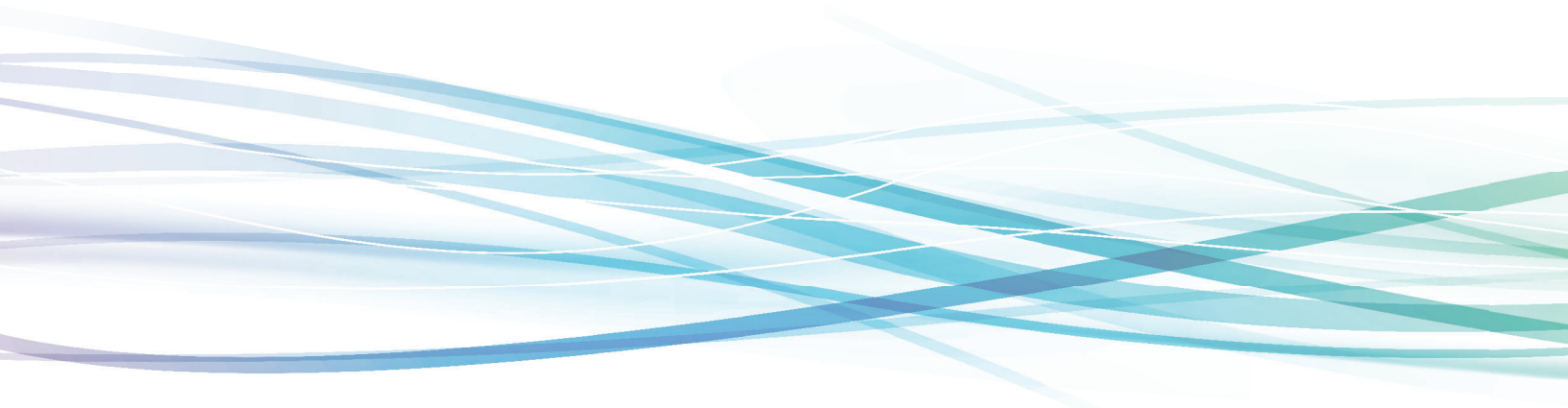


Polaris Vega Application Programming Interface Guide

Applies to Polaris Vega Systems with API Revision G.003.006

IL-1070259 R07
December 2021

Revision Status

Revision Number	Date	Description
1	August 2016	First release
2	August 2017	Updated to document the optional video camera.
3	April 2018	The video camera is no longer supported so related content has been removed. Updated content on device names, starting on page 26. Added content on the frame timestamp and PTP clock scheme, on page 78.
4	June 2019	Updated to document the optional video camera and support for serial communications via the optional LEMO connector. Several chapters are reorganized. This document release coincides with the release of Polaris Vega Firmware Package 006.
5	November 2019	Documented P**aram.Tracking.Frame Frequency**, **Param.Tracking.Track Frequency**, **Info.Status.Tracking.Measured Frame Frequency,** and **Info.Status.Tracking.Measured Track Frequency**. Documented Event 6 “PTP Master has changed”. Changed `WARNING` references to `WARNING01` in the PINIT command. Added `WARNING06` to Table 6-2 on page 183.
6	February 2020	Updated figures in Important Concepts chapter.
7	December 2021	Updated to incorporate Polaris Vega API revision G.003.006.

Part Number: IL-1070259

Published by:

Northern Digital Inc.
103 Randall Dr.
Waterloo, Ontario, Canada N2V 1C5

Telephone: + (519) 884-5142
Toll Free: + (877) 634-6340
Global: + (800) 634 634 00
Facsimile: + (519) 884-5184
Website: www.ndigital.com

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

Northern Digital Inc. has taken due care in preparing this document and the programs and data on the electronic media accompanying this document including research, development, and testing.

This document describes the state of Northern Digital Inc.'s knowledge respecting the subject matter herein at the time of its publication, and may not reflect its state of knowledge at all times in the future. Northern Digital Inc. has carefully reviewed this document for technical accuracy. If errors are suspected, the user should consult with Northern Digital Inc. prior to proceeding. Northern Digital Inc. makes no expressed or implied warranty of any kind with regard to this document or the programs and data on the electronic media accompanying this document.

Northern Digital Inc. makes no representation, condition or warranty to the user or any other party with respect to the adequacy of this document or accompanying media for any particular purpose or with respect to its adequacy to produce a particular result. The user's right to recover damages caused by fault or negligence on the part of Northern Digital Inc. shall be limited to the amount paid by the user to Northern Digital Inc. for the provision of this document. In no event shall Northern Digital Inc. be liable for special, collateral, incidental, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claim for lost profits, data, fees or expenses of any nature or kind.

Product names listed are trademarks of their respective manufacturers. Company names listed are trademarks or trade names of their respective companies.

The Polaris Vega System includes software that is distributed under the GPL v2 licence. For details on the GPL v2 licence refer to http://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

Table of Contents

About This Guide

This guide describes the Polaris Vega Application Programming Interface, including the commands and parameters supported by the Position Sensor, Video Camera, and System Control Unit. For information about which versions of the API and which devices in the system support a particular command, see the command's description in the Command Details chapter, starting on page 62.

The recommended way to query the firmware and API revisions is by using the following command/parameter combination:

```
GET Features.Firmware.API Revision
```

For backwards compatibility, the APIREV (page 67) command is still supported.

Note **This guide is specific to the API for Vega. For information on previous versions of the API for earlier Polaris systems, see the "*Polaris Application Program Interface Guide*" (IL-1070101) available on the NDI support site at https://support.ndigital.com.**

Warnings and Cautions

Warnings

In all NDI documentation, warnings are marked by this symbol. Follow the information in the accompanying paragraph to avoid personal injury.

1. Do not connect the Polaris Vega System to a host computer or network that is not IEC 60950 and/or IEC 60601 approved. If you connect the system to a non-approved host computer or network you may increase leakage currents beyond safe limits and cause personal injury.
2. When using reply option 0800 with the BX (page 69) or TX (page 159) commands, you must take appropriate action to detect the following events: the tool or marker is out of volume, the bump sensor has been tripped, or the system is outside of the optimal operating temperature range. You must determine whether these events are detrimental to your application. If one or more of the events listed occurs, reply option 0800 enables the system to return data that may lead to inaccurate conclusions and may cause personal injury.
3. No options exist for filtering data returned from the BX2 (page 77) command on the basis of system or tool status or location in the volume. Complete system and tool status information is always included in the reply and it is the application's responsibility to interpret this data and ignore those measurements that fall outside of application requirements and constraints. Failure to do so may lead to inaccurate conclusions that may cause personal injury.

Contact Information

If you have any questions regarding the content of this guide or the operation of this product, please contact us:

Head Office
Waterloo, ON Canada
☎ +1 (877) 634-6340
✉ support@ndigital.com
🔗 www.ndigital.com

Shelburne, VT USA
☎ +1 (802) 985-1114
✉ support@ndigital.com
🔗 www.ndigital.com

Radolfzell, Germany
☎ +49 7732 8234 0
✉ support@ndieurope.com
🔗 www.ndieurope.com

Hong Kong, China
☎ + (852) 2802-2205
✉ support@ndigital.com
🔗 www.ndigital.cn

NDI is committed to continuous improvements in the quality and versatility of its software and hardware. To obtain the best results with your NDI system, regularly check for updated information on the NDI support site at https://support.ndigital.com.

1 List of Commands

- Table 1-1 lists the API commands supported by the Vega Position Sensor.
- Table 1-2 lists the API commands supported by the Vega System Control Unit.
- Table 1-3 lists the API commands supported by the Vega Video Camera.

Table 1-1 List of Commands - Vega Position Sensor

Command	Page	Description
3D	63	Returns the latest 3D position of either a single marker or multiple markers.
APIREV	67	Returns the API revision number that functions with your system.
BEEP	68	Sounds the system beeper.
BX	69	Returns the latest tool transformations, individual marker positions, and system status in binary format.
BX2	77	Returns various levels of data on the latest tool transformations, individual marker positions, and system status in binary format.
COMM	87	Sets the serial communication settings of the system. (Serial communication only.)
DFLT	89	Restores the user parameters to factory default values.
DSTART	91	Starts Diagnostic mode.
DSTOP	92	Stops Diagnostic mode.
ECHO	93	Returns exactly what is sent with the command.
GET	94	Returns the user parameter values.
GETINFO	96	Returns descriptive information about the user parameters.
GETLOG	98	Returns the contents of a system log file.
INIT	100	Initializes the system.
IRATE	102	Sets the illuminator rate.
IRED	104	Turns the markers on a wired tool on or off.
LED	106	Changes the state of visible LEDs on a wired tool.
PDIS	108	Disables the reporting of transformations for a particular port handle.
PENA	109	Enables reporting of transformations for a particular port handle.
PFSEL	111	Sets which tool faces to use to track a multi-faced tool.
PHF	113	Releases system resources from an unused port handle.
PHINF	114	Returns port handle status, and information about the tool associated with the port handle, including physical port location.
PHRQ	119	Assigns a port handle to a tool.
PHSR	121	Returns the number of assigned port handles and the port status for each one. Assigns a port handle to a wired tool.
PINIT	124	Initializes a port handle.
PPRD	126	Reads data from the SROM device in a wired tool. (Only applicable to the System Control Unit.)
PPWR	128	Writes data to the SROM device in a wired tool. (Only applicable to the System Control Unit.)
PSEL	130	Selects an SROM device as the target for reading or writing with PPRD or PPWR.
PSRCH	131	Returns a list of valid SROM device IDs for a wired tool or GPIO device.

Table 1-1 List of Commands - Vega Position Sensor (Continued)

Command	Page	Description
PURD	133	Reads data from the user section of the SROM device in a wired tool. (Only applicable to the System Control Unit.)
PUWR	135	Writes data to the user section of a tool SROM device in a wired tool. (Only applicable to the System Control Unit.)
PVWR	137	Assigns a tool definition file to a wireless tool, overrides a tool definition file in a wired tool, and can be used to test a tool definition file before permanently recording the tool definition file onto the SROM device.
RESET	139	Resets the system (can specify either a hard reset or a soft reset).
SAVE	140	Saves all non-volatile user parameters that have been changed.
SET	141	Sets user parameter values.
SFLIST	143	Returns information about the supported features of the system.
STREAM	149	Initiates a streaming response to a specified command.
SYSLOG	151	Writes data to the device log file.
TCTST	153	Returns diagnostics on the active markers of a wired tool.
TSTART	153	Starts Tracking mode.
TSTOP	156	Stops Tracking mode.
TTCFG	157	Sets up a configuration for a wired tool so that you can test the tool without using a tool definition file.
TX	159	Returns the latest tool transformations, individual marker positions, and system status in text format.
USTREAM	168	Stops streaming of the indicated command
VCAP	169	Captures IR image data from the sensors and/or video camera image data.
VGET	175	Retrieves data previously captured with VSNAP.
VER	173	Returns the firmware revision number of critical processors installed in the system.
VSEL	178	Selects a characterized measurement volume.
VSNAP	179	Captures one complete frame sequence of video data from the sensors.

Table 1-2 List of Commands - Vega System Control Unit

Command	Page	Description
APIREV	67	Returns the API revision number that functions with your system.
BEEP	68	Sounds the system beeper.
ECHO	93	Returns exactly what is sent with the command.
GET	94	Returns the user parameter values.
GETINFO	96	Returns descriptive information about the user parameters.
GETLOG	98	Returns the contents of a system log file.
PPRD	126	Reads data from the SROM device in a wired tool. (Only applicable to the System Control Unit.)
PPWR	128	Writes data to the SROM device in a wired tool. (Only applicable to the System Control Unit.)
PSEL	130	Selects an SROM device as the target for reading or writing with PPRD or PPWR.
PSRCH	131	Returns a list of valid SROM device IDs for a wired tool or GPIO device.
PURD	133	Reads data from the user section of the SROM device in a wired tool. (Only applicable to the System Control Unit.)
PUWR	135	Writes data to the user section of a tool SROM device in a wired tool. (Only applicable to the System Control Unit.)
RESET	139	Resets the system (can specify either a hard reset or a soft reset).
SAVE	140	Saves all non-volatile user parameters that have been changed.
SET	141	Sets user parameter values.
SYSLOG	151	Writes data to the device log file.
VER	173	Returns the firmware revision number of critical processors installed in the system.

Table 1-3 List of Commands - Vega Video Camera

Command	Page	Description
APIREV	67	Returns the API revision number that functions with your system.
DFLT	89	Restores the user parameters to factory default values.
ECHO	93	Returns exactly what is sent with the command.
GET	94	Returns the user parameter values.
GETINFO	96	Returns descriptive information about the user parameters.
GETLOG	98	Returns the contents of a system log file.
INIT	100	Initializes the system.
RESET	139	Resets the system (can specify either a hard reset or a soft reset).
SAVE	140	Saves all non-volatile user parameters that have been changed.
SET	141	Sets user parameter values.
SYSLOG	151	Writes data to the device log file.
VCAP	169	Captures video camera image data.
VER	173	Returns the firmware revision number of critical processors installed in the system.

2 Important Concepts

With the introduction of Polaris Vega, the following programming concepts should be understood:

2.1 General Binary Format

The General Binary Format (GBF) is used consistently in all new commands for Vega to return tracking and video data. Its advantage is that the host does not need to keep the context of the request to be able to parse it correctly. It can also contain various levels of detail corresponding to the reported tracking frame. It is structured as a list of individual, well-defined components. Each component holds the information on its unique type and its options that define the process of parsing its content. It uses little endian byte order and all size byte values are interpreted as unsigned values. Figure 2-2 illustrates the general structure of the format.

All numeric values are four bytes (32 bits) unless otherwise specified. The first field in the payload is a two-byte integer that indicates the number of components the payload contains.

Each component starts with a unique two byte value defining its type, followed by four bytes specifying the size of the component, including the 12 bytes for the header.

By design, the system randomly adds unsolicited "unknown components" as part of the BX2 reply or other commands that use the GBF. These additional components ensure the application programmer implements the parser such that it can handle future extensions.

Item Format Option (two bytes) is specific to the component type. Each type will have its own set of options that provide all the information needed to parse the content of the component. The Item Format Option implies the Item’s size.

Item Count (four bytes) describes the number of following items to parse. After parsing all the specified items, a new component starts with its definition of the component type and the parsing process repeats.

Component IDs are as follows:

01 – Frame Component

02 – 6D Data Component

03 – 3D Data Component

04 – 1D Data Component – buttons

05 – 2D Data Component

06 – reserved

07 – reserved

08 – reserved

09 – reserved

10 – Image Data Component

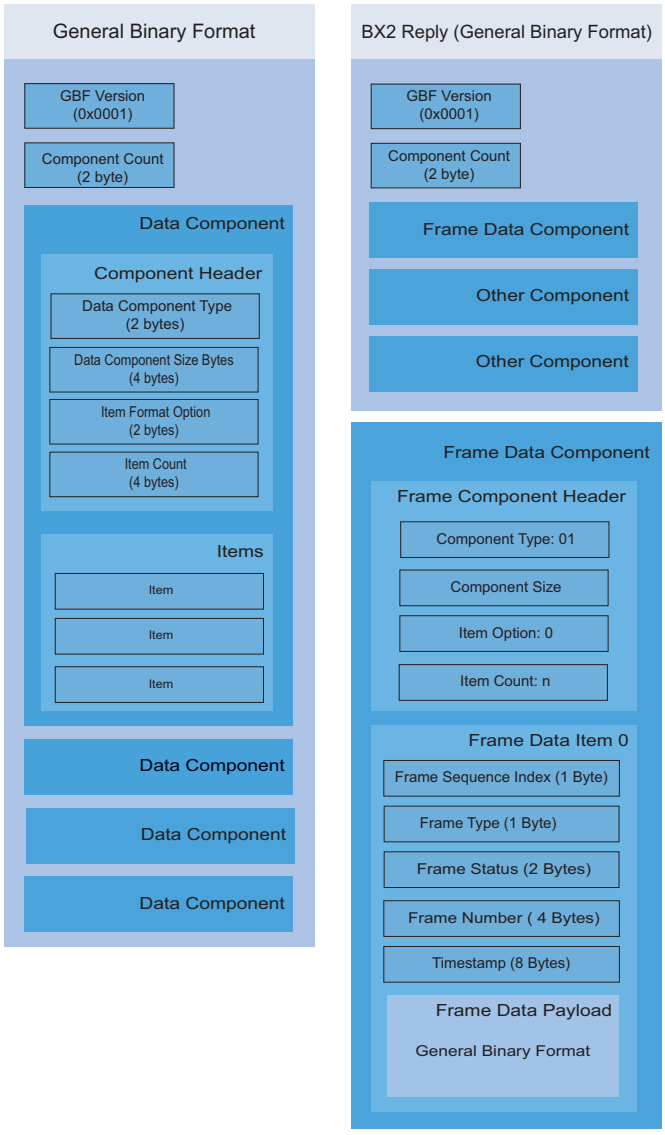
11 to 16 – reserved

17 – Sensor U,V Component

18 – System Alert Component

An example of the GBF structure, with an example of the BX2 command is shown in 2.2.

Figure 2-1 General Binary Format structure and BX2 example



2.2 Extended Binary Header

To facilitate binary replies that have a binary payload greater than 2^{16}-1 bytes long, a new binary header type is introduced. This header has a 32 bit length field and allows for reply lengths up to 2^{32}-1 bytes long. Either binary header may be used in response to any of the “new” binary commands, such as BX2 and VCAP.

This extended binary reply header is intended for use with very large replies. If the reply length is less than 2^{16}-1 bytes long, the original binary header is used. Since TCP packets already include data checksums and to reduce processing time and allow for more efficient memory-to-memory transfer techniques, no CRC will be included in the header or at the end of the data. Thus, the extended header is the same length as the original header.

The format of an extended binary header reply is as follows:

A5C8<four byte Reply Length><command reply>

2.3 Operating Roles for Host Connections

In older Vega firmware (predating API revision G.003.006), NDI promoted the concept of monitor connections, where secondary connections could assume the role of monitors or observers. The host could query the system for status information or tracking data, but no state-changing commands could be issued.

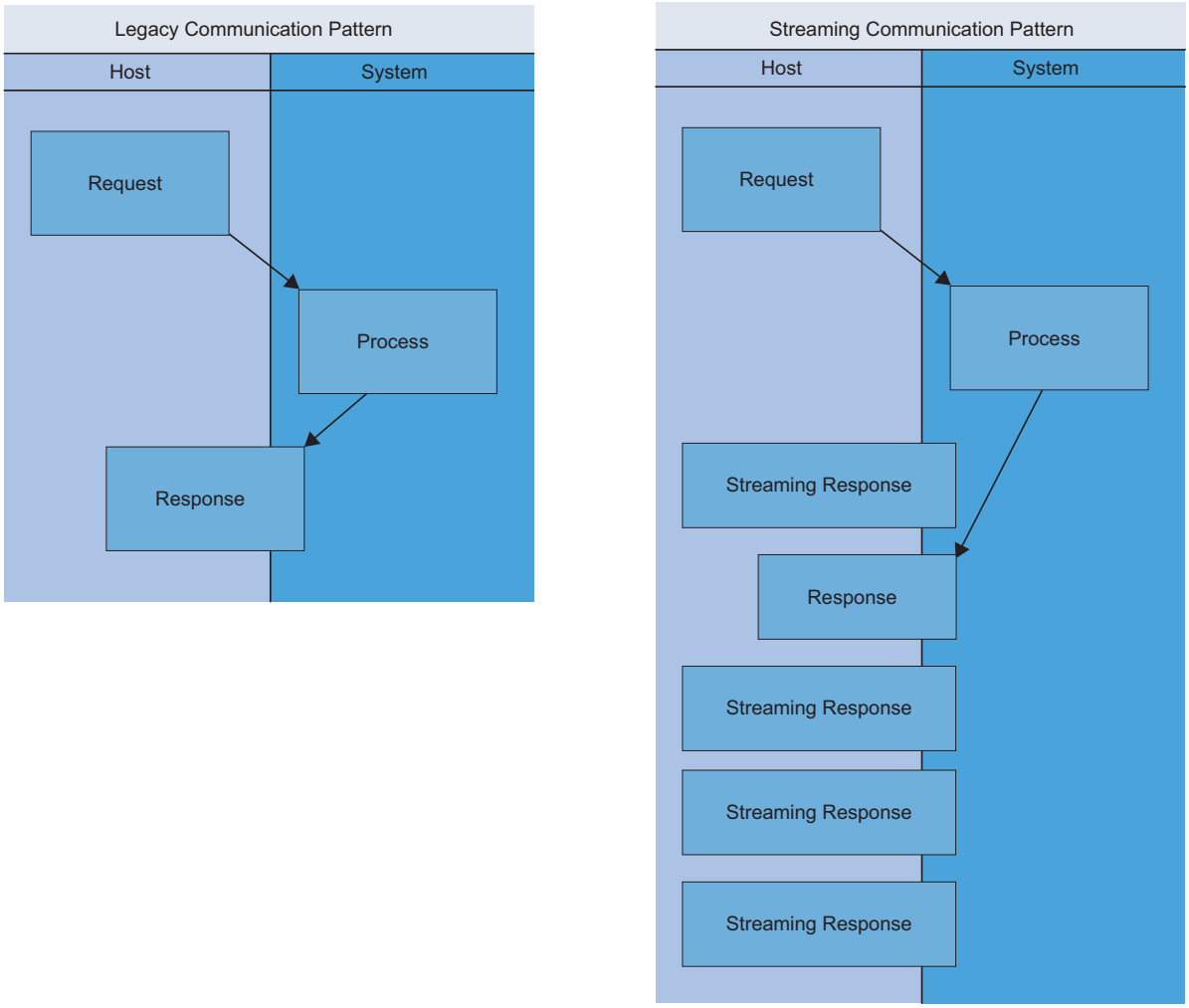
While this concept still exists in API revision G.003.006, NDI does not recommend the use of it as it complicates coordination between different host applications and can lead to unexpected behavior. Instead of using monitor connections, NDI now recommends using streaming connections as described below.

2.4 Data Streaming

With API revision G.003.001 (coinciding with the introduction of Vega) NDI introduced the concept of data streaming. Prior to this, host communication followed a strict response-request scheme where the Polaris system would send only one reply per request. With data streaming, the host can tell the

Vega system to continuously send data, without needing ongoing requests. Figure 2-2 illustrates both concepts:

Figure 2-2 Streaming Response Pattern



With API revision G.003.006 (coinciding with the release of combined firmware revision 008 for Vega), data streaming was further enhanced. Instead of the stream being limited to the existing connection, the STREAM command has been extended to allow the host to specify a new target connection. In addition, the host can now choose between UDP and TCP protocol for streaming. In noisy, unreliable environments where timeliness of data is more important than completeness, UDP streaming can provide advantages over TCP. For details, see the STREAM command and the new options "`--port`" and "`--protocol`".

Definition of Terms

Control Connection or Parent Connection

The main connection between the controlling host application and the Vega system. All setup and configuration of the Vega system is performed on this connection. It is always a TCP connection and the port on the Vega side is defined by the parameter `Param.Network.Host Port` (default = 8765).

Streaming Connections or Child Connections

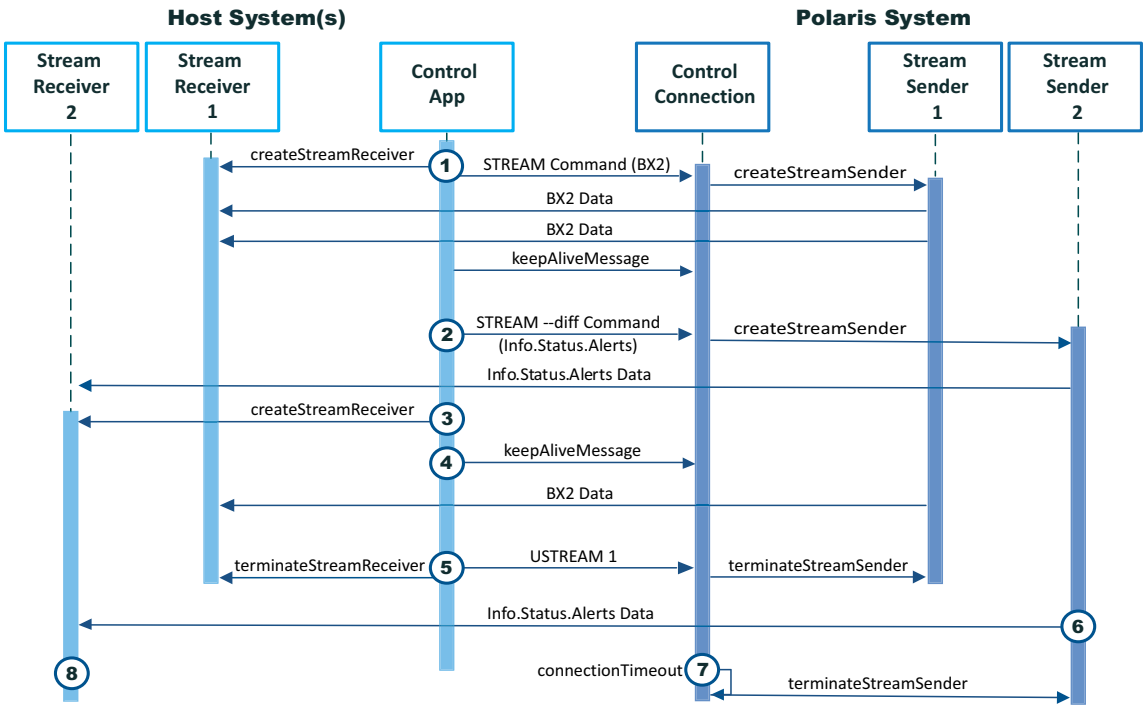
These are additional connections used for streaming data from the system to the host. All streaming connections must be initialized through the control connections and are children of the control connection. Terminating the control connection (intentionally or because of timeout) automatically terminates all streaming connections. An existing streaming connection can be terminated by the control connection at any time using the USTREAM command.

Streaming with Vega

Figure 2-3 illustrates an example of the flow of information between hosts and the Vega system using two BX2 stream requests for streaming tracking data. The diagram assumes that an initial connection between the host control application and the Vega control connection has already been established and that the Vega system is configured.

Note **For clarity, the diagram omits the acknowledgement messages from the Vega control connection to the host control application.**

Figure 2-3 Sequence Diagram



Information Flow Details

1. The host control application creates a stream receiver and issues a STREAM command to the Vega system. Upon receiving the request, Vega creates a stream sender, which immediately starts streaming the data (BX2 in this case) to the specified receiver. By default, the BX2 data is sent at the frame rate of the tracking system.

2. The host control application sends a second STREAM command with the `--diff` option to ensure Vega streams a response only if the response string is different from the previous response. This concept can be used to emulate an event-based notification scheme.

3. If the connection uses a UDP stream, the host might issue the STREAM command for Stream 2 before creating the stream receiver; however, this is not recommended. This cannot occur with TCP streams as they require a connection.
4. To ensure the control connection remains open, the host control application must periodically send traffic, such as a keepalive message, to the Vega system. If the host control application does not, the Vega system shuts down the control connection and all derived child connections (see 7).
5. The host control application issues a USTREAM command to stop Stream 1 (BX2 data). Upon receiving the request, the control connection terminates Stream Sender 1. The host control application must terminate Stream Receiver 1.
6. If the value of the `Info.Status.Alerts` parameter changes, Stream Sender 2 streams the updated parameter value.
7. If the host does not send any traffic for longer than the time specified by `Param.Connect.Idle Timeout`, the Vega system terminates the control connection and any derived child connections. The default timeout is set to 300 seconds.
8. If the connection uses UDP, the stream receiver may persist even after Vega terminates the control connection and stream sender. The host must ensure that receivers are terminated when no longer needed.

Data Streaming Considerations

The following list details several considerations when using data streaming:

- UDP does not guarantee data delivery, but can be the better option to stream tracking data. If streaming occurs over a noisy or unreliable network where packets are lost, TCP retransmits the data until the receiver acknowledges the reception. This can lead to “stale” and backed-up data. When the receiver finally gets the data, it can be several hundred milliseconds old. For tracking data this may not be acceptable. In contrast, with UDP, data may get lost, but once a datagram makes it through, it guarantees that the most recent tracking data is available at the receiver.
- A mix of TCP and UDP streaming connections is allowed if it is deemed beneficial.
- UDP streaming connections should not be used for data where the datagram size exceeds the typical size of an Ethernet packet. This is because the implemented protocol does not provide a mechanism for sorting data or detecting dropped packets. Depending on the reply options chosen, most BX2 replies will fit in a single Ethernet packet.
- NDI does not recommend streaming image capture data. The intent of image data in Vega systems is for diagnostics of scenes and should not be used in a “continuous video mode” as it can negatively impact tracking performance.
- In point-to-point networks or networks with little traffic other than the Vega data, there is no benefit to using UDP over TCP.
- You can stream directly on the control connection; however, it has the disadvantage that streaming replies and request-response replies are interleaved.
- Streaming connections (other than control connections) are always unidirectional. The stream receivers on the host side cannot send and the stream senders on the Vega side cannot receive data.
- Stream receivers can be located on a different physical machine than the host control connection. If located on a different machine, when the connection is terminated, the host

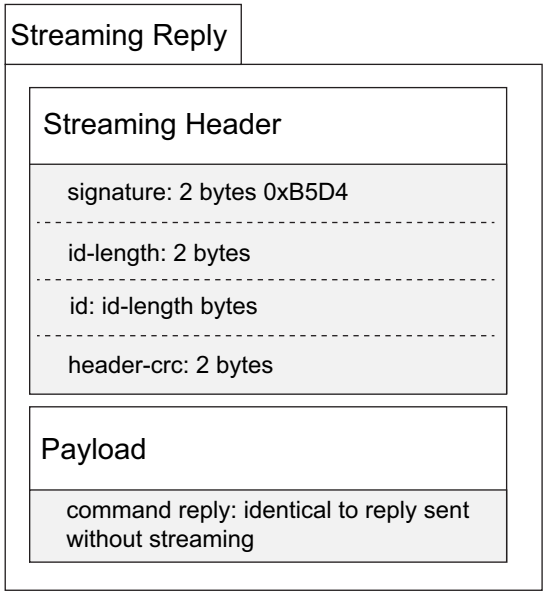
control application must be able to inform the stream receivers that the stream is terminated. Otherwise, the stream receivers might wait indefinitely for more data to arrive.

- The maximum rate of the stream replies when tracking is governed by the frame frequency, independent of the content of the type of streaming data. Actual reporting rate depends on various factors, such as frame sequence (the number of different tool types loaded, etc.), but the reporting rate will always be less than or equal to the frame frequency.

Streaming Reply Structure

Each streaming reply consists of the streaming header and the payload. Figure 2-4 shows the structure:

Figure 2-4 Streaming Reply Structure



Header fields are binary data and store numeric values in little endian byte order. The host can specify an identification string using the `--id` option. If omitted, the stream reply uses the command string to identify the stream.

The payload can be ASCII or binary data, depending on the command used. It is the exact same reply that would be sent if the command-request was done outside of a streaming command.

2.5 Data Averaging

API Revision G.003.006 introduces the concept of data averaging.

Data averaging is available only in combination with the BX2 command. The older BX and TX commands will return an error code when data averaging is enabled and the commands are issued.

Data averaging allows the user to trade data reporting frequency for a reduction in position jitter (noise). Data averaging is available for all Vega systems, but is most useful on Vega XT because of the higher frame rate achievable.

Data averaging is disabled by default, but can be enabled and configured through user parameters. For more information, see Table 4-6 on page 38.

To understand the relationship between averaging depth and reporting rate, review the following terminology:

Term	Description/Comments	Parameters
Frame Frequency (or Frame Rate)	The frequency in [Hz] at which the image sensors are read out.	`Param.Tracking.Frame Frequency` Vega ST/VT: fixed at 60 Hz Vega XT: configurable 60-400 Hz
Track Frequency	The frequency at which unaveraged tracking data is reported. The actual track frequency is influenced by: • The parameter setting • The number of different tool types loaded (active wired/active wireless/ passive) At best (with a single tool type loaded), the track frequency is equal to the frame frequency, but it can also be a fraction (1/2 or 1/3) of that.	`Param.Tracking.Track Frequency` This parameter is an enumeration whose values are derived from the frame frequency and can be: • 1/1 of Frame Frequency • 1/2 of Frame Frequency • 1/3 of Frame Frequency **Note:** If more than one tool type is used, the Track Frequency is always less than the Frame Frequency, even if the parameter is set to **1/1 of Frame Frequency**.
Reporting Frequency	The result of the actual track frequency divided by the averaging depth specified by `Param.Data Averaging.Depth`.	

How Averaging Works on Vega

On Vega, averaging is performed on 3D positions. An averaged 3D position $<\vec{p}>$ is determined by calculating the weighted arithmetic mean of all samples:

$$<\vec{p}> = \sum_{N} w_i \vec{p_i} = \begin{bmatrix} \sum w_i x_i \\ \sum w_i y_i \\ \sum w_i z_i \end{bmatrix}$$

where N is the number of samples to average and w_i are the weights for each sample. In the current implementation, Vega uses equal weights, therefore $w_i = 1/N$ for all i.

For pose tracking (6D data), this means that the data is averaged first on a 3D level (individual markers that are part of the tool) and then the pose is calculated from the averaged 3D positions. All marker filtering that happens during the pose calculation (for example out-of-volume or off-angle filtering) is also done on the averaged 3D positions.

When data averaging is enabled, data is reported only every N samples. Therefore, the relationship between track frequency f_T and reporting frequency f_R is given by: $f_R = f_T / N$.

