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

JRTK,1,LAT,LON,HEIGHT Command

Command Type:

Local Differential and RTK

Description:

Set the receiver's reference position to the coordinates you enter (can issue command to base station or rover)

Command Format:

$JRTK,1,lat,lon,height<CR><LF>

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

Receiver Response:

$>

Additional Information:

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)

$JRTK,1,P<CR><LF>

Command Format:

Receiver Response:

$>

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).

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

Additional Information:

Also see the JRTK,6 command.

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

Additional Information:

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

JRTK,6 Command

Command Type:

Local Differential and RTK

Description:

Display the progress of the base station (can issue command to base station)

Command Format:

$JRTK,6<CR><LF>

Receiver Response:

$JRTK,6,TimeToGo,ReadyTransmit,Transmitting

where:

Response Component	Description
TimeToGo	Seconds left until ready to transmit RTK
ReadyTransmit	Non zero 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 be suspended with $JRTK,5,0.
Transmitting	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.

Example:

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

Additional Information:

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

JRTK,12 Command

Warning! **Hemisphere GNSS recommends that only advanced users employ this command.**

Command Type:

Local Differential and RTK

Description:

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

where 'x' is:

1 = Allow RTK (recommended, and the default)

0 = Do not allow RTK, stay in L-Dif

Receiver Response:

```
$>
```

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.

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

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:

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:

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:

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

JRTK,28 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:

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

JSAVE Command

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

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.

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

JSHOW Commands

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 theAtlas 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)

Example:

See "Receiver Response" section above

Additional Information:

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:

- 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,PORTA** **$JSHOW,ASC,PORTB** **$JSHOW,ASC,PORTC**	**$>JSHOW,ASC,RTCM,1** **$>JSHOW,ASC,RTCM,1** **$>JSHOW,ASC,CMR,1,OTHER** **$>JSHOW,ASC,D1,1,PORTC**

$JSHOW	**$>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:

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:

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:

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:

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

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

JSIGNAL Command

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>

Additional Information:

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

JSMOOTH 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.

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

JSYSVER Command

Command Type:

General Operation and Configuration

Description:

Returns the boot loader version from the GPS card

Command Format:

$JSYSVER<CR><LF>

Receiver Response:

$>SYSVER,v

where 'v' is the boot loader version

Example:

Response when the boot loader version is 75

$>SYSVER,75

Additional Information:

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

JT Command

Command Type:

General Operation and Configuration

Description:

Query the receiver for its GPS engine type

Command Format:

$JT<CR><LF>

Receiver Response:

$>JT,xxxx

where xxxx indicates the GPS engine and mode:

JT Command Response (xxxx)	GPS Engine	Mode
DF2b	Eclipse	WAAS, RTK Base
DF2g	Eclipse	L-band
DF2r	Eclipse	RTK Rover
DF3g	Eclipse II	WAAS, RTK Base
DF3i	Eclipse II	e-Dif
DF3r	Eclipse II	RTK Rover
MF3g	miniEclipse	WAAS, RTK Base
MF3i	miniEclipse	e-Dif
MF3r	miniEclipse	RTK Rover
SX2a	Crescent Vector	WAAS RTK
SX2b	Crescent	Base
SX2g	Crescent	WAAS
SX2i	Crescent	e-Dif
SX2r	Crescent	Rover

Example:

When you issue the $JT<CR><LF>command a typical responsemay 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).

Additional Information:

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

JTAU Commands

JTAU Command Overview

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

JTAU,COG Command

Note:

The JATT,COGTAU command provides identical functionality but works only with Crescent Vector products.

Command Type:

GPS

Description:

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 200.1 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.

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

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 200.2 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/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.

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

JTIMING Commands

$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.

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.

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

$JTIMING,HALTCLOCKSTEER,YES 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 issuing command to enable/disable GNSS clock steering:

```
$>
```

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.

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

PCSI Commands

PCSI,1 Command (Status Line A, Channel 0 command)

Command Type:

Beacon Receiver

Description:

Hemisphere GNSS proprietary NMEA 0183 query

Query the SBX for a selection of parameters related to the operational status of its primary channel

Command Format:

$PCSI,1<CR><LF>

Receiver Response:

$PCSI,ACK,1

$PCSI,CS0,PXXX-Y.YYY,SN,fff.f,M,ddd,R,SS,SNR,MTP,WER,ID,H,T,G

where:

Response Component	Description
CS0	Channel 0
PXXX-Y.YYY	Resident SBX firmware version
SN	SBX receiver serial number
fff.f	Channel 0 current frequency
M	Frequency mode (A = automatic, M = manual, D = database)
ddd	MSK bit rate
R	RTCM rate mode (A = automatic, M = manual, D = database)
SS	Signal strength
SNR	Signal-to-noise ratio
MTP	Message throughput
WER	Word Error Rate - Percentage of bad 30-bit RTCM words in the last 25 words
ID	Beacon ID to which the receiver’s primary channel is tuned
H	Health of the tuned beacon [0-7]
T	$PCSI,1 status output period [0-99]
G	AGC gain in dB (0 to 48 db)

Additional Information:

Optionally you can modify the Status Line A query to request the output of the response message once every period at a specified output rate. It has the following format, where 'T' is the output period in seconds:

$PCSI,1,T<CR><LF>

The response will be:

$PCSI,ACK,1

$PCSI,CS0,PXXXY.YYY,SN,fff.f,M,ddd,R,SS,SNR,MTP,WER,ID,H,T,G

You can stop the output of the message by either of the following:

· Cycling receiver power

· Issuing the **$PCSI,1<CR><LF>** query without the output period field

The response message has the same format as discussed above. In addition to this modified version of the Status Line A command, an additional 'S' field may be placed after the 'T' field, resulting in the following command:

$PCSI,1,T,S<CR><LF>

The 'S' field is not a variable and specifies that the output of the Status Line A message should continue after the power has been cycled. To return the receiver to the default mode (in which message output ceases after receiver power is cycled) send the **$PCSI,1<CR><LF>** query to the receiver.

You may send the $PCSI,1 query through either serial port for reporting of the full status of the primary receiver channel. The query response is returned to the port from which you issued the command. When querying the primary receiver channel using the secondary serial port, no interruptions in RTCM data output will occur on the primary port provided the SBX has acquired a valid beacon.

The response is different depending on whether you are connected directly to the SBX-4 or not.

· If connected directly (by hardware or JCONN), the response will be both an acknowledgement as well as the full PCSI,1 message.

· If connected through a Crescent receiver (such as the R110) you may see the full PCSI,1 message. Consider PCSI,1,1 to generate periodic output.

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

PCSI,1,1 Command (Beacon Status command)

Command Type:

Beacon Receiver

Description:

Obtain PCSI,CS0 beacon status data from an SBX engine when interfaced to the receiver Port D. When you send this command through either Port A, B, or C it is automatically routed to Port D. The resulting PCSI,CS0 message is returned to the same port from which the command was sent at the desired rate.

Command Format:

$PCSI,1,1<CR><LF>

Receiver Response:

$PCSI,CS0,Pxxx-y.yyy,SN,fff.f,M,ddd,R,SS,SNR,MTP,WER,ID,H,T,G

where:

Response Component	Description
CS0	Channel 0
PXXX-Y.YYY	Resident SBX firmware version
SN	SBX receiver serial number
fff.f	Channel 0 current frequency
M	Frequency mode (A = automatic, M = manual, D = database)
ddd	MSK bit rate
R	RTCM rate mode (A = automatic, M = manual, D = database)
SS	Signal strength
SNR	Signal-to-noise ratio
MTP	Message throughput
WER	Word Error Rate - Percentage of bad 30-bit RTCM words in the last 25 words
ID	Beacon ID to which the receiver's primary channel is tuned
H	Health of the tuned beacon (0-7)
T	$PCSI,1 status output period (0-99)
G	AGC gain in, dB (0 to 48)

Example:

$PCSI,CS0,P030-0.000,19001,313.0,D,100,D,18,8,80,0,63,0,1,48

Additional Information:

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

PCSI,2 Command (Status Line B, Channel 1 command)

Command Type:

Beacon Receiver

Description:

Hemisphere GNSS proprietary NMEA 0183 query

Query the SBX to output a selection of parameters related to the operational status of its secondary channel

Command Format:

$PCSI,2<CR><LF>

Receiver Response:

$PCSI,ACK,2

$PCSI,CS1,PXXX-Y.YYY,SN,fff.f,M,ddd,R,SS,SNR,MTP,WER,ID,H,T

Where:

Response Component	Description
CS1	Channel 1
PXXX-Y.YYY	Resident SBX firmware version
SN	SBX receiver serial number
fff.f	Channel 1 current frequency
M	Frequency mode (A = automatic, M = manual, D = database)
ddd	MSK bit rate
R	RTCM rate mode (A = automatic, M = manual, D = database)
SS	Signal strength
SNR	Signal to noise ratio
MTP	Message throughput
WER	Word error rate - Percentage of bad 30-bit RTCM words in the last 25 words
ID	Beacon ID to which the receiver's secondary channel is tuned
H	Health of the tuned beacon (0-7)
T	$PCSI,1 status output period (0-99)

Example:

$PCSI,ACK,2

$PCSI,CS1,P030-0.004,770737,291.0,D,200,D,-7,2,0,100,1024,8,0

Additional Information:

Optionally you can modify the Status Line B query to request the output of the response message once every period. It has the following format, where T is the output period in seconds:

$PCSI,2,T<CR><LF>

The response will be:

$PCSI,ACK,2

$PCSI,CS0,PXXX-Y.YYY,SN,fff.f,M,ddd,R,SS,SNR,MTP,WER,ID,H,T

The response message has the same format as discussed above. The Status Line B message output cannot be set to remain active after the power of the SBX has been cycled.

The $PCSI,2 query may be sent through the either serial port for reporting of the full status of the secondary receiver channel. The response to the query is returned to the port from which the command was issued. When querying the secondary receiver channel using the secondary serial port, no interruptions in RTCM data output will occur on the primary port provided that SBX has acquired a valid beacon.

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

PCSI,3,1 Command (Receiver Search Dump command)

Command Type:

Beacon Receiver

Description:

Hemisphere GNSS proprietary NMEA 0183 query

Query the SBX to output the search information used for beacon selection in Automatic Beacon Search mode. The output has three frequencies per line.

Command Format:

$PCSI,3,1<CR><LF>

Receiver Response:

$PCSI,ACK,3,1

$PCSI,tag1,freq1,ID1,chan1,snr1,ss1,tag2,freq2,ID2,chan2,snr2,ss2, tag3,freq3,ID3,chan3,snr3,ss3

where:

Response Component	Description
tag	Channel number with a range of 1 to 84
freq	Channel frequency (kHz * 10)
ID	Beacon ID
chan	Channel information
snr	SNR (dB)
ss	Signal Strength (dBuV/m)

Example:

```
$PCSI,ACK,3,1

$PCSI,01,2835,209,0E,00,-0009,02,2840,339,0E,00,- 0012,03,2845,006,0E,00,0009

$PCSI,04,2850,342,0E,00,-0010,05,2855,547,0E,00,-0005,06,2860,109,0E,00,- 0011

$PCSI,07,2865,188,0E,00,-0007,08,2870,272,0E,00,-0004,09,2875,682,0E,00,- 0006

$PCSI,10,2880,645,0E,00,-0007,11,2885,256,0E,00,-0009,12,2890,000,06,00,- 0012

$PCSI,13,2895,132,0E,00,-0009,14,2900,281,0E,00,-0010,15,2905,634,0E,00,- 0008

$PCSI,16,2910,172,0E,00,-0007,17,2915,006,0E,00,-0009,18,2920,546,0E,00,- 0014

$PCSI,19,2925,358,0E,00,-0008,20,2930,479,0E,00,-0009,21,2935,358,0E,00,-

                0011

                $PCSI,22,2940,853,0E,00,-0005,23,2945,588,0E,00,-0015,24,2950,210,0E,00,-     0011

                $PCSI,25,2955,000,06,00,-0011,26,2960,663,0E,00,-0010,27,2965,596,0E,00,-     0009
```

```
$PCSI,28,2970,000,06,00,-0009,29,2975,917,0E,00,-0009,30,2980,000,06,00,-      0016
$PCSI,31,2985,343,0E,00,-0013,32,2990,546,0E,00,-0010,33,2995,546,0E,00,-      0010
$PCSI,34,3000,172,0E,00,-0014,35,3005,006,0E,00,- 0011,36,3010,1006,0E,00,-0009

$PCSI,37,3015,006,0E,00,-0015,38,3020,300,0E,00,-0013,39,3025,277,0E,00,-      0100
$PCSI,40,3030,479,0E,00,-0010,41,3035,006,0E,00,-0012,42,3040,050,0E,00,-      0008
$PCSI,43,3045,000,06,00,-0014,44,3050,172,0E,00,-0013,45,3055,000,06,00,-      0011
$PCSI,46,3060,000,06,00,-0011,47,3065,000,06,00,-0014,48,3070,000,06,00,-      0010
$PCSI,49,3075,000,06,00,-0012,50,3080,006,0E,00,-0015,51,3085,000,06,00,-      0015
$PCSI,52,3090,300,0E,00,-0007,53,3095,000,06,00,-0013,54,3100,000,06,00,-      0013
$PCSI,55,3105,000,06,00,-0012,56,3110,127,0E,00,-0013,57,3115,000,06,00,-      0012
$PCSI,58,3120,596,0E,00,-0012,59,3125,051,0E,00,-0009,60,3130,000,06,00,-      0011
$PCSI,61,3135,213,0E,00,-0008,62,3140,000,06,00,-0011,63,3145,000,06,00,-      0015
$PCSI,64,3150,302,0E,00,-0008,65,3155,000,06,00,-0009,66,3160,000,06,00,-      0003
$PCSI,67,3165,000,06,00,-0013,68,3170,000,06,00,- 0011,69,3175,612,0E,01,0000

$PCSI,70,3180,000,06,00,-0015,71,3185,000,06,00,-0008,72,3190,000,06,00,-      0009
$PCSI,73,3195,000,06,00,0011,74,3200,1002,0E,01,-0002,75,3205,067,0E,00,-      0008
$PCSI,76,3210,001,0E,00,-0008,77,3215,000,06,00,-0009,78,3220,132,0E,00,-      0009
$PCSI,79,3225,000,06,00,-0010,80,3230,339,0E,00,-0013,81,3235,000,06,00,-      0011

$PCSI,82,3240,000,06,00,-0010,83,3245,202,0E,00,-0007,84,3250,006,0E,00,- 0002
```

Additional Information:

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

PCSI,3,2 Command (Ten Closest Stations command)

Command Type:

Beacon Receiver

Description:

Display the ten closest beacon stations

Command Format:

$PCSI,3,2<CR><LF>

Receiver Response:

$PCSI,ACK,3,2

$PCSI,3,2,StationID,name,freq,status,time,date,distance,health,WER

$PCSI,3,2, ...

$PCSI,3,2, ...

$PCSI,3,2, ...

$PCSI,3,2, ...

...

Where:

Response Component	Description
StationID	Specific ID number for beacon stations (appears in the last field of the GPGGA message)
name	Name of station
freq	Frequency, in kHz (scaled by 10), on which the station is transmitting. In the first line of the Example below, 2870 indicates 287.0 kHz.
status	0 (operational), 1 (undefined), 2 (no information), 3 (do not use)
time	Not implemented. Currently displayed at 0
date	Not implemented. Currently displayed at 0
distance	Calculated in nautical miles
health	-1 (not updated), 8 (undefined), 0-7 (valid range)
WER	-1 (not updated), 0-100 (valid range)

Example

```
$PCSI,ACK,3,3
$PCSI,3,2, 849,Polson                   MT,2870,0,210,0,0,-1,-1
$PCSI,3,2, 848,Spokane                  WA,3160,0,250,0,0,-1,-1

$PCSI,3,2, 907,Richmond                 BC,3200,0,356,0,0,-1,-1
$PCSI,3,2,      888,Whidbey Is.           WA,3020,0,363,0,0,-1,-1
$PCSI,3,2,      887,Robinson Pt.          WA,3230,0,383,0,0,-1,-1
$PCSI,3,2,      874,Billings              MT,3130,0,389,0,0,-1,-1
$PCSI,3,2,      871,Appleton              WA,3000,0,420,0,0,-1,-1
$PCSI,3,2,      908,Amphitrite Pt         BC,3150,0,448,0,0,-1,-1
$PCSI,3,2,      886,Fort Stevens          OR,2870,0,473,0,0,-1,-1
$PCSI,3,2,      909,Alert Bay             BC,3090,0,480,0,0,-1,-1
```

Additional Information:

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

PCSI,3,3 Command (Station Database command)

Command Type:

Beacon Receiver

Description:

Display the contents of the beacon station database

Command Format:

$PCSI,3,3<CR><LF>

Receiver Response:

$PCSI,ACK,3,3

$PCSI,3,3,IDref1,IDref2,StationID,name,freq,lat,long,datum,status

$PCSI,3,3, ...