Figure 2-5 Data Averaging - Example

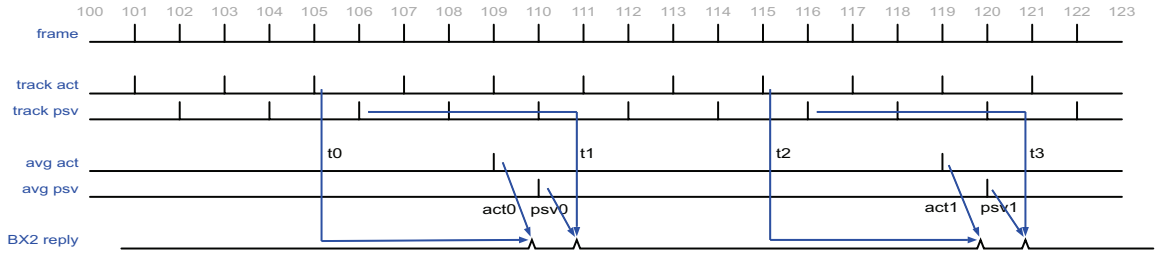


Figure 2-5 shows an example of the internal timing and how the data is reported when averaging is enabled.

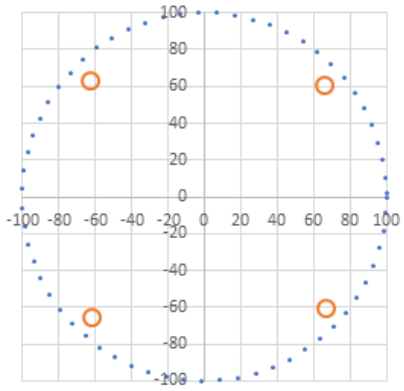
In the example, the system is set up for hybrid tracking (active and passive tools) and averaging depth is set to N=5. The “track act” and “track psv” lines respectively show how the system alternates between taking samples for active and passive tools. In the example, active tools are tracked during odd frames and passive tools are tracked during even frames. After 5 samples have been collected for a given tool type, the average is calculated (“avg act” and “avg psv”, respectively) and the data can be reported via a BX2 reply. The latency between the last sample taken and the data being reported to the host depends on the system (shorter for Vega XT than for Vega ST/VT) and the frame frequency (XT only, shorter for higher frame frequencies).

When averaging is enabled, the timestamp that is reported in the BX2 frame component is adjusted to the centre of the averaging period. In the example, the reply containing the data for “act 0” will contain the timestamp “t0”, which coincides with the time when the sample in frame 105 was taken. Similarly, the BX2 reply for “psv0” contains the timestamp “t1” coinciding with frame 106. For even numbered sample depth, the timestamp would land in the middle of two frames.

Automatic Disengagement of Averaging for Fast Moving Objects

Averaging introduces latency to the data reporting, which can be detrimental to some applications. For non-linear trajectories, averaging can, in extreme cases, introduce measurement artefacts as shown in the example below.

Figure 2-6 Measurement Artefacts - Example



In this example, an object moves on a circular trajectory of 100 mm radius with 1 m/s speed. The object is sampled at 100 Hz (blue dots) and is averaged over 16 frames (orange circles). Due to the measurement artefact, the apparent radius of the circle would be less than 90 mm.

To mitigate these effects, the `Param.Data Averaging.Threshold Distance` parameter limits the amount that a measured 3D position can deviate from its estimated position. If the distance between the measured and estimated positions exceeds the limit, averaging is temporarily disengaged. The default is 0.5 mm.

The system calculates the estimated position using linear extrapolation of the two preceding measurements.

Figure 2-7

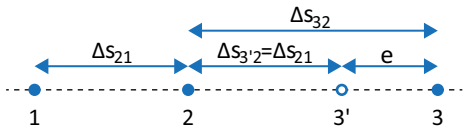


Figure 2-7 illustrates this using a marker moving in a straight line left to right with constant acceleration. Positions 1 and 2 indicate the two most recent positions, with a measured distance of Δs_{21}. Using linear extrapolation, the system estimates the marker's next position to be 3' (where $\Delta s_{3'2}$ is equivalent to Δs_{21}). However, due to its acceleration, the marker's actual measured position is 3.

The distance Δs_{21} depends on the velocity of the marker at position 1 and its acceleration.

$$\Delta s_{21} = v_1 \Delta t + \frac{1}{2} a \Delta t^2$$

The actual distance between points 2 and 3 is given by the following:

$$\Delta s_{32} = v_2 \Delta t + \frac{1}{2} a \Delta t^2 = (v_1 + a\Delta t)\Delta t + \frac{1}{2} a \Delta t^2 = v_1 \Delta t + \frac{3}{2} a \Delta t^2$$

From this, the estimated position error, e, is calculated:

$$e = \Delta s_{32} - \Delta s_{3'2} = \Delta s_{32} - \Delta s_{21} = a \Delta t^2$$

Therefore, the estimated position error depends on the acceleration and the interval between samples (and, therefore, the track frequency).

Table 2-1 on page 15 shows examples for the estimated position error at different track frequencies using acceleration values of 1 m/s and 9.81 m/s.

Table 2-1 Estimated Position Error Examples

acceleration [m/s^2]		1	9.81
f_{track} [Hz]	Δt [ms]	e [mm]	e [mm]
20	50.0	2.500	24.525
60	16.7	0.278	2.725
120	8.3	0.069	0.681
250	4.0	0.016	0.157
400	2.5	0.006	0.061

Using the default 0.5 mm threshold value, the highlighted cells show combinations where data averaging is disengaged because the estimated position error exceeds the threshold.

For the 1 m/s^2 acceleration example, data averaging generally stays engaged, except at very low track frequencies (where data averaging typically would not be used).

For the free-fall example, data averaging disengages for low to medium track frequencies, but stays engaged at high track frequencies.

Note **These examples do not consider position jitter. Position jitter is a function of the distance of the marker from the camera and can reach up to 0.2mm (RMSE) at the back of the extended pyramid volume. Keep this in mind when adjusting the value of** `Param.Data Averaging.Threshold Distance` **as too small a value may cause data averaging to disengage unintentionally due to noise.**

When averaging is disengaged, the port status in BX2 indicates that averaging is only partially completed for this particular reporting frame. Note that this is available for BX2 and the 6D component only. If the threshold distance is exceeded, 3D data reported in the 3D component of BX2 will automatically disengage averaging, but there is no status indication for that. For details, see BX2.

3 Communicating with an NDI System

3.1 Connection Requirements

The system must communicate with a host computer to pass measurement information to another application running on the host computer. Connection requirements are detailed below.

The ethernet connection must handle the bandwidth of data being sent. The bandwidth is dependent on the amount of data being requested from the Vega system.

The ethernet connection must be compliant with IEEE 802.3at type 2 and secure from any unauthorized connections.

The severity of all connection-related hazardous situations is the responsibility of the system integrator because there is no essential performance of the Vega system.

Do not connect the Polaris Vega system to a host computer or network that is not IEC 60950 and/or IEC 60601 approved. If you connect the system to a non-approved host computer or network you may increase leakage currents beyond safe limits and cause personal injury.

Note **Operation on an open or uncontrolled network could limit communication bandwidth, increase latency or otherwise interfere with the normal operation of the Vega system and introduce risks which should be analyzed. Changes to the network including connection, disconnection or updates to any equipment may also affect operation of the system.**

3.2 Communication Overview

The two methods of communication with the Vega systems are request-response and streaming. Both methods are described below.

Request-Response Communication

In request-response communication, from the application perspective, the Vega system is a serial device, which is listening for incoming commands. Upon receiving a command, the system performs some action and returns the status of this action. The system never initiates communication with the application.

Immediately after sending a command, the application can begin to poll the serial buffer for a reply. Most commands reply almost instantly. After reaching the end of the reply, the application can send another command. There may be some delay in the response of the INIT command, and the commands used to read from and write to an SROM device in a wired tool.

Streaming Communication

The Vega system introduces an option to continuously stream command responses for each new frame of data. The STREAM command initiates streaming response and the USTREAM command terminates the streaming response. For details, see “Data Streaming” on page 7.

3.3 Operating Modes

The system has three modes of operation: Setup, Tracking, and Diagnostic. Some commands will only work if they are sent while the system is in a specific mode of operation. If a command is sent when the system is in a mode not valid for that command, the system returns `ERROR0C`.

Setup

Setup mode allows you to configure the system and tools. Tasks done while the system is in Setup mode may include initializing the system, writing to the SROM on a tool, or checking the system revision.

A wireless tool must have a port handle assigned to it (PHRQ) before the application can load a tool definition file (PVWR). Both conditions must be satisfied before the tool can be enabled (PENA).

The system enters the Setup mode on successful power up, on sending a reset, or on exiting from Tracking or Diagnostic modes.

Tracking

In Tracking mode, the system measures the positions and orientations of tools in real time and returns the information to the host computer when requested. The BX2 and BX commands are the most commonly used commands in Tracking mode.

The system enters Tracking mode on successful TSTART command and exits Tracking mode on TSTOP command.

Diagnostic

Diagnostic mode allows you to control and observe active tools, but not track them.

The system enters Diagnostic mode on successful DSTART command and exits Diagnostic mode on DSTOP command.

3.4 General Syntax

Commands must be sent from the host computer to the system in one of the two following formats. To ensure the integrity of data transmission, NDI recommends using format 1, as well as verifying the returned CRC on the host computer.

Format 1

<Command><:><Parameter1><Parameter2>...<ParameterN><CRC16><CR>

A <:> must be sent with every command even if no parameters are required. There are no characters or spaces separating the parameters or the individual parts of the commands, except in user parameter names and string values used with the SET, GET, GETINFO, DFLT, and SYSLOG commands. Commands and parameters are not case-sensitive, except for user parameter names and string values used with the SET, GET, GETINFO, DFLT, and SYSLOG commands and in POSIX-style parameters (which must be separated from each other by one or more spaces).

This format requires a 16-bit CRC (Cyclic Redundancy Check) value and therefore may be more useful in application software. The application software can incorporate a CRC calculation and add it to the command each time a command is sent to the system. Including a CRC provides a communications check to ensure that there are no communication problems between the system and the host computer. The CRC is used in both the commands and replies. It is based on all the characters in the command, up to the CRC itself. It is calculated using the polynomial $x^{16} + x^{15} + x^2 + 1$. For sample code to calculate the CRC, see “Sample C Routines” on page 189.

Format 2

<Command><SPACE><Parameter1><Parameter2>...<ParameterN><CR>

A <SPACE> may be sent with every command; it need not be sent if no parameters are required. There are no characters or spaces separating the parameters or the individual parts of the commands, except in user parameter names and string values used with the SET, GET, GETINFO, DFLT, SYSLOG commands and in POSIX-style parameters (which must be separated from each other by one or more spaces). Commands and parameters are not case-sensitive, except for user parameter names and string values used with the SET, GET, GETINFO, DFLT, and SYSLOG commands.

It is not necessary to calculate a CRC value when using this format, so this format is useful for sending commands to the system in an application such as a terminal program.

3.5 Receiving System Replies

Binary Replies

Commands BX, BX2, GETLOG, and VCAP return binary replies. All other commands return ASCII replies.

If a complete command is received by the system, replies are sent back in the format:

<A5C4><Reply Length 2bytes><Header CRC 2 bytes><command reply><CRC16>

The system always returns <CRC16> in the reply regardless of whether the command was sent in format 1 or format 2 unless the reply is an Extended Binary Reply. The <Reply> will be either the requested data or ERROR<error code>. The <error code> is a two-digit hexadecimal error number. For a list of the error messages associated with error numbers, see “Error Code Definitions” on page 181.

Binary replies are returned in little endian format. For example, a 32-bit reply is returned in the format:

Reply byte	n	n + 1	n + 2	n + 3
Bits	7 - 0	15 - 8	23 - 16	31 - 24

Extended Binary Reply

In order to facilitate binary replies that have a binary payload greater than 65535 bytes long, a new binary header type is introduced. This header has a 32 bit length field and allows for reply lengths up to 2^32-1 bytes long. Either binary header may be used in response to any of the “new” binary commands, currently BX2 and VCAP.

This extended binary reply is intended for use with very large replies. If the reply length is less than 65535 bytes long, then the original binary header is used. Since TCP packets already include data checksums and to reduce processing time and allow for more efficient memory-to-memory transfer techniques, no CRC will be included in the header or at the end of the data. Thus, the extended header is the same length as the original header.

The format of an extended binary reply is as follows:

```
A5C8<4 byte Reply Length><command reply>
```

ASCII Replies

All commands return ASCII replies except BX, BX2, GETLOG, and VCAP, which return binary replies.

If a complete command is received by the system, replies are sent back in the format:

```
<Reply><CRC16><CR>
```

The system always returns <CRC16> in the reply regardless of whether the command was sent in format 1 or format 2. The <Reply> will be either the requested data, OKAY, WARNING<warning code>, or ERROR<error code>.

The <error code> is a two-digit hexadecimal error number. For a listing of all the error messages associated with error numbers, see “Error Code Definitions” on page 181.

For a listing of warning codes and definitions, see “Warning Code Definitions” on page 183.

3.6 Best Practices

This section provides guidelines on how to write an application in order to minimize updates required when there are changes to the API. If your application is written correctly, it will still work when additions are made to the API; you will only need to update your application if you wish to take advantage of the new features.

- Firmware versions prior to CFR008 were more tolerant in how API commands were formatted. For example, in firmware prior to CFR008, the following was acceptable:

  ```
  PINIT 01 \r   // whitespace between '01' and '\r'
  ```

  Firmware after CFR008 no longer accepts this and will return "ERROR07" (Invalid number of parameters). Command syntax from firmware CFR008 and later is as follows:

  ```
  PINIT 01\r   // (no space between '01' and '\r')
  ```

- Ignore the value of any returned field that is listed as “reserved” in the API guide. The values of reserved fields may change in future API releases.
- Program the application to allow all possible values of a returned field, not only the values that are currently defined. This allows for future expansion. For example, if a field returns one character, but currently only characters 0 and 1 are defined, do not write your application such that 0 and 1 are the only acceptable values; more values may be defined in the future.
- Use the frame number, and not the host computer clock, to identify when data was collected. The frame number is incremented by 1 at the base frame rate. Associating a time from the host computer clock to replies from the system assumes that the duration of time between raw data collection and when the reply is received by the host computer is constant. This is not necessarily the case. The frame number is returned with the command BX (page 69), TX (page 159), BX2 (page 77), and VCAP (page 169).
- Use both the shape type and the shape parameters to represent the characterized measurement volume graphically. There may be multiple volumes with the same shape type. All volumes of the same shape type use the shape parameters the same way. You can read the shape type and shape parameters from the `Features.Volumes.*` parameters.
- When checking the firmware revision, check only the combined firmware revision, not the firmware revision of the individual components. The combined firmware revision ensures that all components in a system have compatible firmware. To check the combined firmware revision, read the value of the user parameter `Config.Combined Firmware Revision` or use the command VER 5 (page 173). For information on reading user parameters, see “User Parameters” on page 25.
- When checking for protocol compatibility, check for the API revision instead of the combined firmware revision. An application written for a particular API revision will function with any system that supports that API revision. For details, see the command APIREV (page 67).
- Use `GET Device.*` to determine which devices are in the system configuration, instead of programming device names directly into the application. This will allow the addition or removal of devices without breaking the application. When setting or reading a user parameter value for every hardware device in the system, create a loop to repeat the action for every device name determined using `GET Device.*`. For instructions on how to determine the device names of the hardware devices in your system and how to access user parameters using device names, see “Device Names” on page 26.

- Read the timeout values of the API commands from the user parameter `Info.Timeout.<command name>`; do not program the timeout values directly into the application. For information on user parameters, see “User Parameters” on page 25.
- Do not use the system log to record minor system events. The system log is intended for major milestones only and may not have enough space to accommodate numerous minor entries. For minor entries, use the user parameters `Param.User.String0` to `Param.User.String4`. These parameters can be used for any purpose; the system does not make use of them. For example, an incoming inspection result might be a major milestone to be saved in the system log, while a cleaning schedule might be a minor entry to be saved in a user parameter. For information on these user parameters, see “User-Defined User Parameters” on page 36.

3.7 Port Handles

About Port Handles

The system assigns each tool a port handle. Using the commands below, port handles are two characters in hexadecimal format, 0x01 to 0xFF. (BX2, for example, returns port handles as 4 characters.)

Port handles can be assigned to tools only while the system is in Setup mode.

Port Handle Commands

The following commands are used for port handles:

Command	Description
PHSR (page 121)	Returns the number of assigned port handles and the port status for each one. Assigns a port handle to a wired tool.
PHRQ (page 119)	Assigns a port handle to a tool. PHRQ is followed by PVWR.
PVWR (page 137)	Assigns a tool definition file to a tool, overrides a tool definition file in a wired tool, and can be used to test a tool definition file before permanently recording the tool definition file onto the SROM device of a wired tool.
PINIT (page 124)	Initializes a port handle. PENA calls PINIT.
PHINF (page 114)	Returns port handle status, and information about the tool associated with the port handle, including physical port location.
PHF (page 113)	Releases system resources from an unused port handle. This is required if a tool is disconnected. If a tool is disconnected and then reconnected, the system assigns it a new port handle. The old handle is reported as disabled and should be freed using PHF.
PENA (page 109)	Enables reporting of transformations for a particular port handle.
PDIS (page 108)	Disables the reporting of transformations for a particular port handle.

The order in which these commands are used for wired tools is detailed in Figure 3-1 on page 23. For wireless tools, see Figure 3-2 on page 24.

Disabled Transformations

A transformation may be reported as DISABLED if:

- the port handle was not enabled with PENA (page 109),
- the port handle has been disabled with PDIS (page 108), or
- a wired tool has been disconnected and the port handle has not been freed.

Unoccupied Port Handle

A port handle may be reported as UNOCCUPIED if:

- the tool has been disconnected and port handle information is requested using PHINF (page 114), or
- you have requested a port handle with PHRQ (page 119) but you have not yet used PVWR (page 137) to associate a tool definition file with the port handle.

Flow Charts for Port Handle Usage

Figure 3-1 details the logic for using port handles with wired tools.

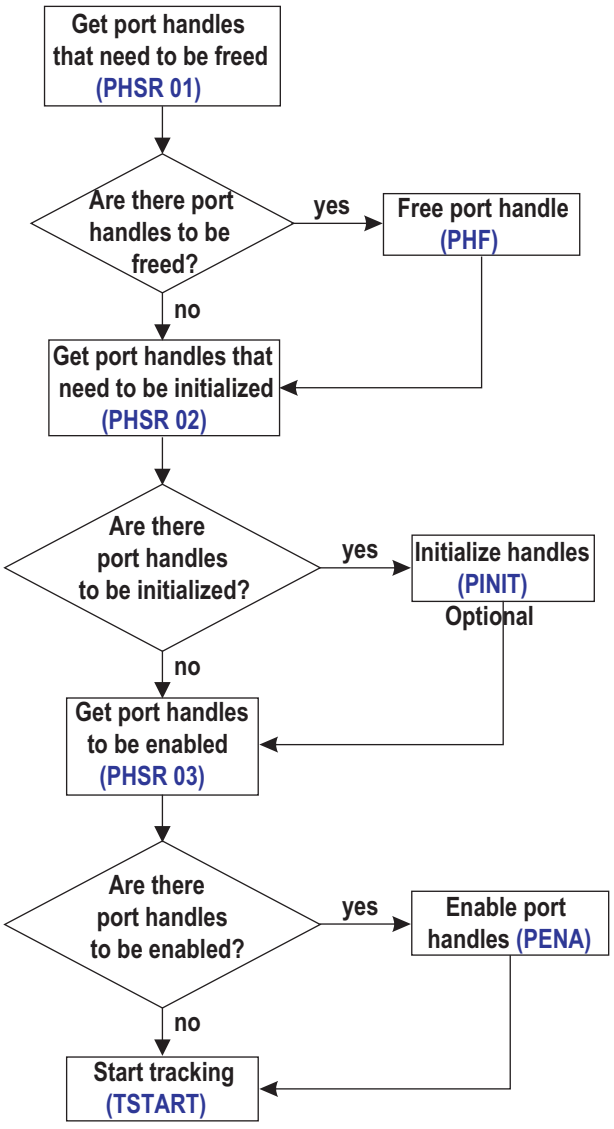


Figure 3-1 Flow Chart for Port Handle Usage - Wired Tools

Figure 3-2 details the logic for using port handles with wireless tools.

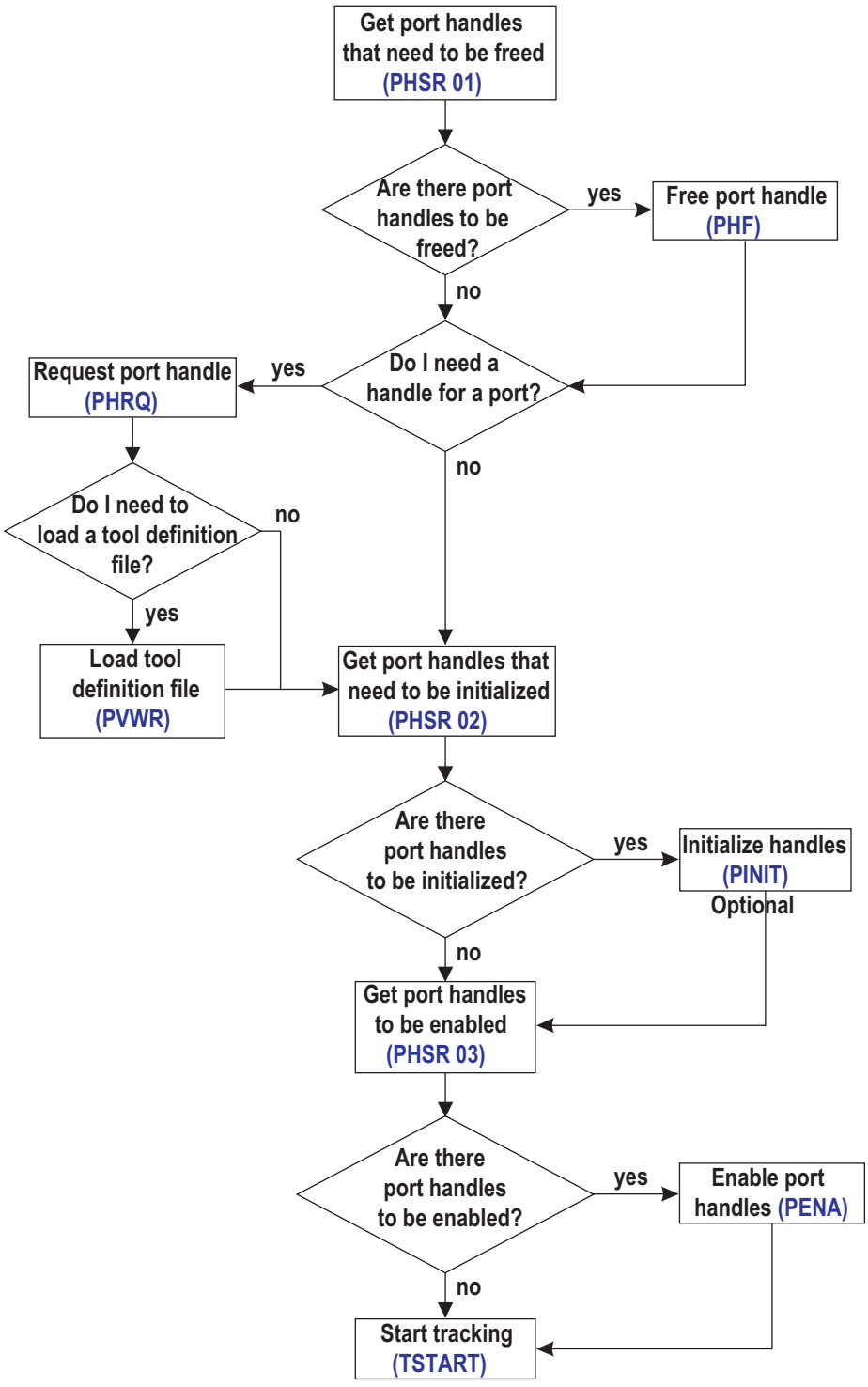


Figure 3-2 Flow Chart for Port Handle Usage - Wireless Tools

4 User Parameters

The following sections provide detailed information on the parameter system used by the Polaris Vega API:

4.1 About User Parameters

User parameters store values for different aspects of the Vega system. Some user parameters store values for the full system configuration, others store values pertaining to a particular hardware device in the system. Some user parameters are read-only parameters that store useful information about the system; some user parameter values can be changed to allow you to configure the system.

For a full list of user parameters, see page 36.

User parameters fall into the following categories:

- Image Capture User Parameters: These user parameters are used in conjunction with the VSNAP or VGET commands to store settings and values related to image capture. For example, background or illuminated frame.
- Settings User Parameters: These user parameters store settings for each hardware device in the system. For example, the settings user parameters store the illuminator rate and the available characterized measurement volumes.
- Information User Parameters: These user parameters store status information for each hardware device in the system and command timeout values.
- Features User Parameters: These user parameters store information about the features of each hardware device in the system.
- System Configuration User Parameters: These user parameters store information about the configuration of the system. These user parameters describe the configuration of the entire system, not a particular device.
- Hardware Device Information User Parameters: These user parameters store information about which hardware devices are part of the system.
- Network User Parameters: These user parameters store information about the network settings of the system.

- Clock User Parameters: These parameters store information about the system clock, including the day, month, year, hour, minutes and seconds.
- Video Camera Parameters: These parameters store information about the configuration of the optional Vega Video Camera Unit.
- Bump Sensor User Parameters: These parameters store information about the bump sensor.

4.2 User Parameter Commands

The following commands are used with the user parameters:

Command	Description
DFLT (page 89)	Restores the user parameters to factory default values.
GET (page 94)	Returns user parameter values.
GETINFO (page 96)	Returns user parameter values and descriptive information about the user parameters, including use details, possible values, and access rules.
SAVE (page 140)	Saves all non-volatile user parameters that have been changed.
SET (page 141)	Sets user parameter values.

For more details, see the individual commands.

4.3 Device Names

Each device in the system configuration has a unique device name, its own set of user parameters, and its own log file.

Note **For information on the log files, see GETLOG (page 98) and SYSLOG (page 151).**

Determining the Devices in the System Configuration

Use the GET command to determine which hardware devices are in your system. To ensure future compatibility if more devices are integrated into your system, your application should read the list of devices every time you connect to a system or whenever a component is connected or disconnected.

Note **The list of devices does not update while the system is in tracking mode. The list of devices will not show changes until the system exits tracking mode.**

The most general method of reading the list of devices to ensure consistent behaviour in the future is as follows:

Command:

```
GET Device.*
```

Reply:

```
Device.Type.0=PS
Device.Type.1=VCU
Device.Instance.0=0
Device.Instance.1=0
Device.Address.0=local
Device.Address.1=
Device.Port.0=8765
Device.Port.1=0
```

The reply returns information about every device in the system configuration and includes the four parameters shows in the example above.

- **Device.Type.X** describes the type of connected device:

Device.Type Parameter	Hardware Device
PS	Position Sensor
VCU	Video Control Unit

- **Device.Instance.X** describes the instance of that type of device in the configuration.

Parameters with the same X index value (for example, Device.Type.0 and Device.Instance.0) describe the same device. For more information, see Table 4-10 on page 50.

Constructing Device Names

To construct the device name for a particular device, use the following syntax:

```
<Device.Type.X>-<Device.Instance.X>
```

For the configuration in the example above, the device names are PS-0 and VCU-0.

Accessing User Parameters Using Device Names

To ensure that the user parameters for the correct device are accessed, prefix the parameter with the device name. All references to user parameters for a device can be made using the device name. If you omit the device name, the system defaults to the parameters for the first Position Sensor in the configuration (PS-0). To access the user parameters for a particular device, use the following syntax:

```
<Device.Type.X>-<Device.Instance.X>.<User Parameter>
```

For example, to check the frame frequency of the Position Sensor, use:

```
GET PS-0.Param.Tracking.Frame Frequency.
```

To view information about the parameters supported by the device, use the following commands:

```
GET PS-0.*
GETINFO PS-0.*
```

Note **For command details, see “GET” on page 94 and “GETINFO” on page 96.**

The system configuration user parameters (beginning with Config) and the hardware device user parameters (beginning with Device) describe the configuration of the entire system. Do not prefix these user parameters with a device name.

4.4 Alerts User Parameters

The alerts user parameters describe the status of a particular hardware device in the system.

Alerts User Parameters

Table 4-1 describes the alerts user parameters.

Table 4-1 Alerts User Parameters

User Parameter	Description
`Info.Status.Alerts`	This user parameter describes the current state of the hardware device. For Position Sensor alerts, see Table 4-2. For System Control Unit alerts, see Table 4-3. For Video Camera alerts, see Table 4-4. The bit corresponding to a particular alert is set when the system first detects the condition. This is accompanied by system response in Table 4-2 or Table 4-3. The bit is cleared when the condition no longer exists. **Note:** The “bump detected” bit is cleared only when you set the `Param.Bump Detector.Clear` Position Sensor user parameter to “1”.
`Info.Status.New Alerts`	Read this user parameter when the diagnostic pending bit is set (bit 8 in the BX or TX System Status component). When an alert is set or cleared, this user parameter lists the current alerts status. Reading this parameter clears both this parameter and the diagnostic pending bit. The bit corresponding to a particular alert is set when the system first detects the condition and is cleared when the system first detects that the condition has been resolved. This is accompanied by system response in Table 4-2 or Table 4-3. Reading this user parameter clears it.
`Param.Simulated Alerts`	Simulates the `Info.Status.Alerts` parameter, for the hardware device specified, for testing purposes. To test the response of a particular alert, set the value of this parameter to the value of the alert. See Table 4-2, Table 4-3, or Table 4-4.

Position Sensor Alerts

Table 4-2 describes the Position Sensor alerts that are returned by the `Info.Status.Alerts` and `Info.Status.New Alerts` user parameters. The returned value is an integer, which you must convert to an 8-character hexadecimal number. The hexadecimal number is made up of the following individual alert values OR'd together:

Table 4-2 Position Sensor Alerts

Hexadecimal Value	Alert	System Response	Log to File	Position Sensor LED Indication	BX2 Code
0x00000001	Non-recoverable parameter fault The system parameter file or some other critical file is missing or has been corrupted (CRC check failed).	INIT returns ERR0R15 See page 181.	yes	Error LED: on Power LED: off	Fault 1
0x00000002	Sensor parameter fault The sensor parameters were not programmed properly or cannot be read by the system. Not in use.	INIT returns ERR0R15 See page 181.	yes	Error LED: on Power LED: off	Fault 2
0x00000004	Not in use.				
0x00000008	Not in use.				
0x00000010	Illuminator voltage fault The illuminator voltage is outside of operating range. This may be caused by a hardware failure.	Sets diagnostic pending bit (bit 8) in TX or BX system status.	yes	Error LED: on Power LED: off	Fault 5
0x00000020	Illuminator current fault The illuminator current is outside of operating range. This may be caused by a hardware failure.	Sets diagnostic pending bit (bit 8) in TX or BX system status.	yes	Error LED: on Power LED: off	Fault 6
0x00000040	Left sensor temperature fault The left sensor temperature cannot be read.	INIT returns ERR0R15 See page 181. Sets diagnostic pending bit (bit 8) in TX or BX system status. The system will not return tracking data, even if reply option 0800 in TX/BX is used.	yes	Error LED: on Power LED: off	Fault 7

Table 4-2 Position Sensor Alerts (Continued)

Hexadecimal Value	Alert	System Response	Log to File	Position Sensor LED Indication	BX2 Code
0x00000080	Right sensor temperature fault The right sensor temperature cannot be read.	INIT returns ERR0R15 See page 181. Sets diagnostic pending bit (bit 8) in TX or BX system status. The system will not return tracking data, even if reply option 0800 in TX/BX is used.	yes	Error LED: on Power LED: off	Fault 8
0x00000100	Main temperature fault The main board temperature cannot be read.	INIT returns ERR0R15 See page 181. Sets diagnostic pending bit (bit 8) in TX or BX system status. The system will not return tracking data, even if reply option 0800 in TX/BX is used.	yes	Error LED: on Power LED: off	Fault 9
0x00000200	One of the image sensors on the Position Sensor is not functioning. This may be caused by an internal hardware failure.	INIT returns ERR0R15 See page 181. Sets diagnostic pending bit (bit 8) in TX or BX system status. The system will not return tracking data, even if reply option 0800 in TX/BX is used.	yes	Error LED: on Power LED: off	Fault 10
0x00000400	Reserved				
0x00010000	Firmware is running in Safe Mode. The system has not been able to boot properly two or more times in a row.	INIT returns ERR0R15 See page 181.	no	Error LED: flashing Power LED: on	Alert 17
0x00020000	System Control Unit fault or alert detected. There is a fault or alert present in the System Control Unit's Info.Status.Alerts parameter.	INIT returns 0KAY Sets diagnostic pending bit (bit 8) in TX or BX system status.	no	Error LED: on Power LED: on	Alert 16

Table 4-2 Position Sensor Alerts (Continued)

Hexadecimal Value	Alert	System Response	Log to File	Position Sensor LED Indication	BX2 Code
0x00040000	A Type 1 low power PSE is detected and there is insufficient power for tracking. A Type 2 PSE compliant with the 802.3at standard that outputs up to 30W must be used.	INIT returns ERR0R15 See page 181.	no	Error LED: on Power LED: on	N/A
0x00080000	System Control Unit is configured but not present. This may mean that the System Control Unit is not functioning, has not powered up yet, or has been disconnected. It can also mean that the system is misconfigured. Active tools will not be available for tracking.		no	Error LED: on Power LED: on	Alert 10
0x00100000	System battery fault The system battery power is too low. This may be caused by a depleted or disconnected battery. This battery powers the bump sensor and the system clock.	Sets diagnostic pending bit (bit 8) in TX or BX system status. Need reply option 0800 in TX or BX to return data.	yes	Error LED: on Power LED: on	Alert 1
0x00200000	Bump detected The bump sensor has detected a bump.	Sets diagnostic pending bit (bit 8) in TX or BX system status. Need reply option 0800 in TX or BX to return data.	yes	Error LED: on Power LED: on	Alert 2
0x00400000	Video camera not functioning. The video camera unit signals a fault or fails to respond to the Position Sensor. If this alert is raised you must restart the system.		yes	Error LED: on Power LED: on	Alert 15
0x00800000	Incompatible firmware. The combination of firmware on the Position Sensor is not compat ible. This may be caused by a failed attempt to update the firmware.	INIT returns ERR0R2E See page 181.	yes	Error LED: flashing Power LED: on	Alert 3
0x01000000	Recoverable parameter fault. The user parameter file has been corrupted (CRC check failed) or is missing. To correct this problem, check that the settings of the user parameters are set correctly and save them using SAVE (page 140).	INIT returns ERR0R15 See page 181.	yes	Error LED: on Power LED: on	Alert 4

Table 4-2 Position Sensor Alerts (Continued)

Hexadecimal Value	Alert	System Response	Log to File	Position Sensor LED Indication	BX2 Code
0x02000000	Not in use.				
0x08000000	PTP clock is not synced. Position Sensor's PTP clock is not synced with other devices on the same network.		no	Error LED: on Power LED: on	Alert 14
0x20000000	The Position Sensor temperature is above the recommended operating range. See the user guide for details.	Sets temperature bit (bit 9) in TX or BX system status. Need reply option 0800 in TX or BX to return data.	no	Error LED: on Power LED: on	Alert 8
0x40000000	The Position Sensor temperature is below the recommended operating range.See the user guide for details.	Sets temperature bit (bit 9) in TX or BX system status. Need reply option 0800 in TX or BX to return data.	no	Power LED: flashes during warm-up when system is first powered on. Error LED: on	Alert 9
0x80000000	Reserved				

System Control Unit Alerts

Table 4-3 describes the System Control Unit alerts that are returned by the `Info.Status.Alerts` and `Info.Status.New Alerts` user parameters. The returned value is an integer, which you must convert to an 8-character hexadecimal number. The hexadecimal number is made up of the following individual alert values OR'd together:

Note **The Vega System Control Unit only incorporates one (Status) LED located on the rear of the System Control Unit.**

Table 4-3 System Control Unit Alerts

Hexadecimal Value	Alert	System Response	Log to File	System Control Unit LED Indication
0x00000001	Non-recoverable parameter fault. The system parameter file or some other critical file is missing or has been corrupted (CRC check failed).	INIT returns ERR0R15 See page 181.	yes	Rear LED: amber
0x00000002 to 0x00000008	Reserved			
0x00000010	Internal strober communication fault. The System Control Unit can detect the internal strober, but cannot communicate with it.	Sets diagnostic pending bit (bit 8) in TX or BX system status.	yes	Rear LED: amber
0x00000020 to 0x00000040	Reserved			
0x00000080	Not in use.			
0x00000100	Strober fault raised. The System Control Unit has detected a fault raised by the strober. There could be a voltage monitor fault or an active marker current monitor fault.	INIT returns ERR0R15 See page 181.	yes	Rear LED: amber
0x00000200 to 0x00400000	Reserved			

Table 4-3 System Control Unit Alerts (Continued)

Hexadecimal Value	Alert	System Response	Log to File	System Control Unit LED Indication
0x00010000	Firmware is running in Safe Mode. The system has not been able to boot properly two or more times in a row.	INIT returns ERR0R15 See page 181.	no	Rear LED: amber flash
0x00800000	Incompatible firmware. The combination of firmware on the System Control Unit is not compatible. This may be caused by a failed attempt to update the firmware.	INIT returns ERR0R2E See page 181.	yes	Rear LED: amber flash
0x01000000	Recoverable parameter fault The user parameter file has been corrupted (CRC check failed) or is missing. To correct this problem, check that the settings of the user parameters are set correctly, and save them (use SAVE (page 140)).	INIT returns ERR0R15 See page 181.	yes	Rear LED: amber flash
0x02000000	Not in use.			
0x04000000	Reserved			
0x08000000	PTP clock not synced. The System Control Unit's PTP clock is not synced with other devices on the same network.		no	Rear LED: amber flash
0x10000000	System Control Unit fan not functioning as expected.		no	Rear LED: amber flash
0x20000000	System Control Unit battery voltage low. This may be caused by a depleted or disconnected battery.		yes	Rear LED: amber flash
0x40000000	Strober alert raised. A strober parameter is missing.		yes	Rear LED: amber flash
0x80000000	Not in use.			