$PCSI,3,3, ...

$PCSI,3,3, ...

$PCSI,3,3, ...

...

where:

Response Component	Description
IDref1	Beacon reference ID (primary)
IDref2	Beacon reference ID (secondary)
StationID	Specific ID number for beacon stations (appears in the last field of the GPGGA message)
name	Name of station
freq	Frequency, in kHz (scaled by 10), on which the station is transmitting. In the first line of the Example below, 2950 indicates 295.0 kHz.
lat	Scaled by 364 (+ve indicates N and -ve indicates S)
long	Longitude is scaled by 182 (+ve indicates N and -ve indicates S)
datum	1 (NAD83), 0(WGS84)
status	0 (operational), 1(undefined), 2 (no information), 3, (do not use)

Example

```
$PCSI,ACK,3,3                                   AK,2950,20554,-24221,1,0
$PCSI,3,3,0282,0283,0891,Level Island
$PCSI,3,3,0306,0307,0906,Sandspit               BC,3000,19377,-23991,1,0
$PCSI,3,3,0278,0279,0889,Annette Is.            AK,3230,20044,-23951,1,0
$PCSI,3,3,0300,0301,0909,Alert Bay              BC,3090,18412,-23099,1,0
```

$PCSI,3,3,0302,0303,0908,Amphitrite Pt BC,3150,17806,-22850,1,0

$PCSI,3,3,0270,0271,0885,C. Mendocino CA,2920,14718,-22641,1,0

$PCSI,3,3,0272,0273,0886,Fort Stevens OR,2870,16817,-22559,1,0

$PCSI,3,3,0304,0305,0907,Richmond BC,3200,17903,-22407,1,0

$PCSI,3,3,0276,0277,0888,Whidbey Is WA,3020,17587,-22331,1,0

Additional Information:

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

PCSI,4 Command (Wipe Search command)

Description:

Clear search history in Auto mode

Command Format:

$PCSI,4<CR><LF>

Receiver Response:

$PCSI,ACK,4

Example:

Additional Information:

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

PCSI,5 Command (Set Baud Rates command)

Command Type:

Beacon Receiver

Description:

Set the baud rate of Port0 and Port1

The baud rate for Port0 is saved for next power up; however, the baud rate for Port1 always defaults to 4800.

Note:

This command applies when you connect directly to a beacon board, as this command has no effect when a beacon board is integrated with a GNSS receiver.

Command Format:

$PCSI,5,portrate0,portrate1<CR><LF>

Where:

- portrate0 = desired baud rate for Port0
- portrate1 = desired baud rate for Port1

Receiver Response:

$>

Example:

Additional Information;

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

PCSI,6 Command (Reboot command)

Command Type:

Beacon Receiver

Description:

Reboot SBX receiver

Command Format:

$PCSI,6<CR><LF>

Receiver Response:

See example below.

Example:

When sending this command your response will appear similar to the below:

```
$PCSI,S/N:00019001

$PCSI,FCFGcrc,B5E5,CCFGcrc,B5E5,Pass

$PCSI,FGLBcrc,19BC,CGLBcrc,19BC,Pass

$PCSI,FLSHcrc,0531 Pass

$PCSI,FSTAcrc,56C3 Base,2FB2,B077
```

Additional Information:

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

PCSI,7 Command (Swap Modes command)

Command Type:

Beacon Receiver

Description:

Swap modes on the receiver (allowing you to output RTCM and PCSI on the desired ports—Port 0 and Port 1)

Note:

This command applies when you connect directly to a beacon board, as this command has no effect when a beacon board is integrated with a GNSS receiver.

Command Format:

$PCSI,7,mode<CR><LF>

where mode is:

- 1 = PCSI on Port1 and RTCM on Port0
- 2 = PCSI on Port0 and RTCM on Port1

Receiver Response:

$PCSI,ACK,7,mode

For example, when sending the following command...

$PCSI,7,1<CR><LF>

... the response is:

$PCSI,ACK,7,1

Example:

Additional Information:

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

Bin Messages

Binary Messages Code

This section provides the code for the binary messages used by Hemisphere GNSS.

```
// BinaryMsg.h

#ifndef __BinaryMsg_H__

#define __BinaryMsg_H__

#ifdef __cplusplus

extern "C" {

#endif

/*

 * Copyright (c) 2020 Hemisphere GNSS, 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 applicable 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.

 */

//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_TYPE108 0x0174006C /* 108 = 0x6C, 372 = 0x174 = total size in bytes -8 -2 -2 */
#define BIN_MSG_HEAD_TYPE104 0x01180068 /* 104 = 0x68, 280 = 0x118 */
#define BIN_MSG_HEAD_TYPE103 0x00400067 /* 103 = 0x67,  64 = 0x158 */
#define BIN_MSG_HEAD_TYPE102 0x01580066 /* 102 = 0x66, 344 = 0x158 */
#define BIN_MSG_HEAD_TYPE101 0x01C00065 /* 101 = 0x65, 448 = 0x1C0 */
#define BIN_MSG_HEAD_TYPE100 0x01040064 /* 100 = 0x64, 260 = 0x104 */
#define BIN_MSG_HEAD_TYPE98 0x00440062  /* 98 = 0x62, 68  = 0x44  */
#define BIN_MSG_HEAD_TYPE97 0x001C0061  /* 97 = 0x61, 28  = 0x1C  */
#define BIN_MSG_HEAD_TYPE96 0x012C0060  /* 96 = 0x60, 300 = 0x12C */  //GPS L1CA phase observables
#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_TYPE91 0x0198005B  /* 91 = 0x5B, 408 = 0x198 = total size in bytes -8 -2 -2*/
#define BIN_MSG_HEAD_TYPE89 0x00500059  /* 89 = 0x59, 80  = 0x50  */
#define BIN_MSG_HEAD_TYPE80 0x00280050  /* 80 = 0x50, 40  = 0x28  */
#define BIN_MSG_HEAD_TYPE76 0x01C0004C  /* 76 = 0x4C, 448 = 0x1C0 = total size in bytes -8 -2 -2*/
#define BIN_MSG_HEAD_TYPE71 0x01C00047  /* 71 = 0x47, 448 = 0x1C0 = total size in bytes -8 -2 -2*/
#define BIN_MSG_HEAD_TYPE61 0x0140003D  /* 61 = 0x3D, 320 = 0x140 */
#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 */

#define BIN_MSG_HEAD_TYPE69 0x012C0045  /* 69 = 0x45, 300 = 0x12C */ //Glonass

#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

#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_TYPE30 0x0080001E  /* 30 = 0x1E, 208 = 0xD0  */  //BeiDou subframe words --- similar to
GPS

#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

#define BIN_MSG_HEAD_TYPE39 0x019C0027  /* 39 = 0x27, 412 = 0x19C */  //BeiDou Channel Data for SLXMON

#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_TYPE10 0x0194000A  /* 10 = 0xA, 404 = 0x194 = total size in bytes -8 -2 -2*/

//#define BIN_MSG_HEAD_TYPE12 0x0194000C  /* 12 = 0xC, 404 = 0x194 = total size in bytes -8 -2 -2*/

#define BIN_MSG_HEAD_TYPE12 0x019A000C  /* 12 = 0xC, 410 = 0x19A = total size in bytes -8 -2 -2*/  //RFR_160506
-- added 6 bytes

#define BIN_MSG_HEAD_TYPE13 0x0194000D  /* 13 = 0xD, 404 = 0x194 = total size in bytes -8 -2 -2*/

#define BIN_MSG_HEAD_TYPE17 0x02100011  /* 17 = 0x11, 528 = 0x210 = total size in bytes -8 -2 -2*/

//#if defined(_RXAIF_PLOT_MESSAGES_)

   #define BIN_MSG_HEAD_TYPE11   0x0064000B  /* 11 = 0x0B, 100 = 0x64 = total size(112) in bytes -8 -2 -2*/

//#endif

#define BIN_MSG_HEAD_TYPE209 0x014C00D1 // 209 = 0xD1, 332 = 0x14C

#define BIN_MSG_HEAD_TYPE122 0x0050007A //122 = 0x7A, 80 = 50

#endif

#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

#define cBPM_SCAT_MEMSIZE 100

#define cBPM_SPEC_AN_MEMSIZE 128  //Must be a power of 2  (for a 4096 FFT we need 32 of these)

//RFR_140829  #define cBPM_STRIP_MEMSIZE 50

#define cBPM_STRIP_MEMSIZE 95

//#if defined(_RXAIF_PLOT_MESSAGES_)

  #define cBPM_AIFSCAT_MEMSIZE 16

//#endif


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_fSpare01;           // Spare                              [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 */
#define CHANNELS_SBAS_E    3
/****************************************************/
/*  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
/****************************************************/
/*  SBinaryMsg59   for USING_GPSL2CL               */
/****************************************************/
typedef struct
{
   SUnionMsgHeader  m_sHead;
   unsigned char  m_byNavMode;         /* Nav Mode FIX_NO, FIX_2D, FIX_3D (high bit =has_diff) */ //1 byte
   char           m_cUTCTimeDiff;      /* whole Seconds between UTC and GPS   */              //1 byte
   unsigned short m_wGPSWeek;          /* GPS week */                                         //2 bytes
   unsigned long  m_ulMaskSatsUsedL2P; /* L2P SATS Used, bit mapped 0..31 */                  //4 bytes
   double         m_dGPSTimeOfWeek;    /* GPS tow  */                                         //8 bytes
   SChannelL2CData m_asChannelData[CHANNELS_12]; /* channel data */                           //20*12 bytes
   unsigned short m_wCheckSum;         /* sum of all bytes of the header and data */
   unsigned short m_wCRLF;             /* Carriage Return Line Feed */
} SBinaryMsg59;                         /* length = 8 + 260 + 2 + 2 = 272 */
//-****************************************************
//-*  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, 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
/****************************************************/
/*  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 */
/****************************************************/
/*  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).

 */


#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) */

/****************************************************/

/*  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 */

/****************************************************/

/*  SMsg71DataL1                                    */
```

```
/****************************************************/

typedef struct

{

    unsigned char  bySV;                    /* satellite being tracked, 0 == not tracked  */

    unsigned char  byStatus;                 /* Status bits (code carrier bit frame...)  */

    unsigned char  byStatusL1P;      /* 0-8 lower 8 bits of L1P SNR/32768, if zero and

   if upper two bits of m_wSNR_codeph_L1P are zero

  then L1P is not tracking */

    unsigned char  byStatusL2P;      /* Status bits (code carrier phase ...)  */

    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 */

} SMsg71DataL1; /* 20 bytes */


/****************************************************/

/*  SMsg71DataL1PL2P                              */

/****************************************************/

typedef struct

{

    /* L1P and L2P Data */

 //   unsigned long  codeph_SNR_L1P; NOT USED YET /* 0-22 = L1 code phase (23 bits), 28-32 = SNR/8192, upper 4
bits */

    unsigned long  codeph_SNR_L2P;              /* 0-22 = L2P code phase (23 bits), 28-32 = SNR/8192, upper 4 bits */

    unsigned long  ulCarrierCycles_SNR_L2P;      /* 0-23 = carrier cycles, 24-32 = SNR/8192 lower 8 bits */

    unsigned short wDCOPhaseB10_L2P;            /* 0-11 = DCO phase, 12-15 = Spare */

    unsigned short m_wSNR_codeph_L1P;           /* 0-13 = lower 14 bits of L1P code,  14-15 SNR/32768 Upper 2 bits */

                                       /* To get full L1P code, use upper bits form L2P and adjust by

                                         +/- 2**14 if necessary */

} SMsg71DataL1PL2P; /* 12 bytes */

/****************************************************/
```

```
  /*  SBinaryMsg71                              */
  /*  Comment: Transmits data from Takemeas.c         */
  /*          debugging structure for Dual Freq.     */
  /***************************************************/
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 */
    SMsg71DataL1PL2P s91L2PData[CHANNELS_12];    /* 12*12 = 144 bytes */
    SMsg71DataL1     s91Data[CHANNELS_12_PLUS];  /* 14*20 = 280 bytes */
    unsigned short   m_wCheckSum;          /* sum of all bytes of the header and data */
    unsigned short   m_wCRLF;              /* Carriage Return Line Feed */
} SBinaryMsg71;                            /* length = 8 + (448) + 2 + 2 = 460 */
////////////////////////////////////////////////////
//  SBinaryMsg10
//  Comment: Transmits scatter plot data from
//          buffacc.c
//
////////////////////////////////////////////////////
//-- maxes out 16 types, you have to expand some fields if you add any more types
enum eBIN10_TYPE {eBIN10_GPSL1CA       = 0,
                  eBIN10_GPSL1P        = 1,
                  eBIN10_GPSL2P        = 2,
                  eBIN10_GLONASSL1     = 3,
                  eBIN10_GLONASSL2     = 4,
                  eBIN10_GPSL2CL       = 5,
                  eBIN10_GPSL5Q        = 6,
                  eBIN10_GALILEO_E1BC  = 7,
                  eBIN10_GALILEO_E5A   = 8,
                  eBIN10_GALILEO_E5B   = 9,
                  eBIN10_BEIDOU_B1     = 10,
                  eBIN10_BEIDOU_B2     = 11,
                  eBIN10_BEIDOU_B3     = 12,
                  eBIN10_GPSL1C       = 13,
                  eBIN10_QZSS_L1CA     = 14,
                  eBIN10_QZSS_L1C      = 15,
                  eBIN10_QZSS_L2C      = 16,
                  eBIN10_QZSS_L5       = 17,
                  eBIN10_BEIDOU_B1BOC  = 18,  // USING_BEIDOU_PHASE3
                  eBIN10_BEIDOU_B2A    = 19,  // USING_BEIDOU_PHASE3
                  eBIN10_BEIDOU_B2B    = 20,  // USING_BEIDOU_PHASE3
                  eBIN10_BEIDOU_B3C    = 21,  // USING_BEIDOU_PHASE3
                  eBIN10_BEIDOU_ACEBOC = 22,
                  eBIN10_GALILEO_E6    = 23,
                  eBIN10_GALILEO_ALTBOC= 24,
                  eBIN10_GLONASS_G1OC  = 26,
                  eBIN10_GLONASS_G2OC  = 27,
                  eBIN10_GLONASS_G3OC  = 28,
                  eBIN10_QZSS_LEX      = 29
                  };
typedef struct
{
    SUnionMsgHeader  m_sHead;              // 8 bytes
    short m_anScatterPlotDataI[cBPM_SCAT_MEMSIZE]; //100*2 = 200 bytes
    short m_anScatterPlotDataQ[cBPM_SCAT_MEMSIZE]; //100*2 = 200 bytes
    unsigned short m_wChannel;
    unsigned short m_wSigType;           // one of eBIN10_TYPE
    unsigned short   m_wCheckSum;          // sum of all bytes of the header and data
    unsigned short   m_wCRLF;              // Carriage Return Line Feed
} SBinaryMsg10;                        // length = 8 +200 +200 +2 +2 +2 +2 = 416

//////////////////////////////////////////////////
//  SBinaryMsg12
//  Comment: Transmits power detector data from
//          buffacc.c

//

////////////////////////////////////////////////////

//-- maxed out 16 types, you have to expand some fields if you add any more types

enum eBIN12_TYPE {eBIN12_GPSL1CA       = 0,

                  eBIN12_GPSL1P        = 1,

                  eBIN12_GPSL2P        = 2,

                  eBIN12_GLONASSL1     = 3,

                  eBIN12_GLONASSL2     = 4,

                  eBIN12_GPSL2CL       = 5,

                  eBIN12_GPSL5Q        = 6,

                  eBIN12_GALILEO_E1BC   = 7,

                  eBIN12_GALILEO_E5A    = 8,

                  eBIN12_GALILEO_E5B    = 9,

                  eBIN12_BEIDOU_B1      = 10,

                  eBIN12_BEIDOU_B2      = 11,

                  eBIN12_BEIDOU_B3      = 12,

                  eBIN12_GPSL1C        = 13,

                  eBIN12_QZSS_L1CA      = 14,

                  eBIN12_QZSS_L1C       = 15,

                  eBIN12_QZSS_L2C       = 16,

                  eBIN12_QZSS_L5        = 17,

                  eBIN12_BEIDOU_B1BOC   = 18,  // USING_BEIDOU_PHASE3

                  eBIN12_BEIDOU_B2A     = 19,  // USING_BEIDOU_PHASE3

                  eBIN12_BEIDOU_B2B     = 20,  // USING_BEIDOU_PHASE3

                  eBIN12_BEIDOU_B3C     = 21,  // USING_BEIDOU_PHASE3

                  eBIN12_BEIDOU_ACEBOC  = 22,

                  eBIN12_GALILEO_E6     = 23,

                  eBIN12_GALILEO_ALTBOC = 24,

                  eBIN12_GLONASS_G1OC   = 26,

                  eBIN12_GLONASS_G2OC   = 27,

                  eBIN12_GLONASS_G3OC   = 28,

                  eBIN12_QZSS_LEX       = 29

                 };
```

```
typedef struct

{

   SUnionMsgHeader  m_sHead;                   // 8 bytes

   int             m_aiPowerDetectData[cBPM_STRIP_MEMSIZE]; //100*4 = 400 bytes //HACK for
dumpIRQBufferToPort  (CrLI can go negative)

   int             m_iThresh_Acq;             //RFR_140829

   int             m_iThresh_Loss;            //RFR_140829

   float           m_fCarrierNCO;             //RFR_140902

   float           m_fCodeNCO;                //RFR_140902

   unsigned short   m_wCarrierBin;             //RFR_140902   [7:0]CarrierBinNum RFR_160506 (break status out to its own
field)

   unsigned short   m_wStatus;                 //RFR_160506   [15:0]CXBF  RFR_160506 added 2-bytes <****

   long            m_lDelayCodeLossIfFramSyn;  //RFR_160506   added 4-bytes

   unsigned short   m_wCodeMovement;           //RFR_140902

   unsigned short   m_wChannel;

   unsigned short   m_wSigType;                // one of eBIN12_TYPE

   unsigned short   m_wCheckSum;               // sum of all bytes of the header and data

   unsigned short   m_wCRLF;                   // Carriage Return Line Feed

} SBinaryMsg12;                            // length = 8 +200 +4 +4 +2 +2 +2 +2= 224


#if defined(ENABLE_PLOT_PERFMETER)

//////////////////////////////////////////////////////

//  SBinaryMsg13

//  Comment: Transmits Performance Numbers

//////////////////////////////////////////////////////

enum eBIN13_TYPE {eBIN13_IDLE          = 0,

              eBIN13_GPISR         = 1,

              eBIN13_BUFFACCUM      = 2,

              eBIN13_TASK_00       = 3,

              eBIN13_TASK_01       = 4,

              eBIN13_SPARE         = 5

             };

typedef struct

{

   SUnionMsgHeader  m_sHead;               // 8 bytes
   int          m_aiPerformanceMetric[cBPM_STRIP_MEMSIZE]; //100*4 = 400 bytes

   int          m_iMetricMaxScale;    //maximum possible value for m_aiPowerDetectData (allows plotting percentage)

   unsigned int   m_uNumDataPointsInPkt;     //num points in m_aiPowerDetectData (might not be the max
cBPM_STRIP_MEMSIZE because could be too slow for some metrics)

   int          m_iMetricFiltered;

   int          m_iMetricPeak;

   unsigned short m_wSpare3;

   unsigned short m_wSpare4;

   unsigned short m_wSpare5;


   unsigned short m_wStatType;          // one of eBIN13_TYPE

   unsigned short m_wCheckSum;          // sum of all bytes of the header and data

   unsigned short m_wCRLF;              // Carriage Return Line Feed

} SBinaryMsg13;                        // length = 8 +200 +4 +4 +2 +2 +2 +2= 224

#endif  //ENABLE_PLOT_PERFMETER


//#if defined(_RXAIF_PLOT_MESSAGES_)

////////////////////////////////////////////////////

//  SBinaryMsg11

//  Comment: Transmits scatter plot data for RXGNSS_AIF statistics

//

////////////////////////////////////////////////////

enum eBIN11_TYPE {eBIN11_COUNTS=0,eBIN11_VALUES};

typedef struct

{

   SUnionMsgHeader  m_sHead;              // 8 bytes

   unsigned short   m_awScatterPlotDataValues[cBPM_AIFSCAT_MEMSIZE];   //16*2 = 32 bytes

   unsigned short   m_awScatterPlotDataCntMag[cBPM_AIFSCAT_MEMSIZE];   //16*2 = 32 bytes

   unsigned short   m_awScatterPlotDataCntDCoff[cBPM_AIFSCAT_MEMSIZE]; //16*2 = 32 bytes

   unsigned short   m_wChannel;           // aif_sel 0: AIF_A, 1: AIF_B, ...

   unsigned short   m_wSigType;           // one of eBIN11_TYPE

   unsigned short   m_wCheckSum;          // sum of all bytes of the header and data

   unsigned short   m_wCRLF;              // Carriage Return Line Feed

} SBinaryMsg11;                          // length = 8 +32 +32 +32 +2 +2 +2 +2 = 112
```

```
////////////////////////////////////////////////////
//  SBinaryMsg17
//  Comment: Transmits spectrum analyzer plot data
//          We may change this notation a bit when
//          we get the channelizer working
//
////////////////////////////////////////////////////
//-- maxes out 16 types, you have to expand some fields if you add any more types
enum eBIN17_TYPE {eBIN17_GPSL1CA       = 0,
                  eBIN17_GPSL1P        = 1,
                  eBIN17_GPSL2P        = 2,
                  eBIN17_GLONASSL1     = 3,
                  eBIN17_GLONASSL2     = 4,
                  eBIN17_GPSL2CL       = 5,
                  eBIN17_GPSL5Q        = 6,
                  eBIN17_GALILEO_E1BC  = 7,
                  eBIN17_GALILEO_E5A   = 8,
                  eBIN17_GALILEO_E5B   = 9,
                  eBIN17_BEIDOU_B1     = 10,
                  eBIN17_BEIDOU_B2     = 11,
                  eBIN17_BEIDOU_B3     = 12,
                  eBIN17_GPSL1C        = 13,
                  eBIN17_QZSS_L1CA     = 14,
                  eBIN17_QZSS_L1C      = 15,
                  eBIN17_QZSS_L2C      = 16,
                  eBIN17_QZSS_L5       = 17};

typedef struct
{
  SUnionMsgHeader  m_sHead;              // 8 bytes
  float m_aiSpecAnPlotData[cBPM_SPEC_AN_MEMSIZE]; //128*4 = 512 bytes
  float m_wSampleFreq;
```

```
    unsigned short m_wStage;                //Index so you know what frequencies are in the message
    unsigned short m_wChannel;
    unsigned short m_wSigType;              // one of eBIN17_TYPE
    unsigned short m_wCheckSum;             // sum of all bytes of the header and data
    unsigned short m_wSpare1;
    unsigned short m_wSpare2;
    unsigned short m_wSpare3;
    unsigned short m_wCRLF;                 // Carriage Return Line Feed
} SBinaryMsg17;                          // length = 8 + 512 + 4 +2 +2 +2 +2 +2 +2 +2 +2 = 540


//#endif



/****************************************************/
/*  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 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
                                  // 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
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 31:6 = 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
                                  // 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

  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 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
                             // 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
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 31:6 = 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 = 0:L1CA, 1:L2P,  2:L2C, 3:L5
                                        // 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

                                        //[3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS

   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

/****************************************************/
/*  SBinaryMsg30                                */
/****************************************************/
typedef SBinaryMsg95 SBinaryMsg30;  //BeiDou iono and correctors

/****************************************************/
/*  SBinaryMsg45                                */
/****************************************************/
typedef SBinaryMsg95 SBinaryMsg45;  //Galileo 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)
                                           0 & 1 = Two almanac SFs, 3= SF 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 struct
{
    unsigned char m_ucPRN;        /* PRN number */

    // For m_ucSatUpperX, m_ucSatUpperY, or m_ucSatUpperZ, the following bit definitions hold:
    // [B1,B0]  =  [B33,B32] of 34 bit absolute value of m_ulSatX, m_ulSatX, or m_ulSatX.
    //            LSB = [B32] = 33554432 meter
    // [B2]     =  sign bit for m_ulSatX, m_ulSatY, or m_ulSatZ,  If set, pos is negative.
    // [B6..B3] =  [B19,B18,B17,B16] of 20 bit absolute value of m_wSatVx, m_wSatVy, or m_wSatVz.
    //            LSB = [B16] = 512 meter/sec
    // [B7]     =  sign bit for m_wSatVx, m_wSatVy, or m_wSatVz.  If set, vel is negative.

    unsigned char m_ucSatUpperX;  /* [B7..B0], see above */
    unsigned char m_ucSatUpperY;  /* [B7..B0], see above */
    unsigned char m_ucSatUpperZ;  /* [B7..B0], see above */

    unsigned short m_wSpare;

    unsigned short m_wSatVx;      /* [B15..B0] lower 16 Bits of 20 bit absolute value, LSB =  1/128 meter/sec */
    unsigned short m_wSatVy;      /* [B15..B0] lower 16 Bits of 20 bit absolute value, LSB =  1/128 meter/sec */
    unsigned short m_wSatVz;      /* [B15..B0] lower 16 Bits of 20 bit absolute value, LSB =  1/128 meter/sec */

    unsigned long m_ulSatX;       /* [B31..B0] lower 32 Bits of 34 bit absolute value, LSB =  1/128 meter */
    unsigned long m_ulSatY;       /* [B31..B0] lower 32 Bits of 34 bit absolute value, LSB =  1/128 meter */
    unsigned long m_ulSatZ;       /* [B31..B0] lower 32 Bits of 34 bit absolute value, LSB =  1/128 meter */
} SSatXYZ;  // 24 bytes



typedef struct
{
    SUnionMsgHeader    m_sHead;
    double             m_dTow;      /* GPS Time in seconds of the receiver clock when the m_asSatXYZ is calculated */
    SSatXYZ            m_asSatXYZ[CHANNELS_L1_E]; /* 20*15 = 360 bytes */
    unsigned short     m_wWeek;     /* GPS Week Number, */
    unsigned char      m_ucSysID;   /* GPS=0, GLONASS=1, GALILEO=2, BEIDOU=3 */
    unsigned char      m_ucSpare01; /* padding to 4-byte align */
    unsigned short     m_wCheckSum; /* sum of all bytes of the header and data */
    unsigned short     m_wCRLF;     /* Carriage Return Line Feed */
} SBinaryMsg108;   /* length = 8 + (8+360+4) + 2 + 2   =   8 + (372) + 2 + 2 = 384 */
/****************************************************/
/*  SBinaryMsg122                                  */
/****************************************************/
typedef struct
{
    SUnionMsgHeader  m_sHead;
    double           m_dGPSTimeOfWeek;     // GPS tow                              [8 bytes]
    unsigned short   m_wGPSWeek;           // GPS week                             [2 bytes]
    unsigned char    m_byPhxPosType;       // Phoenix position type                [1 bytes]
    unsigned char    m_byCorSource;        // Phoenix correction source            [1 byte ]
    unsigned short   m_wBaseStationId;     // Base station ID                      [2 bytes]
    unsigned char    m_bySatUsedCount[6];  // Satellites used per system           [6 bytes]
                                           //  [GPS GLN GAL BDS SBAS QZSS]
    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_fAgeOfDiff;         // age of differential, seconds         [4 bytes]
    float            m_fVNorth;            // North-South Velocity +North m/s      [4 bytes]
    float           m_fVEast;           // East-West Velocity +East m/s             [4 bytes]
    float           m_fVUp;             // Vertical Velocity +up  m/s              [4 bytes]
    float           m_fCovNN;           // Covariance North-North                   [4 bytes]
    float           m_fCovNE;           // Covariance North-East                    [4 bytes]
    float           m_fCovNU;           // Covariance North-Up                      [4 bytes]
    float           m_fCovEE;           // Covariance East-East                     [4 bytes]
    float           m_fCovEU;           // Covariance East-Up                       [4 bytes]
    float           m_fCovUU;           // Covariance 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 */
} SBinaryMsg122;                       /* length = 8 + 80 + 2 + 2 = 92 */
//xx #if defined(WIN32) || (__ARMCC_VERSION>=300441)  // all compilers that we use today
    #pragma pack(pop)
//xx #endif
#ifdef __cplusplus
}
#endif
#endif // __BinaryMsg_H_
```

Last updated: v1.11/ November 15, 2018

Bin1 Message

Message Type:

Binary

Description:

GNSS position message (position and velocity data)

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	Values
AgeOfDiff	Age of differential, seconds. Use Extended AgeOfDiff first. If both = 0, then no differential	Byte	1	0 to 255
NumOfSats	Number of satellites used in the GPS solution	Byte	1	0 to 12
GPSWeek	GPS week associated with this message	Unsigned short	2	0 to 65535
GPSTimeOfWeek	GPS tow (sec) associated with this message	Double	8	0.0 to 604800.0
Latitude	Latitude in degrees north	Double	8	-90.0 to 90.0
Longitude	Longitude in degrees East	Double	8	-180.0 to 180.0
Height	Altitude above the ellipsoid in meters	Float	4	
VNorth	Velocity north in m/s	Float	4	
VEast	Velocity east in m/s	Float	4	
Vup	Velocity up in m/s	Float	4	
StdDevResid	Standard deviation of residuals in meters	Float	4	Positive

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 If bit 7 is set (left-most bit), then this is a manual position	Unsigned short	2	Bits 0 through 6 = Navmode Bit 7 = Manual mark
Extended AgeOfDiff	Extended age of differential, seconds. If 0, use 1 byte AgeOfDiff listed above	Unsigned short	2	0 to 65536

Structure:

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 eastm/s */
float               m_fVUp;             /*  Velocity up  m/s */
  float Residuals */  m_fStdDevResid;   /*   (m), Standard Deviation of

  unsigned short      m_wNavMode;
  unsigned short      m_wAgeOfDiff;     /*   age of diff using 16
                                             bits                 */
  unsigned short      m_wCheckSum;      /*   sum of all bytes of the
  datalength */

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:

JBIN

Topic Last Updated: v1.08 / June 9, 2017

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.

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	Values
MaskSatsTracked	Mask of satellites tracked by the GPS. Bit 0 corresponds to the GPS satellite with PRN 1.	Unsigned long	4	Individual bits represent satellites
MaskSatsUsed	Mask of satellites used in the GPS solution. Bit 0 corresponds to the GPS satellite with PRN 1.	Unsigned long	4	Individual bits represent satellites
GpsUtcDiff	Whole seconds between UTC and GPS time (GPS minus UTC)	Unsigned short	2	Positive
HDOPTimes10	Horizontal dilution of precision scaled by10 (0.1 units)	Unsigned short	2	Positive
VDOPTimes10	Vertical dilution of precision scaled by 10 (0.1 units)	Unsigned short	2	Positive
WAASMask	PRN and tracked or used status masks	Unsigned short	2	*See following*

- 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

Structure:

```
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 UTC) */

m_wGpsUtcDiff;       /* GPS/UTC time difference (GPS minus


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 datalength

*/

unsigned short     m_wCRLF;            /* Carriage Return Line Feed */
```

Additional Information:

Message has a BlockID of 2 and is 16 bytes, excluding the header and epilogue

Related Commands:

JBIN

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

Bin3 Message

Message Type:

Binary

Description:

Lat/Lon/Hgt, Covariances, RMS, DOPs and COG, Speed,Heading

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	Values
GPSTimeOfWeek	GPS tow (sec) associated with this message	Double	8	0.0 to 604800.0
GPSWeek	GPS week associated with this message	Unsigned short	2	0 to 65535
SATS Tracked	Number of satellites tracked in the GPS solution	Unsigned short	2	
NumOfSats	Number of satellites used in the GPS solution	Byte	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	unsigned char	1	Bits 0 through 6 = Navmode Bit 7 = Manual mark

	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			
Spare		unsigned char	1	
Latitude	Latitude in degrees north	Double	8	-90.0 to 90.0
Longitude	Longitude in degrees East	Double	8	-180.0 to 180.0
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 vecto	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	Unsigned short	4	

	Bits 4 – 7 = status.ePitch Bits 8 – 11 = status.eRoll Where status can be 0=INVALID, 1=GNSS, 2=Inertial, 3=Magnetic			
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:

```
typedef struct

{

SUnionMsgHeader    m_sHead;                //

Double           m_dGPSTimeOfWeek;          //GPS tow unsigned short             m_wGPSWeek;             // GPS week
unsigned short             m_wNumSatsTracked;       // SATS Trackedunsigned
short             m_wNumSatsUsed;                  // SATS Used

unsigned char      m_byNavMode;       // Nav Mode (same as message 1) unsigned char      m_bySpare00;                 //
Spare

double             m_dLatitude;       // Latitude degrees, -90..90 doublem_dLongitude;      // Longitude degrees, -180..180

float             m_fHeight;        // (m), Altitude ellipsoid float m_fSpeed;         //Horizontal Speed             m/s
```

float m_fVUp; // Vertical Velocity +up m/s float m_fCOG; // Course over Ground, degrees

float m_fHeading; // Heading (degrees), Zero unlessvector float m_fPitch; // Pitch (degrees), Zero unless vector float

m_fSpare01; // Roll

unsigned short m_wAgeOfDiff; // Age of differential (s).