Video Camera Alerts

Table 4-4 describes the Video Camera Unit alerts the `Info.Status.Alerts` and `Info.Status.New.Alerts` user parameters return. When connected directly to the Video Camera Unit (port 8766), no prefix is required. If connected through the Position Sensor (port 8765),

use the prefix “VCU-0.”. The returned value is an integer, which you must convert to an 8-character hexadecimal number. The hexadecimal number is made up of the following individual alert values OR’d together:

Table 4-4 Video Camera Alerts

Hexadecimal Value	Alert	System Response	Log to File	PSU LED Indication
0x00000001 0x00000002	Internal video camera error	If this condition persists for more than 10 seconds, the system reboots the video camera.	no	None

4.5 Complete List of User Parameters

The following tables list the user parameters for the Vega system. To view a complete list of user parameters for your system, use the command GET * (for parameter names and values) or GETINFO * (for parameter names, values, and usage details).

User-Defined User Parameters

You can use the user parameters `Param.User.String0` to `Param.User.String4` to store user-defined information; the system does not make use of them. For example, use them to keep track of the system maintenance or cleaning schedule.

Image Capture User Parameters

The following user parameters are used in conjunction with the VSNAP or VGET commands. These parameters apply to the Position Sensor only.

Table 4-5 Image Capture User Parameters used with VSNAP or VGET

User Parameter	Description	Access Rules
`Cmd.VGet.Color Depth`	Image capture returned bits per pixel.	RW

Table 4-5 Image Capture User Parameters used with VSNAP or VGET (Continued)

User Parameter	Description	Access Rules
`Cmd.VGet.Compression`	Image capture returned data compression.	RW
`Cmd.VGet.End X`	Image capture end column.	RW
`Cmd.VGet.Sample Option`	Image capture sample option for the stride. For details, see VGET.	RW
`Cmd.VGet.Sensor.Color Depth`	Number of bits per pixel on the image sensors.	R
`Cmd.VGet.Sensor.Height`	Number of vertical pixels on the image sensors.	R
`Cmd.VGet.Sensor.Number`	Number of horizontal image sensors in the tracking system.	R
`Cmd.VGet.Sensor.Width`	Number of horizontal pixels on the image sensors.	R
`Cmd.VGet.Start X`	Image capture start column.	RW
`Cmd.VGet.Stride`	Image capture horizontal pixel step.	RW
`Cmd.VGet.Threshold.Background`	Background suppression threshold [% full scale].	R
`Cmd.VGet.Threshold.Shutter Time`	Exposure time for threshold calculations [usec].	RW

Table 4-5 Image Capture User Parameters used with VSNAP or VGET (Continued)

User Parameter	Description	Access Rules
`Cmd.VGet.Threshold.Trigger`	Spot detection trigger threshold [% full scale].	R
`Cmd.VSnap.Background Frame`	Forces the collection of a background frame with illuminators off. Can be set in Setup mode only.	RW
`Cmd.VSnap.Frame Types`	Enumeration of tool classes reported in a frame sequence.	R
`Cmd.VSnap.Illuminated Frame`	Forces the collection of a frame with illuminators on. Can be set in Setup mode only.	RW
`Cmd.VSnap.Manual Shutter`	Exposure time for illuminated and background frames [usec]	RW

Settings User Parameters

The following user parameters store settings for indicated hardware devices.

Table 4-6 System Settings User Parameters

User Parameter	Description	Access Rules	Devices
`Param.Data Averaging.Depth`	Number of samples to be averaged (binned) Allowed range is 1 to 16	RWS	PSU
`Param.Data Averaging.Threshold Distance`	Maximum sample-to-sample distance of a measured 3D position before averaging is stopped Allowed range is 0.1 mm to 10 mm	RWS	PSU

Table 4-6 System Settings User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
`Param.Default Wavelength.Return Warning`	Enables/disables returning a warning on PINIT if the default wavelength was selected for the tool corresponding to the port handle.	RW	PSU
`Param.Exposure.Shutter Time.Other`	Exposure time for illuminated and background frames [us].	RWS	PSU
`Param.Exposure.Time Slot.Passive`	Time slot within the frame, to coordinate multiple position sensors. The exposure time of the position sensor in slot 2 will be shifted so it does not illuminate at the same time as other position sensors.	RW	PSU
`Param.Laser.Laser Status`	Starts/stops firing the positioning laser. Use this parameter when the Positioning Laser keyed feature is enabled. For details, See “Positioning Laser” on page 187. The laser turns off automatically after 35 s.	RW	PSU
`Param.Laser.Switch Status`	Indicates the status of the external laser switch port. Values: • 0: Open • 1: Closed	R	PSU
`Param.Simulated Alerts`	Simulates the `Info.Status.Alerts` parameter for testing purposes.	RWS	PSU SCU
`Param.Serial Break Option`	Desired action upon a serial break. Only used for serial communications.	RWS	PSU
`Param.System Beeper`	Enables/disables the beeper sequence on system reset.	RWS	PSU SCU
`Param.Tracking.Available Volumes`	Available characterized measurement volumes.	R	PSU

Table 4-6 System Settings User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
Param.Tracking.Background Frame	Forces the collection of a background frame with illuminators off. Takes effect on next DSTART or TSTART.	RW	PSU
Param.Tracking.Frame Frequency	The configured frame frequency, in Hz. Can be set in Setup mode only.	RWS	PSU
Param.Tracking.Frame Sequence	Sequence of frame types currently being tracked.	R	PSU
Param.Tracking.Illuminated Frame	Forces the collection of a frame with illuminators on. Takes effect on next DSTART or TSTART.	RW	PSU
Param.Tracking.Selected Volume	Selects a characterized measurement volume. Can be set in Setup mode only.	Prior to firmware CFR007: RW Firmware CFR007 and later: RWS	PSU
Param.Tracking.Sensitivity.Active	Background IR sensitivity level (1-lowest, 7-highest) for wired active tools. Can be set in Setup mode only.	RWS	PSU
Param.Tracking.Sensitivity.Active Wireless	Background IR sensitivity level (1-lowest, 7-highest) for active wireless tools. Can only be set in Setup mode.	RWS	PSU

Table 4-6 System Settings User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
`Param.Tracking.Sensitivity.Passive`	Background IR sensitivity level (1-lowest, 7-highest) for wireless passive tools. Can only be set in Setup mode.	RWS	PSU
`Param.Tracking.Track Frequency`	The configured tool tracking frequency. Values: • 0: 1/3 of the frame frequency • 1: 1/2 of the frame frequency • 2: 1/1 of the frame frequency Can be set in Setup mode only.	Prior to firmware CFR007: RW Firmware CFR007 and later: RWS	PSU
`Param.User.String0`	User-defined string (up to 63 chars).	RWS	PSU SCU
`Param.User.String1`	User-defined string (up to 63 chars).	RWS	PSU SCU
`Param.User.String2`	User-defined string (up to 63 chars).	RWS	PSU SCU
`Param.User.String3`	User-defined string (up to 63 chars).	RWS	PSU SCU

Table 4-6 System Settings User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
`Param.User.String4`	User-defined string (up to 63 chars).	RWS	PSU SCU
`Param.Video Camera.Direct Connection`	Determines whether port 8766 should be opened for direct API communication to the video camera.	RWS	PSU
`Param.Video Camera.PSU Control`	Determines whether the video camera should be logically tied to the Position Sensor or treated as a separate device. When enabled, the video camera is registered in the Position Sensor's device table. The following commands sent to the Position Sensor are relayed to the VCU: `INIT`, `DFLT *`, SAVE, `GET *`, `GETINFO *`, `VCAP`. VCU parameters may be accessed via the Position Sensor by using a "VCU-0." prefix. When disabled, the video camera is not registered in the Position Sensor's device table. The commands listed above are not relayed to the video camera and video camera parameters cannot be accessed via the Position Sensor. **Note:** Regardless of this setting, if a RESET 0/1 is issued to the Position Sensor, it is always relayed to the video camera.	RWS	PSU

Information User Parameters

The following user parameters store status information for the hardware devices indicated in the Hardware Device column, and command time out values.

Table 4-7 Information User Parameters

User Parameter	Description	Access Rules	Devices
`Info.Partitions.Update In Progress`	Indicates that a firmware upgrade is in progress.	R	PSU, SCU, VCU
`Info.Status.Alerts`	System hardware and operating status flags. For details, see "Alerts User Parameters" on page 28.	R	PSU, SCU, VCU
`Info.Status.Bits Per Second`	The data rate of the video camera in bits per second, when streaming video.	R	VCU
`Info.Status.Bump Detected`	Indicates if the system has detected a bump.	R	PSU
`Info.Status.Frames Per Second`	The current frame rate of the video camera in frames per second when streaming video.	R	VCU
`Info.Status.Gravity Vector`	Gravity directional vector reported in Position Sensor coordinate space.	R	PSU
`Info.Status.New Alerts`	System hardware and operating status flags. For details, see "Alerts User Parameters" on page 28.	R	PSU, SCU, VCU

Table 4-7 Information User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
Info.Status.New Log Entry	Indicates a new system log entry has been made. To clear, set to 'False' (0).	RW	PSU, SCU
Info.Status.PTP.Clock State	PTP Clock Master/Slave state.	R	PSU, SCU
Info.Status.PTP.Master Offset	PTP Clock master offset in µs.	R	PSU, SCU
Info.Status.PTP.Sync State	PTP Clock sync state.	R	PSU, SCU
Info.Status.System Mode	System operating mode.	R	PSU
Info.Status.Tracking.Measured Frame Frequency	The actual frame rate at which the system takes images of the measurement volume, in Hz.	R	PSU
Info.Status.Tracking.Measured Track Frequency	The actual tracking rate at which the system reports transformations for all the tools being tracked, in Hz.	R	PSU
Info.Timeout.<command>	Time out for the specified command (sec). For the SCU, only the following commands have timeout values: APIREV, COMM, DFLT, ECHO, GET, GETINFO, GETLOG, INIT, SYSLOG, RESET, SAVE, SET, VER.	R	PSU, SCU

Table 4-7 Information User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
`Info.Video Connections.Address.0`	The IP address of the client that is receiving the video image stream. Only valid when streaming.	R	VCU

Features User Parameters

The following user parameters store information about the features for the indicated hardware devices.

Table 4-8 Features User Parameters

User Parameter	Description	Access Rules	Devices
`Features.Firmware.API Revision`	Current API revision information.	R	PSU SCU VCU
`Features.Firmware.Available Combined Firmware Revisions`	List of combined firmware revisions loaded in the device.	R	PSU SCU
`Features.Firmware.Bootloader Version`	Current bootloader revision number.	R	PSU SCU VCU
`Features.Firmware.Build Number`	Current firmware build revision number.	R	PSU SCU VCU

Table 4-8 Features User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
`Features.Firmware.Combined Firmware Revision`	Current combined firmware revision of the device.	R	PSU SCU VCU
`Features.Firmware.Major Version`	Current firmware major revision number.	R	PSU SCU VCU
`Features.Firmware.Maximum Versions`	Number of firmware revisions that may be stored in the device simultaneously.	R	PSU SCU
`Features.Firmware.Minor Version`	Current firmware minor revision number.	R	PSU SCU VCU
`Features.Firmware.Package Number`	Current firmware package number.	R	PSU SCU VCU
`Features.Firmware.Safeloader Version`	Current safeloader firmware revision number.	R	PSU SCU VCU

Table 4-8 Features User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
Features.Firmware.Version	Current firmware revision number.	R	PSU SCU VCU
Features.Hardware.Characterization Date	Date of characterization.	R	VCU
Features.Hardware.Manufacture Date	Date of manufacture.	R	PSU SCU VCU
Features.Hardware.Main Board Revision	Functional revision number of main board.	R	PSU
Features.Hardware.Model	Name of hardware device model.	R	PSU SCU VCU
Features.Hardware.Modules.Date.<idx>	Manufacturing date system sub-component.	R	PSU
Features.Hardware.Modules.Part Number.<idx>	Part number of system sub-component.	R	PSU
Features.Hardware.Modules.Serial Number.<idx>	Serial number of system sub-component.	R	PSU

Table 4-8 Features User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
Features.Hardware.Modules.Type.<idx>	Type of system sub-component.	R	PSU
Features.Hardware.OEM Number	Hardware device customer number.	R	PSU SCU VCU
Features.Hardware.Part Number	Product part number.	R	PSU SCU VCU
Features.Hardware.Serial Number	Hardware device serial number.	R	PSU SCU VCU
Features.Keys.Active Keys.<idx>	List of active feature keys. For details, see page 185.	R	PSU SCU VCU
Features.Keys.Disabled Keys	Comma-separated list of disabled keys; change takes effect on next reset. For details, see page 185.	RWS	PSU SCU VCU

Table 4-8 Features User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
Features.Keys.Installed Keys.<idx>	'Value' is the name of the installed feature.	R	PSU SCU VCU
Features.Tools.Active Ports	Maximum number of wired active tools that can be enabled simultaneously.	R	PSU
Features.Tools.Enabled Tools	Maximum number of tools that can be enabled simultaneously.	R	PSU
Features.Tools.Passive Ports	Maximum number of passive tools that can be enabled simultaneously.	R	PSU
Features.Tools.Wireless Ports	Maximum number wireless active tools that can be enabled simultaneously.	R	PSU
Features.Volumes.Index.<idx>	Indicates the volume that is being referred to.	R	PSU
Features.Volumes.Name.<idx>	The volume name.	R	PSU
Features.Volumes.Param<n>.<idx>	Shape parameters as described in SFLIST, where n is between 0 to 9.	R	PSU
Features.Volumes.Shape.<idx>	The shape type.	R	PSU

System Configuration User Parameters

The following user parameters store information about the configuration of the system. These user parameters describe the configuration of the entire system, not a particular device.

Table 4-9 System Configuration User Parameters

User Parameter	Description	Access Rules
`Config.Combined Firmware Revision`	Current combined firmware revision of the system.	R
`Config.Multi Firmware.Available Combined Firmware Revisions`	List of combined firmware revisions loaded in the system.	R
`Config.Multi Firmware.Load Combined Firmware Revision`	Combined firmware revision to load on next reset (selection automatically saves when set). Use this parameter when the Multi Firmware keyed feature is enabled. For details, see “Multi Firmware Feature” on page 186.	RW
`Config.Multi Firmware.Update Combined Firmware Revision`	Combined firmware revision to replace on next upgrade or downgrade. Use this parameter when the Multi Firmware keyed feature is enabled. For details, see “Multi Firmware Feature” on page 186.	RWS

Hardware Device Information User Parameters

The following user parameters store information about the hardware devices in the system. For information on how to use the hardware device user parameters, see “Device Names” on page 26.

Table 4-10 Hardware Device User Parameters

User Parameter	Description	Access Rules
`Device.Address.<idx>`	The network address of the device (or “local” if that is the device you are talking to).	R

Table 4-10 Hardware Device User Parameters (Continued)

User Parameter	Description	Access Rules
`Device.Instance.<idx>`	Instance of this type of device in the system configuration.	R
`Device.Port.<idx>`	The network port to connect to the device.	R
`Device.Type.<idx>`	Type of device in the system configuration.	R

Network User Parameters

The following user parameters store information about the system network settings.

Table 4-11 Network User Parameters

User Parameter	Description	Access Rules	Devices
`Info.Connect.Index`	Index of this connection in connection table.	R	PSU SCU VCU
`Info.Connect.isAuth`	Indicates if this connection is authenticated.	R	PSU SCU VCU
`Info.Connect.isMaster`	Indicates if this connection is master.	R	PSU SCU VCU
`Info.Connections.Address.<idx>`	Remote IP address.	R	PSU SCU VCU
`Info.Connections.isAuth.<idx>`	Indicates if this connection is authenticated.	R	PSU SCU VCU

Table 4-11 Network User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
Info.Connections.isMaster.<idx>	Indicates if this connection is master.	R	PSU SCU VCU
Info.Connections.Port.<idx>	Remote IP port.	R	PSU SCU VCU
Info.Connections.Replies.<idx>	Number of API replies.	R	PSU SCU VCU
Info.Connections.Requests.<idx>	Number of API requests.	R	PSU SCU VCU
Info.Connections.Rx Bytes.<idx>	Received byte count.	R	PSU SCU VCU
Info.Connections.Streams.<idx>	Number of active streams.	R	PSU SCU VCU

Table 4-11 Network User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
Info.Connections.Tx Bytes.<idx>	Transmitted byte count.	R	PSU SCU VCU
Param.Connect.Idle Timeout	Seconds of inactivity before a connection is automatically closed (0 = never).	RWS	PSU SCU VCU
Param.Connect.Master Hosts	Comma-separated list of hosts allowed to become configuration masters. If left blank, any host can become a configuration master.	RWS	PSU SCU VCU
Param.Connect.Master Timeout	Seconds of inactivity before another connection is allowed to become master. If 0, another connection can never become master.	RWS	PSU SCU VCU
Param.Connect.Monitor Hosts	Comma-separated list of hosts allowed to connect. If “none”, there are no monitor hosts. If left blank, access is unrestricted.	RWS	PSU SCU VCU
Param.Connect.SCU Hostname	Host name or address for SCU connection.	RWS	PSU
Param.Connect.SCU Port	TCP port for SCU connections.	RWS	PSU

Table 4-11 Network User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
`Param.Network.Host Name`	Ethernet interface hostname. The default is blank.	RWS	PSU SCU
`Param.Network.Host Port`	Ethernet interface port number. The default is 8765.	RWS	PSU SCU
`Param.Network.IP Method`	Method of receiving IP address.	RWS	PSU SCU
`Param.Network.MAC Address`	Ethernet interface MAC address.	R	PSU SCU
`Param.Network.Service Name`	Service name advertised in DNS-SD. Values: • m: model • h: host name • n: serial number • t: tracking group.	RWS	PSU SCU
`Param.Network.Static.Gateway`	Requested interface IPv4 gateway.	RWS	PSU SCU
`Param.Network.Static.IP Address`	Requested ethernet interface IPv4 address.	RWS	PSU SCU

Table 4-11 Network User Parameters (Continued)

User Parameter	Description	Access Rules	Devices
`Param.Network.Static.Subnet Mask`	Requested ethernet interface IPv4 subnet mask.	RWS	PSU SCU

Clock User Parameters

The following user parameters store information about the system clock.

Table 4-12 Clock User Parameters

User Parameter	Description	Access Rules	Devices
`Param.Clock.Date Time`	System's real time clock (RTC) formatted according to ISO 8601 (UTC)	RW	PSU SCU
`Param.Clock.Time`	System time in seconds since epoch, with fractional seconds. Initialized by the RTC clock during boot. If the system operates as a PTP slave, the system time is synchronized to the PTP Master clock and the values of `Param.Clock.Time` and `Param.Clock.Date Time` will differ.	R	PSU SCU

Video Camera Parameters

The following user parameters store information about the optional Vega Video Camera Unit. For details on the video camera, see the user guide that accompanied your system.

Note **When issuing commands for VCU parameters through the Position Sensor, the prefix "VCU-0." is required before the parameter. For example, when** `Param.Video Camera.PSU Control` **= Enabled. The prefix "VCU-0." is not required when connected directly to the VCU on port 8766. For example, when** `Param.Video Camera.Direct Connection` **= Enabled.**

Table 4-13 Video Camera User Parameters

User Parameter	Description	Access Rules
`Param.Allow Streaming`	Enables or disables video output when a direct connection to the video camera is enabled. Video output is disabled by default.	RWS
`Param.[Binning X | Binning Y]`	The binning factor in the X and Y direction used to produce the image. The 1024x768 resolution uses binning.	R
`Param.Disconnect Clients`	Set to “1” to force the current streaming client to disconnect.	W
`Param.Effects.Brightness`	An integer added to all of the pixels equally. This is not based on color.	RWS
`Param.Effects.Color Saturation`	A multiplier to control the color saturation. A value of 0 creates a black and white image, while a value greater than 1 increases the intensity of the colors.	RWS
`Param.Effects.Contrast`	A floating point gain multiplier and is applied to all colors.	RWS
`Param.Effects.Edge Enhancement`	Enable/Disable the edge enhancement. 0= disabled, 1=enabled.	RWS

Table 4-13 Video Camera User Parameters (Continued)

User Parameter	Description	Access Rules
Param.Effects.Edge Enhancement Sharpness	Controls the sharpness of the edge enhancement algorithm. Edge Enhancement is applied to the Luminance (Y) or black and white component of the image. Edges in the image are detected and the luminance is adjusted to make the edges appear sharper.	RWS
Param.Effects.Gamma Correction	Gamma correction applies a non-linear gain to increase the gain in dark areas of an image while keeping the light areas relatively untouched. Additionally, the user can select a user defined mode and specify their own gamma correction value. Values: • **Off:** No Gamma correction • **Built-in:** Use built-in hardware gamma correction • **User Defined:** Uses the Gamma Value to create a correction table • **User-Defined Enhanced**: Uses the Gamma Value to create a correction table For details on the Gamma Correction and Gamma Value parameters, see “Gamma Correction” on page 200.	RWS
Param.Effects.Gamma Value	The value used to generate the User-Defined (i.e. output = input(1/gamma)) or User-Defined Enhanced (i.e. output = input(1/gamma) * (1 – (3/(input+3))+input*0.01) gamma curve.	RWS
Param.Effects.Noise Filter	Turns the noise filter on or off. 0= disabled, 1=enabled.	RWS
Param.Effects.Noise Filter Smoothness	Controls the amount of smoothing provided by the noise filter.	RWS
Param.Exposure And Gain.Auto	Enable/disables the auto exposure/gain mode.	RWS
Param.Exposure And Gain.Auto Brightness Target	The desired level of brightness of the image. Expressed as a percentage of the maximum pixel value.	RWS

Table 4-13 Video Camera User Parameters (Continued)

User Parameter	Description	Access Rules
Param.Exposure And Gain. Auto Maximum Exposure	The maximum exposure at which point the system starts increasing gain. Lower values limit motion-blur at the cost of increased noise.	RWS
Param.Exposure And Gain.Exposure Time	Exposure time in microseconds. While auto exposure/gain is enabled, this parameter is read-only.	RWS
Param.Exposure And Gain.Meter.Mode	Choose either Weighted ROI, Segmented, Spotlight, or Center Weighted. For detailed information about each node, see “Exposure and Gain Control - Meter Modes” on page 201.	RWS
Param.Exposure And Gain.Meter.ROI [Center X|Center Y|Width|Height]	Defines the rectangle of interest (Region 0) when using Weighted ROI metering method. The Width and Height are used to define the size of Region 3 when using Segmented metering method. These values are ignored in Spotlight or Center Weighted metering modes. All values are expressed as a percentage of the resolution size.	RWS
Param.Exposure And Gain.Meter.ROI Pixels [Left|Top|Width|Height]	Returns the pixel locations of the actual rectangle of interest when using the Weighted ROI and Segmented metering methods.	RWS
Param.Exposure And Gain.Meter.ROI Weighting	The percentage weighting applied to Region 0 when using Weighted ROI metering method. The remaining percentage is applied to Region 1.	RWS
Param.Exposure And Gain.System Gain	Each gain component can range from 0 to 128 in increments of 0.125 if gain is between 1 and 4, 0.250 if gain is between 4.25 and 8, and 1.000 if gain is between 8 and 128. While auto exposure/gain is enabled, this parameter is read-only.	RWS
Param.Frame Rate	The sensor's output frame rate in frames per second.	RWS
Param.[Left | Top]	Different resolutions use different areas of the image sensor to produce a video. These parameters report the left and top offset of the sensor area.	R
Param.Lens.6D.[q0|qx|qy|qz|tx|y|z]	Describes the location (in mm) and orientation (quaternion) of the video camera in measurement coordinate space.	RWS

Table 4-13 Video Camera User Parameters (Continued)

User Parameter	Description	Access Rules
`Param.Lens.Distortion.[k1|k2|k3|p1|p2]`	Lens distortion parameters for Zhang's method (in OpenCV).	RWS
`Param.Lens.Pinhole.[U0,V0,fx,fy]`	Lens pinhole parameters for the currently selected resolution. These parameters are used to relate the 3D scene into 2D sensor space. U0[x pixels], V0[y pixels] , fx[mm], fy[mm]	RWS
`Param.Resolution`	One of: • 1024x768 • 1920x1088 • 2048x1536 Changing the resolution takes effect only when not streaming.	RWS
`Param.White Balance.Auto`	Turn auto white balance on or off. • 0: Use fixed color gains. • 1: Automatically adjust color gains.	RWS
`Param.White Balance.[Auto Blue Green Factor | Auto Red Green Factor]`	Multiplier to skew the image more or less blue/red when auto white balance is enabled. Values > 1 make the image more blue/red, values <1 make it less blue/red.	RWS
`Param.White Balance.[Red Gain|Green Gain|Blue Gain]`	The gains applied to the red, green, and blue channels. The parameters are read-only while auto white balance is enabled.	RWS
`Param.White Balance.[Red Offset|Green Offset|Blue Offset]`	The offsets applied to the red, green, and blue channels. The parameters are read-only while auto white balance is enabled.	RWS

Bump Sensor User Parameters

The following user parameters store information about the bumper sensor status. For details on the bump sensor, see the user guide that accompanied your system.

Table 4-14 Bump Sensor User Parameters

User Parameter	Description	Access Rules
`Param.Bump Detector.Bump Detection`	This user parameter enables the bump detector. Values: “1” bump detector enabled (default), “0” bump detector disabled.	RWS
`Param.Bump Detector.Bumped`	This user parameter indicates when the system has detected a bump. The system sets this user parameter to “1” upon detecting a bump. The system resets this user parameter to “0” once you have set the `Param.Bump Detector.Clear` user parameter to “1.”	R
`Param.Bump Detector.Clear`	Set this user parameter to clear all bumps detected up to that point. This clears the “bump detected” bit in the `Info.Status.Alerts` user parameter and sets the `Info.Status.Bump Detected` and `Param.Bump Detector.Bumped` user parameters to “0”. Values: “1” clears all detected bumps. The system will automatically reset this user parameter to “0”.	W

5 Command Details

Before sending any commands to the Polaris Vega system, read the user guide that accompanied your system to ensure that you have a full understanding of the system functionality.

3D

Returns the latest three-dimensional marker position of a single marker or multiple markers.

Compatibility

Supported by the Position Sensor since G.001.

Deprecated.

Operating Mode

Diagnostic, Tracking

Prerequisite Command

IRED (page 104), only for active markers in Diagnostic mode

Syntax

`3D<SPACE><Port Handle><Reply Option><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters. Specifies which type of marker the system reports data for. For details, see **"Usage Notes" on page 66**. The specified port handle must be initialized (PINIT) and enabled (PENA).
`Reply Option`	Specifies which information will be returned. The reply options cannot be OR'd. **Valid Values** • 1: Single marker 3D data, with error value • 2: Single marker 3D data, with error value and out-of-volume information • 3: Single marker 3D data, with line separation value • 4: Single marker 3D data, with line separation value and out-of-volume information • 5: 3D data for up to 50 markers, with line separation and out-of-volume information

Replies

Upon Success:

```
<Number of Visible Markers><LF>
  <Reply Option n Data><CRC16><CR>
```

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Reply Component	Description
`Number of Visible Markers`	Three characters (a sign and two decimal digits). The number of markers detected by the system. For reply options 1 to 4, only one marker can be in view. If more than one marker is in view, the system returns 00 for the number of markers.
`Reply Option n Data`	The data specific to the requested reply option. For details, see the reply option information below: • Reply option 1: 3D data for a single marker, with error value • Reply option 2: 3D data for a single marker, with error value and out-of-volume information • Reply option 3: 3D data for a single marker, with line separation value) • Reply option 4: 3D data for a single marker, with line separation value and out-of-volume information • Reply option 5: 3D data for up to 50 markers, with line separation and out-of-volume information

Reply Option 1 - 3D data for a single marker, with error value

`<Reply Option 1 Data> = <Tx><Ty><Tz><Error Value>`

Reply Component	Description
`Tx, Ty, Tz`	Nine characters each (a sign and eight decimal digits, with an implied decimal in the position XXXX . XXXX). Position of the marker, in the coordinate system of the Position Sensor.
`Error Value`	Four characters (a sign and three decimal digits, with an implied decimal in the position X . XX). The normalized error number associated with the calculation for this marker position. **Possible Values:** +000 (best case) to +100 (worst case)

Reply Option 2 - 3D data for a single marker, with error value and out-of-volume information

`<Reply Option 2 Data> = <Tx><Ty><Tz><Error Value><Out of Volume>`

Reply Component	Description
`Tx, Ty, Tz`	Nine characters each (a sign and eight decimal digits, with an implied decimal in the position XXXX . XXXX). Position of the marker in the coordinate system of the Position Sensor.
`Error Value`	Four characters (a sign and three decimal digits, with an implied decimal in the position X . XX). The normalized error number associated with the calculation for this marker position. **Possible Values:** +000 (best case) to +100 (worst case)
`Out of Volume`	One hexadecimal character. Indicates whether the marker is outside the characterized measurement volume. **Possible Values** • 0: The marker is inside the characterized measurement volume. • 1: The marker is out of volume.