unsigned short m_wAttitudeStatus; // Attitude Status, zero unless vector

// bit {0-3} = status.eYaw

// bit {4-7} = status.ePitch

// bit {8-11} = status.eRoll

// where status can be 0=INVALID,

// 1=GNSS, 2=Inertial, 3=Magnetic

float m_fStdDevHeading_d; // Yaw stddev (degrees), zero unless vector

float m_fStdDevPitch_d; // Pitch stddev (degrees), zero unless vector

float m_fHRMS; // Horizontal RMS

float m_fVRMS; //Vertical RMS

float m_fHDOP; // Horizontal DOP

float m_fVDOP; // Vertical DOP

float m_fTDOP; // Time DOP

float m_fCovNN; // Covaraince North-North

float m_fCovNE; // Covaraince North-East

float m_fCovNU; // Covaraince North-Up

float m_fCovEE; // Covaraince East-East

float m_fCovEU; // Covaraince East-Up

float m_fCovUU; // Covaraince Up-Up

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:

Related Commands:

JBIN

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

Bin5 Message

Message Type:

Binary

Description:

Base station information

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	Values
Latitude	Latitude of base station in degrees north	Double	8	-90.0 to 90.0
Longitude	Longitude of base station in degrees east	Double	8	-180.0 to 180.0
Height	Base station altitude in meters	Float	4	
BaseID	Base station ID	Unsigned short	2	0 to 65535
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:

```
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:

Related Commands:

JBIN

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

Bin 6 Message

Message Type:

Binary

Description:

Manual Mark Tag

Message Format:

The message is output when the manual mark is triggered.

Message Component	Description	Type	Bytes	Values
Time of Week	GPS tow (sec) associated with this message	Double	8	0.0 to 604800.0
GPS Week	GPS week associated with this message	Unsigned short	2	0 to 65535
Spare	Spare	Unsigned short	2	

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:

Related Commands:

Topic Last Updated: June 1, 2018

Bin16 Message

Message Type:

Binary

Description:

Generic GNSS observations (see notes on message 76)

Command Format to Request Message:

$JBIN,16,r<CR><LF>

Where:

- '16' = Bin16 message
- 'r' = message rate in Hz (1 or 0)

Message Format:

Message Component	Description	Type	Bytes	Values
Tow	Time in seconds	double	8	
Week	GPS week number	Unsigned short	2	Individual bits represent satellites
Spare1	Not used at this time	Unsigned short	2	Future Use
PageCount	Page information	Unsigned long	4	*See following*

· [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

AllSignalsIncluded_01	Bit mask of all signals included in the set of pages	Unsigned long	4	*See following*

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 31:27 = spare

Message	Description	Type	Bytes	Values
AllSignalsIncluded_02	Bit mask of all signals included in the set of pages	Unsigned long	4	*See following*

- bit 0 = QZS:L1CA included
- bit 1 = spare
- bit 2 = QZS:L2C included
- bit 3 = QZS:L5 included
- bit 4 = QZS:L1C included
- bit 31:5 = spare

Obs[16]	16 sets of observations	Structure array	16*12 = 192	
CodeMSBsPRN	Array of 16 32-bit words	Array of unsigned longs	16*4=64	

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

ChanSignalSYS	Array of 16 16-bit words	Array of unsigned shorts	16*2=32	

· [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

GAL Signal ID: E1BC=0, E5A=1, E5B=2

BDS Signal ID: B1I=0, B2I=1, B3I=2

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

CheckSum	Sum of all bytes of header and data	Unsigned short	2	
CRLF	Carriage return line feed	Unsigned short	2	

Structure:

```
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 7:4 = spare

                                        // bit 8  = GLO:G1C or GLO:G1P

included

 included
```

// bit 9 = GLO:G2C or GLO:G1P

// bit 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 31:27 = spare

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 31:5 = 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)

PRN,

satellite

Log_Base_2(X+1) units of 1/100th sec,

// bit 7:0 (8 bits) = satellite

//   = 0 if no

// bit 12:8 (5 bits) =

//   where X = Time, in

tracking was last stressed

bits meters

67108864 meters

//   since carrier phase

//   or cycle slipped

// bit 31:13 (19 bits) = upper 19

// of code pseudorange LSB = 256

//   MSB =

unsigned short m_awChanSignalSYS[CHANNELS_gen]; // Array of 16, 16 bit words (32 bytes)

first channel)

G1, B1I, B2I, B3I, etc) L2C=2, L5=3

G2C/G2P=1 E5B=2

L5=3, L1C=4 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS

//[15,14] spare bits

//[13] = 1 if GLONASS P-Code

//[12,11,10,9,8] = Channel (0 is the

//[7,6,5,4] = Signal ID (L1CA, L5,

// GPS Signal ID: L1CA=0, L2P=1,

// GLO Signal ID: G1C/G1P=0,

// GAL Signal ID: E1BC=0, E5A=1,

// BDS Signal ID: B1I=0, B2I=1, B3I=2

// QZS Signal ID: L1CA=0, L2C=2,

//[3,2,1,0] = GNSS System,

unsigned short and data (2 bytes)	m_wCheckSum;	/// sum of all bytes of the header
unsigned short (2 bytes)	m_wCRLF;	// Carriage Return Line Feed

} SBinaryMsg16;   // length = 8 +

(8+2+2+4+4+4+192+64+32=312) + 2 + 2 = 324

Additional Information:

Related Commands:

JBIN

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

Bin19 Message

Message Type:

Binary

Description:

GNSS diagnostic information

Command Format to Request Message:

$JBIN,19,r<CR><LF>

Where:

- '19' = Bin19 message
- 'r' = message rate in Hz (1 or 0)

Message Format:

Message Component	Description	Type	Bytes	Values
SecOfWeek	Time of Week	long	4	
GPSWeek	GPS Week Number	unsigned short	2	
NavMode	Nav Mode	unsigned char	1	0-255
UTCTimeDiff	Whole seconds between UTC and GPS time	char	1	
PageCount	Information about the paging of the BIN19 message.	unsigned long	4	Bits [16,17,18,19,20,21] Number of Pages = N Bits [22,23,24,25,26,27] Page Number [0...N-1]
AllSignalsIncludes01	Bitmask of all signals includes in this 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 8 = GLO:G1C or GLO:G1P included bit 9 = GLO:G2C or GLO:G1P included bit 16 = GAL:E1BC included bit 17 = GAL:E5A included bit 18 = GAL:E5B included bit 24 = BDS:B1I included bit 25 = BDS:B2I included bit 26 = BDS:B3I included
AllSignalsIncluded02	Continued bitmask of all signals included in this set of pages.	unsigned long	4	bit 0 = QZS:L1CA included bit 2 = QZS:L2C included bit 3 = QZS:L5 included bit 4 = QZS:L1C included

Spare		unsigned short	2	
ChannelData[16]	Detailed data for each signal included.	SGENERICchanData[]	320	
ChanSignalSYS	Information about the type of signal represented by each entry in ChannelData	unsigned short[]	32	[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 = 0: L1CA, 1: L2P, 2: L2C, 3: L5 GLO Signal ID = 0: G1C/G1P, 1: G2C/G2P GAL Signal ID = 0: E1BC, 1: E5A, 2:E5B BDS Signal ID = 0: B1I, 1: B2I, 2:B3I QZS Signal ID = 0: L1CA, 1: xxx, 2:L2C, 3: L5, 4: L1C [3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS
CheckSum	Sum of all bytes of header and data	Unsigned short	2	
CRLF	Carriage return line feed	Unsigned short	2	

Structure:

//==

// SGENERICchanData

//==

typedef struct

{

unsigned char m_bySV; // Bit (0-6) = SV slot, 0 == not trackedunsigned 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 angleunsigned char m_byLastMessage; // last message processedunsigned char m_bySlip; // cycle slip on chan 1 char m_cFlags; //

// [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 unsignedshort m_wAllocType;             //


} SGENERICchanData; // (20 bytes)


//------------------------------------------------------------------------------

// SBinaryMsg19

//  Generic GNSS message for signal tracking status

//------------------------------------------------------------------------------

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 7:4 = spare

// bit 8  = GLO:G1C or GLO:G1P included

// bit 9  = GLO:G2C or GLO:G1P included
// bit 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 31:27 = spare  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 31:5 = 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 = 0: L1CA,                    1: L2P,             2: L2C, 3: L5

                                                 //     GLO Signal ID = 0: G1C/G1P,             1: G2C/G2P

                                                 //     GAL Signal ID = 0: E1BC,                    1: E5A,             2:E5B

                                                 //     BDS Signal ID = 0: B1I,                     1: B2I,             2:B3I

                                                 //     QZS Signal ID = 0: L1CA,                    1: xxx,             2:L2C,  3: L5,

4: L1C


                                                 //[3,2,1,0] = GNSS System, 0=GPS,1=GLO,2=GAL,3=BDS,4=QZS


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 =
```

Additional Information:

Related Commands:

Topic Last Updated: v1.08 / June 9, 2017

Bin22 Message

Message Type:

Binary

Description:

QZSS Almanac

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** alAlmwords[0] — t_{oa} alAlmwords[1] — $\sqrt{A}$ alAlmwords[2] — e alAlmwords[3] — ω alAlmwords[4] — M_0 alAlmwords[5] — Ω_0 alAlmwords[6] — $\dot{\Omega}$ alAlmwords[7] — δ_i alAlmwords[8] — SV Health alAlmwords[9] — WN_a	**Long[12]**	**12*4 = 48**

alAlmwords[0]	t_{oa}
alAlmwords[1]	$\sqrt{A}$
alAlmwords[2]	e
alAlmwords[3]	ω
alAlmwords[4]	M_0
alAlmwords[5]	Ω_0
alAlmwords[6]	$\dot{\Omega}$
alAlmwords[7]	δ_i
alAlmwords[8]	SV Health
alAlmwords[9]	WN_a

	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:

```
//----------------------------------------------------------------------------
// SBinaryMsg22
//  QZSS 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
unsigned short       m_wCheckSum;      // sum of all bytes of the header and data
unsigned short       m_wCRLF;          // Carriage Return Line Feed
} SBinaryMsg22; // length = 8 + (4+48=52) + 2 + 2 = 64
```

Additional Information:

Related Commands:

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

Bin32 Message

Message Type:

Binary

Description:

BeiDou Almanac

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** alAlmwords[0] | t_{oa} alAlmwords[1] | $\sqrt{A}$ alAlmwords[2] | e alAlmwords[3] | ω alAlmwords[4] | M_0 alAlmwords[5] | Ω_0 alAlmwords[6] | $\dot{\Omega}$ alAlmwords[7] | δ_i alAlmwords[8] | [a_0 |a_1 |Health] Bits [30:20|19:9|8:0] alAlmwords[9] | WN_a	**Long[12]**	**12*4 = 48**

	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:

```
//-----------------------------------------------------------------------------

// SBinaryMsg32

//  BeiDou 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

unsigned short           m_wCheckSum;          // sum of all bytes of the header and data

 unsigned short           m_wCRLF;                 // Carriage Return Line Feed

} SBinaryMsg32; // length = 8 + (4+48=52) + 2 + 2 = 64
```

Additional Information:

Related Commands:

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

Bin 34 Message

Message Type:

Binary

Description:

BeiDou -> GPS, ->GLO, -> GAL, ->UTC time offset parameters

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	Values
A0UTC, A1UTC	BDT clock bias relative to UTC	Int	4 x 2=8	
A0GPS, A1GPS	BDT click bias relative to GPS time	Unsigned short	2 x 2=4	
A0GAL, A1GAL	BDT clock bias relative to Galileo system time	Unsigned short	2 x 2=4	
A0GLO, A1GLO	BDT clock bias relative to GLONASS time	Unsigned 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		Short	2	Future use
Spare2		Short	2	Future use
Spare3		Short	2	Future use

Structure:

```
 //-----------------------------------------------------------------------------
// SBinaryMsg34  --- BeiDou -> GPS, ->GLO, -> GAL, ->UTC time offset parameters
// Information is in  D1
//-----------------------------------------------------------------------------
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
```

Additional Information:

Related Commands:

Topic Last Updated: v1.10 June 1, 2018

Bin35 Message

Message Type:

Binary

Description:

BeiDou ephemeris information

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	Values
SV	Satellite to which this data belongs	Unsigned short	2	
Spare1	Not used at this time	Unsigned short	2	Future use
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:

```
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_BeidouNav[30]; /* Unparsed BeiDou navigation words.
*/

Read needed. */
 /* Each of the BeiDou nav words contains one 32- bit signed or unsigned word.
as a signed or unsigned long as
unsigned short     m_wCheckSum;     /* sum of all bytes of the header and
data */
    unsigned short                  m_wCRLF;     /*    Carriage Return Line Feed */
    } SBinaryMsg35;                              /*    length = 8 + (128) + 2 + 2 = 140 */
```

Additional Information:

Message has a BlockID of 35 and is 128 bytes, excluding the header and epilogue

Related Commands:

JBIN

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

Bin36 Message

Message Type:

Binary

Description:

BeiDou code and carrier phase information (all frequencies)

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:

Message Component	Description	Type	Bytes	Values
Tow	Time in seconds	Double	2	
Week	GPS week number	Unsigned short	2	
Spare1	Spare 1 (zero)	Unsigned short	2	
FreqPage	See following	Unsigned long	4	
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)				
Obs[CHANNELS_20]	20 sets of BeiDou observations	SObsPacket	20 x 12 = 240	
1CodeMSBsPRN[CHANNELS_20]	*See following*	Unsigned long	20 x 4 = 80	

- 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

Structure:

```
 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 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
                              // 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
  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 31:6 = 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
                                        // 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

  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:

Related Commands:

JBIN

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

Bin42 Message

Message Type:

Binary

Description:

Galileo Almanac

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** alAlmwords[0] — WN_a alAlmwords[1] — t_{oa} 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] — M_0 alAlmwords[9] — a_{f0} alAlmwords[10] — a_{f1}	**Long[12]**	**12*4 = 48**

	<table><tr><td>alAlmwords[11]</td><td>[E5a$_{HS}$ | E5b$_{HS}$ | E1B$_{HS}$] Bits [2|2|2]</td></tr></table> **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 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
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
```

Additional Information:

Related Commands:

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

Bin44 Message

Message Type:

Binary

Description:

Galileo time conversion parameters

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	Values
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	Future use
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	Unsigned short	2	0 = GGTO Invalid 1 = GGTO Valid.

CheckSum	Sum of all bytes of header and data	Unsigned short	2	
CRLF	Carriage return line feed	Unsigned short	2	

Structure:

```
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 leapseconds. short m_dtlsf;               // Scheduled futre (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 GGTOpolynomial. double        m_A1G;     // 1st order term of GGTO
polynomial. unsigned long              m_t0G;          // Reference time of week for GGTO.
unsignedshort  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:

JBIN

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

Bin45 Message

Message Type:

Binary

Description:

Galileo ephemeris information

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	Values
SV	Satellite to which this data belongs	Unsigned short	2	
Spare1	Not used at this time	Unsigned short	2	Future use
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:

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 datalength */
          unsigned short   m_wCRLF;           Carriage Return Line Feed */

          } SBinaryMsg95;                     length = 8 + (128) + 2 + 2 = 140 */
                                          /*


                                          /*
```

Additional Information:

Message has a BlockID of 45 and is 128 bytes, excluding the header and epilogue

Related Commands:

JBIN

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

Bin62 Message

Message Type:

Binary, GLONASS

Description:

GLONASS almanac information

Command Format to Request Message:

$JBIN,62,r<CR><LF>

where:

'62' = Bin62 message

'r' = message rate in Hz (1 or 0)

Message Component	Description	Type	Bytes	Values
SV	Satellite to which this data belongs	Byte	1	
Ktag_ch	Proprietary data	Byte	1	
Spare1	Spare, keeps alignment to 4 bytes	Unsigned short	2	
Strings[3]	GLONASS almanac data (36 bytes) • 0 & 1 = Two almanac SFs • 3= SF 5	SGLONASS string	36	

Structure:

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 almanacdata (36 bytes)

0 & 1 = Two almanac SFs, 3= SF 5*/ unsigned short m_wCheckSum; /* sum of all bytes of the datalength */ unsigned short m_wCRLF; /* Carriage Return Line Feed */

} SBinaryMsg62; /* length = 8 + (40) + 2 + 2 = 52 */

Additional Information:

Related Commands:

JBIN

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

Bin65 Message

Message Type

Binary, GLONASS

Description:

GLONASS ephemeris information

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	Values
SV	Satellite to which this data belongs	Byte	1	
Ktag	Satellite K Number + 8	Byte	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:

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. */ unsignedshort 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 datalength */
unsigned short    m_wCRLF;       /*   Carriage Return Line Feed */
```

} SBinaryMsg65; /* length = 8 + (68) + 2 + 2 = 80 */

Additional Information:

Related Commands:

JBIN

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

Bin66 Message

Message Type:

Binary, GLONASS

Description:

GLONASS L1/L2 code and carrier phase information

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	Values
Tow	Time in seconds	Double		
Week	GPS week number	Unsigned short		
Spare1	Spare 1 (zero)	Unsigned short		
Spare2	Spare 2 (zero)	Unsigned long		
L1Obs[CHANNELS_12]	12 sets of L1 (GLONASS) observations	SObsPacket		
L2Obs[CHANNELS_12]	12 sets of L2 (GLONASS) observations	SObsPacket		
L1CodeMSBsSlot[CHANNELS_12]	*See following*	Unsigned long		
• 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				

Structure:

typedef struct

{

SUnionMsgHeader m_sHead;

double m_dTow; /* 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_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

spare

*/

bit 12:8 (5 bits) =

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 thedatalength */ unsignedshort m_wCRLF; /* Carriage Return Line Feed */

} SBinaryMsg66; /* length = 8 + (352) + 2 + 2 = 364 */

Additional Information:

Related Commands:

JBIN

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

Bin69 Message

Message Type:

Binary, GLONASS

Description:

GLONASS L1/L2diagnostic information

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	Values
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:

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 */ unsignedchar   m_bySpare01;   /* spare 1 */ unsignedchar   m_bySpare02;   /* spare 2 */

unsigned short   m_wCheckSum;   /* sum of all bytes of the datalength */ unsigned short m_wCRLF;   /* Carriage Return Line Feed */

} SBinaryMsg69;   /* length = 8 + 300 + 2 + 2 = 312 */

Additional Information:

Related Commands:

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 themessage.

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:

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

4. 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).

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:

Message Component	Description	Type	Bytes	Values
TOW	Predicted GPS time in seconds	Double	8	
Week	GPS week number	Unsigned short	2	
Spare1		Unsigned long	2	

Spare2		Unsigned long	4	
L2PSatObs[12] *(array for next 3 fields)*	*L2 satellite observation data*	*Structure array*	*12 x 12 =* *144*	
CS_TT_W3_SNR	*See following*	Unsigned long	4	

- 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

P7_Doppler_FL	*See following*	Unsigned long	4	

- 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

CodeAndPhase	*See following*	Unsigned long	4	

- 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)

L1CASatObs[15] *(array for next 3 fields)*	*L1 satellite code observation data*	*Structure array*	*15 x 12 =* *180*
CS_TT_W3_SNR	*See following*	Unsigned long	4

- 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

- Bits 24-31 (8 bits)

Cycle Slips; increments by 1 every cycle slip with natural roll-over after 255

P7_Doppler_FL	*See following*	Unsigned long	4

- 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				
CodeAndPhase	*See following*	Unsigned long	4	
• 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)				
L1CACodeMSBsPRN[15]	L1CA code observation	Array of 15 Unsigned long	15 x 4 = 60	*See following*
• 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				
L1PCode[12]	L1(P) code observation data	Array of 12 Unsigned long	12 x 4 = 48	*See following*
• Bits 0-15 (16 bits) 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				

wCeckSum	Sum of all bytes of header and data	Unsigned short	2	
wCRLF	Carriage return line feed	Unsigned short	2	

Structure:

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 15words.

satellite

Spare

upper

bit 7:0 (8 bits) =

PRN, 0 if no satellite bit 12:8 (5 bits) =

bit 31:13 (19 bits) =

19 bits of L1CA LSB

= 256 meters

MSB = 67108864 meters */

unsigned long m_auL1Pword[CHANNELS_12]; /* array of 12 words relating to L1(P)

16 range.

SNR_value

code. Bit 0-15 (16 bits) lower bits of the L1P code pseudo

LSB = 1/256 meters MSB

= 128 meters

Bits 16-27 (12 bits) = L1P

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 datalength */
                unsigned short   m_wCRLF;       /*   Carriage Return Line Feed */
```

} SBinaryMsg76;   /* length = 8 + (448) + 2 + 2 = 460 */

Additional Information:

Related Commands:

JBIN

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

Bin80 Message

Message Type:

Binary

Description:

SBAS data frame information

Command Format to Request Message:

Message Format:

Message Component	Description	Type	Bytes	Values
PRN	Broadcast PRN	Unsigned short	2	
Spare	Not used at this time	Unsigned short	2	Future use
MsgSecOfWeek	Seconds of week for message	Unsigned long	4	
WaasMsg[8]	250-bit WAAS message (RTCA DO0229). 8 unsigned longs, with most significant bit received first.	Unsigned long	4 x 8 = 32	

Additional Information:

$JBIN,80,r<CR><LF>

where:

· '80' = Bin80 message

Structure

```
typedef struct
{
SUnionMsgHeader    m_sHead;
unsigned short m_wPRN; unsigned short            /* Broadcast PRN */
m_wSpare;
unsigned long    m_ulMsgSecOfWeek;
long             m_aulWaasMsg[8]; short          /* spare (zero) */
m_wCheckSum;
unsigned short m_wCRLF;
                                                 /* Seconds of Week For Message */ unsigned
                                                 /* Actual 250 bit waas message*/ unsigned
                                                 /* sum of all bytes of the datalength */
```

```
                                          /* Carriage Return Line Feed */

} SBinaryMsg80;                           /* length = 8 + (40) + 2 + 2 = 52 */
```

'r' = message rate in Hz (1 or 0)

Message Component	Description	Type	Bytes	Values
PRN	Broadcast PRN	Unsigned short	2	
Spare	Not used at this time	Unsigned short	2	Future use
MsgSecOfWeek	Seconds of week for message	Unsigned long	4	
WaasMsg[8]	250-bit WAAS message (RTCA DO0229). 8 unsigned longs, with most significant bit received first.	Unsigned long	4 x 8 = 32	

Message has a BlockID of 80 and is 40 bytes, excluding the header and epilogue

Related Commands:

JBIN

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

Bin89 Message

Message Type:

Binary

Description:

SBAS satellite tracking information (supports three SBAS satellites)

Command Format to Request Message:

Message Format:

$JBIN,89,r<CR><LF>

Where:

· '89' = Bin89 message

Message Component	Description	Type	Bytes	Values
GPSSecOfWeek	GPS tow integer sec	Long		
MaskSBASTracked	SBAS satellites tracked, bit mapped 0..3	Byte		
MaskSBASUSED	SBAS satellites used, bit mapped 0..3	Byte		
Spare	Spare	Unsigned short		
ChannelData[CHANNELS_SBAS_E]	SBAS channel data	SChannelData		

'r' = message rate in Hz (1 or 0)

Structure:

typedef struct

{

SUnionMsgHeader long

m_sHead;

m_lGPSSecOfWeek; /* GPS tow integer sec */

unsigned char m_byMaskSBASTracked; /* SBAS Sats Tracked, bit mapped 0..3 */

unsigned char unsigned short

m_byMaskSBASUSED; /* SBAS Sats Used, bit mapped 0..3 */ m_wSpare; /* spare */

SChannelData unsigned short unsigned short

} SBinaryMsg89;

m_asChannelData[CHANNELS_SBAS_E]; /* SBAS channel data */ m_wCheckSum; /* sum of all bytes of thedatalength */ m_wCRLF; /* Carriage Return Line Feed */

/* length = 8 + 80 + 2 + 2 = 92 */

Additional Information:

Related Commands:

JBIN

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

Bin92 Message

Message Type:

Binary

Description:

QPS Almanac

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 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]	t_{oa}
alAlmwords[1]	$\sqrt{A}$
alAlmwords[2]	e
alAlmwords[3]	ω
alAlmwords[4]	M_0
alAlmwords[5]	Ω_0
alAlmwords[6]	$\dot{\Omega}$
alAlmwords[7]	δ_i
alAlmwords[8]	SV Health
alAlmwords[9]	WN_a

Structure:

```
//----------------------------------------------------------------------------
// SBinaryMsg92
//  QPS 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
unsigned short            m_wCheckSum;           // sum of all bytes of the header and data
 unsigned short            m_wCRLF;                  // Carriage Return Line Feed
} SBinaryMsg92; // length = 8 + (4+48=52) + 2 + 2 = 64
```

Additional Information:

Related Commands:

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

Bin93 Message

Message Type:

Binary

Description:

SBAS ephemeris information

Command Format to Request Message:

Message Format:

$JBIN,93,r<CR><LF>

where:

· '93' = Bin93 message

Message Component	Description	Type	Bytes	Values
SV	Satellite to which this data belongs	Unsigned short	2	
Spare	Not used at this time	Unsigned short	2	Future use
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	

'r' = message rate in Hz (1 or 0)

Structure:

```
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 datalength */ 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 45 bytes, excluding the header and epilogue

Related Commands:

JBIN

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

Bin94 Message

Message Type:

Binary

Description:

Ionospheric and UTC conversion parameters

Command Format to Request Message:

Message Format:

$JBIN,94,r<CR><LF>

Where:

'94' = Bin94 message

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

Message Component	Description	Type	Bytes	Values
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, second 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	
Spare	Not used at this time	Short	2	Future use

When set to on the message is sent once and then sent again whenever satellite information changes

Structure:

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 GPSweek. */ 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 shortm_wSpare1;    /* spare 4 (zero)*/

unsigned short m_wCheckSum; /* sum of all bytesof the datalength */ 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:

JBIN

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

Bin95 Message

Message Type:

Binary

Description:

GPS ephemeris information

Command Format to Request Message:

Message Format:

$JBIN,95,r<CR><LF>

Where:

'95' = Bin95 message

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

Message Component	Description	Type	Bytes	Values
SV	Satellite to which this data belongs	Unsigned short	2	
Spare1	Not used at this time	Unsigned short	2	Future use
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:

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 messagewords. */ unsignedlong 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 datalength */
unsigned short      m_wCRLF;              Carriage Return Line Feed */

} SBinaryMsg95;                           length = 8 + (128) + 2 + 2 = 140 */
                                    /*


                                    /*
```

Additional Information:

Message has a BlockID of 95 and is 128 bytes, excluding the header and epilogue

Related Commands:

JBIN

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

Bin96 Message

Message Type:

Binary

Description:

GPS L1 code and carrier phase information

Command Format to Request Message:

Message Format:

$JBIN,96,r<CR><LF>

Where:

'96' = Bin96 message

Message Component	Description	Type	Bytes	Values
Spare1	Not used at this time	Unsigned short	2	Future use
Week	GPS week number	Unsigned short	2	
TOW	Predicted GPS time in seconds	Double	8	
UNICS_TT_SNR_PRN[12]	*See following*	Unsigned long	4	

- Bits 0-7 (8 bits)

Pseudorandom noise; PRN is 0 if no data

- Bits 8-15 (8 bits)

Signal-to noise ratio (SNR); SNR=10.0 *log10* (0.8192*SNR)

- Bits 16-23 (8 bits)

PhaseTrackTime (PTT); in units of 1/10 sec; range=0 to 25 sec (if greater than 25 see UIDoppler_FL[12] below)

- Bits 24-31 (8 bits)

CycleSlip Counter (CSC); increments by 1 every cycle with natural rollover after 255

UIDoppler_FL[12]	*See following*	Unsigned long	4	

- 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.

PseudoRange[12]	Pseudorange	Double	8	

'r' = message rate in Hz (20, 10, 2, 1, or 0)

Phase[12]	Phase (m) L1 wave = 0.190293672798365	Double	8	

Structure:

```
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 datalength      */

                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:

JBIN

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

Bin97 Message

Message Type

Binary

Description:

Processor statistics

Command Format to Request Message:

Message Format:

$JBIN,97,r<CR><LF>

Where:

'97' = Bin97 message

Message Component	Description	Type	Bytes	Values
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	Positive
MissedSubFrame	Total number of missed sub frames in the navigation message since power on	Unsigned short	2	Positive
MaxSubFramePnd	Max sub frames queued for processing at any one time	Unsigned short	2	Positive
MissedAccum	Total number of missed code accumulation measurements in the channel tracking loop	Unsigned short	2	Positive
MissedMeas	Total number missed pseudorange measurements	Unsigned short	2	Positive
Spare 1	Not used at this time	Unsigned long	4	Future use
Spare 2	Not used at this time	Unsigned long	4	Future use
Spare 3	Not used at this time	Unsigned long	4	Future use
Spare 4	Not used at this time	Unsigned short	2	Future use
Spare 5	Not used at this time	Unsigned short	2	Future use

'r' = message rate in Hz (20, 10, 2, 1, 0, or .2)

Structure:

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 datalength */ unsignedshort 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:

JBIN

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

Bin98 Message

Message Type:

Binary

Description:

GPS satellite and almanac information

Command Format to Request Message:

$JBIN,98,r<CR><LF>

Where:

- '98' = Bin98 message

'r' = message rate in Hz (1 or 0)

Message Component	Description	Type	Bytes	Values
AlmanData[8]	SV data, 8 at a time	SSVAlmanData		See following
LastAlman	Last almanac processed	Byte	1	0 to 31
IonoUTCVFlag	Flag that is set when ionosphere modeling data is extracted from the GPS sub frame 4	Byte	1	0 = not logged 2 = valid
Spare	Not used at this time	Unsigned short	2	Future use

Message Format:

Structure

```
typedef struct
{
SUnionMsgHeader        m_sHead;

SSVAlmanData           m_asAlmanData[8];    /*  SV data, 8 at a time */
unsigned char          m_byLastAlman;       /*  last almanac processed */
unsigned char unsigned m_byIonoUTCVFlag;    /*  iono UTC flag */ spare */
short                  m_wSpare;
                                                sum of all bytes of the datalength */
unsigned short         m_wCheckSum;
                                            /*
```

```
                                        /*

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:

JBIN

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

Bin99 Message

Message Type:

Binary

Description:

GPS L1 diagnostic information

Command Format to Request Message:

$JBIN,99,r<CR><LF>

Where:

'99' = Bin99 message

Message Component	Description	Type	Bytes	Values
NavMode	Navigation mode data (lower 3 bits hold the GPS mode, upper bit set if differential is available)	Byte	1	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
UTCTimeDiff	Whole seconds between UTC and GPS time (GPS minus UTC)	Byte	1	Positive
GPSWeek	GPS week associated with this message	Unsigned short	2	0 to 65536
GPSTimeofWeek	GPS tow (sec) associated with this message	Double	8	0.0 to 604800.0
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	-32768 to 32768
Spare	Not used at this time	Unsigned short	2	Future use

'r' = message rate in Hz (1 or 0)

Structure:

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 shortm_wGPSWeek; /* GPS week */

double m_dGPSTimeOfWeek; /* GPS tow */

SChannelData m_asChannelData[CHANNELS_12]; /* channel data*/ short m_nClockErrAtL1; /* clock error at L1, Hz */

unsigned shortm_wSpare; /* spare */

unsigned short m_wCheckSum; /* sum of all bytes of the datalength */ unsignedshort 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:

JBIN

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

Bin100 Message

Message Type:

Binary

Description:

GPS L2 diagnostic information

Command Format to Request Message:

Message Format:

$JBIN,100,r<CR><LF>

where:

'100' = Bin100 message

'r' = message rate in Hz (1 or 0)

Message Component	Description	Type	Bytes	Values
NavMode	Navigation mode data (lower 3 bits hold the GPS mode, upper bit set if differential is available)	Byte	1	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
UTCTimeDiff	Whole seconds between UTC and GPS time (GPS minus UTC)	Byte	1	Positive

GPSWeek	GPS week associated with this message	Unsigned short	2	0 to 65535
MaskSatsUsedL2P	L2P satellites used, bit mapped 0..31	Unsigned long		
GPSTimeofWeek	GPS tow (sec) associated with this message	Double	8	0.0 to 604800.0
MaskSatsUsedL1P	L1P satellites used, bit mapped 0..31	Unsigned long		
sChannelData[CHANNELS_12]	L2 channel data	SChannelData	12 x 24 = 288	

Structure

```
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 mapped0..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 datalength */ unsigned short
m_wCRLF;                /* Carriage Return Line Feed */

} SBinaryMsg100;                        /* length = 8 + 260 + 2 + 2 = 272 */
```

Additional Information:

Message has a BlockID of 100 and is 260 bytes, excluding the header and epilogue

Related Commands:

JBIN

Topic Last Updated: v1.08 / June 9, 2017

Bin122 Message

Message Type:

Binary

Description:

Alternate position solution data

Command Format to Request Message:

$JBIN,122,r<CR><LF>

Where:

- '122' = Bin122message
- 'r' = message rate in Hz

Message Format

Message Component	Description	Type	Bytes	Values
GPSTimeOfWeek	GPS tow (sec) associated with this message	Double	8	to 604800.0
GPSWeek	GPS week associated with this message	Unsigned short	2	0 to 65535
PosType	Type of position 0: Autonomous 1: SBAS 2: Differential phase solution 3: Differential code solution 4: RTK (fixed vs. float is not specified) 5: RTK Fixed 6: RTK Float 7: Tracer 8: Manual 9: Atlas (fixed vs. float not specified) 10: SureFix 11: FastFix	Unsigned char	1	
Correction source	Source of corrections 0: No correction 1: SBAS 2: eDif 3: Atlas	Unsigned char	1	

	6: RTCM unspecified version 7: RTCM 2.3 8: RTCM 3 9: ROX 11: CMR			
BaseID	Base station ID	Unsigned short	2	0 to 65535
SatUsedCount	Sats used in each system [GPS, GLN, GAL, BDS, SBAS, QZSS]	Unsigned char array	6*1 = 6	
Latitude	Latitude in degrees north	Double	8	-90.0 to 90.0
Longitude	Longitude in degrees east	Double	8	-180.0 to 180.0
Height	Altitude above ellipsoid in meters	Float	4	
AgeOfDiff	Age of differential, in seconds	Float	4	
VNorth	North-South velocity, +North m/s	Float	4	
VEast	East-West velocity, +East m/s	Float	4	
VUP	Vertical velocity, +up m/s	Float	4	
CovNN	Covariance North-North	Float	4	
CovNE	Covariance North-East	Float	4	
CovNU	Covariance North-Up	Float	4	
CovEE	Covariance East-East	Float	4	
CovEU	Covariance East-Up	Float	4	
CovUU	Covariance Up-Up	Float	4	

Structure:

typedef struct

{

SUnionMsgHeader m_sHead;

```
    double          m_dGPSTimeOfWeek;     // GPS tow                              [8 bytes]
    unsigned short   m_wGPSWeek;          // GPS week                              [2 bytes]
    unsigned char    m_byPhxPosType;       // Phoenix position type                  [1 bytes]
    unsigned char    m_byCorSource;       // Phoenix correction source               [1 byte ]
    unsigned short   m_wBaseStationId;     // Base station ID                          [2 bytes]
    unsigned char    m_bySatUsedCount[6];  // Satellites used per system              [6 bytes]
                                  //  [GPS GLN GAL BDS SBAS QZSS]
    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_fAgeOfDiff;        // age of differential, seconds            [4 bytes]
    float           m_fVNorth;           // North-South Velocity +North m/s          [4 bytes]
    float           m_fVEast;            // East-West Velocity +East m/s            [4 bytes]
    float           m_fVUp;              // Vertical Velocity +up  m/s              [4 bytes]
    float           m_fCovNN;            // Covariance North-North                  [4 bytes]
    float           m_fCovNE;            // Covariance North-East                   [4 bytes]
    float           m_fCovNU;            // Covariance North-Up                     [4 bytes]
    float           m_fCovEE;            // Covariance East-East                    [4 bytes]
    float           m_fCovEU;            // Covariance East-Up                      [4 bytes]
    float           m_fCovUU;            // Covariance 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 */
} SBinaryMsg122;                    /* length = 8 + 80 + 2 + 2 = 92 */
```

Additional Information:

Related Commands:

JBIN

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

Bin209 Message

Message Type:

Binary

Description:

SNR and status for all GNSS tracks

Command Format to Request Message:

Message Format:

$JBIN,209,r<CR><LF>

Where:

'209' = Bin209 message

'r' = message rate in Hz

Message Component	Description	Type	Bytes	Values
GPSTimeofWeek	GPS tow (sec) associated with this message	Double	8	0.0 to 604800.0
GPSWeek	GPS week associated with this message	Unsigned short	2	0 to 65535
UTCTimeDiff	Whole Seconds between UTC and GPS	char	1	
Page	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	Unsigned char	1	
sSVData	SNR data			

Structure:

```
typedef struct

{

        SUnionMsgHeader     m_sHead;                    //

        double              m_dGPSTimeOfWeek;           // GPS tow

        unsigned short      m_wGPSWeek;                 // GPS week

        char                m_cUTCTimeDiff;             // Whole Seconds between UTC and GPS

        unsigned char       m_byPage;

// 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

    SSVSNRData              m_asSVData[40];             // SNR data

    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
```

Additional Information:

Related Commands:

JBIN

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

SSVSNRData209

Message Type:

Binary

Description:

Satellite Data for Bin 209

Command Format to Request Message:

Message Format:

Message Component	Description	Type	Bytes
Status_SYS_PRNID	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, 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	Unsigned Short	2
m_chElev	Elevation angle, LSB = 1 deg	char	1
m_byAzimuth	1/2 the Azimuth angle, LSB = 2 deg	Unsigned char	1
m_ulSNR3_SNR2_SNR1	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	Unsigned long	4

Structure:

```
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, 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
```

Related Commands:

JBIN

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

Bin309 Message

Message Type:

Binary

Description:

SNR and status for all GNSS tracks

Command Format to Request Message:

$JNIN,309,r<CR><LF>

where:

'309' = Bin309 message

'r'=message rate in Hz

Message Format:

Message Component	Description	Type	Bytes	Values
GPSTimeofWeek	GPS tow (sec) associated with this message	Double	8	0.0 to 604800.0
GPSWeek	GPS week associated with this message	Unsigned short	2	0 to 65535
UTCTimeDiff	Whole Seconds between UTC and GPS	char	1	
Page	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	Unsigned char	1	
sSVData309	SNR data	SSVSNRData309	30 * 16 = 480	

Structure:

```
typedef struct

{

   unsigned shortm_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, 7 = SBAS, 8 = IRNSS
                                        //    Bit 11-15   Spare
                                         //    m_wSYS_PRNID = 0 ==> unfilled data
   unsigned shortm_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
                                        //       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
                                        //    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 shortm_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
```

Additional Information:

Related Commands:

JBIN

Topic Last Updated: 3.0 / December 30, 2019

SSVSNRData309

Message Type:

Binary

Description:

Satellite Data for Bin309

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, 7 = SBAS, 8 = IRNSS 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 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 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 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_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 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 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:

```
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, 7 =
SBAS, 8 = IRNSS

                                       //   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

                     //      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

                    //    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
```

Related Commands:

JBIN

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

NMEA 0183 Supported Messages

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

GLMLA Message

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.

Command Format to Request Message:

$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 other port when you send the command with it (without the brackets)

Message Format:

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

where:

Message Component	Description
A.A	Total number of sentences
B.B	Sentence number
CC	Satellite ID (satellite slot) number
D.D	Calendar day count within the four year period beginning with the previous leap year
EE	Generalized health of the satellite and carrier frequency number respectively
FFFF	Eccentricity
GG	DOT, rate of change of the draconitic circling time
HHHH	Argument of perigee
IIII	16 MSB of system time scale correction
JJJJJJ	Correction to the average value of the draconitic circling time
KKKKKK	Time of the ascension node, almanac reference time
MMMMMM	Greenwich longitude of the ascension node
NNNNNN	Correction to the average value of the inclination angle
PPP	LSB of system time scale correction

QQQ	Course value of the time scale shift

Example:

Additional Informatio:

Similar to the GPS message GPALM

Related Commands:

JASC,GL

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

GPALM Message

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

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:

Similar to the GLONASS message GLMLA

Related Commands:

JASC,GP

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

GPDTM Message

GPDTM Message

Message Type:

Data

Description:

Datum reference

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:

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:

JASC,GP

Topic Last Updated:

GPGGA Message

GPGGA Message

Message Type:

Data

Description:

Detailed GNSS position information (most frequently used NMEA 0183 data message)

Command Format to Request Message:

$JASC,GPGGA,r[,OTHER]<CR><LF>

where:

●□□□ 'r' = messager ate (in Hz) of 20, 10, 5, 4, 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:

$GPGGA,HHMMSS.SS,DDMM.MMMMM,K,DDDMM.MMMMM,L,N,QQ,PP.P,AAAA.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 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)
N	Quality indicator; value is: • 0 = no position • 1 = undifferentially corrected position (autonomous) • 2 = differentially corrected position (SBAS, DGPS,Atlas DGPSservice, L- Dif and e-Dif) • 4 = RTK fixed integer (Crescent RTK, Eclipse RTK),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

GNGGA - GNSS information GLGGA - GLONASS 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:

JASC,GP, JASC,GN, JASC,GL, JNMEA,GGAALLGNSS

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

GPGLL Message

GPGLL Message

Message Type:

Data

Description:

Latitude and longitude data

Command Format to Request Message:

$JASC,GPGLL,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 thebrackets)

Message Format:

$GPGLL,DDMM.MMMMM,S,DDDMM.MMMMM,S,HHMMSS.SS,S*CC<CR><LF>

Where:

Message Component	Description
DDMM.MMMMM	Latitude in degrees, minutes, and decimal minutes
S	S = N (North latitude) or S (South latitude)
DDDMM.MMMMM	Longitude in degrees, minutes, and decimal minutes
S	S = E (East longitude) or W (West longitude)
HHMMSS.SS	UTC time in hours, minutes, and seconds of GNSS position
S	Status, S = A (valid) or V (invalid)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

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.

Related Commands:

JASC,GP, JASC,GN, JASC,GL, JNMEA,GGAALLGNSS

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

GPGNS Message

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

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 positionfix • D = Differential. Satellite system used in differential mode in positionfix • 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.

Related Commands:

JASC,GP, JASC,GN, JASC,GL, JNMEA,GGAALLGNSS

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

GPGRS Message

GPGRS Message

Message Type:

Data

Description:

Supports Receiver Autonomous Integrity Monitoring (RAIM)

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:

Message Component	Description
HHMMSS.SS	UTC time
M	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
X.X ... X.X	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.
GSID	GNSS system ID, value is 1 (GPS)
SID	Signal ID, value is 1 (L1 C/A)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Commands:

JASC,GP

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

GNGSA Message

GNGSA Message

Message Type:

Data

Description:

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.

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 ... OO,P.P,Q.Q,R.R,GSID*CC<CR><LF>

where:

Message Component	Description
A	Satellite acquisition mode (M = manually forced to 2D or 3D, A = automatic swap between 2D and 3D)
B	Position mode (1 = fix not available, 2 = 2D fix, 3 = 3D fix)
CC to OO	Satellites used in the position solution, a null field occurs if a channel is unused
P.P	Position Dilution of Precision (PDOP) = 1.0 to 9.9
Q.Q	Horizontal Dilution of Precision (HDOP) 1.0 to 9.9
R.R	Vertical Dilution of Precision (VDOP) = 1.0 to 9.9
GSID	GNSS system ID, value is 1 (GPS), 2 (GLONASS), 3 (GALILEO), 5 (BEIDOU)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

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

Related Commands:

JASC,GP, JASC,GN, JASC,GL

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

GPGST Message

GPGST Message

Message Type:

Data

Description:

GNSS pseudorange error statistics and position accuracy

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:

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the GPS position
A.A	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.
B.B	Standard deviation of semi-major axis of error ellipse, in meters
C.C	Standard deviation of semi-minor axis of error ellipse, in meters
D.D	Error in Eclipse’s semi major axis origination, in decimal degrees, true north
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
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Command:

JASC,GP

Topic Last Updated: v1.01 / September 23, 2010

GPGSV Message

GPGSV Message

Message Type:

Data

Description:

GPS satellites in view

Null fields occur where data is unavailable due to the number of satellites tracked.

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:

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

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 Command:

JASC,GP, JASC,GL, BEIDOU

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

GLGSV Message

GLGSV Message

Message Type:

Data

Description:

GLONASS satellites in view

Null fields occur where data is unavailable due to the number of satellites tracked.

Command Format to Request Message:

$JASC,GLGSV,r[,OTHER]<CR><LF>

where:

- 'r' = message rate in Hz of 05, .1, .5, .2, 1, 2, 5, 10, 20

●(and without the brackets) and enacts a change on the other port when you send the command with it (without the brackets)

Message Format:

$GLGSV,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
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

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

GAGSV – GALILEO information GQGSV – QZSS information

If you request GNGSV the receiver will respond with GPGSV messages only.

Related Commands:

JASC,GP, JASC,GL, BEIDOU

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

GAGSV Message

GAGSV Message

Message Type:

Data

Description:

Galileo satellites in view

Null fields occur where data is unavailable due to the number of satellites tracked.

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>

where:

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

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:

JASC,GP, JASC,GL, BEIDOU

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

GBGSV Message

GBGSV Message

Message Type:

Data

Description:

BeiDou satellites in view

Null fields occur where data is unavailable due to the number of satellites tracked.

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
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

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

GAGSV – GALILEO information GQGSV – QZSS information

If you request GNGSV the receiver will respond with GPGSV messages only.

Related Commands:

JASC,GP, JASC,GL, BEIDOU

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

GPHDG/HEHDG Message

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.

Command Format to Request Message:

$JASC,GPHDG,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:

$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:

Message Component	Description
s.s	Magnetic sensor reading, in degrees
d.d	Magnetic deviation, in degrees
D	E = Easterly deviation, W = Westerly deviation
v.v	Magnetic variation, in degrees
V	E = Easterly deviation, W = Westerly deviation
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

You can change the HDG message header to either GP or HE using the JATT,NMEAHE command.

Related Commands:

JASC,GP

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

GPHEB Message

GPHEB Message

Description:

Heave value in meters

Command Format to Request Message:

$JASC,GPHEV,1<CR><LF>

Message Format:

$GPHEV,H,*CC<CR><LF>

where:

Message Component	Description
H	Heave value, in meters
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Commands:

JASC,GP

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

GPRMC Message

GPRMC Message

Message Type:

Data

Description:

Contains recommended minimum specific GNSS data

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:

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the GPS position
A	Status (A = valid, V = invalid)
DDMM.MMM	Latitude in degrees, minutes, and decimal minutes
N	Latitude location (N = North latitude, S = South latitude)
DDDMM.MMM	Longitude in degrees, minutes, and decimal minutes
W	Longitude location (E = East longitude, W = West longitude)
Z.Z	Ground speed, in knots
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

Additional Information:

Related Commands:

JASC,GP

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

GPHDT/HEHDT Message

GPHDT/HEHDT Message

Description:

True heading of the vessel. This is the direction that the vessel (antennas) is pointing and is not necessarily the direction of vessel motion (the course over ground).

Command Format to Request Message:

$JASC,GPHDT,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:

$GPHDT,X.X,T*CC<CR><LF>

or

$HEHDT,X.X,T*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

Additional Information:

You can change the HDT message header to either GP or HE using the JATT,NMEAHE command.

Related Commands:

JASC,GP

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

GPHDM/HEHDM Message

GPHDM/HEHDM Message

Message Type:

Data

Description:

Magnetic heading of the vessel derived from the true heading calculated

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
M	Indicates magnetic heading
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

You can change the HDM message header to either GP or HE using the JATT,NMEAHE command.

Related Commands:

JASC,GP

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

GPROT/HEROT Message

GPROT/HEROT Message

Message Type:

Data

Description:

Vessel’s rate of turn (ROT) information

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:

Message Component	Description
X.X	Rate of turn in °/min (negative when the vessel bow turns to port)
A	Flag indicating the data is valid
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

You can change the ROT message header to either GP or HE using the JATT,NMEAHE command.