Reply Option 3 - 3D data for a single marker, with line separation value

<Reply Option 3 Data> = <Tx><Ty><Tz><Line Separation>

Reply Component	Description
Tx, Ty, Tz	Nine characters each (a sign and eight decimal digits, with an implied decimal in the position XXXX . XXXX). Position of the marker in the coordinate system of the Position Sensor.
Line Separation	Four characters (a sign and three decimal digits, with an implied decimal in the position X . XX). The minimum distance (in mm) between the two lines of sight calculated from the marker image on the left and right sensor to the IR source. **Possible Values:** +000 (best case) to +999 (worst case)

Reply Option 4 - 3D data for a single marker, with line separation value and out-of-volume information

<Reply Option 4 Data> = <T_x><T_y><T_z><Line Separation><Out of Volume>

Reply Component	Description
Tx, Ty, Tz	Nine characters each (a sign and eight decimal digits, with an implied decimal in the position XXXX . XXXX). Position of the marker, in the coordinate system of the Position Sensor.
Line Separation	Four characters (a sign and three decimal digits, with an implied decimal in the position X . XX). The minimum distance (in mm) between the two lines of sight calculated from the marker image on the left and right sensor to the IR source. **Possible Values:** +000 (best case) to +999 (worst case)
Out of Volume	One hexadecimal character Indicates whether the marker is outside the characterized measurement volume. **Possible Values** • 0: The marker is inside the characterized measurement volume. • 1: The marker is out of volume.

Reply Option 5 - 3D data for up to 50 markers, with line separation value and out-of-volume information

<Reply Option 5 Data> =
<T_{x1}><T_{y1}><T_{z1}><Line Separation 1><Out of Volume 1><LF>
<T_{x50}><T_{y50}><T_{z50}><Line Separation 50><Out of Volume 50><LF>

Reply Component	Description
Txn, Tyn, Tzn	Nine characters each (a sign and 8 decimal digits, with an implied decimal in the position XXXX . XXXX). Position of the n^{th} marker in the coordinate system of the Position Sensor. The system reports up to 50 3D positions, including phantom markers. If the system detects more than 50 IR sources, it report the first 50 only. The IR sources are not reported in any particular order.

Reply Component	Description
`Line Separation n`	Four characters (a sign and three decimal digits, with an implied decimal in the position X . XX). Line separation of the nth marker. The minimum distance (in mm) between the two lines of sight calculated from the marker image on the left and right sensor to the IR source. **Possible Values:** +000 (best case) to +999 (worst case)
`Out of Volume n`	One hexadecimal character. Indicates whether the nth marker is outside the characterized measurement volume. **Possible Values** • 0: The marker is inside the characterized measurement volume. • 1: The marker is out of volume.

Usage Notes

- The specified port handle must be enabled using PENA (page 109).
- You may need to use the 3D command about ten times if it is sent immediately after using IRED (page 104). This allows time for the system to implement the activation signature and optimize the signal by adjusting the range control.
- Reply Options 1 to 4: You cannot have more than one marker in view. Any other IR sources in view will prevent the system from returning marker data.
- Reply Option 5: The system does not distinguish between real markers, phantom markers, or other IR sources. You must determine whether the reported marker positions are valid. For more information on phantom markers, see the user guide that accompanied the system.
- The 3D command returns data regardless of the bump status, temperature status, and other system status conditions. Before trusting the marker positions returned by the 3D command, you should check these conditions by reading the `Info.Status.Alerts` user parameter. To check the value of user parameters use GET (page 94). You can use BX (page 69) or TX (page 159) to request 3D data that is filtered when the bump status, temperature status, or other system conditions are not ideal.
- Reply Option 1 and Reply Option 2: The system will not calculate an error and will return an error value of +000.

Example

Command:

```
3D 011
```

Reply:

```
+01-12345678+12345678-12345678+0954B7B
```

In this case, one marker is in view.

APIREV

Returns the API revision number that functions with your system.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Deprecated. Use the parameter `Features.Firmware.API Revision` instead.

Operating Mode

All modes

Syntax

`APIREV<SPACE><CR>`

Replies

Upon Success:
`<Family>.<Major revision number>.<Minor revision number><CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Reply Component	Description
`Family`	One ASCII character. This character is always G. Other types of NDI measurement systems use other characters.
`Major revision number`	Three ASCII characters. The major revision number is incremented whenever there is an incompatible change in the API. (Whenever a command is deprecated or when its response is changed in a way that may break an application.)
`Minor revision number`	Three ASCII characters. The minor number is incremented whenever there is an addition to the API that is compatible with all existing applications and usage. (Compatible changes are additions to the API command or option set that will not affect any existing applications.)

Example

Command:
`APIREV`

Reply:
`G.003.0016379`

BEEP

Sounds the system beeper.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Operating Mode

All modes

Syntax

`BEEP<SPACE><Number of Beeps><CR>`

Parameter	Description
`Number of Beeps`	**Valid Values:** 1 to 9

Replies

Upon Success:
`<Beep Status><CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Reply Component	Description
`Beep Status`	**Possible Values** • 0: The system is busy beeping. • 1: Beeping has started.

Usage Notes

- The beep duration is shorter than the beep used for reset and fatal error conditions.
- Disabling the system beeper by setting the value of the user parameter `Param.System Beeper` does not affect the BEEP command.
- The system will never return a beep status of 0. If you send the BEEP command while the system is busy beeping, the system will return a beep status of 1, but will not initiate the second sequence of beeps.

Example

Command:
`BEEP 1`

Reply:
`1D4C1`

BX

Returns the latest tool transformations, individual marker positions, and system status in binary format.

Compatibility

Supported by the Position Sensor since G.001.

BX2 is recommended for new application development.

Operating Mode

Tracking

Syntax

`BX<SPACE><Reply Option><CR>`

Parameter	Description
`Reply Option`	Optional. Specifies which information will be returned. If no reply option is specified, the system returns information for reply option 0001. The reply options are hexadecimal numbers that can be OR'd. If multiple reply options are used, the replies are returned for each port handle in order of increasing option value, with the following exceptions: Reply option 0800 is not reported separately from the other options; it simply enables the system to return certain information in the other options. Reply option 1000 is reported after all handle-specific options but before the <system status> and <CRC16>. **Valid Values** • 0001: Transformation data (default) • 0002: Tool and marker information • 0004: 3D position of a single stray active marker • 0008: 3D positions of markers on tools • 0800: Transformations not normally reported • 1000: 3D positions of stray passive markers • 2000: Appends extended stray passive marker status

Replies

Note **The reply for the BX command is binary data.**

Upon Success:

```
<Start Sequence><Reply Length><Header CRC><01(Number of Handles)>
  <Handle 1><Handle 1 Status><Reply Opt 0001 Data>...<Reply Opt 0008 Data>
  ...
  <Handle n><Handle n Status><Reply Opt 0001 Data>...<Reply Opt 0008 Data>
  <Reply Option 1000 Data>
  <System Status><CRC16>
```

Note **If a handle status is "disabled," the system does not return any of** `<Reply Option 0001 Data>...<Reply Option 0008>` **for that port handle.**

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Reply Component	Description
`Start Sequence`	Two bytes: A5C4. Indicates the start of the BX reply.
`Reply Length`	Two bytes. Indicates the number of bytes in the reply body between `<Header CRC>` and `<CRC16>`, exclusive.
`Header CRC`	Two bytes. CRC16 of `<Start Sequence>` and `<Reply Length>`.
`Number of Handles`	One byte. The number of port handles for which information is returned.
`Handle n`	One byte. The port handle whose information follows.
`Handle Status`	One byte. **Possible Values** • 01: Valid • 02: Missing • 04: Disabled
`Reply Option m Data`	The data specific to the requested reply option. For details, see the reply option information below: • Reply option 0001: Transformation data (default) • Reply option 0002: Tool and marker information • Reply option 0004: Latest 3D position of single stray active marker) • Reply option 0008: 3D position of markers on tools • Reply option 0800: Reporting all transformations • Reply option 1000: 3D position of stray passive markers • Reply option 2000: Appends extended stray passive marker status.
`System Status`	Two bytes. The status of the system. **Bits in use in bit field** • Bit 0: System communication synchronization error • Bit 3: Recoverable system processing exception • Bit 6: A port handle has become occupied • Bit 7: A port handle has become unoccupied • Bit 8: Diagnostic pending • Bit 9: Temperature (system is not within operating temperature range) • Bit 10: Hardware configuration changed (e.g. VCU or SCU has connected or disconnected) Bits 1, 2, 4, 5, and 11 to 15 are reserved.

Note **The "diagnostic pending" bit is set when an alert is detected or cleared. To view the alerts status and clear the diagnostic pending bit, use GET (page 94) to check the `Info.Status.New Alerts` user parameter for every hardware device in the system. For details, see "Usage Notes" on page 76. For API revision G.001.003 and earlier, the diagnostic pending bit did not indicate when an alert was cleared.**

Reply Option 0001 - Transformation Data

`<Reply Option 0001 Data> = <Q0><Qx><Qy><Qz><Tx><Ty><Tz><Error><Port Status>`
`<Frame Number>`

or

`<Reply Option 0001 Data> = <Port Status><Frame Number>`

Reply Component	Description
`Q0, Qx, Qy, Qz`	Four bytes each. Rotational components of the transformation, quaternion, unitless, reported as IEEE 32-bit, single precision, floating point numbers. The value for Q0 is always non-negative.
`Tx, Ty, Tz`	Four bytes each. Translational components of the transformation, in mm, reported as IEEE 32-bit, single precision, floating point numbers.
`Error`	Four bytes. The error is an RMS value, given in mm. It is the result of the least squares minimization between the marker geometry in the tool definition file and the data from the tool's markers measured by the system. Reported as IEEE 32-bit, single precision, floating point number.
`Port Status`	Four bytes. **Bits in use in bit field** • Bit 0: Occupied • bit 1: Switch 1 closed • Bit 2: Switch 2 closed • Bit 3: Switch 3 closed • Bit 4: Initialized • Bit 5: Enabled • Bit 6: Out of volume • Bit 7: Partially out of volume • Bit 8: Algorithm limitation (processing requires more buffer than is available) • Bit 9: IR interference (a large bright IR object) • Bit 12: Processing exception (same as tool information bit 7 in reply option 0002) • Bit 14: Fell behind while processing (same as tool information bit 3 in reply option 0002) • Bit 15: Data buffer limitation (too much data; for example, too many markers) Bits 10, 11, 13, and 16 to 31 are reserved.
`Frame Number`	4 byte unsigned number. The frame number is an internal counter related to data acquisition, which is derived from the PTP time. The frame number corresponds to the frame in which the raw data, used to calculate the accompanying transformation, was collected.

Note **If the handle status is "missing," the system returns only the port status and the frame number.**
- Tools are reported as missing if a transformation cannot be determined.
- In the event of a system error that prevents tracking, all tools are reported as missing.

Reply Option 0002 - Tool and Marker Information

`<Reply Option 0002 Data> = <Tool Information><Marker Information>`

Reply Component	Description
`Tool Information`	One byte. **Bit field** • Bit 0: Bad transformation fit • Bit 1: Not enough acceptable markers for transformation • Bit 2: IR interference—environmental IR is interfering with the system (combination of port status bits 9 and 15 in reply option 0001) • Bit 3: Fell behind while processing (same as port status bit 14 in reply option 0001) • Bit 4: Tool face used • Bit 5: Tool face used • Bit 6: Tool face used • Bit 7: Processing exception (same as port status bit 12 in reply option 0001)
`Marker Information`	Ten bytes (four bits per marker). See below for an example. **Possible Values** • 0000: Not used because it was missing • 0001: Not used because it exceeded the maximum marker angle • 0010: Not used because it exceeded the maximum 3D error for the tool • 0011: Used to calculate the transformation • 0100: Used to calculate the transformation, but it is out of volume • 0101: Not used because it was outside the characterized measurement and was not needed to calculate a transformation.

Example - Marker Information: A tool with markers located at T, R, C, and A, where all four markers were used to determine the calculation, would have the following reply:

Marker	T	S	R	Q	...	D	C	B	A
Reply	0011	0000	0011	0000		0000	0011	0000	0011

Reply Option 0004 - 3D Position of Single Stray Active Marker

`<Reply Option 0004 Data> = <Status><`T_x`><`T_y`><`T_z`>`

or

`<Reply Option 0004 Data> = <Status>`

Reply Component	Description
Status	One byte. The status of the stray active marker. A stray marker on an active tool is not fixed with respect to the other markers that make up the tool. **Bits in use in bit field** • Bit 0: Valid stray active marker • Bit 1: Marker is missing • Bit 3: Marker is out of volume Bits 2 and 4 to 7 are reserved.
Tx, Ty, Tz	Four bytes each. Position of the marker in the coordinate system of the Position Sensor, reported as IEEE 32-bit, single precision, floating point numbers. The marker position is reported only if the marker status is “valid” or “out of volume” and reply option 0800 is used.

Note **If no stray active marker is defined (for example, for wireless port handles, or wired tools with no stray marker defined in the tool definition file), the status is 00 and no position information is returned. If the marker is missing, or if the marker is out of volume and reply option 0800 is not used, the system returns only the status.**

Reply Option 0008 - 3D Position of Markers On Tools

<Reply Option 0008 Data> = <Number of Markers><Out of Volume><T_{xn}><T_{yn}><T_{zn}>

Reply Component	Description
Number of Markers	One byte. Number of markers used in tool transformations.
Out of Volume	One byte/eight markers (one bit per marker). The bit is set when the marker is outside the characterized measurement volume (see example below). Reply size = (number of markers)/8, rounded up to the nearest integer.
Txn, Tyn, and Tzn	Four bytes each. Position of the n^{th} marker, reported in the coordinate system of the Position Sensor, reported as IEEE 32-bit, single precision, floating point numbers. The system reports the positions of markers used in tool transformations and markers that exceeded the maximum marker angle or maximum 3D error specified in the tool definition file. For more information, see “Usage Notes” on page 76.

Example - Out of Volume The information is returned in the format illustrated in the following example: one bit per marker, in little endian format. In this example, there are nine markers, all of which are out of volume:

Marker Number					9				8	7	6	5	4	3	2	1
Bit Field	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Reply	0				1				F				F			
Reply Byte	n								n+1							

Reply Option 0800 - Reporting All Transformations

This option enables the reporting of transformations or translations in situations where translations or transformations are calculated, but by default are not reported by the system. Such situations include:

- The tool or marker is outside of the characterized measurement volume.
- The bump sensor has been tripped.
- The system is outside of the optimal operating temperature range.
- Other system conditions are not ideal. For a full list of these conditions, see “Alerts User Parameters” on page 28.

This reply option must be OR’d with reply option 0001 to obtain transformations for tools in the situations listed above. It must be OR’d with reply options 0004, 0008, or 1000 to obtain position information for markers in the situations listed above.

When using reply option 0800 with the BX command, you must take appropriate action to detect the events listed above, and determine whether they are detrimental to your application. If one or more of the events listed above occurs, reply option 0800 enables the system to return data that may lead to inaccurate conclusions and may cause personal injury.

Appropriate action to detect the events listed above includes:

- reading the out-of-volume flag in reply options 0001 and 0002 when tracking tools
- reading the out-of-volume information in reply options 0004, 0008, and 1000 when tracking stray markers
- reading the temperature flag in the system status
- reading the diagnostic pending bit in the system status
- reading the `Info.Status.New Alerts` user parameter for every hardware device in the system when the diagnostic pending bit is set. For details, see “Usage Notes” on page 76.

Reply Option 1000 - 3D Position of Stray Passive Markers

<Reply Option 1000 Data> = <Number of Markers><Out of Volume><T_{xn}><T_{yn}><<T_{zn}><2000 reply option data if requested>

Reply Component	Description
Number of Markers	One byte. Number of stray markers.
Out of Volume	One byte/eight markers (one bit per marker). The bit is set when the marker is outside the characterized measurement volume (see example below). Reply size = (number of markers)/8, rounded up to the nearest integer.
Txn, Tyn, Tzn	Four bytes each. Position of the n^{th} marker in the coordinate system of the Position Sensor, reported as IEEE 32-bit, single precision, floating point numbers.

Note **At least one passive port handle must be enabled, to activate the illuminators on the Position Sensor. If no passive port handles are enabled, <Number of Markers> will return 00 and no other data will be returned.**

Stray passive markers are markers which are not used to calculate any of the transformations for any enabled, passive tools. Stray active wireless tool markers are not reported.

Example - Out of Volume The information is returned in the format illustrated in the following example: one bit per marker, in little endian format. In this example there are nine markers, all of which are out of volume:

Marker Number					9	8	7	6	5	4	3	2	1
Bit Field	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
Reply	0				1				F				F		
Reply Byte	n								n+1						

Reply Option 2000 - Extended Stray Passive Marker Status

<Reply Option 2000 Data> = <extended marker information status>

Reply Component	Description
extended marker information status	Four bits / stray marker. Bit 0 is currently used with the following values: • 0: The marker does not share lines of site with any other marker and is therefore unlikely to be a phantom marker. • 1: The marker shares lines of site with other markers and is therefore possibly a phantom marker. Bits 1 to 3 are reserved.

This reply option can be used in conjunction with reply option 1000 to help identify stray passive markers that could be phantom markers. See the user guide that accompanied your system for information about phantom markers.

Usage Notes

- The BX reply format requires fewer characters than the text format; this allows transformations to be reported more quickly. For replies in text format, use TX (page 159).
- Replies are returned in little endian format.
- By default, transformations are not reported if the tool is partially or wholly out of the characterized measurement volume, if the bump sensor has been tripped, if the system is outside of the optimal operating temperature range, or if certain other alerts have occurred. For details, see "Alerts User Parameters" on page 29. To report these transformations, you must use reply option 0800 OR'd with the desired reply option(s). The accuracy of these transformations is unknown.
- Reply Option 0001:
  - When the "diagnostic pending" bit is set in the system status, use GET (page 94) to read the `Info.Status.New Alerts` user parameter for every hardware device in the system. The act of reading these parameters clears the parameters and the "diagnostic pending" bit. For more information on alerts and their associated user parameters, see "Alerts User Parameters" on page 28.
  - For wired tools, bits 1, 2, and 3 in the port status report switch status.
- Reply Option 0008: Markers are returned in alphabetical order according to how they are labelled in the tool definition file. For example, for a tool with markers labelled A, G, M and S, the system will return the marker positions in the order A G M S. Reply option 0008 only returns data for markers that the system detects. To identify which marker is which, compare the reply option 0008 data to the data returned with reply option 0002. The marker order is the same for both replies; each marker that does not have a <marker information> status of 0000 ("missing") in reply option 0002 corresponds to a marker in reply option 0008.
- System Status:
  - The external IR bit (bit 1) and system CRC error bit (bit 2) are not used by the system.
  - In API revision G.001.004 and later, the diagnostic pending bit (bit 8) is set whenever an alert is detected or cleared. In API revision G.001.003 and earlier, the diagnostic pending bit is set only when an alert is detected.
- Reply Option 0002: Reply 0010 means that the marker was not used because it exceeded the maximum 3D error for the tool.

Example

Command:

```
BX 0801
```

Reply:

```
A5C4005723130201013F3AF3CABE5B7209BF1C07713E635592C39E831F43332973C50051133DA5BD9
F00000031000002CC02013EA1B5D03D137D21BD787C673F72394A4286B6CB43606EF4C50468C13ED4
E74100000031000002CD000059C9
```

This is the hexadecimal representation of the binary data being returned. This example returns data for two tools.

BX2

Returns the latest tool transformations, individual marker positions, and system status in the General Binary Format.

The BX2 command provides a flexible way of providing measurement data at various levels of detail. The reply can contain a single frame or multiple frames. Each frame can contain various levels of measurement data details, such as 6D, 3D or 2D data.

- It does not repeat already reported information.
- It works with the STREAM command to keep latency to a minimum and avoid missing or repeating information.
- Addresses the problem of providing system wide failures and warning in the multi-connection environment.

Compatibility

Supported by the Position Sensor since G.003.

Operating Mode

Tracking

Syntax

`BX2<SPACE><Reply Options><CR>`

Reply Option	Description
`--6d=tools|none`	Specifies whether 6D information for tools is returned. The default is `tools`.
`--3d=none|tools|strays|all`	Specifies which 3D information is returned (none, tool 3Ds, stray 3Ds, or all 3Ds). The default is `none`. If selected, 3Ds are returned for all frame types, not just passive frames.
`--2d=none|tools|strays|all`	Specifies which 2D (line of sight) information is returned. The default is `none`.
`--sensor=none|tools|strays|all`	Specifies which scaled sensor UV information is returned. Scaled UV can be used to visualize the images on the sensors and also provide diagnostic information related to UV brightness. The default is `none`.
`--1d=buttons|none`	Specifies whether buttons are reported. The default is `buttons`.

Replies

Upon Success:

```
<Start Sequence><2 byte Reply Length><Header CRC><GBF Version>
  <Component Count><Frame Component 1>...<Frame Component N><Data CRC>
```

or

```
<Extended Binary Start Sequence><four byte Reply Length><GBF
  Version><Component Count><Frame Component 1>...<Frame Component N>
```

On Error:

`ERROR<error code>`

For error code definitions, see page 181.

Frame Component: 0x0001

The Frame Status field contains error status information related to the frame. In all cases, a value of zero indicates no errors or fault conditions.

All other measurement data details are included as part of the frame data payload. The payload itself follows the General Binary Format.

Frame Data Item		
`Frame Type`	One byte	Frame Types are as follows: • 0: DUMMY • 1: ACTIVE_WIRELESS • 2: PASSIVE • 3: ACTIVE • 4: LASER • 5: ILLUMINATED • 6: BACKGROUND • 7: MAGNETIC
`Frame Sequence Index`	Two byte	
`Frame Status`	Two bytes	For bits 0 to 15, the field uses the same codes as the 6D Port/Tool Status, but only the ones which are applicable to the frame as a whole.
`Frame Number`	Four bytes	
`Frame Timestamp*`	Eight bytes	struct timespec • Bytes 0-3: seconds since start of epoch • Bytes 4-7: nanoseconds
`Frame Data Payload`	Variable	General Binary Format

*The frame timestamp is reported in Coordinated Universal Time (UTC). The absolute accuracy of the timestamp depends on the quality of the PTP Master. In the absence of a Grandmaster or Boundary Clock, one of the PTP devices on the network becomes PTP Master. The timebase of all PTP Slaves are adjusted to that of the PTP Master. While this ensures accurate synchronization between all PTP devices, it does not guarantee that the absolute timestamp will be correct.

6D Data Component: 0x0002

The payload consists of a variable number of 6D data items. Each item has a tool handle (two bytes in the interest of data alignment) followed by a 2 byte bit-field of port/tool status. If the tool is not missing, this will be followed by a transformation in the same format as the BX command (q0, qx, qy, qz, tx, ty, tz, error)

6D Tool Data Item		
`Tool Handle`	Two bytes	
`Status`	Two bytes	See below
`Q0, Qx, Qy, Qz, Tx, Ty, Tz, Error`	Four bytes each	

Port/Tool Status is as follows:

Bit 0-7	Error codes as described in Port/Tool Status Error Codes.
Bit 8	Transform missing
Bit 9	Averaging Enabled
Bit 10-12	Reserved
Bit 13-15	Indicates which face of a multi-face tool is being tracked

Table 5-1 Port/Tool Status Error Codes

Error Code	Description
0	Enabled
3	Tool is partially out of the characterized measurement volume
5	Data is partially averaged; averaging depth is less than specified
9	Tool is out of the characterized measurement volume
The following error codes are reported only if the tool is missing.	
13	Too few markers detected
14	IR interference (a large bright IR object)
17	Bad transformation fit
18	Data buffer limitation (too much data; for example, too many markers)
19	Algorithm limitation (processing requires more buffer than is available)
20	Fell behind while processing
21	Position sensors out of synch
22	Processing exception
31	Tool is missing
32	Tracking is not enabled for this tool
33	Tool has been unplugged from the System Control Unit

3D Data Component: 0x0003

The 3D component payload consists of a variable number of 3D items. Each item has a 4 byte handle reference to the port handle of the tool to which the 3D's belong. If the 3D is "stray", the handle reference will be -1. Location information will be equivalent to the 3D information in the BX command. If the marker is missing its 3D item then information will not be present.

Note **To provide additional diagnostic information for active and passive tools, all defined markers are reported with appropriate status and index (whether they are visible or not) for tools defined with up to 20 markers. For tools with more than 20 defined markers, only visible markers will be reported.**

3D Tool Data Item		
`Tool Handle Reference`	Two bytes	0xffff for "stray" 3D
`Number of 3Ds`	Two bytes	

3D Data Item		
Status	One byte	See below
-reserved-	One byte	
Marker Index	Two bytes	index of marker on tool, sequential # for strays
X, Y, Z	Two bytes each	

For data alignment, the marker status field is one byte as follows:

0x00	OK
0x01	Missing (missing markers may not be reported in component at all)
0x02	Not used: exceeded max marker angle
0x03	Not used: exceeded max 3D error for tool
0x04	Not used: Out of Volume
0x05	Out of Volume – used in 6D
0x06	Possible phantom marker (in volume, applies to stray markers only)
0x07	Saturated (in or out of volume, not used in 6D)
0x08	Saturated and out of volume (not used in 6D)
0x09 to 0xFF	Reserved

1D Button Component: 0x0004

The 1D button component consists of a variable number of button state items. Each item contains a port handle (tool) reference or, in the case of non-tool buttons, a dummy tool reference corresponding to the frame in which the button was sampled. Button states are one byte each. Use 0 for open and 1 for closed.

Currently, the only supported states are CLOSED and OPEN. In future, button processing on the firmware may be enhanced to support additional states, such as PRESSED, RELEASED, CLICKED, DOUBLE-CLICKED, or HOLD.

1D Button Data Item		
Tool Handle Reference	Two bytes	0xffff for “stray” (non-tool) buttons
Number of buttons	Two bytes	
Button Data Item		
Button data	Four bytes	One byte for each 4 buttons (little endian)

2D Data Component: 0x0005

The 2D data component is as follows:

Tool Line of Sight (LOS) Item		
Tool Handle Reference	Two bytes	0xffff for “stray” LOS
Number of Sensors	Two bytes	
LOS Sensor Item		
Sensor Reference	Two bytes	Index number of the sensor

Number of LOS Items	Two bytes	
LOS Item		
Status	Four bytes	See below
Base X,Y,Z	Four bytes each	vector from origin to sensor
LOS X,Y,Z	Four bytes each	LOS vector from sensor
Number Marker References (N)	Two bytes	0-65535
Marker References	Two bytes each	Index of 3D item reported for tool handle
LOS Item Padding	((N+1)*2)%4 bytes	Zero or two bytes of padding to ensure each item ends in a 4 byte boundary

The marker status field is four bytes as follows

0x00	OK
0x07	Saturated (in volume, not used in 6D)

Sensor UV Component: 0x0011

This component is returned if the –sensor option is used. It returns simplified uv data that can be used for diagnostics.

Component Item		
Tool Handle Reference	Two bytes	0xffff for “stray” LOS
Number of Sensors	Two bytes	
UV Sensor Item		
Sensor Reference	Two bytes	Index number of the sensor
Number of UV Items	Two bytes	
Scaled UV Item		
Scaled U, scaled V	One byte each	0-255 upper left of view is 0,0
Width U, Height V	One byte each	
Energy	Four bytes	Sum intensity of all centroid pixels
Peak Intensity	Two bytes	0-65535
Number Marker References (N)	Two bytes	
Marker References	Two bytes each	Refers to 3D index for tool in 3D item
UV Item padding	((N+1)*2)%4 bytes	0 or two bytes of padding to ensure each item ends in a 4 byte boundary

System Alert Component 0x0012

The System Alert Component returns all current system faults, alerts, and events. The component header indicates the number of items. An item consists of a 2 byte type followed by a 2 byte code.

Faults are conditions that indicate the system is unable to function correctly. Generally the unit must be returned to NDI for repair. Alerts are conditions that may impact measurement performance but can be resolved on their own or without physical repair. Events may also impact performance or

system behaviour, but they are a normal part of operations and do not indicate a system malfunction.

Component Item		
Condition Type	One byte	0: Fault 1: Alert 2:Event
- Reserved -	One byte	
Condition Code	Two bytes each	

Faults

The BX2 component 0x0012 returns the following faults:

Error Code	Description
1	Non-recoverable parameter fault
2	Sensor parameter fault
3	Main voltage fault
4	Sensor voltage fault
5	Illuminator voltage fault
6	Illuminator current fault
7	Sensor 0 temperature fault (left)
8	Sensor 1 temperature fault (right)
9	Main temperature fault
10	Sensor fault

Alerts

The BX2 component 0x0012 returns the following alerts:

Error Code	Description
1	Battery fault
2	Bump detected
3	Firmware incompatible
4	Non-fatal parameter fault
5	Not used
6	Not used
7	Not used
8	Temperature high
9	Temperature low
10	System Control Unit disconnected
11	Not used
12	Not used
13	Not used

Error Code	Description
14	PTP synchronization fault
15	Video camera not functioning
17	Firmware running in Safe Mode

For detailed information on system faults and alerts, see “Position Sensor Alerts” on page 30.

Events

The following events are returned as part of the BX2 component 0x0012:

Event Code	Description	How the event is cleared
1	Active tool connected	PHSR by the master connection
2	Active tool disconnected	PHSR by the master connection
5	Hardware configuration changed (e.g. VCU or System Control Unit has connected or disconnected)	PHSR by the master connection
6	The PTP Master has changed	Automatically cleared after being reported in a BX2 response

Usage Notes

The `<Reply>` is either Requested Data or `ERROR<error code>`. The BX2 command can be used alone to generate one reply for each BX2 request or it can be used with the STREAM command to generate a continuous, non-repeating stream of tracking data.

Data returned by the BX2 command is reported in a new binary format. See “General Binary Format” on page 5. The content is wrapped in the same Binary Reply Format as the BX command. When streamed, the entire response will be preceded by the Streaming Reply Format header. See “Data Streaming” on page 7.

No options exist for filtering data returned from the BX2 (page 77) command on the basis of system or tool status or location in the volume. Complete system and tool status information is always included in the reply and it is the application’s responsibility to interpret this data and ignore those measurements that fall outside of application requirements and constraints. Failure to do so may lead to inaccurate conclusions that may cause personal injury.

BX2 Binary Data structured in General Binary Format contains one or more tracking frames, similar to the BX command. It contains up to one full frame sequence of previously unreported data. Each frame will be contained in the Frame Data Component. See “General Binary Format” on page 5.

Frame data component will contain various level of tracking data according to the specified BX2 commands. Each type of the tracking data such as 6D, 3D or 2D will be reported again in the General Binary Data Format as separate components.

The content of the single frame of data contains various levels of tracking detail. Each lower level of information references the higher-level information, see Figure 5-1.

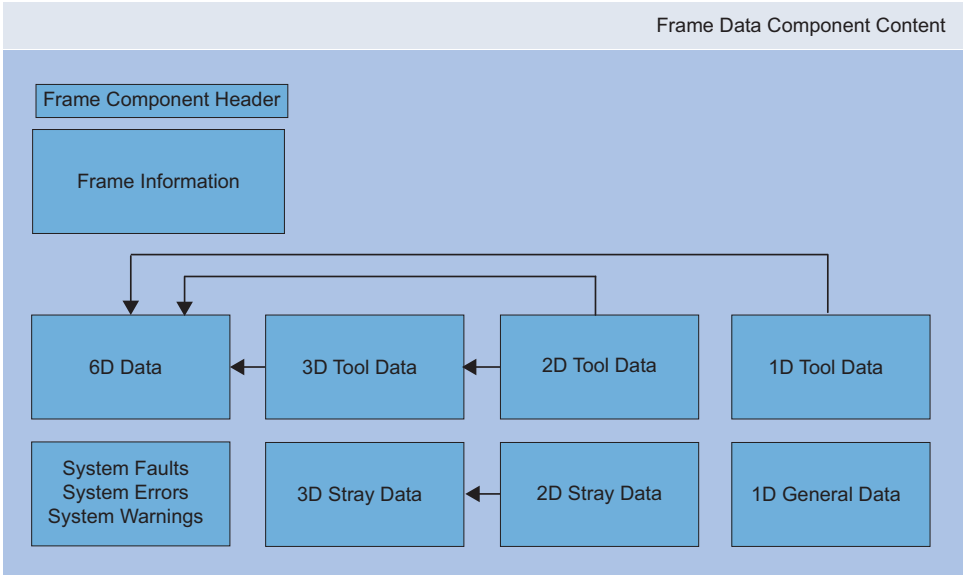


Figure 5-1 Frame Component Overview

Example

6D tools (2 passive tools loaded and tracking)

Command:

```
BX2 --6d=tools --1d=none
```

Reply:

```
C4 A5 64 00 07 D3 01 00 01 00 01 00 60 00 00 00 00 00 01 00 00 00 02 00 00
00 BF 05 2E 38 CB 74 75 57 12 A2 D9 2A 01 00 02 00 12 00 0C 00 00 00 00 00
00 00 00 00 02 00 34 00 00 00 00 00 02 00 00 00 03 00 00 20 79 3A 7E 3F 76
F0 37 BD DE 39 DE BD 83 3A B2 BA 10 95 6A 42 C0 05 F6 C2 BD CA 8C C4 6F 7B
CE 3C 04 00 0D 01 F3 7D
```

This string decodes as:

```
Start sequence          2 Bytes     Unitless    0xA5C4
Reply length            2 Bytes     Bytes       0x0064      100 Bytes
Header CRC              2 Bytes     Unitless    0xD307

GBF version             2 Bytes     Unitless    0x0001      Version 1
Component count         2 Bytes     Unitless    0x0001      1 Component

  Component 1

  Component type        2 Bytes     Unitless    0x0001      Frame
 Component
```

```
 Component size          4 Bytes     Bytes       0x00000060  96 Bytes
 Item Format Option      2 Bytes     Unitless    0x0000
 Item count              4 Bytes     Count       0x00000001  1 Item to
parse

   Frame Item 1

   Frame Type            1 Byte      Type        0x02        Passive
   Sequence Index        1 Byte                  0x00
   Frame Status          2 Bytes                 0x0000      OK
   Frame Number          4 Bytes     Count       0x382E05BF  942540223
decimal
   Timestamp             8 Bytes     Seconds     0x577574CB  Jun 30 2016
15:36:43
                                     Nanoseconds 0x2AD9A212

   GBF version           2 Bytes     Unitless    0x0001      Version 1
   Component count       2 Bytes     Unitless    0x0002      2 Data
Components

     Data Component 1

     Component type      2 Bytes     Unitless    0x0012      System Alerts
     Component size      4 Bytes     Count       0x0000000C  12 Bytes
     Item Format Option  2 Bytes     Unitless    0x0000
     Item count          4 Bytes     Count       0x00000000  No System
Alert Items

     Data Component 2

     Component type      2 Bytes     Unitless    0x0002      6D Data
Component
     Component size      4 Bytes     Bytes       0x00000034  52 Bytes
     Item Format Option  2 Bytes     Unitless    0x0000
     Item count          4 Bytes     Count       0x00000002  Two 6D Data
Items

       6D Data Item 1

       Tool Handle       2 Bytes                 0x0003      Handle 3
       Handle status     2 Bytes                 0x2000      OK, Face 1
                                                             Transform not
missing
       Q0                4 Bytes     Unitless    0x3F7E3A79  +0.993079722
       Qx                4 Bytes     Unitless    0xBD37F076  -0.044907056
       Qy                4 Bytes     Unitless    0xBDDE39DE  -0.108508810
       Qz                4 Bytes     Unitless    0xBAB23A83  -0.001359776
       Tx                4 Bytes     mm          0x426A9510  58.645568848
mm
```

Ty	4 Bytes	mm	0xC2F605C0	-123.0112305 mm
Tz	4 Bytes	mm	0xC48CCABD	-1126.335571 mm
Error	4 Bytes	mm	0x3CCE7B6F	0.0252053421 mm RMS
6D Data Item 2				
Tool Handle	2 Bytes		0x0004	Handle 4
Handle status	2 Bytes		0x010D	Too Few Markers Transform missing
CRC16	2 Bytes	Unitless	0x7DF3	

COMM

Sets the serial communication settings for the system.

Compatibility

Supported by the Position Sensor since G.001. Used for serial communications only with Position Sensors equipped with a serial communications port (LEMO connector).

Operating Mode

All modes

Syntax

`COMM<SPACE><Baud Rate><Data Bits><Parity><Stop Bits><Hardware Handshaking><CR>`

Parameter	Description
`Baud Rate`	The data transmission rate between the system and the host computer, in bits per second. The default baud rate is 9600 bps. **Valid Values** • 0: 9600 bps • 1: 14 400 bps • 2: 19 200 bps • 3: 38 400 bps • 4: 57 600 bps • 5: 115 200 bps • 6: 921 600 bps • 7: 1 228 739 bps
`Data Bits`	To use any command that returns binary data (BX, BX2, GETLOG, or VCAP), the data bits must be set to eight bits. The default is eight data bits. **Valid Values** • 0: Eight bits • 1: Seven bits
`Parity`	The default is no parity. **Valid Values** • 0: None • 1: Odd • 2: Even
`Stop Bits`	The default is one stop bit. **Valid Values** • 0: One bit • 1: Two bits
`Hardware Handshaking`	The default is no hardware handshaking. **Valid Values** • 0: Off • 1: On

Replies

Upon Success:

OKAY<CRC16><CR>

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Usage Notes

- The system serial communication parameters have a default setting of 00000 (i.e. 9600 baud, 8 data bits, no parity, 1 stop bit, hardware handshaking off).
- To use any command that returns binary data (BX, GETLOG, or VCAP), you must set the data bits to 0 (8 bits).
- If you change the baud rate using the COMM command, you must also change your host computer baud rate; otherwise, a system reset or other unexpected communication behaviour will occur. The host application should wait approximately 100 ms after receiving the OKAY reply from the system before changing its own communication parameters.
- NDI strongly recommends using hardware handshaking when using the higher baud rates.
- Most Windows applications do not allow you to choose 1.2 Mbaud. To allow you to communicate at this speed, NDI has aliased 19 200 baud to 1.2 Mbaud when using a USB connection. Thus, to communicate at 1.2 MB:

  a) Connect the system using a USB connection (this is the only option for passive systems).

  b) Set the system to 1.2 Mbaud (<baud rate> parameter value 7).

  c) Set the application on the host computer to 19 200 baud. The virtual COM driver maps the communications speed to 1.2 Mbaud, so the application will actually communicate with the system at 1.2 Mbaud.

Do not set the System to 19 200 baud when using a USB connection; if the system is set to 19 200 baud, it will be unable to communicate with the host computer, because setting the host application to 19 200 baud will result in the aliased rate of 1.2 Mbaud.

Example

Command:

COMM 30001

Reply:

OKAYA896

This changes the serial communication parameters to 38400 baud, 8 data bits, no parity, 1 stop bit, hardware handshaking on.

DFLT

Restores the user parameters to factory default values.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`DFLT<SPACE><User Parameter Name><CR>`

Parameter	Description
`User Parameter Name`	A string, identifying the name of the user parameter. May include a trailing wild card character (*) Use `DFLT *` to restore all user parameters to default values. User parameter names are case-sensitive.

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- The user parameter name may include a trailing wild card character (*).
- To return all user parameters to their default values, use `DFLT` *****.
- The user parameter values set using the DFLT command persist until the system is reset or initialized. To save the user parameters at their factory default values, after using the DFLT command, use SAVE (page 140).
- To view a list of user parameters and their current values, use `GET *`.
- User parameter names are case-sensitive.
- For more information on user parameters, see “User Parameters” on page 25.

Example

Command:

```
DFLT *
```

Reply:

```
0KAYA896
```

DSTART

Starts Diagnostic mode.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`DSTART<SPACE><Reply Option><CR>`

Parameter	Description
`Reply Option`	80 (Optional)

Replies

Upon Success:
`0KAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

The frame number is reported in reply option 0001 of TX (page 159) and BX (page 69). In the Vega system, the frame number is derived from the PTP time and reply option 80 is ignored.

To facilitate active tool setup from a monitor connection, DSTART returns OKAY when in diagnostic mode.

Example

Command:
`DSTART`

Reply:
`0KAYA896`

DSTOP

Stops Diagnostic mode.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Diagnostic

Prerequisite Command

DSTART (page 91)

Syntax

DSTOP<SPACE><CR>

Replies

Upon Success:
OKAY<CRC16><CR>

On Error:
ERROR<Error Code><CRC16><CR>
For error code definitions, see page 181.

Usage Notes

If executed from setup mode, it returns OKAY.

Example

Command:
DSTOP

Reply:
OKAYA896

ECHO

Returns exactly what is sent with the command.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`ECHO<SPACE><Any ASCII characters><CR>`

Replies

Upon Success:
Exactly what is sent with the command, with `<CRC16><CR>`.

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

The ECHO command can handle a maximum of ~50,000 characters. Exceeding this number causes the system to return error 02.

Example

Command:
`ECHO Testing!`

Reply:
`Testing!A81C`

GET

Returns the user parameter values.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`GET<SPACE><User Parameter Name><CR>`

Parameter	Description
`User Parameter Name`	A string identifying the name of the user parameter. May include a trailing wild card character (*). To return all user parameter values, use `GET *`. User parameter names are case-sensitive.

Replies

Upon Success:

```
<User Parameter Name>=<value><LF> (repeated for each user parameter name,
  but no line feed after the last parameter)
  <CRC16><CR>
```

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Reply Component	Description
`User Parameter Name`	Variable size Full name of the user parameter
`Value`	Value of the user parameter

Usage Notes

- The user parameter name may include a trailing wild card character (*).
- Use `GET *` to return the names and values of all user parameters.
- Numeric user parameter values are returned as decimal strings.
- User parameter names are case-sensitive.
- For descriptive information about each user parameter, including type, attributes, and possible values, use GETINFO (page 96).

For more information on user parameters, see “User Parameters” on page 25.

Example

Command:

```
GET Info.Status.New Alerts
```

Reply:

```
Info.Status.New Alerts=08B32
```

GETINFO

Returns descriptive information about the user parameters.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`GETINFO<SPACE><User Parameter Name><CR>`

Parameter	Description
`User Parameter Name`	A string, identifying the name of the user parameter. May include a trailing wild card character (*). To return all user parameter values, Use `GETINFO *`. User parameter names are case-sensitive.

Replies

Upon Success:

```
<User Parameter Name>=<Value>;<Type>;<Attribute>;<Minimum>;<Maximum>;
  <Enumeration>;<Description><LF> (repeated for each user parameter, but
  no line feed after last parameter)
  <CRC16><CR>
```

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Reply Component	Description
`User Parameter Name`	Variable size. Full name of the user parameter.
`Value`	Variable size. Value of the user parameter.
`Type`	One hexadecimal character. Describes the data type. **Possible Values** • 0: Boolean • 1: Integer • 2: Float • 3: String

Reply Component	Description
`Attribute`	1 to 4 hexadecimal characters Describes the access rules. **Bit field** • Bit 0: Read • Bit 1: Write • Bit 2: Save • Bit 3: Volatile (may change frequently) • Bit 4: Keyed (cannot be changed unless key is supplied) - Not used in Vega • Bit 5: Enabled keyed parameter - Not used in Vega • Bits 6 to 7: Reserved (may not all be set to 0) • Bit 8: Table parameter • Bit 9-15: Reserved (may not all be set to 0)
`Minimum`	Minimum allowed value of the user parameter. For a string, the minimum number of characters allowed. If minimum and maximum are 0, no range check is performed.
`Maximum`	Maximum allowed value of the user parameter. For a string, the maximum number of characters allowed. If minimum and maximum are 0, no range check is performed.
`Enumeration`	Comma-separated enumeration list. This is a list of possible values that the user parameter can take, and corresponds to the values in the `<Value>` field (the first item in the list corresponds to value 0, the second item corresponds to value 1, etc.).
`Description`	Describes the user parameter's function.

Usage Notes

- The user parameter name may include a trailing wild card character (*).
- Use `GETINFO *` to return information for all user parameters.
- Numeric user parameter values are returned as decimal strings.
- User parameter names are case-sensitive.
- For a list of user parameters and values without descriptive information, use the GET command.

For more information on user parameters, see "User Parameters" on page 25.

Example

Command:

```
GETINFO Info.Status.Bump Detected
```

Reply:

```
Info.Status.Bump Detected=0;1;800D;0;1;False,True;Indicates if the system
  has detected a bump49CB
```

The system returns descriptive information for the specified parameter.

GETLOG

Returns the contents of the device's event log.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`GETLOG<SPACE><Offset><Length><Logname><CR>`

Parameter	Description
`Offset`	Eight hexadecimal character string. Specifies the offset of the data requested within the file.
`Length`	Four hexadecimal character string. Specifies the requested amount of data, in bytes. Up to 50 kilobytes can be requested at one time.
`Logname`	String identifying the name of the log. Log names are case-sensitive. In API revision G.001.003 and earlier and G.003.001 and later, the log file is named sysinfo. API revision G.001.004 up to but not including G.003.001, the log file is named \<Device Name>\sysinfo. For device name details, see "Device Names" on page 26.

Replies

Upon Success:

`<Header><Length><Header CRC><Data><Data CRC>`

Note **The reply for the GETLOG command is binary data.**

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Reply Component	Description
`Header`	Two bytes: A5C4. Indicates the start of the GETLOG reply.
`Length`	Two bytes. The number of bytes of data being returned.

Reply Component	Description
`Header CRC`	Two bytes CRC16 for header.
`Data`	Up to 50 kilobytes of binary data.
`Data CRC`	2 hexadecimal characters. CRC16 of the <Data> section.

Usage Notes

- To read the entire log file:
  a) Start with an offset of 0 and request 50 kilobytes of data.
  b) Increment the offset by 50 kilobytes, and request another 50 kilobytes of data.
  c) Repeat step b) until the reply length of the data is less than the amount you requested. This indicates that you have reached the end of the log file.
- Replies are returned in little endian format.
- To write to a log, use SYSLOG (page 151).
- The log name is **sysinfo**.

Example

Command:

```
GETLOG 000000000800sysinfo
```

INIT

Initializes the system.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`INIT<SPACE><CR>`

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- During power up or system reset, the system configuration is determined. The configuration includes firmware revisions and the characterized measurement volumes for which the Position Sensor has been calibrated. The INIT command ensures that the system configuration was determined successfully.
- The system will automatically return to Setup mode after using the INIT command.
- The INIT command sets any modified user parameters back to the saved values. To prevent modified values from being reset, send the SAVE command before sending INIT.
- If `ERROR2E` or `ERROR15` is returned, there may be a system fault that is indicated by the alerts in the `Info.Status.New Alerts` or `Info.Status.Alerts` user parameter on one or more devices. Use GET to read these user parameters. For details, see “Alerts User Parameters” on page 28.
- If a Monitor mode connection issues the `INIT` command:
  a) If the system is already in the Setup mode with no tools loaded the system, the response is OKAY.
  b) If the system is already initialized but is in Tracking or Diagnostics mode, or if there are tools loaded, in API version G.003.005 and later the response is WARNING06. In versions before G.003.005, the response is WARNING01.
  c) If the system is not initialized, the response is `ERROR39` (Permission denied).
- The INIT command automatically terminates all active streams.

Example

Command:

```
INIT
```

Reply:

```
OKAYA896
```

IRATE

Sets the illuminator rate.

Compatibility

Supported by the Position Sensor since G.001.

Deprecated. To set the illuminator rate for Vega, use the command SET (page 141) to set the user parameter `Param.Tracking.Track Frequency`.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`IRATE<SPACE><Illuminator Rate><CR>`

Parameter	Description
`Illuminator Rate`	Sets the number of times per second that the illuminators emit IR. **Valid Values** • 0: 1/3 of the frame frequency • 1: 1/2 of the frame frequency • 2: 1/1 of the frame frequency

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Example

Command:

```
IRATE 0
```

Reply:

```
OKAYA896
```

IRED

Turns the markers on a wired tool on or off.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Diagnostic

Prerequisite Command

PENA (page 109)

Syntax

`IRED<SPACE><Port Handle><Marker Activation Signature><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters.
`Marker Activation Signature`	Eight hexadecimal characters (32 bits). One bit for each marker. Set the bits corresponding to the markers you want to activate. See example in Usage Notes. **Bit field** • Bit 0: Marker A • Bit 1: Marker B • Bit 2: Marker C ..... • Bit 19: Marker T Bits 20 to 31 are reserved.

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

There are 20 marker positions, labelled “A” to “T.” To specify that a marker should be turned on, set the bit corresponding to that marker to 1. For example, to activate markers B, G, M, and T, set the bit field as follows:

Marker Location		T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
Bit	31- 20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit Value	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0
Activation Signature Parameter Value	0 0 0	8				1				0				4				2			

Example

Command:

```
IRED 0A00081042
```

Reply:

```
OKAYA896
```

LED

Changes the state of visible LEDs on a wired tool.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

All modes

Prerequisite Command

INIT (page 100)

Syntax

`LED<SPACE><Port Handle><LED Number><State><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters
`LED Number`	Specifies the LED. **Valid Values:** 1 to 3
`State`	Sets the state of the specified LED. **Valid Values** • B: Blank (not on) • F: Flash • S: Solid on

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

The visible LEDs are only activated while the system is in Tracking and Diagnostic modes.

Example

Command:

```
LED 0A1S
```

Reply:

```
0KAYA896
```

PDIS

Disables the reporting of transformations for a particular port handle.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

PENA (page 109)

Syntax

`PDIS<SPACE><Port Handle><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Example

Command:
`PDIS 01`

Reply:
`OKAYA896`

PENA

Enables the reporting of transformations for a particular port handle.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Syntax

PENA<SPACE><Port Handle><Tool Tracking Priority><CR>

Parameter	Description
Port Handle	Two hexadecimal characters
Tool Tracking Priority	Describes the type of tool. **Valid Values** • S: Static: a static tool is considered to be relatively immobile, e.g. a reference tool. • D: Dynamic: a dynamic tool is considered to be in motion, e.g. a probe. • B: Button box: a button box can have and LEDs, but no markers. No transformations are returned for a button box tool, but switch status is returned.

Replies

Upon Success:

OKAY<CRC16><CR>

or

WARNING02<CRC16><CR>

Indicates that the tool you are trying to enable is a unique geometry tool that doesn't meet the unique geometry requirements.

WARNING03<CRC16><CR> Indicates that the tool you are trying to enable is a unique geometry tool that conflicts with another unique geometry tool already loaded and enabled.

WARNING04<CRC16><CR>

Indicates that the tool you are trying to enable is a unique geometry tool that doesn't meet the unique geometry requirements, and conflicts with another unique geometry tool already loaded and enabled.

WARNING05<CRC16><CR>

Returned when the system selects a default marker wavelength to track a tool if the tool's definition file did not specify a marker wavelength.

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Usage Notes

- The system does not make use of the tool tracking priority. You must still specify a value, but it does not matter which tool tracking priority you choose.
- When the PENA command is issued, the system compares the tool being enabled with currently enabled tools for conflicting unique geometry constraints. This process is almost instantaneous. If the tool doesn't meet the unique geometry constraints, or conflicts with a tool that is already enabled, the system issues a WARNING02, WARNING03, or WARNING04.
- The system will still enable the tool when the system returns WARNING02, WARNING03 or WARNING04; however, the tool may not track properly since the unique geometry is compromised.
- For more information on unique geometry tools and unique geometry constraints, see the *Polaris Tool Design Guide*.

Example

Command:

PENA 01D

Reply:

OKAYA896

PFSEL

Sets which tool faces to use to track a multi-faced tool.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

PINIT (page 124)

Syntax

`PFSEL<SPACE><Port Handle><Face Selection Mask><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters
`Face Selection`	Two hexadecimal characters (eight bits) Set the bits corresponding to the faces you wish to track.

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- When a tool is initialized, the face selection defaults to a value of 0xFF, so all faces are tracked by default.
- To include a tool face to be tracked, set the corresponding bit. For example, if you wish to track faces 0 and 5, the face selection value is 0x21, as shown in the following table:

Tool Face Number	7	6	5	4	3	2	1	0
Bit Value	0	0	1	0	0	0	0	1
Face Selection Hexadecimal Value	2				1			

- If the system returns error code 23, the face selection did not include any of the valid faces of the selected tool.

Example

Command:

```
PFSEL 0121
```

Reply:

```
0KAYA896
```

PHF

Releases system resources from an unused port handle.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

PHRQ (page 119)

Syntax

PHF<SPACE><Port Handle><CR>

Parameter	Description
Port Handle	Two hexadecimal characters

Replies

Upon Success:
OKAY<CRC16><CR>

On Error:
ERROR<Error Code><CRC16><CR>
For error code definitions, see page 181.

Usage Notes

- The PHF command should be used whenever a tool is disconnected. This optimizes the use of system resources. If PHF is not used, the system will be unable to assign a port handle after the maximum number of port handles has been reached.
- If a tool is disconnected then reconnected, it is assigned a new port handle. The old port handle is no longer in use and should be freed using PHF.

Example

Command:
PHF 01

Reply:
OKAYA896
This frees port handle 01, so it is no longer assigned.

PHINF

Returns port handle status, information about the tool associated with the port handle, and the physical location of a port handle.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

All modes

Prerequisite Command

PHSR (page 121) or PHRQ (page 119)

Syntax

```
PHINF<SPACE><Port Handle><Reply Option><CR>
```

Parameter	Description
`Port Handle`	2 hexadecimal characters
`Reply Option`	Optional. Specifies which information will be returned. If no reply option is specified, the system returns information for reply option 0001. The reply options are hexadecimal numbers that can be OR'd. If multiple reply options are used, the replies are returned in order of increasing option value. **Valid Values** • 0001: Tool information (default) • 0002: Wired tool electrical information • 0004: Tool part number • 0008: Switch and visible LED information • 0010: Tool marker type and wavelength • 0020: Physical port location

Replies

Upon Success:
If there is a tool assigned tool definition file to the port handle:

```
<Reply Option 0001 Data><Reply Option 0002 Data>...<Reply Option 0020
Data><CRC16><CR>
```

Note **The physical location of a port handle is the only information available unless PHINF has been preceded by PINIT (page 124).**

If no tool definition file is assigned to the port handle:

```
UNOCCUPIED<CRC16><CR>
```

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Reply Option 0001 - Tool Information

<Reply Option 0001 Data> = <Tool Type><Manufacturer's ID><Tool Revision><Serial Number><Port Status>

Reply Component	Description	
Tool Type	Eight characters <Tool Type> = <Main Type><Number of Switches><Number of Visible LEDs><Reserved><Subtype>	
	Main Type	Two hexadecimal characters **Possible Values** • 01:Reference • 02: Probe • 03: Button box or foot switch • 04: Software-defined • 05: Microscope tracker • 06: Reserved • 07: Calibration device • 08:Tool Docking Station • 09: Isolation box • 0A: C-arm tracker • 0B: Catheter 0D to FF are reserved.
	Number of Switches	One character
	Number of Visible LEDs	One character
	Reserved	Two characters
	Subtype	Two characters
Manufacturer's ID	12 characters	
Tool Revision	Three characters	
Serial Number	Eight hexadecimal characters (32 bits) **Bit field** • Bits 0 to 9: Sequence number (one-based) • Bits 10 to 18: Day of year (zero-based, e.g. Jan 1 is day 0 and Dec 31 is day 364) • Bits 19 to 22: Month (zero-based) • Bits 23 to 31: Year (year is <current year> - 1900, e.g. the year 2009 is 109)	

Reply Component	Description	
`Port Status`	Two hexadecimal characters (8 bits) **Bit field** • Bit 0: Tool-in-port • Bit 1:Switch 1 closed • Bit 2: Switch 2 closed • Bit 3: Switch 3 closed • Bit 4: Port initialized • Bit 5: Port enabled • Bit 6: Reserved • Bit 7: Tool-in-port from current sensing	

Reply Option 0002 - Wired Tool Electrical Information

Reply Component	Description
`Reply Option 0002 Data`	Eight hexadecimal characters. Wired tool electrical information. The electrical current is tested for two conditions: over and under. An “over” current condition indicates that there is a short circuit in either the cable or the marker. An “under” current condition indicates that there is either a break in the cable or the marker has burnt out. **Bit field** • Bit 0: Marker A failed • Bit 1: Marker B failed • Bit 2: Marker C failed … • Bit 19: Maker T failed. • Bit 30: Under • Bit 31: Over Bits 20 to 29 are reserved.

You can test the electrical current of all the markers on a tool using TCTST (page 153).

Reply Option 0004 - Tool Part Number

Reply Component	Description
`Reply Option 0004 Data`	20 characters The part number of the tool.

Reply Option 0008 - Switch and Visible LED Information

Reply Component	Description
`Reply Option 0008 Data`	Two hexadecimal characters (8 bits) This option reports the information found in the tool description. It is not information sensed by the hardware. **Bit field** • Bit 0: Tool-in-port switch supported • Bit 1: Switch 1 supported • Bit 2: Switch 2 supported • Bit 3: Switch 3 supported • Bit 4: Tool tracking LED supported • Bit 5: LED 1 line 1 supported • Bit 6: LED 2 line 2 supported • Bit 7: LED 3 line 3 supported

Reply Option 0010 - Tool Marker Type and Wavelength

Reply Component	Description
`Reply Option 0010 Data`	Two hexadecimal characters (8 bits) **Bits 0 to 2 give information on the marker wavelength:** • 000: 9x0 nm • 001: 880 nm • 010: 930 nm • 100: 870 nm • 111: 850 nm **Bits 3 to 7 give information on the marker type:** • 00001: NDI active • 00010: NDI ceramic • 00011: Unknown active • 00100: Unknown passive • 00101: Passive sphere • 00110: Passive disc • 00111: NDI Radix Bits 00000 and 01000 to 11111 are reserved.

Reply Option 0020 - Physical Port Location

```
<Reply Option 0020 Data> = <Hardware Device><System Type><Tool Type>
  <Port Number><Reserved>
```

Reply Component	Description
`Hardware Device`	Eight characters For passive or active wireless tools this is the Position Sensor serial number. For Vega active tools, this is STB-0.
`System Type`	One character **Possible values:** Reserved

Reply Component	Description
`Tool Type`	One character **Possible values** • 0: Wired • 1: Wireless
`Port Number`	Two ASCII characters **Possible values** • 01 to 03: Used for Vega wired tools • 00: Used for Vega wireless tools
`Reserved`	Two characters

Usage Notes

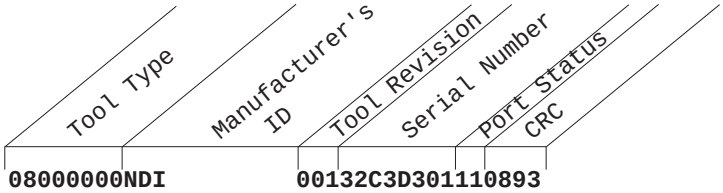
- The physical location of a port handle is the only information available unless PHINF has been preceded by PINIT (page 124) or PENA (page 109).
- Port handles for tools that have been disconnected will be reported as UNOCCUPIED and no additional information will be returned.
- Reply option 0001: For wired tools, bits 1, 2, and 3 in the port status report status.
- Reply option 0008: For wired tools, bits 1, 2, and 3 report status, and bits 5, 6, and 7 report LED status.
- Reply option 0010: A value of 010 for marker wavelength can be returned only for tools characterized using NDI 6D Architect version 2.02 or later. Tools characterized with earlier versions of NDI 6D Architect will have a value of 000 for a marker wavelength of 930 nm.
- Reply option 0040: This option is not supported by the hybrid Vega system.

Example

Command:

```
PHINF 040001
```

Reply:



PHRQ

Assigns a port handle to a tool.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

```
PHRQ<SPACE><Hardware Device><System Type><Tool Type><Port Number><Dummy Tool><CR>
```

Parameter	Description
`Hardware Device`	Eight characters. The hardware device must match the one returned by PHINF (page 114) reply option 0020, or use wild card characters (*). For active tools connected to the system, specifying all wildcards will default to hardware device STB-0 (the tool ports on the System Control Unit).
`System Type`	One character. **Valid Values:** Use a wild card character (*).
`Tool Type`	One character. This must be specified for wireless tools. **Valid Values** • 0 or *: Wired • 1: Wireless (passive or active wireless)
`Port Number`	Two characters. The physical port number where a wired tool is plugged in. This must be specified for wired tools. **Valid Values** • 01 to 03: Used for hybrid Vega wired tools • 00 or **: Used for wireless tools

Parameter	Description
Dummy Tool	Two characters If specified, will auto-generate a non-trackable dummy tool. Useful for 3D stray-marker tracking. If Tool Type is **Wired**, a value of 01 or 02 adds an active wired dummy tool. If Tool Type is **Wireless**, valid values are as below: • **: Do not load a dummy tool. Requires tool definition to be loaded with PVWR (page 137) subsequent commands. • 01: Adds passive dummy tool • 02: Adds active wireless dummy tool

Replies

Upon Success:

<Port Handle><CRC16><CR>

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Usage Notes

- Use PHRQ to assign a port handle to a wireless tool or to a wired tool that has neither a tool-in-port diode or a marker in position A of the tool wiring matrix. If a wired tool has a tool-in-port diode or a marker in position A of the tool wiring matrix, use PHSR (page 121) to detect the tool and assign it a port handle.
- **Wireless tools**: You must specify the tool type. All other parameters may be left as wild card characters (*).
- **Wired tools**: You must specify the port number. All other parameters may be left as wild card characters (*).
- After using PHRQ, you must use PVWR (page 137) to assign a tool definition file to the tool. If you do not assign a tool definition file to the tool, the port handle will be reported as unoccupied when it is initialized with PINIT (page 124) or PENA (page 109).

Example

Command:

PHRQ *********1****

Reply:

04D715

This requests a port handle for a wireless tool.

PHSR

Returns the number of assigned port handles and the port status for each one. Assigns a port handle to a wired tool.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

All modes

Prerequisite Command

INIT (page 100)

Syntax

PHSR<SPACE><Reply Option><CR>

Parameter	Description
Reply Option	Specifies which information will be returned. If no reply option is specified, the system returns information for reply option 00. The reply options cannot be OR'd. **Valid Values** • 00: Reports all allocated port handles (default) • 01: Reports port handles that need to be freed • 02: Reports port handles that are occupied, but not initialized or enabled • 03: Reports port handles that are occupied and initialized, but not enabled • 04: Reports enabled port handles

Replies

Upon Success:

<Number of Port Handles>
<1st Port Handle><1st Port Handle Status>
<2nd Port Handle><2nd Port Handle Status>
...
<nth Port Handle><nth Port Handle Status>
<CRC16><CR>

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Reply Component	Description
`Number of Port Handles`	Two hexadecimal characters. The number of allocated port handles of the type specified in the reply option. If no reply option is specified, the number returned is the total number of allocated port handles.
`n^th Port Handle`	Two hexadecimal characters. Specifies the port handle whose status follows.
`n^th Port Handle Status`	Three hexadecimal characters (12 bits). **Bit field:** • Bit 0: Occupied • Bit 1: Switch 1 closed • Bit 2: Switch 2 closed • Bit 3: Switch 3 closed • Bit 4: Initialized • Bit 5: Enabled • Bit 6: Reserved • Bit 7: Tool detected from current sensing • Bit 8 to 11: Reserved

Usage Notes

- When you send the PHSR command, the system will detect and assign port handles to any wired tools that do not already have a port handle assigned (i.e. any wired tools that were plugged in after the last PHSR call). It will then return the requested port handle information.
- The system will detect a wired tool if the tool has a tool-in-port diode, or a marker in position A of the tool wiring matrix. If you are using a wired tool that does not meet this criteria, you will need to request a port handle for the tool using PHRQ.
- If you unplug a wired tool while the system is in tracking mode, the port handle will be reported as “disabled” in the replies to the BX and TX commands. If you reconnect the tool, it will need a new port handle.
- If you connect a wired tool to the system while the system is in tracking mode, you will have to take the following steps before the system will report the tool:
  a) Exit tracking mode (TSTOP).
  b) Assign, initialize, and enable a port handle for the tool as outlined in Figure 3-1 on page 23.
  c) Re-enter tracking mode (TSTART).
- PHSR will report wireless tool ports as unoccupied if you have requested a port handle using PHRQ (page 119) but have not yet associated a tool definition file for the port handle (using PVWR (page 137)).
- To obtain a port handle for a wireless tool, use PHRQ.
- PHSR will only return the number of assigned port handles and their status when executed in tracking or diagnostic mode from a master connection, or when executed in any mode from a monitor connection.

Examples

Command:
PHSR

Reply:
001414
In this case, there are no occupied port handles.

Command:
PHSR

Reply:
0101031F1AF
In this case, there is one occupied port handle, which is initialized and enabled.

PINIT

Initializes a port handle.

Compatibility

Supported by the Position Sensor since G.001.

Deprecated. PENA now initializes tools that have not been initialized by PINIT.

Operating Mode

Setup

Prerequisite Command

PVWR (page 137) or PHSR (page 121)

Syntax

`PINIT<SPACE><Port Handle><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters

Replies

Upon Success:
`OKAY<CRC16><CR>`
or

`WARNING01` (Indicates that a non-fatal tool error has been encountered, e.g. a burnt out marker.)

or

`WARNING05` is returned when the system selects a default marker wavelength to track a tool (if the tool's tool definition file did not specify a marker wavelength).

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- If the tool description is drawn from a tool definition file that has been loaded using PVWR (page 137), initialization involves unpacking and verifying the tool definition file. This process is almost instantaneous.
- If the tool description is drawn from an SROM device, initialization involves reading, unpacking, and verifying the tool definition file contents, and testing electrical current through all the markers to detect burnt out markers. This process takes approximately two seconds if

successful, or several seconds longer if a problem is encountered and retries are attempted by the system.

- The port handle will still initialize when the system returns WARNING01 or WARNING05.
- The System Control Unit will load and parse active tool info when a tool is plugged in. PENA will load and parse passive tool info if not done so yet.

Example

Command:

```
PINIT 01
```

Reply:

```
0KAYA896
```

This initializes port handle 01.

PPRD

Reads data from the SROM device in a wired tool.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Used in hybrid Vega systems only.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`PPRD<SPACE><Port Handle><SROM Device Address><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters.
`SROM Device Address`	Four hexadecimal characters. **Valid Values:** 0x0000 to 0x07C0

Replies

Upon Success:
`<SROM Device Data><CRC16><CR>`
The SROM device data is 64 bytes (128 hexadecimal characters) of data.

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- The SROM device is a 2-KB write-once device that must be read in 64-byte chunks. An SROM device is considered blank if its contents are all 0xFFs.
- PPRD reads 64 bytes of data from the SROM device starting at a specified SROM device address.

Example

Command:
`PPRD 010000`

Reply:

```
0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0
123456789ABCDEF0123456789ABCDEF0123456789ABCDEF66A5
```

PPWR

Writes data to the SROM device in a wired tool.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Used in hybrid Vega systems only.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`PPWR<SPACE><Port Handle><SROM Device Address><SROM Device Data><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters.
`SROM Device Address`	Four hexadecimal characters. **Valid Values:** 0x0000 to 0x07C0
`SROM Device Data`	64 bytes (128 hexadecimal characters) of data

Replies

Upon Success:

`OKAY<CRC16><CR>`

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Usage Notes

- PPWR writes 64 bytes of data to the SROM device starting at a specified SROM device address.
- The data must be formatted into unsigned ASCII characters. Each byte of binary data can be represented by two hexadecimal characters, which are then sent to the system in ASCII (4 bits per ASCII character).
- The tool description section of tool SROM device is a 1-Kbyte, write-once area that must be written in 64-byte chunks. If the information being written to the system is less than 64 bytes in size, then the remainder of the chunk must be padded out with ones to maintain the 64-byte size before being written to the system. To write to the second 1-Kbyte section, use the PUWR command.

- An SROM device is considered blank if its contents are all 0xFFs.
- The recommended procedure to follow for updating an SROM device is:
  a) Read the contents of the SROM device using PPRD (page 126).
  b) Modify the data.
  c) Write the modified data back to the SROM device using PPWR (page 128).

Example

Command:

```
PPWR 0100C0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000009731
```

Reply:

```
0KAYA896
```

PSEL

Selects an SROM device as the target for reading or writing with PPRD or PPWR.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Used in hybrid Vega systems only.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`PSEL<SPACE><Port Handle><SROM Device ID><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters.
`Tool SROM Device ID`	16 characters. To determine the SROM device ID, use PSRCH (page 131).