Related Commands:

JASC,GP

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

GPRRE Message

GPRRE Message

Message Type:

Data

Description:

Satellite range residuals and estimated position error

Command Format to Request Message:

$JASC,GPRRE,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:

$GPRRE,N,II,RR ... II,RR,HHH.H,VVV.V*CC<CR><LF>

where:

Message Component	Description
N	Number of satellites used in position computation
II	Satellite number
RR	Range residual, in meters
HHH.H	Horizontal position error estimate, in meters
VVV.V	Vertical position error estimate, in meters
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Commands:

JASC,GP

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

GPVTG Message

GPVTG Message

Message Type:

Data

Description:

Course over ground and ground speed

Command Format to Request Message:

$JASC,GPVTG,r[,OTHER]<CR><LF>

Where:

· 'r' = messager ate 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:

$GPVTG,TTT,T,MMM,M,NNN.NN,N,KKK.KK,K,X*CC<CR><LF>

where:

Message Component	Description
TTT	True course over ground (COG) in degrees (000 to 359)
T	True course over ground indicator (always 'T')
MMM	Magnetic course over ground in degrees (000 to 359)
M	Magnetic course over ground indicator (always 'M')
NNN.NN	Speed over ground in knots
N	Speed over ground in knots indicator (always 'N')
KKK.KK	Speed over ground in km/h
K	Speed over ground in km/h indicator (always 'K')

X	Mode
	
	A = Autonomous mode D = Differential mode
	
	E = Estimated (dead reckoning) mode M = Manual input mode
	
	S = Simulator mode N = Data not valid

*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Sample message output:

$GPVTG,103.85,T,92.79,M,0.14,N,0.25,K,D*1E

Additional Information:

Related Commands:

JASC,GP

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

GPZDA Message

GPZDA Message

Message Type:

Data

Description:

UTC time and date information

Command Format to Request Message:

$JASC,GPZDA,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:

$GPZDA,HHMMSS.SS,DD,MM,YYYY,XX,YY*CC<CR><LF>

Where:

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the GPS unit
DD	Day (0 to 31)
MM	Month (1 to 12)
YYYY	Year
XX	Local zone description in hours (-13 to 13)
YY	Local zone description in minutes (0 to 59)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Commands:

JASC,GP

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

PASHR Message

PASHR Message

Message Type:

Vector, Data

Description:

Time, true heading, roll, pitch, and heave data in one message

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)

· ',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 th 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><

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

Additional Information:

Related Commands:

JASC,PASHR

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

HGNSS Proprietary Messages

PSAT, ATTSTAT Message

PSAT,ATTSTAT Message

Message Type:

Data

Description:

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:

Message Component	Description
S	ID of the secondary antenna
MSEP	custom separation between antennas manually entered (when the value is MOV, it means MOVEBASE is on)
CSEP	auto GPS antenna separation
Heading	Heading
TYPE	Heading indicator, value is: N= Heading used GNSS G=Heading used gyroscope
Pitch	pitch
Roll	roll
Q	The current setting of antenna directivity, value is P= antennas placed front and back, output pitch R= antennas placed left and right, output roll
N	The number of satellite used by the secondary antenna

SYS	Systems in use: • GPS: L1, L2, L5 • GLONASS: G1, G2 • BDS: B1,B2 B3 • Galileo: E5a, E5b, E5a+b, E6
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: v2.0/April 30, 2019

PSAT, GBS Message

PSAT,GBS Message

Message Type

Data

Description:

Used to support Receiver Autonomous Integrity Monitoring (RAIM)

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:

Message Component	Description
HHMMSS.SS	UTC time in hours, minutes, and seconds of the GGA or GNS fix associated with this sentence
KK.K	Expected error in latitude
LL.L	Expected error in longitude
AA.A	Expected error in altitude
ID	ID number of most likely failed satellite
P.PPPPP	Probability of HPR fault
B.B	Estimate of range bias, in meters, on most likely failed satellite
S.S	Standard deviation of range bias estimate
FLAG	Based on horizontal radius: 0 = Good 1 = Warning 2 = Bad or Fault
GSID	GNSS system ID, value is 1 (GPS)
SID	Signal ID, value is 1 (L1 C/A)
*CC	Checksum
<CR>	Carriage return

<LF>	Line feed

Additional Information:

Related Commands:

JASC,GP

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

PSAT, HPR Message

PSAT,HPR Message

Message Type:

Data

Description:

Proprietary NMEA message that provides the true heading,pitch, roll, and time in a single message

During normal operation heading and pitch are derived from GPS and roll comes from the inertial sensor. While coasting heading is based on gyro and pitch/roll are from the inertial sensor.

Command Format to Request Message:

$JASC,GPHPR,r[,OTHER]<CR><LF>

where:

- 'r' = message rate in Hz of 20, 10, 2, 1, 0 or .2 (0 turnsoff 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:

Message Component	Description
TIME	UTC time (HHMMSS.SS)
HEADING	Heading (degrees)
PITCH	Pitch (degrees)
ROLL	Roll (degrees)
TYPE	N = GPS derived heading G = gyro heading
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Commands:

JASC,GP

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

PSAT, INTLT Message

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.

Command Format to Request Message:

$JASC,INTLT,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,INTLT,PITCH,ROLL*CC<CR><LF>

where:

Message Component	Description
PITCH	Pitch (degrees)
ROLL	Roll (degrees)
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Additional Information:

Related Commands:

JASC,GP

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

PSAT, BLV Message

PSAT, BLV Message

Message Type:

Data, Local Differential and RTK

Description:

Contains RTK fix progress information

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 thebrackets)

Message Format:

$PSAT,BLV,HHMMSS.SS,DATE,A.A,B.B,C.C,ID,STATE,number,pdop*CC<CR><L F>

where:

Message Component	Description
HHMMSS.SS	UTC time (HHMMSS.SS)
DATE	Date (day-month-year)
A.A	North component of base to rover vector (m)
B.B	Esat component of base to rover vector (m)
C.C	Up component of base to rover vector (m)
ID	Base station ID
STATE	Quality indicator; value is: • 0 = no position • 1 = undifferentially corrected position (autonomous) • 2 = differentially corrected position (SBAS, DGPS, Atlas DGPS service, L-Dif and e-Dif) • 4 = RTK fixed integer (Crescent RTK, Eclipse RTK) ,Atlas high precision services converged 5 = RTK float, Atlas high precision services converging
NUMBER	Number of used satellite

PDOP	PDOP
*CC	Checksum
<CR>	Carriage return
<LF>	Line feed

Example:

$PSAT,BLV,000151.00,051115,-0.001,0.002,-0.003,0333,4,20,1.2*52

Additional Information:

Related Commands:

JASC, PSAT, BLV

Topic Last Updated: v1.08 / June 9, 2017

PSAT, FVI Message

PSAT,FVI Message

Message Type:

Data, Local Differential and RTK

Description:

Contains much more special information

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, DDMM.MMMM, DDDMM.MMMM, 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
DDMM.MMMM	Latitude in degrees and decimal minutes
DDMM.MMMM	Longitude in degrees, and decimal minutes
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)
LU.UUU	Up component of master to slave vector (m)
ZONE	projection area
UEEE.EEEE	East to positon 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 = undifferentially corrected position (autonomous) • 2 = differentially corrected position (SBAS, DGPS ,Atlas DGPS service, L-Dif ande-Dif) • 4 = RTK fixed integer (Crescent RTK, Eclipse RTK), 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:

JASC,PSAT,FVI

Topic Last Updated: v1.08 / June 9, 2017

PSAT, RPTKPROG Message

PSAT,RTKPROG Message

Message Type:

Data, Local Differential and RTK

Description:

Contains RTK fix progress information

$JASC,PSAT,RTKPROG,r[,OTHER]<CR><LF>

Command Format to Request Message:

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>

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:

$PSAT,RTKPROG,1,1,24,243.0,0.0,0.0,3*4F<CR><LF>

- Ready to enter RTK ambiguity fix
- Receiver running in RTK ambiguity fix mode
- 24 satellites used to fix summer-1 is 243.0, summer-2 is 0, summer-3 is 0
- Bit mask is 3 (GPS + GLONASS)

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:

JASC,PSAT,RTKPROG

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

PSAT, RTKSTAT Message

PSAT,RTKSTAT Message

Message Type

Data, Local Differential and RTK

Description:

Contains the most relevant parameters affecting RTK

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, L5 • GLONASS: G1, G2 • BDS: B1,B2 B3 • Galileo: E5a, E5b, E5a+b, E6
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,),0,1,0.011,000

- 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 command

JQUERY,RTKSTAT message

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

PSAT, VCT Message

PSAT,VCT Message

Message Type:

Data, Local Differential and RTK

Description:

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:

Related Command:

JASC,PSAT,VCT

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

TSS1 Message

TSS1 Message

Message Type

Vector, Data

Description:

Heave, pitch,and roll message in the commonly used TSS1 message format

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:

Message Component	Description
XX	Horizontal acceleration (hex value), in 3.83 cm/s², with a range of zero to 9.81 m/s²
AAAA	Vertical acceleration (hex value - 2's complement), in 0.0625 cm/s², with a range of –20.48 to +20.48 m/s²
S	Space character
M	Space if positive; minus if negative
HHHH	Heave, in centimeters, with a range of –99.99 to +99.99 meters
Q	Status flag Value Description 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 thiscondition. F 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.
M	Space if positive; minus if negative

RRRR	Roll, in units of 0.01 degrees (ex: 1000 = 10°), with a range of –99.99° to +99.99°
S	Space character
M	Space if positive; minus if negative
PPPP	Pitch, in units of 0.01 degrees (ex: 1000 = 10°), with a range of –99.99° to +99.99°
<CR>	Carriage return
<LF>	Line feed

Example:

: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°

Related Commands

JASC,PTSS1

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

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.

The following is an example of how the RTCM data appears on screen:

```
mRMP@PJfeUtNsmMFM{nVtIOTDbA^xGh~kDH`_FdW_yqLRryrDuh
cB\@}N`ozbSD@O^}nrGqkeTlpLLrYpDqAsrLRrQN{zW|uW@H`z]~aG
xWYt@I`_FxW_qqLRryrDCikA\@Cj]DE]|E@w_mIroMNjkKOsmMFM{ WDw
W@HVEbA^xGhLJQH`_F`W_aNsmMFM[WVLA\@S}amz@iIIuP
qx~IZhTCpLLrYpdP@kOsmMFM[kVDHwVGbA^P{WWuNt_SW_yMs
mMnqdrhcC\@sE^ZfC@}vJmNGAHJVhTCqLRryrdviStW@H_GbA^ P{wxu[k
```

All Hemisphere GNSS receivers support RTCM v2.x Type 1, Type 5, Type 6, and Type 9 messages for DGPS positioning.

Hemisphere GNSS receivers do not support RTCM v2.x messages for RTK positioning. However RTCM v3.x messages (Type 1001 through 1008) are suitable for RTK positioning.

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

Firmware

About Firmware

Hemisphere GNSS products are built on one of three receiver platforms, each of which has specific firmware applications available.

- Crescent - WAAS, e-Dif,Atlas service, L-Dif/RTK base, L-Dif/RTK rover
- Crescent Vector - WAAS, RTK rover
- Eclipse - WAAS/RTK base, RTK rover Atlas high precision services

Some products may require purchasing a subscription code to unlock specific functionality. See Subscription Codes for more information.

As its name suggests, firmware is somewhere between hardware and software. Like software, it is a computer program which is executed by a microprocessor or a micro-controller. But it is also tightly linked to a piece of hardware, and has little meaning outside of it.

Within the context of GNSS, the hardware is the GNSS receiver and it is the receiver's processor that executes the firmware. The receiver's processor supports two simultaneous versions of firmware but only one version operates at a given time. The two versions—referred to as applications—may have different functionality.

Use the JAPP command to change between two receiver applications.

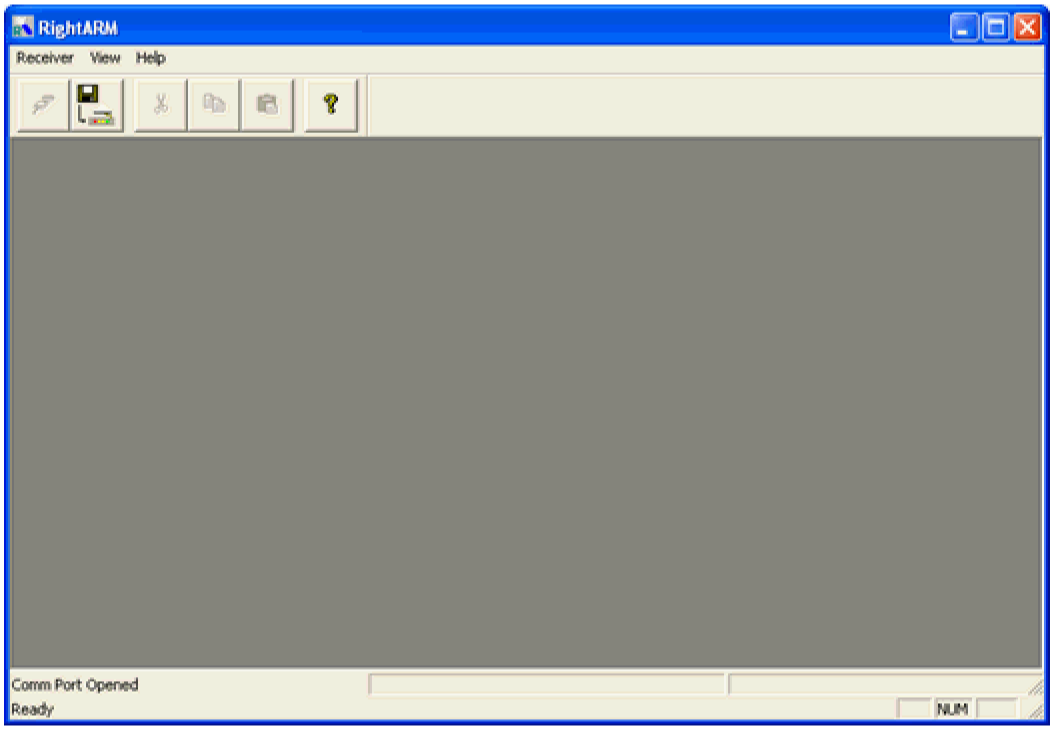
Topic last updated: v1.07/ February 16, 2017

Using RightARM to Load Firmware

RightARM is Hemisphere GNSS software that allows you to load the various GNSS receiver firmware options and updates as they are provided by Hemisphere GNSS.

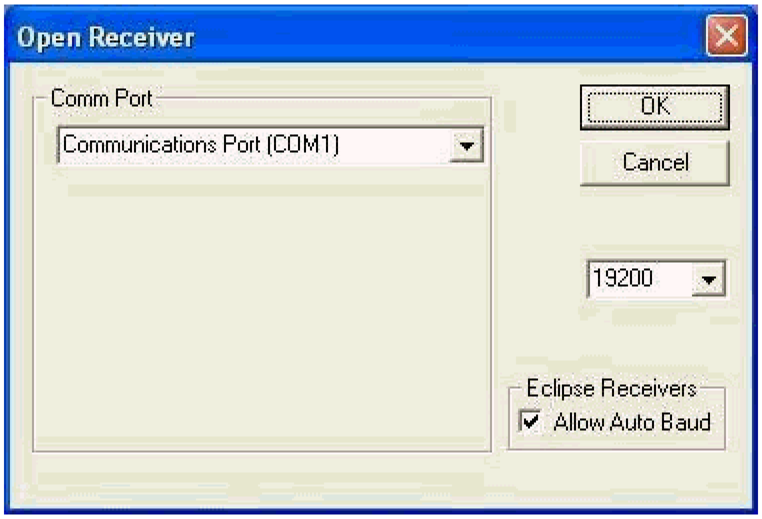
To load the firmware:

1. Download the latest version of RightARM from http://www.hemispheregnss.com.
2. Install RightARM application on your computer.
3. Connect the receiver to your computer and power on the receiver.
4. Double-click the RightARM icon to launch the program. The following screen appears.



Comm Port Opened
Ready

1. Click the **Open Receiver** button or select **Receiver > Connect**. The Open Receiver window appears, so you can identify a connected receiver.

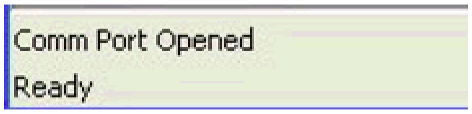


Communications Port (COM1)

19200

[✓] Allow Auto Baud

2. Select the **Comm Port** on your computer to which the receiver is connected, select the 19200 baud rate for the receiver, and then click **OK**.

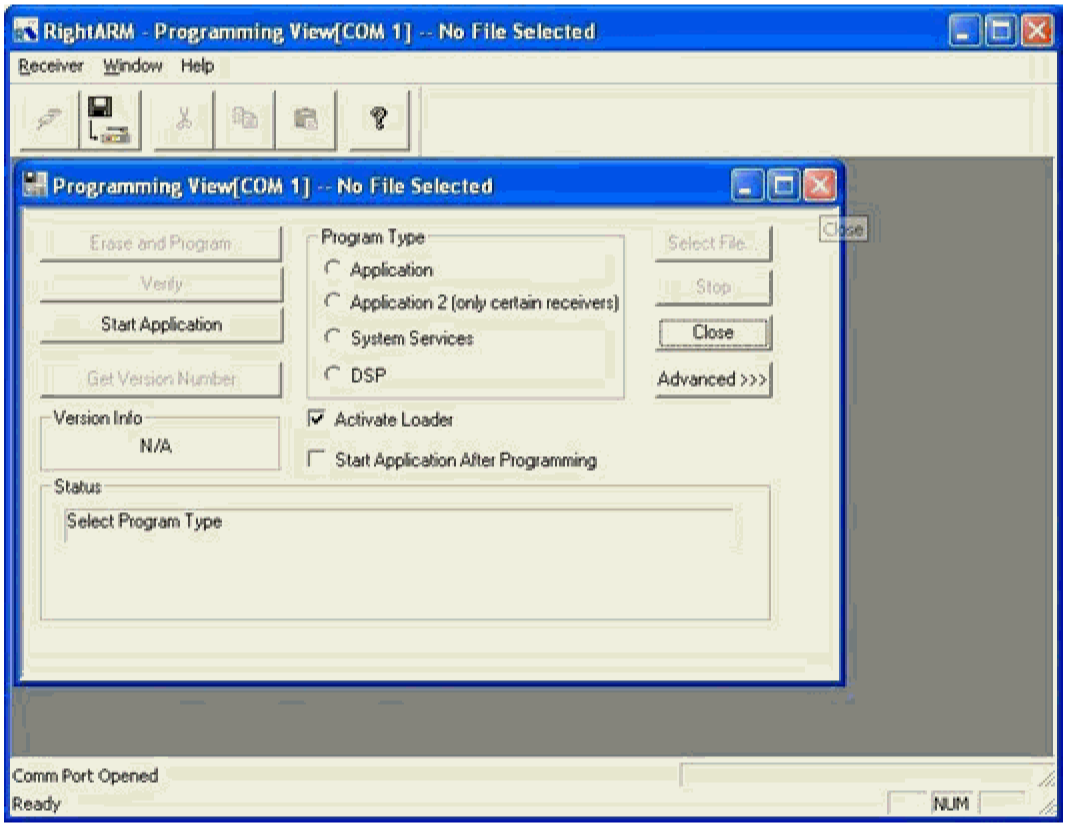


Comm Port Opened
Ready

Note: You must set the baud rate to 19200.

When RightARM has successfully connected to the receiver the following message appears in the lower left corner of the screen.

3. Click the **Programming View** button . The Programming View window appears, enabling you to select different firmware programming options.



4. Select the Program Type you want to install and then click **Select File**. The Open window appears.

5. Select the required firmware file from the location where you saved it on your computer and click Open**. "File Loaded" appears in the status window on the Programming View window.**

6. Click the **Erase and Program** button to erase the firmware that is currently installed on the receiver in the selected application location and install the newly selected file in its place. "Erasing...Please Wait" appears in the Status field and a progress bar below this message indicates the programming progress. Once the new firmware has been successfully loaded on the receiver "Programming Done" appears in the Status field.

7. Once the appropriate firmware has been loaded, click the **Close** button to close the Programming View window.

8. Exit RightARM, turn off your receiver, and then disconnect the receiver from your computer.

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

Subscriptions Codes

This section covers:

o Finding the serial number and inputting a subscription code (e-Dif, RTK, 20 Hz or 10Hz, etc.) into a Hemisphere GNSS receiver

o Viewing the status and interpreting the $JI subscription date codes

o The difference between the receiver's response to the $JK and $JI commands

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

Subscribing to an Application

Activating an application code on a Hemisphere GNSS receiver requires the following: Serial communication cable to connect the Hemisphere GNSS receiver to the serial COM port on the computer·□□□

Download SLXMon from the www.hemispheregnss.com and install it on your PC or use a generic terminal program such as HyperTerminal

Load the application to which to subscribe onto the Hemisphere GNSS receiver (see Using RightARM to Load Firmware)

Purchase the application subscription code from Hemisphere GNSS or an authorized Hemisphere GNSS representative

To activate the application on a Hemisphere GNSS receiver:

1. Connect the Hemisphere GNSS receiver to the serial COM port on the computer.

Start SLXMon.

2. Select **File > Connect** and then select the appropriate Comm Port and Baud Rate to open communication with the receiver.

3. Select **Control > View Command Page**.

4. In the Receiver Command Page window type **$JAPP** in the Message box and then click **Send**.

5. Confirm which applications are loaded onto the receiver and the order in which they appear in the Reply box.

Example Response (in Reply box):

$>JAPP,WAAS,DIFF

where WAAS (SBAS, EGNOS, MSAS) is the number one application (or application number 1) and DIFF (same as e-Dif) is the "other" application (or application number 2)

If DIFF is listed as application number 2 in the $JAPP response then type the following command in the Message box: $JAPP,O where 'O' is the "other" application in the example. This swaps the two applications so that DIFF is be the current application.

Type the following command in the Message box:

$JI

The first number in the response is the serial number of the receiver.

Example Response (in Reply box):

$>JI,810133,1,3,09031998,01/06/1998,12/31/2018,3.5,31

The serial number is 810133. You will need to provide it to Hemisphere GNSS with your request for an e-Dif subscription code.

Type the following command in the Message box after receiving the subscription code from Hemisphere GNSS:

$JK,nnnn

where 'nnnn' is the subscription number. The receiver will respond with "subscription accepted."

Topic Last Updated: v1.7 / 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

Tables 1 and 2 below provide subscription information for Crescent and Eclipse receivers, where the data rate and subscription are indicated by the 'date' returned by the JK command. For Eclipse II receivers, refer to Eclipse II Subscription Codes. The part of the date that indicates the data rate and subscription code is called the Additive Code. The last two digits in the subscription expiration date's 'year' comprise the Additive Codes, that is, the available data output rate from the receiver, plus the subscriptions—the enabled GPS differential correction sources.

Table 3 outlines the components of the Crescent, Eclipse, and Eclipse II Additive Codes. The subscription codes have different additive components for Crescent, Eclipse, and Eclipse II.

Table 1: Crescent Subscription Codes

Date Code (Additive Code)	Hex Code	Maximum Data Rate	Subscription Description
3000 (0)	HEX 0	10 Hz	SBAS enabled
3001 (1)	HEX 1	20 Hz	SBAS enabled
3002 (0+2)	HEX 2	10 Hz	SBAS, e-Dif enabled
3003 (1+2)	HEX 3	20 Hz	SBAS, e-Dif enabled
3004 (0+4)	HEX 4	10 Hz	SBAS, RTK Rover enabled
3005 (1+4)	HEX 5	20 Hz	SBAS, RTK Rover enabled
3006 (0+2+4)	HEX 6	10 Hz	SBAS, RTK Rover, e-Dif enabled
3007 (1+2+4)	HEX 7	20 Hz	SBAS, RTK Rover, e-Dif enabled
3008 (0+8)	HEX 8	10 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Base enabled
3009 (1+8)	HEX 9	20 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Base enabled
3010 (0+2+8)	HEX A	10 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Base, e-Dif enabled
3011 (1+2+8)	HEX B	20 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Base, e-Dif enabled
3012 (0+4+8)	HEX C	10 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Rover, RTK Base enabled
3013 (1+4+8)	HEX D	20 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Rover, RTK Base enabled
3014 (0+2+4+8)	HEX E	10 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Rover, RTK Base, e-Dif enabled
3015 (1+2+4+8)	HEX F	20 Hz	SBAS, L-Dif Rover, L-Dif Base, RTK Rover, RTK Base, e-Dif enabled

Table 2: Eclipse Subscription Codes

Date Code (Additive Code)	Hex Code	Maximum Data Rate	Subscription Description
3000 (0)	HEX 0	10 Hz	SBAS, Atlas enabled
3001 (1)	HEX 1	20 Hz	SBAS,Atlas enabled

3004 (0+4)	HEX 4	10 Hz	SBAS,Atlas , RTK Rover, RTK Base, Raw L1/L2 data enabled
3005 (1+4)	HEX 5	20 Hz	SBAS,Atlas , RTK Rover, RTK Base, Raw L1/L2 data enabled
3008 (0+8)	HEX 8	10 Hz	SBAS,Atlas , RTK Base, Raw L1/L2 data enabled
3009 (1+8)	HEX 9	20 Hz	SBAS,Atlas , RTK Base, Raw L1/L2 data enabled
3016 (0+16)	HEX 10	10 Hz	SBAS,Atlas , Raw L1/L2 data enabled
3017 (1+16)	HEX 11	20 Hz	SBAS,Atlas , Raw L1/L2 data enabled

Table 3: Crescent, Eclipse, and Eclipse II Additive Codes Components

Crescent		Eclipse		Eclipse II	
Code	Description	Code	Description	Code	Description
0	10 Hz	0	10 Hz	0	10 Hz
1	20 Hz	1	20 Hz	1	20 Hz
2	e-Dif	2	n/a	2	e-Dif
4	L-Dif Rover, L-Dif Base, RTK Rover	4	Raw L1/L2 Data, RTK Base, RTK Rover	4	RTK Rover (minimum L1 only)
8	RTK Base	8	Raw L1/L2 Data, RTK Base	8	RTK Base (minimum L1 only)
16	n/a	16	Raw L1/L2 Data	16	Raw Data (minimum L1 only)
32	n/a	32	n/a	32	L2 signals
64	n/a	64	n/a	64	GLONASS signals (minimum L1 only)

Crescent Additive Code Examples

- 10 Hz (SBAS), e-Dif, and RTK is 0+2+4 = 6 (so 30**06**)
- 20 Hz (SBAS), e-Dif, and RTK is 1+2+4 = 7 (so 30**07**)

Comparing the JI and JK Responses

Example 1:

In the following Crescent 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 and 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: v1.03 / January 11, 2012

Eclipse II Subscription Codes

Use the information below to determine your Eclipse II subscription code and its features.

	1	2	4	8	16	32	64		
	0x01	0x02	0x04	0x08	0x10	0x20	0x40		
	20Hz	e-Dif	RTK Rover, RTK Base, Raw Out	RTK Base, Raw Out	Raw Out	L2	GLONASS	Date Code (Additive Code)	Hex Cod
Standard								3000	0
	Y							3001	1
		Y						3002	2
	Y	Y						3003	3
			Y					3004	4
	Y		Y					3005	5
		Y	Y					3006	6
	Y	Y	Y					3007	7
				Y				3008	8
	Y			Y				3009	9
		Y		Y				3010	A
	Y	Y		Y				3011	B
			Y	Y				3012	C
	Y		Y	Y				3013	D
		Y	Y	Y				3014	E
	Y	Y	Y	Y				3015	F
				Y				3016	10
	Y			Y				3017	11
		Y		Y				3018	12
	Y	Y			Y			3019	13
			Y		Y			3020	14
	Y		Y		Y			3021	15
		Y	Y		Y			3022	16

	Y	Y	Y		Y			3023	17
				Y	Y			3024	18
	Y			Y	Y			3025	19
		Y		Y	Y			3026	1A
	Y	Y		Y	Y			3027	1B
			Y	Y	Y			3028	1C
	Y		Y	Y	Y			3029	1D
		Y	Y	Y	Y			3030	1E
	Y	Y	Y	Y	Y			3031	1F
						Y		3032	20
	Y					Y		3033	21
		Y				Y		3034	22
	Y	Y				Y		3035	23
			Y			Y		3036	24
	Y		Y			Y		3037	25
		Y	Y			Y		3038	26
	Y	Y	Y			Y		3039	27
				Y		Y		3040	28
	Y			Y		Y		3041	29
		Y		Y		Y		3042	2A
	Y	Y		Y		Y		3043	2B
			Y	Y		Y		3044	2C

	1	2	4	8	16	32	64		
	0x01	0x02	0x04	0x08	0x10	0x20	0x40		
	20Hz	e-Dif	RTK Rover, RTK Base, Raw Out	RTK Base, Raw Out	Raw Out	L2	GLONASS	Date Code (Additive Code)	Hex Cod
		Y	Y	Y		Y		3045	2D
		Y	Y	Y		Y		3046	2E

	Y	Y	Y	Y		Y		3047	2F
					Y	Y		3048	30
	Y				Y	Y		3049	31
		Y			Y	Y		3050	32
	Y	Y			Y	Y		3051	33
			Y		Y	Y		3052	34
	Y		Y		Y	Y		3053	35
		Y	Y		Y	Y		3054	36
	Y	Y	Y		Y	Y		3055	37
				Y	Y	Y		3056	38
	Y			Y	Y	Y		3057	39
		Y		Y	Y	Y		3058	3A
	Y	Y		Y	Y	Y		3059	3B
			Y	Y	Y	Y		3060	3C
	Y		Y	Y	Y	Y		3061	3D
		Y	Y	Y	Y	Y		3062	3E
	Y	Y	Y	Y	Y	Y		3063	3F
							Y	3064	40
	Y						Y	3065	41
		Y					Y	3066	42
	Y	Y					Y	3067	43
			Y				Y	3068	44
	Y		Y				Y	3069	45
		Y	Y				Y	3070	46
		Y	Y				Y	3071	47
							Y	3072	48
	1	Y		Y			Y	3073	49
	1	Y		Y			Y	3074	4A
				Y			Y	3075	4B
	Y		Y	Y			Y	3076	4C

			Y	Y			Y	3077	4D
	Y	Y	Y	Y			Y	3078	4E
		Y	Y	Y			Y	3079	4F
	Y				Y		Y	3080	50
					Y		Y	3081	51
	Y	Y			Y		Y	3082	52
		Y			Y		Y	3083	53
	Y		Y		Y		Y	3084	54
	Y		Y		Y		Y	3085	55
		Y	Y		Y		Y	3086	56
	Y	Y	Y		Y		Y	3087	57
				Y	Y		Y	3088	58
	Y			Y	Y		Y	3089	59
		Y		Y	Y		Y	3090	5A
	Y	Y		Y	Y		Y	3091	5B
			Y	Y	Y		Y	3092	5C

	1	2	4	8	16	32	64		
	0x01	0x02	0x04	0x08	0x10	0x20	0x40		
	20Hz	e-Dif	RTK Rover, RTK Base, Raw Out	RTK Base, Raw Out	Raw Out	L2	GLONASS	Date Code (Additive Code)	Hex Cod
	Y		Y	Y	Y		Y	3093	5D
		Y	Y	Y	Y		Y	3094	5E
	Y	Y	Y	Y	Y		Y	3095	5F
						Y	Y	3096	60
	Y	Y				Y	Y	3097	61
		Y				Y	Y	3098	62
	Y	Y				Y	Y	3099	63
			Y			Y	Y	3100	64

	Y		Y			Y	Y	3101	65
		Y	Y			Y	Y	3102	66
	Y	Y	Y			Y	Y	3103	67
				Y		Y	Y	3104	68
	Y			Y		Y	Y	3105	69
		Y		Y		Y	Y	3106	6A
	Y	Y		Y		Y	Y	3107	6B
			Y	Y		Y	Y	3108	6C
	Y		Y	Y		Y	Y	3109	6D
		Y	Y	Y		Y	Y	3110	6E
	Y	Y	Y	Y		Y	Y	3111	6F
					Y	Y	Y	3112	70
	Y				Y	Y	Y	3113	71
		Y			Y	Y	Y	3114	72
	Y	Y			Y	Y	Y	3115	73
			Y		Y	Y	Y	3116	74
	Y		Y		Y	Y	Y	3117	75
		Y	Y		Y	Y	Y	3118	76
	Y	Y	Y		Y	Y	Y	3119	77
				Y	Y	Y	Y	3120	78
	Y			Y	Y	Y	Y	3121	79
		Y		Y	Y	Y	Y	3122	7A
	Y	Y		Y	Y	Y	Y	3123	7B
			Y	Y	Y	Y	Y	3124	7C
	Y		Y	Y	Y	Y	Y	3125	7D
		Y	Y	Y	Y	Y	Y	3126	7E
	Y	Y	Y	Y	Y	Y	Y	3127	7F

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

Determining the Receiver Type and Current Application

To determine the current receiver type, use the JT command. Table 1 shows the receiver type indicated by the JT response.

Table 1: $JT Response and Receiver Type

$JT Response	Receiver Type
SX1x	SX-1
SX2x	Crescent
SLXx	SLX2/SLX3
DF2x	Eclipse
DF3x	Eclipse II
MF3x	miniEclipse

The 'x' in the responses represents the receiver's current application. For example, if x = i, as in SX2i, 'i' is the application code for e-Dif.

Table 2 shows the application for the application code in the JT response.

Table 2: $JT Response and Application

$JT Responses with Application Code	Receiver Application
r	RTK rover
b	RTK base
i	e-Dif
g	L-band
g	WAAS
g	Standalone
a	Vector

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

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

Resources

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

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 1800 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

71828 L Street, NW, Suite 805

Washington, D.C. 20036 USA
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
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 Ionosphereic Activity

The following sites are useful in providing details regarding solar and ionospheric activity.

http://iono.jpl.nasa.gov

http://www.spaceweather.com

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