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Example

Command:
`PSEL 010B3876530000005B`

Reply:
`OKAYA896`

PSRCH

Returns a list of valid SROM device IDs for a wired tool or GPIO device.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Used in hybrid Vega systems only.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

PSRCH<SPACE><Port Handle><CR>

Parameter	Description
Port Handle	Two hexadecimal characters

Replies

Upon Success:

<Number of SROM devices><SROM device 1 ID><SROM device 2 ID>...<SROM device 7 ID> <CRC16><CR>

Note **For a single tool or GPIO device, only the first SROM device ID is reported and the remainder are blank. The remaining six positions are reserved for special functionality.**

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Reply Component	Description
Number of SROM devices	One character
SROM device n ID	16 characters

Usage Notes

The SROM device has an embedded ID, which is a unique, 16 character, alphanumeric identifier. The SROM device ID is used to select an SROM device as a target with PSEL (page 130).

Example

Command:

```
PSRCH 01
```

Reply:

```
10B3876530000005B                                                    7FFF
```

There are 96 spaces between B and 7. The spaces are place holders for the SROM device ID numbers 2 to 7.

PURD

Reads data from the user section of the SROM device in a wired tool.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Used in hybrid Vega systems only.

Operating Mode

All modes

Prerequisite Command

INIT (page 100)

Syntax

`PURD<SPACE><Port Handle><User SROM Device Address><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters
`User SROM Device Address`	Four hexadecimal characters **Valid Values:** 0x0000 to 0x03C0

Replies

Upon Success:
`<SROM Device Data><CRC16><CR>`
The SROM device data is 64 bytes (128 hexadecimal characters) of data.

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- The SROM device is automatically selected as the reading target when this command is issued, so you do not need to find and specify the SROM device ID. The SROM device address has an implied offset in the command which places the user information at the correct SROM device address.
- The PURD command returns 64 bytes of data at a time.

Example

Command:
`PURD:010000`

Reply:

```
0022446688AACCEE0022446688AACCEE0022446688AACCEE0022446688AACCEE0022446688AACCEE0
022446688AACCEE0022446688AACCEE0022446688AACCEE3CC0
```

PUWR

Writes data to the user section of the SROM device in a wired tool.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Used in hybrid Vega systems only.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

```
PUWR<SPACE><Port Handle><User SROM device address><User SROM device
Data><CR>
```

Parameter	Description
`Port Handle`	Two hexadecimal characters
`User SROM device address`	Four hexadecimal characters **Valid Values:** 0x0000 to 0x03C0
`User SROM device data`	64 bytes of data to write (128 hexadecimal characters)

Replies

Upon Success:

`OKAY<CRC16><CR>`

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Usage Notes

- The SROM device is automatically selected as the reading target when this command is issued, so you do not need to find and specify the SROM device ID. The SROM device address has an implied offset in the command which places the user information at the correct SROM device address.
- The data must be formatted into unsigned ASCII characters. Each byte of binary data can be represented by two hexadecimal characters, which are then sent to the system in ASCII (4 bits per ASCII character).
- The user section of SROM devices is a 1-Kbyte, write-once area that must be written in 64-byte chunks. If the information being written to the system is less than 64 bytes in size, then

the remainder of the chunk must be padded out with ones to maintain the 64-byte size before being written to the system.

- The recommended procedure to follow for updating an SROM device is outlined below:
  a) Read the contents of the SROM device using PURD (page 133).
  b) Modify the data read.
  c) Write the modified data back to the SROM device using PUWR.

Example

Command:

```
PUWR 0100800000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000A927
```

Reply:

```
0KAYA896
```

PVWR

Assigns a tool definition file to a wireless tool, or overrides the SROM device in a wired tool.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

PHRQ (page 119) or PHSR (page 121)

Syntax

`PVWR<SPACE><Port Handle><Start Address><Tool Definition File Data><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters.
`Start Address`	Four hexadecimal characters. Increment the start address by 64 bytes with each chunk of data sent for a particular port handle. **Valid Values:** 0x0000 to 0x3FC0
`Tool Definition Data`	64 bytes (128 hexadecimal characters) of data

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- Use PVWR
  - To assign a tool definition file to a wireless tool after using PHRQ.
  - To assign a tool definition file to a wired tool, to override the SROM device in the tool.
  - To assign a tool definition file to a wired tool, to test the tool definition file before permanently recording the tool definition file onto the SROM device.
- The data must be formatted into unsigned ASCII characters. Each byte of binary data can be represented by two hexadecimal characters, which are then sent to the system in ASCII (4 bits per ASCII character.

- Data is sent to the system in 64-byte chunks (128 hexadecimal characters). The last chunk must be padded out with zeroes to maintain the 64-byte size before being written to the system.
- If a wireless tool port is the target of this command, the port becomes occupied when the first 64 bytes of information is written. Any previous initialization for the port is lost.
- Use PVWR to assign a tool definition file to a wireless tool after using PHRQ (page 119).
- After using PVWR, enable (PENA) the port handle in order to track the tool.
- To permanently write a tool definition file to an SROM device, use PPWR (page 128).

Example

Command:

```
PVWR 0200004E444900551C000001000000000000010100000001A419335A00000003000000030000
00000000400000000000000000000000000000000000000000000000000000
```

Reply:

```
0KAYA896
```

RESET

Resets the system.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`RESET<SPACE><Reset Option><CR>`

Parameter	Description
`Reset Option`	Optional. Specifies the type of reset. If no reset option is specified, the system performs a RESET 0. The reset options cannot be OR'd. **Valid Values** • 0: Generates a soft reset. Does not power cycle the Position Sensor. • 1: Performs a board-level reset of all hardware devices.

Replies

Upon Success:
`RESET<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Example

Command:
`RESET 0`

Reply:
`RESETBE6F`

SAVE

Saves all non-volatile user parameters that have been changed.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

SAVE<SPACE><CR>

Replies

Upon Success:
OKAY<CRC16><CR>

On Error:
ERROR<Error Code><CRC16><CR>
For error code definitions, see page 181.

Usage Notes

- To restore the user parameters to factory default values, use DFLT (page 89). To save the user parameters at their factory default values, use the SAVE command after using the DFLT command.
- To set user parameter values, use SET (page 141).
- For more information on user parameters, see “User Parameters” on page 25.
- If the Password Protection keyed feature is enabled, before you can save user parameters, you must enter the correct password.

Example

Command:
SAVE

Reply:
OKAYA896

SET

Sets user parameter values.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`SET<SPACE><User Parameter Name>=<Value><CR>`

Parameter	Description
`User Parameter Name`	A case-sensitive string identifying the name of the user parameter.
`Value`	The value to set. Numerical values are decimal unless preceded by 0x. For Boolean values, 1 is true and 0 is false.

Replies

Upon Success:

`OKAY<CRC16><CR>`

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Usage Notes

- To view a list of user parameters and their current values, use `GET *`. For a description of the user parameters, use `GETINFO *`.
- User parameter values set using the SET command persist until the system is reset or initialized. To save the user parameter values, use SAVE (page 140). To reset user parameters to their default values, use DFLT (page 89).
- User parameter names are case-sensitive.
- For more information on user parameters, see “User Parameters” on page 25

Example

Command:

```
SET Param.Tracking.Sensitivity=1
```

Reply:

0KAYA896

This sets the infrared light sensitivity level to level 1 on the first Position Sensor in the configuration.

SFLIST

Returns information about the supported features of the system.

Compatibility

Supported by the Position Sensor since G.001.

Deprecated.

Operating Mode

Setup, diagnostics or tracking

Syntax

`SFLIST<SPACE><Reply Option><CR>`

Parameter	Description
`Reply Option`	Specifies which information will be returned. The reply options cannot be OR'd. **Valid Values** • 00: Summary of supported features • 01: Number of active tool ports • 02: Number of wireless tool ports • 03: Number of measurement volumes and wavelengths; volume shapes and supported wavelengths • 04: The number of wired tool ports available which support tool-in-port detection from current sensing • 05: Number of active wireless tools

The reply options cannot be OR’d.

Replies

Upon Success:
`<Reply Option n Data><CRC16><CR>`

On Error:
ERROR<`Error Code`><CRC16><CR>
For error code definitions, see page 181.

Reply Component	Description
`Reply Option n Data`	The data specific to the requested reply option. See the reply option information below for details: • Reply option 00: Summary of supported features • Reply option 01: Number of active tool ports • Reply option 02: Number of wireless tool ports • Reply option 03: Number of characterized measurement volumes and wavelengths; volume shapes and supported wavelength • Reply option 04: The number of wired tool ports available which support tool-in-port detection from current sensing • Reply option 05: Number of active wireless tools

Reply Option 00 - Supported Features Summary

Reply Component	Description
`Reply Option 00 Data`	Eight hexadecimal characters (32 bits). **Bit field** • Bit 0: Active tool ports available • Bit 1: Passive tool ports available • Bit 2: Multiple volume characterization parameters supported • Bit 3: Tool-in-port from current sensing available • Bit 4: Active wireless tool ports available Bit 5 and bits 7 to 31 are reserved.

Reply Option 01 - Number of Active Tool Ports

Reply Component	Description
`Reply Option 01 Data`	One hexadecimal character. The number of wired tool ports.

Reply Option 02 - Number of Wireless Tool Ports

Reply Component	Description
`Reply Option 02 Data`	One hexadecimal character. The number of wireless tool ports, up to a maximum of 15 (the highest number that can be represented in one hexadecimal digit). To find out the actual number of wireless tool ports, read the parameters `Features.Tools.Passive Ports` (for passive wireless) and `Features.Tools.Wireless Ports` (for active wireless).

Reply Option 03 - Volumes

Note **SFLIST is deprecated; use the `Features.Volume` parameters instead. See** Features User Parameters on page 45.

```
<Reply Option 03 Data> =
  <Number of Volumes>
  <1st Shape Type><1st Shape Parameter><1st Number of Wavelengths
  Supported><1st Supported Wavelengths><LF>
...
  <nth Shape Type><nth Shape Parameter><nth Number of Wavelengths
  Supported><nth Supported Wavelengths><LF>
```

Reply Component	Description
`Number of Volumes`	One hexadecimal character.
`nth Shape Type`	One hexadecimal character. **Possible Values** • 5: Extended Pyramid Shape (volume names: “Pyramid”, “Extended Pyramid”) • 7: Arc Shape (volume name: “Vicra”. Not supported by Vega PSUs)
`nth Shape Parameter`	Ten parameters, seven characters each (a sign and six digits, with an implied decimal in the position XXXX . XX)
`nth Number of Wavelengths Supported`	One hexadecimal character
`nth Supported Wavelengths`	One character per wavelength supported. **Possible Values** • 0: 930 nm (see “Usage Notes” on page 148) • 1: 880 nm • 4: 870 nm • 7: 850 nm

Reply Option 04 - Number of Active Tool Ports Supporting Tool-in-Port Detection From Current Sensing

Reply Component	Description
`Reply Option 04 Data`	One hexadecimal character

Reply Option 05 - Number of Active Wireless Ports

Reply Component	Description
`Reply Option 05 Data`	One hexadecimal character

Vega System - Shape Parameters

Note **SFLIST is deprecated; use the `Features.Volumes` parameters instead. See** Features User Parameters on page 45.

For the pyramid measurement volume, <Shape Parameter> in reply option 03 returns the following values (illustrated in Figure 5-2):

Shape Parameter	Value	Description
D1	-2400 mm	*z*-coordinate of back of volume
D2	-1532 mm	*z*-coordinate where sides of volume change slope
D3	-950 mm	*z*-coordinate of front of volume
D4	572 mm	Half width of volume at *z* = D2
D5	398 mm	Half height of volume *z* = D2
D6	0569.46	Slope of front part of volume sides in the *yz*-plane (scaled by 1000)
D7	0243.03	Slope of back part of volume sides in the *yz*-plane (scaled by 1000)
D8	0297.73	Slope of volume top and bottom in the *xz*-plane (scaled by 1000)
D9	9999.99 mm	Maximum half width of volume (unrestricted)
D10	9999.99 mm	Maximum half height of volume (unrestricted)

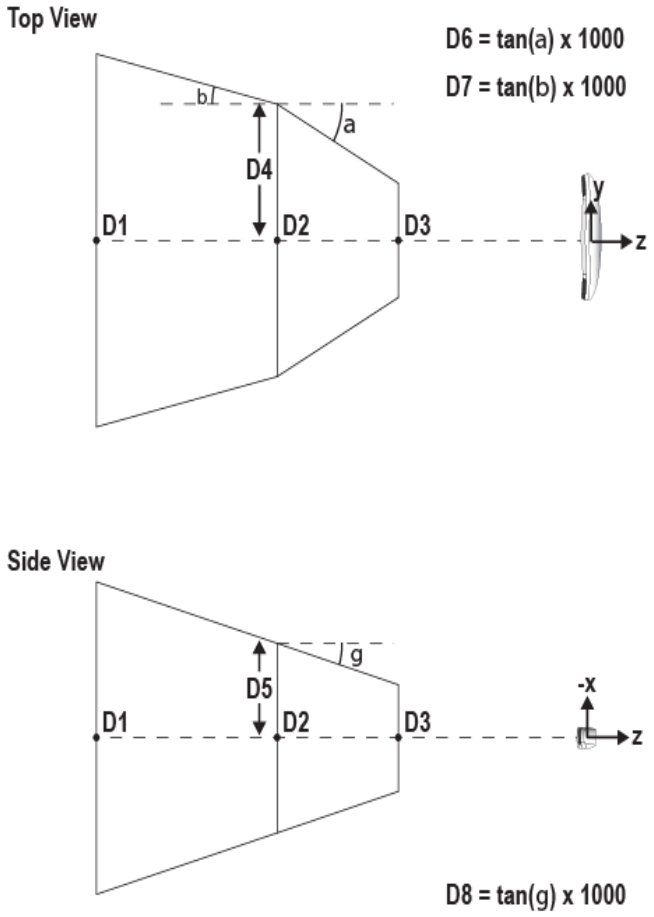


Figure 5-2 Pyramid Volume Parameters (Vega)

For the extended pyramid measurement volume, <Shape Parameter> in reply option 03 returns the following values (illustrated in Figure 5-2 and Figure 5-3):

Parameter	Value	Description
D1	-3000 mm	*z*-coordinate of back of volume
D2	-1532 mm	*z*-coordinate where sides of volume change slope
D3	-950 mm	*z*-coordinate of front of volume
D4	572 mm	Half width of volume at *z* = D2
D5	398 mm	Half height of volume *z* = D2
D6	0569.46	Slope of front part of volume sides in the *yz*-plane (scaled by 1000)
D7	0243.03	Slope of back part of volume sides in the *yz*-plane (scaled by 1000)
D8	0297.73	Slope of volume top and bottom in the *xz*-plane (scaled by 1000)
D9	9999.99 mm	Maximum half width of volume (unrestricted)
D10	735 mm	Maximum half height of volume

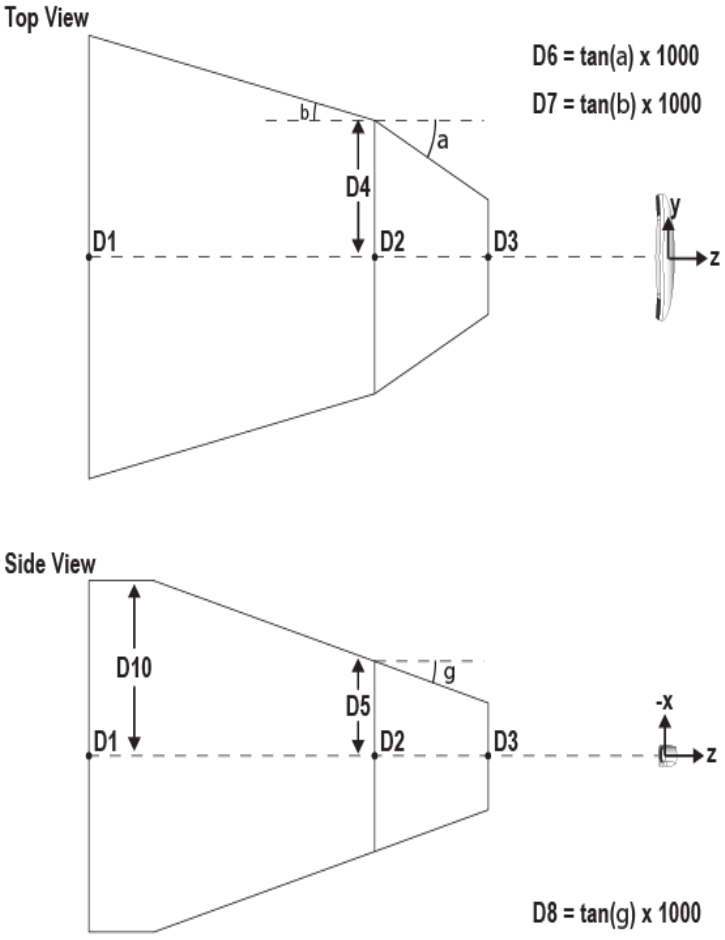


Figure 5-3 Extended Pyramid Volume Parameters (Vega)

Usage Notes

- Use both the shape type and the shape parameters to represent the characterized measurement volume graphically. There may be multiple volumes with the same shape type. All volumes of the same shape type use the shape parameters the same way.
- **Reply option 03**: A characterized measurement volume that supports wavelength value 0 (930 nm) supports the wavelength values of 000 (9*x*0 nm) and 010 (930 nm) returned with PHINF (page 114).

Examples

Command:

```
SFLIST
```

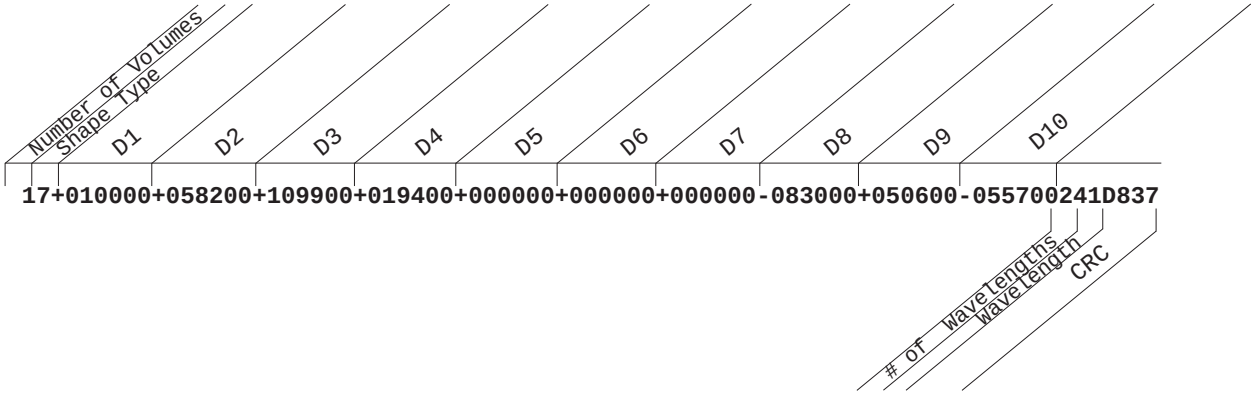
Reply:

```
0000003FEEEC
```

Command:

```
SFLIST 03
```

Reply:



STREAM

Initiates a streaming response to a command.

Compatibility

Supported by the Position Sensor since G.003.

Operating Mode

All modes

Syntax

`STREAM<SPACE><Parameter><CR>`

Parameter	Description
`[--id=<ID string>]`	Optional. An ID string that will be returned in the stream response header. If omitted, the stream uses the command string as the ID. The ID must be unique to the given connection. If it contains spaces, it must be quoted. **Note:** If a new stream is created with the same ID as an existing stream, the existing stream is terminated and replaced with the new stream.
`[--interval=<frame count>]`	Optional. An integer frame count interval that is used to limit the response rate. If omitted, data is sent at the frame rate of the tracking system. See “Usage Notes” on page 150.
`[--diff=true]`	Optional. Indicates that only the differences between the current response and the last streamed response will be sent. If omitted, Vega streams all responses. Currently, this option is valid for ASCII responses to commands such as GET and GETINFO.
`[--protocol=UDP|TCP]`	Optional. Specifies the protocol for the stream. If omitted, the stream is created as a TCP stream. If you set this parameter, you must specify the port using the `--port` parameter.
`[--port=[<ip address>:]<port>]`	Required if you specify the `--protocol` parameter; otherwise, this parameter is optional. Specifies the destination of the stream. If omitted for a TCP stream, the stream is created on the existing control connection. The IP address is optional and can be an IPv4 address or a host name. If omitted, it uses the address of the control connection.
`[--cmd=]<command to stream>`	The command string exactly as it would be if issued separately.

Replies

Upon Success:

`OKAY<CRC16><CR>`

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Note **The response is binary and is similar to the BX binary response with a different header signature. To maintain compatibility with the serial protocol, all binary replies are in little endian format rather than network byte order. The header signature is a 2 byte little endian code. The first byte is 0xD4 and the second byte is 0xB5.**

```
B5D4<Stream ID Length(2 bytes)><Stream ID><Header CRC16><command reply>
```

`<command reply>` **is the unmodified reply for the command that is being streamed exactly as it would appear if the command were given separately without streaming.**

Usage Notes

- For details on data streaming format, see “Data Streaming” on page 7.
- NDI does not recommend using the "--interval" parameter with tracking commands (BX2, BX, TX) since the reporting rate for these commands depends on several factors, such as frame frequency, track frequency, number and types of tools loaded, and averaging depth. If you use these commands in combination with the "--interval" parameter, the parameter specifies the lowest possible interval at which data is reported, but the actual reporting interval can be larger.

Example

Command:

```
STREAM --id="1" --cmd="BX 0803"
```

Reply:

```
0KAYA896
```

The following would continue with updated data replies until `USTREAM` is issued:

```
d4b5010031860ac4a50300...
d4b5010031860ac4a50300...
d4b5010031860ac4a50300...
...
```

SYSLOG

Writes data to the device's event log.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

`SYSLOG<SPACE>\<Device Name>\<Category>=<Message><CR>`
or

`SYSLOG<SPACE><Category>=<Message><CR>`

Parameter	Description
`Device Name`	Selects a hardware device to write to. The device name is ignored if it is specified. For information on device names, see "Device Names" on page 26.
`Category`	A string, up to 12 characters. Specifies the log entry category or source. If you enter more than 12 characters, the system truncate the category to 12 characters.
`Message`	A string, up to 256 characters. Contains the log message. If you enter more than 256 characters, the system truncates the message to 256 characters.

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- The system log in each hardware device is intended to record events central to the life of the device. The system automatically records events such as updates, bump sensor events, and hardware faults in the log.
- To read the log, use GETLOG (page 98).

Example

Command:

```
SYSLOG Test=This is a SYSLOG test!
```

Reply:

```
OKAYA896
```

TCTST

Returns diagnostics on the active markers of a wired tool.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

PINIT (page 124)

Syntax

`TCTST<SPACE><Port Handle><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters

Replies

Upon success:
<Marker A Current><Marker B Current>...<Marker T Current><CRC16><CR>

On Error:
ERROR<`Error Code`><CRC16><CR>
For error code definitions, see page 181.

Reply Component	Description
`Marker n Current`	Two hexadecimal characters. The electrical current of the markers.

Usage Notes

- If the result is less than 0x0A, either there is no marker or there is a problem with the diode that has caused an open circuit.
- If the result is greater than 0x0A, the marker is either okay or it has short-circuited. The exact value cannot be predicted as it depends upon the System Control Unit and the tool design (cable length, number of markers, and marker configuration). This value should be determined on a historical basis for each particular tool design.
- You cannot test a visible LED, since the System Control Unit cannot reliably test the low current of an LED because the LED current result may be corrupted from electrical noise.

Example

Command:

```
TCTST 01
```

Reply:

```
940000000094010000009200000000940000000DF24
```

TSTART

Starts Tracking mode.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`TSTART<SPACE><Reply Option><CR>`

Parameter	Description
`Reply Option`	80 (Optional)

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

The frame number is reported in reply option 0001 of TX (page 159) and BX (page 69). In the Vega system, the frame number is derived from the PTP time and reply option 80 is ignored.

To facilitate the retrieval of tracking data in a monitor connection, TSTART returns OKAY when in tracking mode.

Example

Command:
TSTART

Reply:
OKAYA896

TSTOP

Stops tracking mode.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Tracking

Prerequisite Command

TSTART (page 155)

Syntax

TSTOP<SPACE><CR>

Replies

Upon Success:
OKAY<CRC16><CR>

On Error:
ERROR<Error Code><CRC16><CR>
For error code definitions, see page 181.

Usage Notes

If executed from the Setup mode, it returns OKAY.

Example

Command:
TSTOP

Reply:
OKAYA896

TTCFG

Sets up a configuration for a wired tool, so that you can test the tool without using a tool definition file.

Compatibility

Supported by the Position Sensor since G.001.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`TTCFG<SPACE><Port Handle><CR>`

Parameter	Description
`Port Handle`	Two hexadecimal characters

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

- TTCFG internally sets up a test configuration for a wired tool, so that it can be tested without having a tool definition file. This is useful for testing the wiring in the tool before characterizing the tool. For example, after sending TTCFG, you can:
  - use TCTST to test the current
  - in diagnostic mode, use IRED to individually activate the markers.
- After sending the TTCFG command, you must enable (PENA) the port handle before using any other commands that list these as prerequisites.
- With the test configuration, the tool cannot be tracked.

Example

Command:
`TTCFG 0A`

Reply:
0KAYA896

TX

Returns the latest tool transformations, individual marker positions, and system status in text format.

Compatibility

Supported by the Position Sensor since G.001.

BX2 is recommended for new application development.

Operating Mode

Tracking

Syntax

`TX<SPACE><Reply Option><CR>`

Parameter	Description
`Reply Option`	Optional. Specifies which information to return. If no reply option is specified, the system returns information for reply option 0001. The reply options are hexadecimal numbers that can be OR'd. If multiple reply options are used, the replies are returned for each port handle in order of increasing option value, with the following exceptions: • Reply option 0800 is not reported separately from the other options; it simply enables the system to return certain information in the other options. • Reply option 1000 is reported after all handle-specific options but before the <system status> and <CRC16>. **Valid Values** • 0001: Transformation data (default) • 0002: Tool and marker information • 0004: 3D position of a single stray active marker • 0008: 3D positions of markers on tools • 0800: Transformations not normally reported • 1000: 3D positions of stray passive markers

Replies

Upon Success:

```
<# of Handles><Handle 1><Reply Opt 0001 Data>...<Reply Opt 0008 Data><LF>
  ...
  <Handle n><Reply Option 0001 Data>...<Reply Option 0008 Data><LF>
  <Reply Option 1000 Data><System Status><CRC16><CR>
```

Note **If the port handle is disabled, the system returns the string DISABLED instead of <Reply Option 0001 Data>...<Reply Option 0008 Data>.**

On Error:

`ERROR<Error Code><CRC16><CR>`

For error code definitions, see page 181.

Reply Component	Description
`Number of Handles`	Two hexadecimal characters. The number of port handles information is returned for.
`Handle n`	Two hexadecimal characters. The port handle whose information follows.
`Reply Option m Data`	The data specific to the requested reply option. For details, see the reply option information below: • Reply option 0001: Transformation data (default) • Reply option 0002: Tool and marker information) • Reply option 0004: Latest 3D position of single, stray, active marker • Reply option 0008 (3D position of markers on tools) • Reply option 0800 (reporting all transformations) • Reply option 1000 (3D position of stray passive markers)
`System Status`	Four hexadecimal characters (16 bits) The status of the system. Bit field • Bit 0 System communication synchronization error • Bit 3: Recoverable system processing exception. • Bit 6: A port handle has become occupied • Bit 7: A port handle has become unoccupied • Bit 8: Diagnostic pending • Bit 9: Temperature (system is not within operating temperature range) • Bit 10: Hardware configuration changed (e.g. VCU or SCU connected or disconnected) Bits 1, 2, 4, 5, and 11 to 15 are reserved.

Note **The "diagnostic pending" bit is set whenever an alert is detected or cleared. To view the alerts status and clear the diagnostic pending bit, use GET (page 94) to check the `Info.Status.New Alerts` user parameter for every hardware device in the system. For more details, see "Usage Notes" on page 76. Note that for API revision G.001.003 and earlier, the diagnostic pending bit did not indicate when an alert was cleared.**

Reply Option 0001 - Transformation Data

`<Reply Option 0001 Data> = <Q0><Qx><Qy><Qz><Tx><Ty><Tz><Error><Port Status> <Frame Number>`

`or`

`<Reply Option 0001 Data> = MISSING<Port Status><Frame Number>`

Reply Component	Description
`Q0, Qx, Qy, Qz`	Six characters each (a sign and five decimal digits, with an implied decimal in the position X . XXXX). Rotational component of the transformation, quaternion, unitless. The value for Q0 is always non-negative.

Reply Component	Description
`Tx, Ty, Tz`	Seven characters each (a sign and six decimal digits, with an implied decimal in the position XXXX . XX). Translational components of the transformation, in mm.
`Error`	Six characters (a sign and five decimal digits, with an implied decimal in the position X . XXXX). The error is an RMS value, given in mm. It is the result of the least squares minimization between the marker geometry in the tool definition file and the data from the tool's markers measured by the system.
`Port Status`	Eight hexadecimal characters (32 bits). **Bit field** • Bit 0: Occupied • Bit 1: Switch 1 closed • Bit 2: Switch 2 closed • Bit 3: Switch 3 closed • Bit 4: Initialized • Bit 5: Enabled • Bit 6: Out of volume • Bit 7: Partially out of volume • Bit 8: Algorithm limitation (processing requires more buffer than is available) • Bit 9: IR interference (a large bright IR object) • bit 12: Processing exception (same as tool information bit 7 in reply option 0002) • Bit 14: Fell behind while processing (same as tool information bit 3 in reply option 0002) • Bit 15: Data buffer limitation (too much data; for example, too many markers) Bits 10, 11, 13, and 16 to 31 are reserved.
`Frame Number`	Eight hexadecimal characters. The frame number is an internal counter related to data acquisition, which is derived from the PTP time. The frame number corresponds to the frame in which the raw data, used to calculate the accompanying transformation, was collected.

Note **If a transformation cannot be determined, the system returns the string MISSING, followed by the port status and frame number.**

Reply Option 0002 - Tool and Marker Information

```
<Reply Option 0002 Data> = <Tool Information><Marker Information>
```

Reply Component	Description
`Tool Information`	Two hexadecimal characters (8 bits) **Bit field** • Bit 0: Bad transformation fit • Bit 1: Not enough acceptable markers for transformation • Bit 2: IR interference—environmental IR is interfering with the system (combination of port status bits 9 and 15 in reply option 0001) • Bit 3: Fell behind while processing (same as port status bit 14 in reply option 0001) • Bits 4 to 6: Tool face used • Bit 7: Processing exception (same as port status bit 12 in reply option 0001)

Reply Component	Description
`Marker Information`	20 hexadecimal characters (one per marker) See below for an example. **Possible Values** • 0: Not used; it was missing • 1: Not used; it exceeded the maximum marker angle • 2: Not used; it exceeded the maximum 3D error for the tool • 3: Used to calculate the transformation • 4: Used to calculate the transformation, but it is out of volume • 5: Not used; it was outside the characterized measurement volume and was not needed to calculate a transformation.

Example - Marker Information: A tool with markers located at T, R, C, and A, where all four markers were used to determine the calculation, would have the reply 30300000000000000303, as illustrated:

Marker Letter	T	S	R	Q	...	D	C	B	A
Reply Char (Hex)	3	0	3	0	...	0	3	0	3

Reply Option 0004 - 3D Position of Single Stray Active Marker

`<Reply Option 0004 Data> = <Status><Tx><Ty><Tz>`

or

`<Reply Option 0004 Data> = <Status>`

Reply Component	Description
`Status`	Two hexadecimal characters (8 bits) The status of the stray active marker. A stray marker on an active tool is not fixed with respect to the other markers that make up the tool. **Bit field** • Bit 0: Valid stray active marker • Bit 1: Marker is missing • Bit 2: Reserved • Bit 3: Marker is out of volume • Bits 4 to 7Reserved
`Tx, Ty, Tz`	Seven characters each (a sign and six decimal digits, with an implied decimal in the position XXXX . XX). Position of the marker, reported in the coordinate system of the Position Sensor. The marker position is reported only if the marker status is “valid,” or if the status is “out of volume” and reply option 0800 is used.

Note **If no stray active marker is defined (for example, for wireless port handles or wired tools with no stray marker defined in the tool definition file), the status is 00, and no position information is returned. If the marker is missing, or if the marker is out of volume and reply option 0800 is not used, the system returns only the status.**

Reply Option 0008 - 3D Position of Markers on Tools

<Reply Option 0008 Data> = <Number of Markers><Out of Volume><T_{xn}><T_{yn}><T_{zn}>

Reply Component	Description
Number of Markers	Two hexadecimal characters. Number of markers used in tool transformations.
Out of Volume	One hexadecimal character per 4 markers (one bit per marker). The bit is set when the marker is outside the characterized measurement volume (see example below). Reply size = (number of markers)/4, rounded up to the nearest integer.
Txn, Tyn, Tzn	Seven characters each (a sign and six decimal digits, with an implied decimal in the position XXXX . XX). Position of the n^{th} marker, reported in the coordinate system of the Position Sensor. The system reports the positions of markers used in tool transformations and markers that exceeded the maximum marker angle or maximum 3D error specified in the tool definition file. For more information, see “Usage Notes” on page 165. **Reply size:** If reply option 0800 is not used, reply size = (21 characters) x (number of markers inside the characterized measurement volume). If reply option 0800 is used, reply size = (21 characters) x (total number of markers).

Example - Out of Volume: The information is returned in the format illustrated in the following example: one bit per marker, in little endian format. In this example there are nine markers, all of which are out of volume:

Marker Number				9	8	7	6	5	4	3	2	1
Bit Field	0	0	0	1	1	1	1	1	1	1	1	1
Reply	1				F				F			

Reply Option 0800 - Reporting All Transformations

This option enables the reporting of transformations or translations in situations where translations or transformations are calculated, but by default are not reported by the system. Such situations include:

- The tool or marker is outside of the characterized measurement volume.
- The bump sensor has been tripped.
- The system is outside of the optimal operating temperature range.
- Other system conditions are not ideal. For a full list of these conditions, see “Alerts User Parameters” on page 28.

This reply option must be OR’d with reply option 0001 to obtain transformations for tools in the situations listed above. It must be OR’d with reply options 0004, 0008, or 1000 to obtain position information for markers in the situations listed above.

When using reply option 0800 with the TX command, you must take appropriate action to detect the events listed above, and determine whether they are detrimental to your application. If one or more of the events listed above occurs, reply option 0800 enables the system to return data that may lead to inaccurate conclusions and may cause personal injury.

Appropriate action to detect the events listed above includes:

- reading the out-of-volume flag in reply options 0001 and 0002 when tracking tools
- reading the out-of-volume information in reply options 0004, 0008, and 1000 when tracking stray markers
- reading the temperature flag in the system status
- reading the diagnostic pending bit in the system status
- reading the `Info.Status.New Alerts` user parameter for every hardware device in the system when the diagnostic pending bit is set. For details, see "Usage Notes" on page 165.

Reply Option 1000 - 3D Position of up to 50 Stray Passive Markers

`<Reply Option 1000 Data> = <Number of Markers><Out of Volume><Txn><Tyn><Tzn>`

Reply Component	Description
`Number of Markers`	Two hexadecimal characters. Number of stray markers.
`Out of Volume`	One hexadecimal character per four markers (one bit per marker). The bit is set when the marker is outside the characterized measurement volume (see example below). Reply size = (number of markers)/4, rounded up to the nearest integer.
`Txn, Tyn, Tzn`	Seven characters each (a sign and six decimal digits, with an implied decimal in the position XXXX . XX). Position of the n^{th} marker, reported in the coordinate system of the Position Sensor. **Reply size:** If reply option 0800 is not used, reply size = (21 characters) x (number of markers inside the characterized measurement volume). If reply option 0800 is used, reply size = (21 characters) x (total number of markers).

Note **To activate the illuminators on the Position Sensor, at least one passive port handle must be enabled. If no passive port handles are enabled, <Number of Markers> returns 00 and no other data is returned.**

Stray passive markers are defined as markers which are not used to calculate any of the transformations for any enabled, passive tools. Stray active wireless tool markers are not reported.

Example - Out of Volume The information is returned in the format illustrated in the following example: one bit per marker, in little endian format. In this example there are nine markers, all of which are out of volume:

Marker Number				9	8	7	6	5	4	3	2	1
Bit Field	0	0	0	1	1	1	1	1	1	1	1	1
Reply	1				F				F			

Usage Notes

- The TX format is easier to parse than the binary format; it is useful when troubleshooting or observing data as it is collected. For replies in binary format, use BX2 (page 77).
- By default, transformations will not be reported if the tool is either partially or wholly out of the characterized measurement volume, if the bump sensor has been tripped, or if the system is outside of the optimal operating temperature range. To report these transformations, you must use reply option 0800 OR'd with the desired reply option(s). The accuracy of these transformations is unknown.
- Reply Option 0001:
  - When the "diagnostic pending" bit is set in the system status, use GET (page 94) to read the `Info.Status.New Alerts` user parameter for every hardware device in the system. The act of reading these parameters clears the parameters and the "diagnostic pending" bit. For more information on alerts and their associated user parameters, see "Alerts User Parameters" on page 28.
  - For wired tools, bits 1, 2, and 3 in the port status report switch status.
- Reply Option 0008: Markers are returned in alphabetical order according to how they are labelled in the tool definition file. For example, for a tool with markers labelled A, G, M and S, the system will return the marker positions in the order A G M S. Reply option 0008 only returns data for markers that the system detects. To identify which marker is which, compare the reply option 0008 data to the data returned with reply option 0002. The marker order is the same for both replies; each marker that does not have a <marker information> status of 0 ("missing") in reply option 0002 corresponds to a marker in reply option 0008
- Reply Option 1000: At least one passive tool definition file must be initialized and enabled in order for the system to return stray passive marker data. If no passive tool definition files are enabled, this reply option will return 00.
- System Status: In API revision G.001.004 and later, the diagnostic pending bit (bit 8) is set whenever an alert is detected or cleared. In API revision G.001.003 and earlier, the diagnostic pending bit is set only when an alert is detected.
- Reply Option 0002: Marker information value 2 means that the marker was not used because it exceeded the maximum 3D error for the tool.

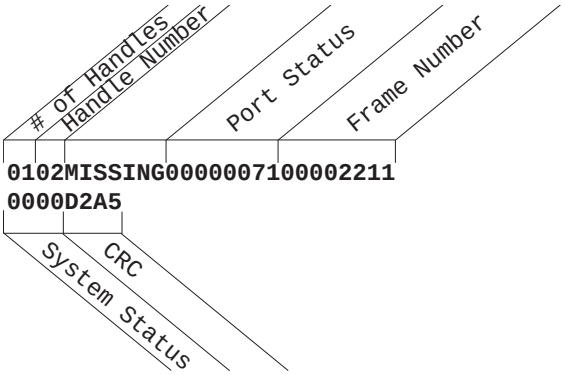
Examples

Example 1

Command:

```
TX 0001
```

Reply:



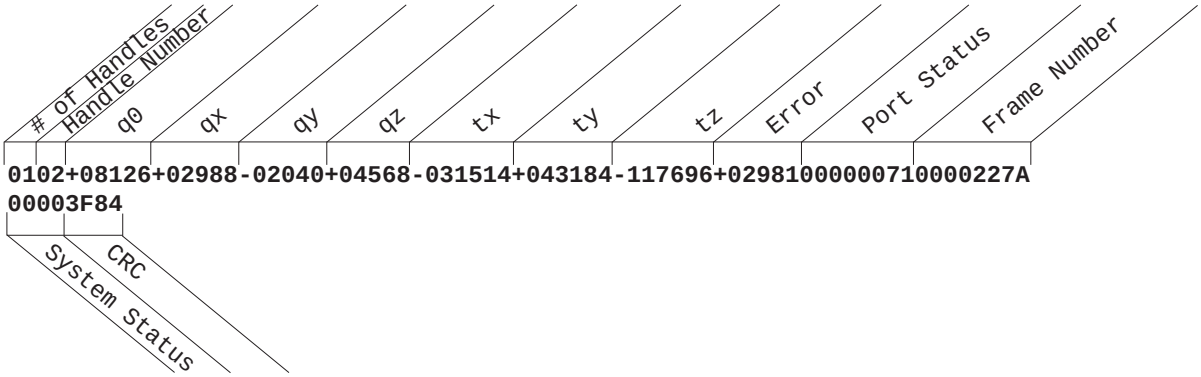
The system reports that there is one tool, which is missing. Notice the port status, which indicates that the tool is occupied, initialized, enabled, and out of volume.

Example 2

Command:

```
TX 0801
```

Reply:



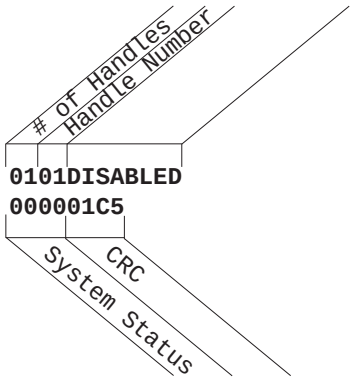
With the 0800 reply option applied, the system reports the missing tool. Notice the port status, which indicates that the tool is occupied, initialized, enabled, and out of volume.

Example 3

Command:

```
TX 0001
```

Reply:



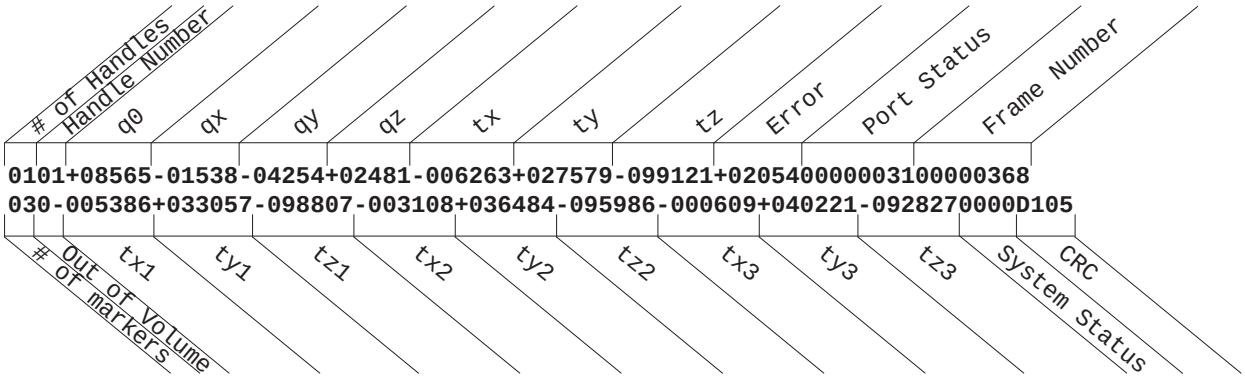
The system reports that there is one tool, whose port handle is disabled. It also reports the system status.

Example 4

Command:

```
TX 1001
```

Reply:



The system reports the transformation for one tool (first line of the reply), and the positions of three stray passive markers (second line of the reply).

USTREAM

Terminates one or all child streams.

Compatibility

Supported by the Position Sensor starting in G.003.

Operating Mode

All modes

Syntax

`USTREAM<SPACE><ID string><CR>`

Parameter	Description
`[--id=]<ID string>`	The ID for the stream. If it contains spaces, it must be quoted. The `--id=` identifier is optional.

Note **If the `STREAM` command did not set a unique identifier, the command string is used as the identifier for the stream and the ID string for the `USTREAM` command is the command string.**

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Note

- With firmware CFR008 and later, issuing the `USTREAM` command with no parameters terminates all existing child streams associated with the parent connection.

Example

Command:
`USTREAM --id="1"`

Reply:
`OKAYA896`
The stream of "d4b5010031860ac4a50300...." messages stops.

VCAP

Captures and returns IR image data from the IR sensors and/or the Vega Video Camera Unit.

Compatibility

Supported by the Position Sensor since G.003. Includes enhanced functionality since G.003.004.

Supported by the Video Camera since V.001.

VGET and VSNAP are recommended when using serial communications due to the latency introduced in VCAP when retrieving an image.

Operating Mode

Tracking

Syntax

VCAP<SPACE><Parameters><CR>

Parameter	Description	Sensor
`--action=capture|return|all`	Specifies whether new images should be only captured, only returned, or both captured and returned. When `capture` is specified, no image data is returned, regardless of any other parameter setting. The default is `all`.	IR + Video
`--type=IR|video|all`	Specifies whether the command action applies to the IR sensors, the video sensor, or both the IR and video sensors simultaneously. When `IR` is specified, video-specific options are ignored. When `video` is specified, IR-specific options are ignored. The default is `IR`.	IR + Video
`--frame=passive|active|activewireless|background|illuminated|all`	Specifies what type of frame in the IR frame sequence to return. When VCAP is sent with no parameters, the next available frame type is returned. The `Param.Tracking.Background Frame` and `Param.Tracking.Illuminated Frame` parameters must be set to 1 before frames are returned. When `all` is specified, all frame indexes of the specified frame type are returned.	IR
`--frameindex=<frame index>|all`	Specifies which frame in the IR frame sequence to return. This is useful when the system is configured with more than one frame of a particular type (e.g. two active frames) and only one of them needs to be returned. When VCAP is sent with no frame index, the next available frame type is returned. When `all` is specified, all frame indexes of the specified frame type will be returned.	IR
`--sensor=<sensor number>|all`	Specifies which IR sensor to capture an image from. By default, or if `all` is specified, both sensors are used. The left sensor (sensor 0) is returned first, followed by the right sensor (sensor 1).	IR

Parameter	Description	Sensor
`--format=RAW|TIFF|PGM`	Specifies the IR image format. By default, RAW is used.	IR
`--depth=<bits-per-pixel>`	Specifies the number of bits to use per pixel. Valid values are 1, 2, 4, 8 and 16. The default is 16.	IR
`--stride=<number of pixels>`	Specifies the pixel-read step size. For example, a stride of 4 means that every fourth pixel is returned. The default is 1 and every pixel is returned.	IR
`--sample=pixel|average|peak`	If stride is greater than 1, this specifies how to sample the intermediate pixels. The default is pixel, where intermediate pixels are ignored.	IR
`--area=<x, y, width, height>`	Specifies the area of the image to be returned. The maximum size of the image is 1920 x 1200. The default is to return the whole image. If the `stride` parameter is defined, the area returned will be a subset of the area that is defined.	IR
`--vid_format=JPEG`	Specifies the video image format. JPEG is the only option supported today. The default is JPEG.	Video
`--vid_exp_bracket_adjust=<incremental positive and negative EV adjustment per additional video image pair>`	Floating-point amount of EVs to adjust per incremental pair of under- and over-exposed video images. One (1) EV is equal to one (1) exposure step (or stop), corresponding to a doubling or halving of exposure. For example, increasing the EV by one(+1) will result in halving the exposure time and decreasing the EV by one(-1) will double the exposure. Ignored if exposure bracketing depth is set to 0. The maximum amount of EVs that may be adjusted by the end of the exposure bracketing depth is 5.0. The default is 0.33.	Video
`--vid_exp_bracket_depth=<number of pairs of video images to capture with increasing incremental adjusted exposure>`	If set to 1, three video images are captured at the following indexes and exposures: • Index 0: normal EV(t_0) • Index 1: normal EV + vid_exp_bracket_adjust * 1EV ($t_1 = t_0 \times 2^{-\text{vid\_exp\_bracket\_adjust}}$) • Index 2: normal EV – vid_exp_bracket_adjust * 1EV ($t_2 = t_0 \times 2^{\text{vid\_exp\_bracket\_adjust}}$) The maximum is 1, the default is 0.	Video
`--vid_read_index=<index of first captured video image to return>`	Index of captured video image at which to begin reading. The default is 0.	Video
`--vid_read_length=all|<number of captured video images to return>`	Number of captured video images to return, starting at the specified read index. Maximum is (1 + 2 * vid_exp_bracket_depth). The default is all.	Video

Replies

Upon Success:

`A5C8<4 byte Reply Length><command reply>` or

`A5C4<2 byte Reply Length><2 byte Header CRC><command reply><2 byte Data CRC>`

Note **The `<command reply>` payload is in the General Binary Format, which is documented in the section "General Binary Format" on page 5.**

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Image Data Component: 0x000A

Image Component Header			
Item Type	One byte	0=RAW, 1=PGM, 2=TIFF, 3=JPEG	IR+Video
Sensor	One byte	Sensor number (0-127 IR, 128 Video)	IR+Video
Frame Type	One byte	Frame type (see BX2)	IR
Frame Index	One byte	Frame sequence/read index	IR+Video
Frame Number	Four bytes	Frame number	IR
Trigger Threshold	Four bytes	Trigger threshold, percentage of full scale (float)	IR
Background Threshold	Four bytes	Background threshold, percentage of full scale (float)	IR
Exposure	Two bytes	Exposure in microseconds	IR
Stride	One byte	Pixel stride count	IR
Image Depth	One byte	Bits per pixel	IR+Video
Image Area	Eight bytes	X, Y, Width, Height (Two bytes each)	IR+Video
Meta data length (M)	Four bytes	Length of optional meta data. Must be multiple of 4	IR+Video
Meta data	M bytes	Optional meta data	IR+Video
Image Item		The image data	IR+Video

PGM format images have the following meta data embedded as comments:

frame_type = `<frame type>`

frame_number = `<frame number>`

sensor = `<sensor number>`

exposure = `<exposure time>`

trigger_threshold = `<% of full scale>`

background_threshold = `<% of full scale>`

stride = `<stride pixel count>`

depth = `<bits per pixel>`

area = <x,y,width,height>

Optional meta data for IR images:

timestamp = `<date time string>`: Date and time when IR image was captured.

Optional meta data for video images:

timestamp = <date time string>: Date and time when video image was captured.

bracket_index=<integer>: The index of the image in the exposure bracket sequence.

exposure = <exposure time us>: Exposure time of captured video image in microseconds. The value of the `Exposure` field in the Image Component Header is set to 0.

Examples

Command:
VCAP

Reply:
9.2 MB of data in GBF format, consisting of two image components (one for each sensor), each showing the entire image (1920 x 1200 pixels x 16 bits of gray scale) in RAW format.

Command:
```
vcap --sensor=0 --stride=2 --format=tiff --depth=8
```

Reply:
576,326 bytes of data in GBF format, consisting of one image component (for the left sensor), showing 960 x 600 pixels (sampling every second pixel in every second row of the entire image), in 8-bit gray scale, in TIFF format.

VER

Returns the firmware revision number of critical processors installed in the system.

Compatibility

Supported by the Position Sensor and System Control Unit since G.001.

Supported by the Video Camera since V.001.

Operating Mode

All modes

Syntax

VER<SPACE><Reply Option><CR>

Parameter	Description
Reply Option	Specifies which information will be returned. Reply options cannot be OR'd. **Valid Values** • 0: System Control Processor • 3: System Control Unit Processor, only supported by hybrid systems • 4: System Control Processor, with enhanced revision numbering. The revision numbering is XXX.YYY, where XXX = major revision and YYY = minor revision. The major revision number is always the same as the revision number for parameter value 0. • 5: Combined firmware revision number. The revision numbering format is XXX. Only the number is reported; there is no information about the type of system. Values 1,2, and 6 are reserved.

Replies

Upon Success:
Reply Options 0, 3, 4:

```
<Type of Firmware><LF>
  <NDI Serial Number><LF>
  <Characterization Date><LF> (included only for Reply Option 0 and 4)
  <Freeze Tag><LF>
  <Freeze Date><LF>
  <Copyright Information><LF>
  <CRC16><CR>
```

Reply Option 5:

```
<Combined Firmware Revision><CRC16><CR>
```

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Usage Notes

- If you send the command VER 5 after the INIT command has replied with ERROR2E, the reply will be ???, because component versions are incompatible.
- You can also obtain the combined firmware revision of the system by using GET (page 94) to read the value of the user parameter Config.Combined Firmware Revision. For more information on user parameters, see “User Parameters” on page 25.
- Reply Option 3: Only supported by hybrid systems.

Examples

Command:

VER 4

Reply:

```
Polaris Vega Control Firmware
  NDI S/N: P9-B0058
  Characterization Date: 06/09/16
  Freeze Tag: Polaris Vega Beta 008.002
  Freeze Date: June 20 2016
  (c) Northern Digital Inc.
  AEBC
```

Command:

VER 5

Reply:

001BDB5

VGET

The VGET command retrieves data previously captured with VSNAP (page 179).

Compatibility

Supported by the Position Sensor since G.001.

VGET and VSNAP are recommended when using serial communications due to the latency introduced in VCAP when retrieving an image.

Operating Mode

All modes

Prerequisite Command

VSNAP (page 179)

Syntax

```
VGET<SPACE><Row><Sensor><Frame Index><Start Column><End
  Column><Stride><CR>
```

Parameter	Description
`Row`	Four hexadecimal characters. Specifies the row of data to retrieve. **Valid Values:** 0 to the value of the user parameter `Cmd.VGet.Sensor.Height` - 1.
`Sensor`	Two hexadecimal characters. Specifies which sensor's data to return. **Valid Values** • 0: Left sensor • 1: Right sensor
`Frame Index`	Two hexadecimal characters. The index into the array of frames captured by VSNAP (page 179). Specifies which frame's data to return. The frame index is zero-based. The tool class for each frame is returned in the VSNAP reply; to retrieve information for a particular tool class, use the frame index for that frame returned in the VSNAP reply.
`Start Column`	Four hexadecimal characters. Optional. Indicates the first column to retrieve. See "Usage Notes" on page 176. **Valid Values:** 0 to the value of the user parameter `Cmd.VGet.Sensor.Width` - 1.
`End Column`	Four hexadecimal characters. Optional. Indicates the last column to retrieve. See "Usage Notes" on page 176. **Valid Values:** 0 to the value of the user parameter `Cmd.VGet.Sensor.Width` - 1.

Parameter	Description
Stride	Two hexadecimal characters. Optional. Indicates the stride count to use from start to end column. A lower stride count results in a higher resolution. The stride options are described in "Usage Notes" on page 176. To specify which stride option to use, set the value of the user parameter Cmd.VGet.Sample Option.

Replies

Upon Success:

<Header><Length><HeaderCRC><Data><DataCRC>

Note **The reply for the VGET command is binary data.**

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Reply Component	Description
Header	Two bytes: A5C4 Indicates the start of the VGET reply.
Length	Two bytes. Indicates data length.
Header CRC	Two bytes. CRC16 for header.
Data	Up to 2048 bytes of binary data. The data is a sequence of grey-scale pixel intensities. The intensity for each pixel is given by x bits of data, where x is the value of the user parameter Cmd.VGet.Color Depth. See "Usage Notes" on page 176. A pixel intensity of 0 is black and an intensity of 2^x-1 is white.
DataCRC	Two bytes. CRC16 of the <Data> section.

Usage Notes

- The VGET command retrieves one row of data. To retrieve an entire image, use a sequence of VGET commands. For a lower resolution image, retrieve fewer rows.
- To use the VGET command, the data bits must be set to 0 (8 bits) using COMM (page 87).
- Replies are returned in little endian format.
- Instead of using the <Start Column>, <End Column>, and <Stride> parameters, you can set the values of the user parameters Cmd.VGet.Start X, Cmd.VGet.End X, and Cmd.VGet.Stride for the start column, end column, and stride, respectively.
- A lower stride count results in a higher resolution. To specify the stride option to use, set the value of the user parameter Cmd.VGet.Sample Option. Options are as follows:
  - Point (Cmd.VGet.Sample Option = 0): For a stride of n, returns every n^{th} pixel.

  - Average (Cmd.VGet.Sample Option = 1): For a stride of n, returns the average of the n pixels.
  - Peak (Cmd.VGet.Sample Option = 2): For a stride of n, returns the maximum value of the n pixels.
- To adjust the number of bits per pixel returned, set the value of the user parameter `Cmd.VGet.Color Depth`. A higher value results in higher picture quality and longer reply length.
- For diagnostic purposes, it may be helpful to color-code the image data according to the internal thresholds used by the system. Since these thresholds are dynamic and vary with exposure time and other factors, they have to be retrieved individually for each frame. To determine the threshold values:
  a) Set the value of the user parameter `Cmd.VGet.Threshold.Shutter Time` to the exposure time for the frame and sensor whose threshold you wish to determine. The exposure time is returned in the VSNAP response.
  b) Read the value of the user parameter `Cmd.VGet.Threshold.Trigger`. This value is a function of the exposure time and the sensitivity level (described in the user guide). In order for a marker to be detected by the system, at least on pixel must exceed this threshold.
  c) Read the value of the user parameter `Cmd.VGet.Threshold.Background`. Ideally only markers exceed this threshold.
- To read user parameter values, use GET (page 94).
- To set user parameter values, use SET (page 141).
- For more information on user parameters, see “User Parameters” on page 25.

Example

Command:

```
VGET 0001000100100002001
```

VSEL

Selects a characterised measurement volume.

Compatibility

Supported by the Position Sensor since G.001.

Deprecated. Use SET (page 141) to set the user parameter `Param.Tracking.Selected Volume` instead.

Operating Mode

Setup

Prerequisite Command

INIT (page 100)

Syntax

`VSEL<SPACE><Volume Number><CR>`

Parameter	Description
`Volume Number`	One hexadecimal character. **Possible Values:** 1 to the maximum returned by SFLIST (page 143)

Replies

Upon Success:
`OKAY<CRC16><CR>`

On Error:
`ERROR<Error Code><CRC16><CR>`
For error code definitions, see page 181.

Usage Notes

To determine which measurement volumes are available, use SFLIST (page 143).

Example

Command:
`VSEL 1`

Reply:
`OKAYA896`

VSNAP

Captures one complete sequence of video data from the sensors. For details, see “Usage Notes” on page 180.

Compatibility

Supported by the Position Sensor since G.001.

VGET and VSNAP are recommended when using serial communications due to the latency introduced in VCAP when retrieving an image.

Operating Mode

Diagnostic, Tracking

Syntax

VSNAP<SPACE><CR>

Reply

Upon Success:

```
<Frame Number><Number of Frames><Number of Sensors>
  <Frame 0 Type><Frame 0 Exposure Time 0><Frame 0 Exposure Time 1>...
  <Frame 1 Type><Frame 1 Exposure Time 0><Frame 1 Exposure Time 1>...
  ...
  <CRC16><CR>
```

On Error:

ERROR<Error Code><CRC16><CR>

For error code definitions, see page 181.

Reply Component	Description
Frame Number	Eight hexadecimal characters. The frame number of the first frame in the sequence.
Number of Frames	Two hexadecimal characters. The number of frames in the sequence.
Number of Sensors	Two hexadecimal characters. The number of sensors for each frame. This is always 2.
Frame n Type	Two hexadecimal characters. The tool class of frame n. The list of possible values is reported as the enumeration list in user parameter Cmd.VSnap.Frame Types. For details on reading the enumeration list of a user parameter, see GETINFO (page 96). A value of 0F indicates a frame that is used only for timing purposes, and contains no useful data.
Frame n Exposure Time m	Four hexadecimal characters. The exposure time of the m^{th} sensor in µsec. Sensor 0 is the left sensor and sensor 1 is the right sensor, from the point of view of the Position Sensor.

VSNAP Usage Notes

Usage Notes

- The VSNAP command captures one complete sequence of video data from the sensors. The captured data is stored in internal memory; to retrieve the data, use VGET (page 175). A complete sequence of data consists of the following frames:
  - A frame for each type of tool loaded (passive or active wireless). This allows you to see exactly what the Position Sensor detects while it is tracking.
  - An illuminated frame (illuminators are activated), if the user parameter `Cmd.VSnap.Illuminated Frame` is enabled (set to 1). This allows you to detect any infrared light sources caused by reflections.
  - A background frame (illuminators are off), if the user parameter `Cmd.VSnap.Background Frame` is enabled (set to 1). This allows you to detect any environmental infrared light.

Note **The `Cmd.VSnap.Illuminated Frame` and `Cmd.VSnap.Background Frame` user parameters can be set only when the system is in Setup mode.**

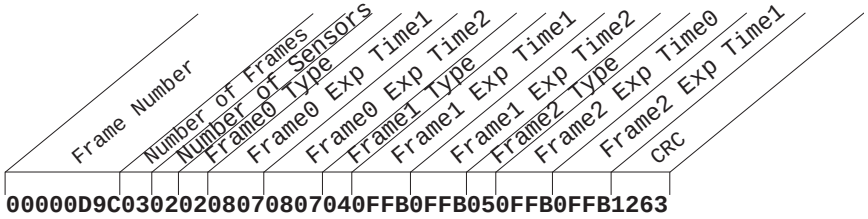
- The exposure time (the amount of time during which the sensors collect light) for the illuminated and background frames is the value of the user parameter `Cmd.VSnap.Manual Shutter`.
- To read user parameter values, use GET (page 94). To set user parameter values, use SET (page 141). For more information on user parameters, see “User Parameters” on page 25.

Example

Command:

VSNAP

Reply:



6 Error and Warning Code Definitions

6.1 Error Code Definitions

If the system receives an invalid command, it responds to the host with the message ERROR<Error Code>. Table 6-1 identifies the error codes and their definitions.

Table 6-1 Error Code Definitions

Error Code	Definition
01	Invalid command.
02	Command too long.
03	Command too short.
04	Invalid CRC calculated for command; calculated CRC does not match the one sent.
05	Time-out on command execution.
06	Unable to set up new communication parameters. This occurs if one of the communication parameters is out of range.
07	Incorrect number of parameters.
08	Invalid port handle selected.
09	Invalid mode selected. Either the tracking priority is out of range, or an incorrect priority was selected (e.g. the tool has markers defined and “button box” was selected).
0A	Invalid LED selected. The LED selected is out of range.
0B	Invalid LED state selected. The LED state selected is out of range.
0C	Command is invalid while in the current mode.
0D	No tool is assigned to the selected port handle.
0E	Selected port handle not initialized. The port handle needs to be initialized before the command is sent.
0F	Selected port handle not enabled. The port handle needs to be enabled before the command is sent.
10	System not initialized. The system must be initialized before the command is sent.
11	Unable to stop tracking. This occurs if there are hardware problems. Please contact NDI.
12	Unable to start tracking. This occurs if there are hardware problems. Please contact NDI.
13	Hardware error: unable to read the SROM device.
14	Invalid Position Sensor characterization parameters.
15	Unable to initialize the system. This occurs if: • the system could not return to Setup mode • there are internal hardware problems. Please contact NDI. • there are internal parameter errors. Use GET to read the Info.Status.Alerts parameter for more details.
16	Unable to start Diagnostic mode. This occurs if there are hardware problems. Please contact NDI.
17	Unable to stop Diagnostic mode. This occurs if there are hardware problems. Please contact NDI.
18	Reserved
19	Unable to read device's version information. This occurs if: • the processor selected is out of range • the system is unable to inquire firmware version information from a processor

Table 6-1 Error Code Definitions (Continued)

Error Code	Definition
1A	Internal system error. This occurs when the system is unable to recover after: • too much IR • a system processing exception
1B	Reserved
1C	Unable to set marker activation signature.
1D	Reserved
1E	Unable to read SROM device data. This occurs if the system is: • unable to auto-select the first SROM device on the given port handle as a target to read from • unable to read a page of SROM device data successfully
1F	Unable to write SROM device data. This can occur if: • the system is unable to auto-select the first SROM device on the given port handle as a target for writing to the SROM device • the system is unable to write a page of SROM device data successfully
20	Reserved
21	Unable to test electrical current on tool.
22	Enabled tools are not supported by selected volume parameters. For example, a Position Sensor cannot track a tool if the volume parameter set does not include the marker wavelength of an enabled tool.
23	Command parameter is out of range.
24	Unable to select measurement volume. This occurs if: • the selected volume is not available • there are internal hardware errors. Please contact NDI.
25	Unable to determine the system's supported features list. This occurs if the system is unable to read all the hardware information.
26-27	Reserved
28	Too many tools are enabled, or the configuration of tools loaded requires too many frames.
29	Reserved
2A	No memory is available for dynamic allocation (heap is full).
2B	The requested port handle has not been allocated.
2C	The requested port handle has become unoccupied.
2D	All handles have been allocated.
2E	Incompatible firmware versions. This can occur if: • a firmware update failed • components with incompatible firmware are connected To correct the problem, update the firmware. If the `Multi Firmware feature` is installed, select a valid combined firmware revision.
2F	Invalid port description.
30	Requested port is already assigned a port handle.
31	Reserved
32	Invalid operation for the device associated with the specified port handle.
33	Feature not available.
34	User parameter does not exist.
35	Invalid value type (e.g. string instead of integer).

Table 6-1 Error Code Definitions (Continued)

Error Code	Definition
36	User parameter value set is out of valid range.
37	User parameter array index is out of valid range.
38	User parameter size is incorrect.
39	Permission denied; file or user parameter is read-only, or a command which requires master mode is attempted from a monitor mode connection.
3A	Reserved
3B	File not found.
3C	Error writing to file.
3D	Error reading from file.
3E-3F	Reserved
40	Tool Definition File Error. This occurs if: • the CRC failed • the file format is invalid
41	Tool characteristics not supported. This occurs when one of the following fields in the tool definition file is outside of the range supported by the system: • number of markers • number of faces • number of groups • number of markers per face (unique geometry tools only)
42	Device not present. This occurs when the command is specific to a device that is not connected to the system.
43-FF	Reserved

6.2 Warning Code Definitions

Table 6-2 Warning Code Definitions

Warning	Definition
WARNING01	A non-fatal tool error has been encountered, e.g. a burnt out marker.
WARNING02	The tool you are trying to enable is a unique geometry tool that doesn't meet the unique geometry requirements.
WARNING03	The tool you are trying to enable is a unique geometry tool that conflicts with another unique geometry tool already loaded and enabled.
WARNING04	The tool you are trying to enable is a unique geometry tool that doesn't meet the unique geometry requirements, and conflicts with another unique geometry tool already loaded and enabled.
WARNING05	The system has selected a default marker wavelength to track a tool (if the tool's tool definition file did not specify a marker wavelength).
WARNING06	A Monitor connection has issued the INIT command while the system is in either Tracking or Diagnostics mode or a tool has been loaded.

WARNING01 and WARNING05 are returned with the PINIT or the PENA command.

WARNING02, WARNING03 and WARNING04 are returned with the PENA command.

WARNING06 is returned with the INIT command.

Appendix A Keyed Features

This section describes how to use the API commands and user parameters with the keyed features. For more information on keyed features, see the user guide that accompanied your system. For more information on user parameters, see “User Parameters” on page 25.

A.1 Disabling and Enabling Keyed Features

Disabling a keyed feature makes that feature unavailable. Enabling a keyed feature makes the feature available. A keyed feature is enabled upon installation.

To disable or enable a keyed feature:

1. Use SET to set the value of the user parameter `Features.Keys.Disabled Keys`.

   The value of this parameter is a comma-separated list. To disable a keyed feature, add its name to the comma-separated list. To re-enable a keyed feature, remove its name from the comma-separated list. For example:

   To disable the Multi Firmware feature, use `SET Features.Keys.Disabled Keys=Multi Firmware`. To re-enable all the installed features keys, use `SET Features.Keys.Disabled Keys=`.
2. Use SAVE to save the settings.
3. Reset the system (RESET). The changed settings take effect upon system reset.

A.2 Multi Firmware Feature

The multi firmware feature allows the system to contain more than one combined firmware revision. When the multi firmware feature is enabled, you can specify which combined firmware revision the system will use on its next reset or power up.

This feature is referred to as “MultiFirmware” in the ToolBox user interface and Polaris Vega API.

Changing the Combined Firmware Revision Currently in Use

1. If necessary, to determine which combined firmware revision is currently in use, use GET (page 94) to read the user parameter `Config.Combined Firmware Revision`.
2. To determine which combined firmware revisions are available, perform one of the following actions:
   a) For API revision G.001.004 and later, use GET (page 94) to read the user parameter `Config.Multi Firmware.Available Combined Firmware Revisions`.
   b) For API revision G.001.003 or earlier, use GETINFO (page 96) to read the user parameter `Config.Multi Firmware.Load Combined Firmware Revision`.
3. To select the desired combined firmware revision, use SET (page 141) to set the value of the user parameter `Config.Multi Firmware.Load Combined Firmware Revision`.

   The enumeration is zero-based.

Example

Step	Details	Example Reply
1	Optional step.	**Command**: `GET Config.Combined Firmware Revision` **Reply**: `Config.Combined Firmware Revision=002<CRC16>`
2a	The list of possible firmware revisions is given in the enumerated list. In this example, the firmware revisions are 002 and 003.	**Command**: `GET Config.Multi Firmware.Available Combined Firmware Revisions` **Reply**: `002,003<CRC16>`
2b	The list of possible firmware revisions is given in the enumerated list returned by GETINFO. In this example, the firmware revisions are 002 and 003.	**Command**: `GETINFO Config.Multi Firmware.Load Combined Firmware Revision` **Reply**: `Config.Multi Firmware.Load Combined Firmware Revision=0;1;3;0;255;002,003; Combined firmware revision to load on next reset (selection automatically saves when set)<CRC16>`
3	For example, to select the second item in the list (revision 003), set the value of the user parameter to 1. This parameter value is automatically saved when set. The selected combined firmware revision is loaded on the next reset.	**Command**: `SET Config.Multi Firmware.Load Combined Firmware Revision=1` **Reply**: `OKAY<CRC16>`

A.3 Positioning Laser

The positioning laser is located in the Vega system Position Sensor and indicates the centre of the characterized measurement volume. This feature allows you to properly position the Position Sensor, or position objects in the measurement volume. The positioning laser feature cannot be purchased after you obtain the system; the laser hardware must be installed when the system is manufactured. For full details on the positioning laser, see the user guide that accompanied your system.

It is possible to activate (turn on) the laser by using an external laser switch connected to a laser switch port. The optional laser switch is not supplied by NDI.

This feature is referred to as “Laser” in the ToolBox user interface and Vega API.

A.4 Serial Communications - LEMO Connector

This feature key is enabled at the factory for Vega ST Systems that have a LEMO connector to support serial communications. There is no need to disable this feature key if ethernet communications are preferred.

- If serial and ethernet connections are made to the Position Sensor at the same time, the serial connection will be granted Master role whenever needed, even if an ethernet connection was already Master.
- When this feature key is enabled, the Position Sensor is configured for a static IP address by default for ethernet communication.

From the application perspective, the Vega system is a serial device, which is listening for incoming commands. Upon receiving a command, the system performs some action and returns the status of this action. The system never initiates communication with the application except on power up or reset, when it returns `RESET<CRC16><CR>`. (If only an SCU is connected it will return `SCUONLY<CRC16><CR>`.)

Immediately after sending a command, the application can begin to poll the serial buffer for a reply. Most commands reply almost instantaneously. After reaching the end of the reply, the application can send another command.

Note **The application must read the complete response from the system before sending another command. Failure to do so may result in an error or in unpredictable system behaviour.**

For full details on serial communications, see the user guide that accompanied your system.

This feature is referred to as “LEMO” in the ToolBox user interface and Vega API.

A.5 Radiation Robustness

This feature key is enabled at the factory for Vega ST Systems that are hardened against neutron radiation. Radiation robustness is achieved through a shielded processor and modified memory management.

This feature is referred to as “RadiationRobust” in the ToolBox user interface and Vega API.

A.6 Video Camera

This feature key is enabled at the factory for Vega VT Systems, which are equipped with a video camera. The video camera is integrated into the Position Sensor. The video camera provides a live, color video stream of the Vega measurement volume but is not used for tracking tools.

This feature is referred to as “VideoCamera” in the ToolBox user interface and Vega API.

Appendix B Sample C Routines

The following sample C routines are included for reference. For more information and sample code, see the Combined API Sample (CAPI).

Table 6-3 Sample C Routines

Routine	Description
CalcCRC16	Calculates a running CRC16 using the polynomial X^16 + X^15 + X^2 + 1.
EulerAngleTrig	Determines the sine and cosine of the Euler angles.
DetermineR	Calculates the 3x3 rotation matrix which corresponds to the given Euler angles.
CvtQuatToRotationMatrix	Determines the rotation matrix that corresponds to the given quaternion values.
DetermineEuler	Calculates the Euler angles given the 3x3 rotation matrix.
CvtQuatToEulerRotation	Determines the rotation in Euler angles (degrees) that corresponds to the given quaternion rotation.

The following defines are used by the sample C routines:

```
/*
   * Conversion factors.
   */
  #define RAD_TO_DEGREES        (180 / 3.1415926)

/*
   * Defined data types.
   */
  typedef float
      RotationMatrix[3][3];

typedef struct Rotation
  {
    float
          fRoll,     /* rotation about the object's z-axis (Euler angle) */
          fPitch,    /* rotation about the object's y-axis (Euler angle) */
          fYaw;      /* rotation about the object's x-axis (Euler angle) */
  } Rotation;

typedef struct QuatRotation
{
    float
          fQ0,
          fQX,
          fQY,
          fQZ;
  } QuatRotation;
```

B.1 CalcCRC16

The following is a sample C routine, for calculating a running 16 bit CRC, as used in communications between the host computer and the Vega system.

```
/*************************************************************
  Name:        CalcCRC16

Input Values:
      int
          data         :Data value to add to running CRC16.
      unsigned int
          *puCRC16     :Ptr. to running CRC16.

Output Values:
      None.

Returned Value:
      None.

Description:
      This routine calculates a running CRC16 using the polynomial
      X^16 + X^15 + X^2 + 1.

*************************************************************/
  void CalcCRC16( int data, unsigned int *puCRC16 )
{
    static int
          oddparity[16] = { 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0 };

    data = (data ^ (*puCRC16 & 0xff)) & 0xff;
      *puCRC16 >>= 8;

    if ( oddparity[data & 0x0f] ^ oddparity[data >> 4] )
      {
         *puCRC16 ^=0xc001

      } /* if */
      data <<= 6;
      *puCRC16 ^= data;
      data <<= 1;
      *puCRC16 ^= data;

} /* CalcCRC16 */
```

B.2 EulerAngleTrig

```
/*****************************************************************
  Name:        EulerAngleTrig

Input Values:
      Rotation
          *pdtRotationAngle :Ptr to struct containing the roll, pitch, yaw
                             Euler angles which define the required rotation.

Output Values:
      Rotation
          *pdtSinAngle :Ptr to struct containing the sine of the roll, pitch,
                        yaw Euler angles.
          *pdtCosAngle :Ptr to struct containing the cosine of the roll, pitch,
                        yaw Euler angles.

Returned Value:
      None.

Description:
      This routine determines the sine and cosine of the Euler angles.

*****************************************************************/
  static void EulerAngleTrig( Rotation *pdtRotationAngle,
                  Rotation *pdtSinAngle,
                  Rotation *pdtCosAngle )
  {
     pdtSinAngle->fRoll=    sin( pdtRotationAngle->fRoll );
      pdtSinAngle->fPitch= sin( pdtRotationAngle->fPitch );
      pdtSinAngle->fYaw =  sin( pdtRotationAngle->fYaw );
      pdtCosAngle->fRoll=  cos( pdtRotationAngle->fRoll );
      pdtCosAngle->fPitch= cos( pdtRotationAngle->fPitch );
      pdtCosAngle->fYaw=   cos( pdtRotationAngle->fYaw );

} /* EulerAngleTrig */
```

B.3 DetermineR

```
/*****************************************************************
  Name:        DetermineR

Input Values:
      Rotation
          *pdtRotationAngle :Ptr to struct containing the roll, pitch, yaw
                             Euler angles which define the required rotation.

Output Values:
      RotationMatrix
          dtRotationMatrix :The 3x3 rotation matrix to be determined.

Returned Value:
      None.

Description:
      This routine calculates the 3x3 rotation matrix which corresponds to the
      given Euler angles.

*****************************************************************/
  void DetermineR( Rotation *pdtRotationAngle, RotationMatrix
           dtRotationMatrix )
{
    Rotation
          dtSinAngle, /* the sine of the roll, pitch, and yaw angles */
          dtCosAngle; /* the cosine of the roll, pitch, and yaw angles */

/*
  * Might as well determine the sine and cosine of the given Euler
  *angles right from the start
  */
  EulerAngleTrig( pdtRotationAngle, &dtSinAngle, &dtCosAngle );

/*
  * Fill in the rotation matrix.
  */
dtRotationMatrix[0][0] = dtCosAngle.fRoll * dtCosAngle.fPitch;
  dtRotationMatrix[0][1] = dtCosAngle.fRoll * dtSinAngle.fPitch *
      dtSinAngle.fYaw - dtSinAngle.fRoll * dtCosAngle.fYaw;
  dtRotationMatrix[0][2] = dtCosAngle.fRoll * dtSinAngle.fPitch *
      dtCosAngle.fYaw + dtSinAngle.fRoll * dtSinAngle.fYaw;
  dtRotationMatrix[1][0] = dtSinAngle.fRoll * dtCosAngle.fPitch;
  dtRotationMatrix[1][1] = dtSinAngle.fRoll * dtSinAngle.fPitch *
      dtSinAngle.fYaw + dtCosAngle.fRoll * dtCosAngle.fYaw;
  dtRotationMatrix[1][2] = dtSinAngle.fRoll * dtSinAngle.fPitch *
      dtCosAngle.fYaw - dtCosAngle.fRoll * dtSinAngle.fYaw;
  dtRotationMatrix[2][0] = - dtSinAngle.fPitch;
  dtRotationMatrix[2][1] = dtCosAngle.fPitch * dtSinAngle.fYaw;
  dtRotationMatrix[2][2] = dtCosAngle.fPitch * dtCosAngle.fYaw;

} /* DetermineR */
```

B.4 CvtQuatToRotationMatrix

```
/*****************************************************************
  Name:        CvtQuatToRotationMatrix

Input Values:
      QuatRotation
          *pdtQuatRot :Ptr to the quaternion rotation.

Output Values:
      RotationMatrix
          dtRotationMatrix :The 3x3 determined rotation matrix.

Returned Value:
      None.

  Description:
      This routine determines the rotation matrix that corresponds
      to the given quaternion.

    Let the quaternion be represented by:

         | Q0 |
      Q = | Qx |
          | Qy |
          | Qz |

    and the rotation matrix by:

         | M00 M01 M02 |
      M = | M10 M11 M12 |
          | M20 M21 M22 |

    then assuming the quaternion, Q, has been normalized to convert
      Q to M we use the following equations:

    M00 = (Q0 * Q0) + (Qx * Qx) - (Qy * Qy) - (Qz * Qz)
      M01 = 2 * ((Qx * Qy) - (Q0 * Qz))
      M02 = 2 * ((Qx * Qz) + (Q0 * Qy))
      M10 = 2 * ((Qx * Qy) + (Q0 * Qz))
      M11 = (Q0 * Q0) - (Qx * Qx) + (Qy * Qy) - (Qz * Qz)
      M12 = 2 * ((Qy * Qz) - (Q0 * Qx))
      M20 = 2 * ((Qx * Qz) - (Q0 * Qy))
      M21 = 2 * ((Qy * Qz) + (Q0 * Qx))
      M22 = (Q0 * Q0) - (Qx * Qx) - (Qy * Qy) + (Qz * Qz)

*****************************************************************/
  void CvtQuatToRotationMatrix( QuatRotation *pdtQuatRot,
                  RotationMatrix dtRotMatrix )
  {
    float
          fQ0Q0,
          fQxQx,
          fQyQy,
          fQzQz,
          fQ0Qx,
```

```
            fQ0Qy,
            fQ0Qz,
            fQxQy,
            fQxQz,
            fQyQz;

      /*
         * Determine some calculations done more than once.
         */
            fQ0Q0 = pdtQuatRot->fQ0 * pdtQuatRot->fQ0;
            fQxQx = pdtQuatRot->fQX * pdtQuatRot->fQX;
            fQyQy = pdtQuatRot->fQY * pdtQuatRot->fQY;
            fQzQz = pdtQuatRot->fQZ * pdtQuatRot->fQZ;
            fQ0Qx = pdtQuatRot->fQ0 * pdtQuatRot->fQX;
            fQ0Qy = pdtQuatRot->fQ0 * pdtQuatRot->fQY;
            fQ0Qz = pdtQuatRot->fQ0 * pdtQuatRot->fQZ;
            fQxQy = pdtQuatRot->fQX * pdtQuatRot->fQY;
            fQxQz = pdtQuatRot->fQX * pdtQuatRot->fQZ;
            fQyQz = pdtQuatRot->fQY * pdtQuatRot->fQZ;

      /*
         * Determine the rotation matrix elements.
         */
            dtRotMatrix[0][0] = fQ0Q0 + fQxQx - fQyQy - fQzQz;
            dtRotMatrix[0][1] = 2.0 * (-fQ0Qz + fQxQy);
            dtRotMatrix[0][2] = 2.0 * (fQ0Qy + fQxQz);
            dtRotMatrix[1][0] = 2.0 * (fQ0Qz + fQxQy);
            dtRotMatrix[1][1] = fQ0Q0 - fQxQx + fQyQy - fQzQz;
            dtRotMatrix[1][2] = 2.0 * (-fQ0Qx + fQyQz);
            dtRotMatrix[2][0] = 2.0 * (-fQ0Qy + fQxQz);
            dtRotMatrix[2][1] = 2.0 * (fQ0Qx + fQyQz);
            dtRotMatrix[2][2] = fQ0Q0 - fQxQx - fQyQy + fQzQz;

   } /* CvtQuatToRotationMatrix */
```

B.5 DetermineEuler

```
/*****************************************************************
  Name:         DetermineEuler

Input Values:
      RotationMatrix
          dtRotationMatrix :The 3x3 rotation matrix to convert.

Output Values:
      Rotation
          *pdtEulerRot :Rotation is Euler angle format.
              Roll, pitch, yaw Euler angles which define the required rotation.

Returned Value:
      None.

Description:
      This routine calculates the Euler angles given the 3x3 rotation matrix.

*****************************************************************/
void DetermineEuler( RotationMatrix dtRotMatrix, Rotation *pdtEulerRot )
{
       float
          fRoll,
          fCosRoll,
          fSinRoll;

       fRoll    = atan2( dtRotMatrix[1][0], dtRotMatrix[0][0] );
       fCosRoll = cos( fRoll );
       fSinRoll = sin( fRoll );

       pdtEulerRot->fRoll  = fRoll;
       pdtEulerRot->fPitch = atan2( -dtRotMatrix[2][0],
                             (fCosRoll * dtRotMatrix[0][0]) + (fSinRoll *
       dtRotMatrix[1][0]) );
       pdtEulerRot->fYaw   = atan2(
                             (fSinRoll * dtRotMatrix[0][2]) -
                             (fCosRoll * dtRotMatrix[1][2]),
                             (-fSinRoll * dtRotMatrix[0][1]) +
                             (fCosRoll * dtRotMatrix[1][1]) );

}      /* DetermineEuler */
```

B.6 CvtQuatToEulerRotation

```
/************************************************************
  Name:        CvtQuatToEulerRotation

Input Values:
      QuatRotation
          *pdtQuatRot :Ptr to the quaternion rotation.

Output Values:
      Rotation
          *pdtEulerRot :Ptr to the determined rotation Euler angles.

Returned Value:
      None.

Description:
      This routine determines the rotation in Euler angles (degrees)that
      corresponds to the given quaternion rotation.

************************************************************************/
  void CvtQuatToEulerRotation( QuatRotation *pdtQuatRot, Rotation *pdtEulerRot )
  {
    RotationMatrix
          dtRotMatrix;

    CvtQuatToRotationMatrix( pdtQuatRot, dtRotMatrix );

    DetermineEuler( dtRotMatrix, pdtEulerRot );

    pdtEulerRot->fYaw   *= RAD_TO_DEGREES;
      pdtEulerRot->fPitch *= RAD_TO_DEGREES;
      pdtEulerRot->fRoll  *= RAD_TO_DEGREES;

} /* CvtQuatToEulerRotation */
```

Appendix C Changes in Implementation

This chapter describes the cumulative changes in commands between API version G.001, which was the version to support Polaris Spectra, and API version G.003, which supports the Polaris Vega platform. This information may be useful when migrating from Spectra to Vega. For more detailed information on revisions to the Vega API, see the *"Polaris Vega Firmware Revision History"*.

This chapter also describes features that have been discontinued or deprecated in recent revisions of API version G.003.

C.1 New Concepts

Several important concepts have been introduced in support of the Vega platform. For information, see "Important Concepts" on page 5.

C.2 New Commands

New commands in introduced between G.001 and G.003 are as follows:

New Command	Description
BX2 (page 77)	Returns various levels of data on the latest tool transformations, individual marker positions, and system status in binary format.
STREAM (page 149)	Initiates a streaming response to a specified command.
USTREAM (page 168)	Stops streaming of the specified command.
VCAP (page 169)	Captures IR image data from the sensors.(Replaces the VGET and VSNAP commands.)

BX2

The BX2 command provides a flexible way of providing measurement data at various levels of detail. The reply can contain a single or multiple frames. Each frame can contain various levels of measurement data detail such as 6D, 3D or 2D data.

- It does not repeat already reported information.
- It works with the STREAM command to keep latency to a minimum and avoid missing or repeating information.
- Addresses the problem of providing system wide failures and warnings in the multi-connection environment.

STREAM

STREAM initiates a streaming response to a command. For details on data streaming, see "Data Streaming" on page 7.

USTREAM

USTREAM terminates the streaming response to a command. For details on data streaming, see "Data Streaming" on page 7.

VCAP

The VCAP command can be used instead of VSNAP and VGET. VCAP is recommended for Vega applications; VSNAP and VGET are supported for backwards compatibility with Spectra.

The command contains options to specify and control the image data returned. The reply contains the image data for a single frame from all sensors with embedded “meta-data” that includes sensor number, frame number, exposure and other relevant information about the frames. Readable parameters provide additional information that will assist in interpreting the image data, such as the makeup of the frame sequence and the number and names of the image sensors.

C.3 Deprecated Commands

The following commands are deprecated. Deprecated commands will no longer be enhanced to support new hardware devices or new API features. Support for deprecated commands may be discontinued in future releases.

Command	Replacement
3D	BX2 command
APIREV	User parameter `Features.Firmware.API Revision`
IRATE	User parameter `Param.Tracking.Track Frequency`
PINIT	PENA command calls PINIT
SFLIST	User parameters: `Features.Tools.Active Ports`, `Features.Tools.Passive Ports`, `Features.Volumes *`, `Features.Tools.Wireless Ports`
VSEL	User parameter `Param.Tracking.Selected Volume`

C.4 Deleted Commands

The following commands are deleted:

Deleted Command	Description
GETIO	Deleted
HCWDOG	Deleted
PSOUT	Deleted
SENSEL	Replaced by User Parameter `Param.Tracking.Sensitivity` (Table 4-6 on page 38)
SETIO	Deleted
SSTAT	Deleted

C.5 Changed Commands

The behaviour of some commands has changed since API version G.001. For details, see the *“Polaris Vega Firmware Revision History”*.

C.6 Password Protection

The current password protection feature is deprecated. NDI is developing a more comprehensive version of this feature to be released with a future firmware revision.

If you choose to enable the current password protection feature, it provides the following security against changes to the system configuration:

- You must enter the correct password before you can save user parameter values, update the firmware, or install, disable, or enable a keyed feature.
- If you do not enter the correct password, you can change user parameter values but cannot save them. Upon system reset or initialization, the parameters return to their previous values.
- If the system is reset or initialized after you enter the password, you must re-enter the password before you can make changes to the system configuration.

To enter the password, use NDI ToolBox or the API command SET `Config.Password=<password>`, where <password> is the correct password. Obtain the password from NDI.

In the ToolBox user interface and Vega API, this feature is referred to as “Password”.

C.7 250 Hz Tracking Rate (MR250)

Previously, the Vega ST Position Sensor supported the option of tracking at 250 Hz with lower latency. This option has been discontinued and is no longer available. The replacement option is the Vega XT.

Appendix D Vega Video Camera Parameter Supplement

This chapter provides supplemental information about video camera parameters and characteristics. It includes the following sections:

For a listing of all video camera parameters, see the `Features.Volumes` parameters in "Features User Parameters" on page 45.

D.1 Gamma Correction

Gamma correction applies a non-linear gain to increase the gain in dark areas of an image while keeping the light areas relatively untouched.

The four supported gamma correction modes are

- Off: No Gamma correction
- Built-in: Use built-in hardware gamma correction
- User Defined: Uses the Gamma Value to create a correction table
- User-Defined Enhanced: Uses the Gamma Value to create a correction table

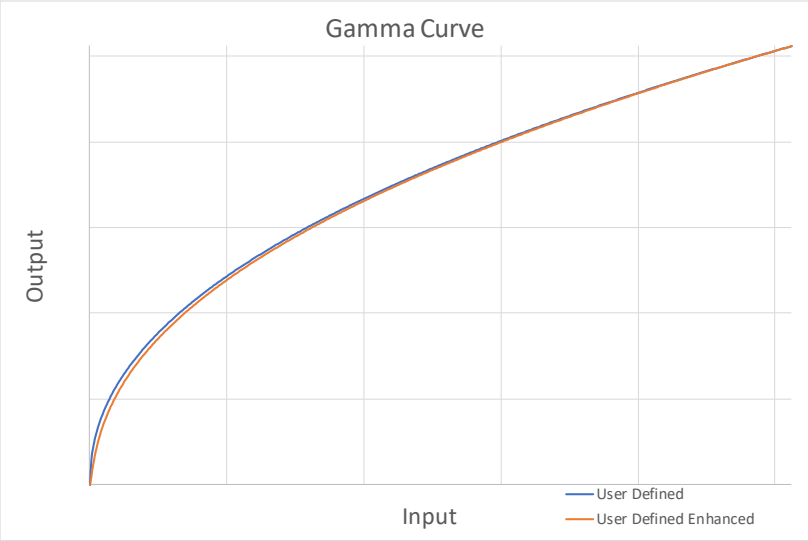
A user specified value (`Param.Effects.Gamma Value`) is used to generate the correction for the User Defined and User-Defined Enhanced modes.

User Defined: output = $\texttt{input}^{(1/\texttt{gamma})}$

User-Defined Enhanced output = $\texttt{input}^{(1/\texttt{gamma})} * (1 - (3/(\text{input}+3))+\text{input}*0.01)$

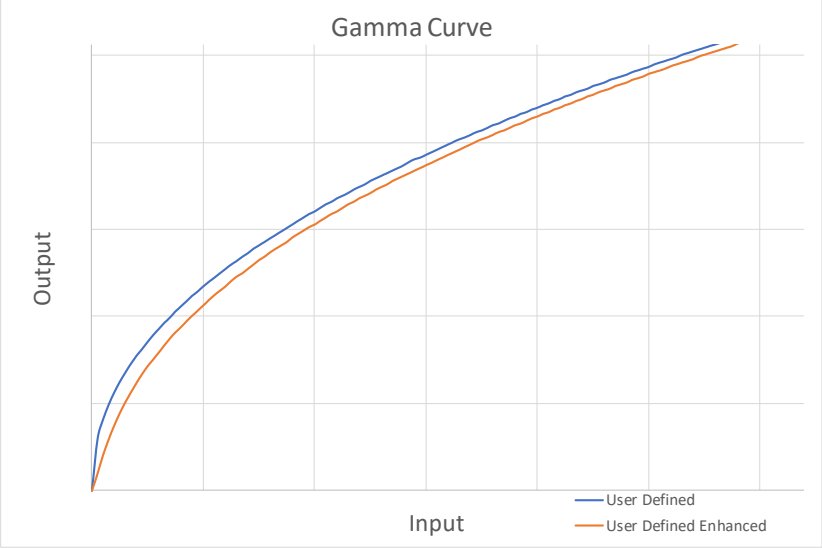
Figure 6-1 shows the User Defined gamma correction curves of 2.2.

Figure 6-1 User Defined and User-Defined Gamma Curve



Zooming in on the lower end of the graph, Figure 6-2, we see that the slope of the User-Defined Enhanced curve near the origin is less steep than the (more common) User Defined gamma curve. This reduces the “fogginess” in dark areas of the image when the gamma value is large.

Figure 6-2 Zoom in on User-Defined Enhanced Gamma Curve



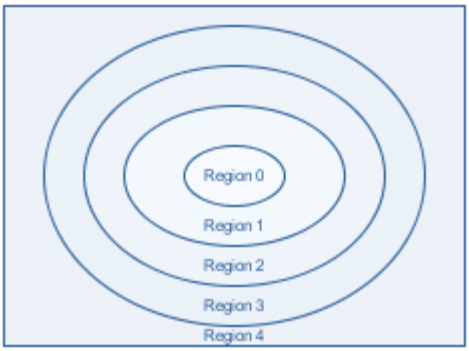
D.2 Exposure and Gain Control - Meter Modes

The Exposure and Gain Control algorithm optimizes the exposure time and/or the system gain to achieve a desired level of brightness in the image. A user-selected meter mode can be specified using the `Param.Exposure And Gain.Meter.Mode`. There are four meter modes to choose from:

- General scenes can usually use the **Center Weighted** mode.
- Backlit scenes should use the **Segmented** mode.
- Spotlighted scenes such as an operating room with a particularly bright area should use the **Spotlight** mode.
- The **Weighted ROI** mode uses a selected zone in the image area to adjust the brightness.

Centre Weighted This metering mode calculates a weighted average based on the regions shown in the diagram below.

Figure 6-3 Regions used for Centre Weighted Meter Mode

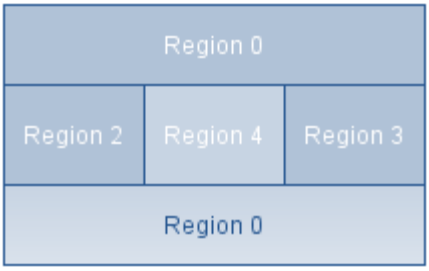


Segmented This metering mode is designed for scenes that have a principal object in a backlit condition. It prevents the background lighting from controlling the exposure.

The frame is divided into five regions. This algorithm assumes that the main object is in the center of the scene. The size of Region 4 is defined by the size of rectangle of interest (`Param.Exposure And Gain.Meter.ROI [Center X|Center Y|Width|Height]`) and is centered in the sensor.

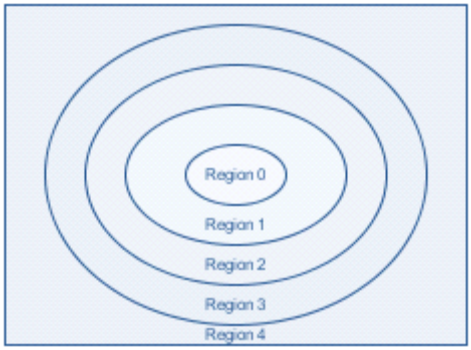
Figure 6-4 represents the Segmented metering system,. The main object is in Region 4. Region 0, Region 2 , and Region 3 are considered the background.

Figure 6-4 Segmented Metering Mode



Spotlight This metering mode is designed to prevent saturation of a spotlight area and may be ideal for surgical applications. The algorithm searches for the brightest center-weighted patch. The sensor is divided into several regions as shown in the figure below and each region is assigned a weight, with the highest weighting in the center of the image and decreasing towards the edges.

Figure 6-5 Regions used for Spotlight Meter Mode



Weighted ROI In the Weighted ROI metering mode, the current brightness value is calculated using a user selected region and a user defined weighting value. In Figure 6-6, Region 0 is the user-selected region and Region 1 is the background. The user-defined weighting value is the weighting between the average brightness of Region 0 and Region 1, expressed as a percentage.

Figure 6-6 ROI Metering Method

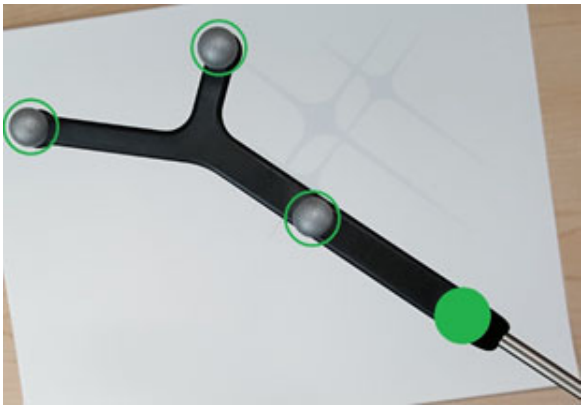
`Param.Exposure And Gain.Meter.ROI Weighting` sets the weighting value and `Param.Exposure And Gain.Meter.ROI [Center X|Center Y|Width|Height]` specifies the location and size of Region 1.

Appendix E Vega Video Camera Alignment

This appendix describes how to align and synchronize the Polaris Vega video camera stream to the real-time tracking stream from the Vega Position Sensor.

The Vega video camera sensor has been characterized and aligned to the tracking system to describe the relationship between objects seen in the video images and objects tracked in the position sensor coordinate space. This feature may be used for augmented reality overlays, a simple example of which is shown in Figure E-1.

Figure E-1 Tracking Data Overlay on Video Stream

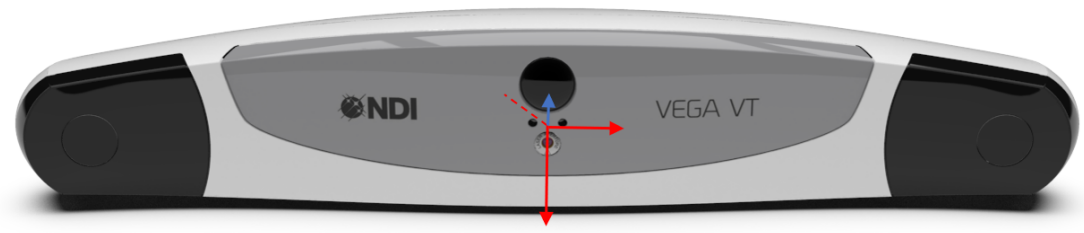


E.1 Spatial alignment

The video camera-to Vega tracking system alignment data is calculated for use with the OpenCV image processing library and its Calib3D (OpenCV, 2017) module, and describes four sets of parameters:

- Orientation and location of the video camera in Vega's measurement space, Figure E-2.
- Radial and tangential distortion coefficients of video camera lens for Zhang's method (Camera Resectioning, n.d.)
- Parameters describing the Pinhole Camera Model (Pinhole camera model, n.d.)
- Resolution-specific alignment parameters

Figure E-2 Video Alignment Pose



The video alignment pose (in blue, above) describes the video camera's pose with respect to the Vega tracking coordinate space (in red, above).

These sets of parameters are in the Vega video camera's parameter system and published through the Vega Application Program Interface, using the parameter names listed in Table E-1 when connected directly to the Video Camera Unit (port 8766). If connected through the PSU (port 8765), Video Camera Unit parameters are forwarded with the name prefix "VCU-0.".

Table E-1 VCU Parameters for Alignment and 3D Reconstruction with OpenCV

Parameter Name	Description
Param.Lens.6D.q0	Quaternion rotation from Camera Coordinate System to Video Coordinate System.
Param.Lens.6D.qx	
Param.Lens.6D.qy	
Param.Lens.6D.qz	
Param.Lens.6D.tx	Translation from Camera Coordinate System to Video Coordinate System [mm].
Param.Lens.6D.ty	
Param.Lens.6D.tz	
Param.Lens.Distortion.k1	Radial lens distortion coefficients for Zhang's method.
Param.Lens.Distortion.k2	
Param.Lens.Distortion.k3	
Param.Lens.Distortion.p1	Tangential lens distortion coefficient for Zhang's method.
Param.Lens.Distortion.p2	
Param.Lens.Pinhole.fu	Horizontal focal length in sensor space [pixels].
Param.Lens.Pinhole.fv	Vertical focal length in sensor space [pixels].
Param.Lens.Pinhole.u0	Horizontal position of pinhole in sensor space [pixels].
Param.Lens.Pinhole.v0	Vertical position of pinhole in sensor space [pixels].
Param.Left	Left coordinate of imaging window on the sensor [pixels]. Changes with selected resolution.
Param.Top	Top coordinate of imaging window on the sensor [pixels]. Changes with selected resolution.
Param.Binning X	Binning scale factor in horizontal direction. Changes with selected resolution.
Param.Binning Y	Binning scale factor in vertical direction. Changes with selected resolution.

E.2 Calculating the Pinhole Model for each Resolution

Whenever the resolution of the video stream is changed, the binning scale factors and top-left window coordinates of the imaging window change. It is necessary to calculate resolution-specific pinhole model parameters for the selected resolution using the following transformations:

u_0 = ([Param.Lens.Pinhole.u0] - [Param.Left]) ÷ [Param.Binning X]

v_0 = ([Param.Lens.Pinhole.v0] - [Param.Top]) ÷ [Param.Binning Y]

f_u = [Param.Lens.Pinhole.fu] ÷ [Param.Binning X]

f_v = [Param.Lens.Pinhole.fv] ÷ [Param.Binning Y]

where

u_0, v_0: Resolution-specific coordinate of the optical axis in the imaging window

f_u, f_v: Resolution-specific horizontal and vertical focal length.

E.3 Projecting Tracking Poses onto the Video Image

Given alignment, distortion, and pinhole model parameters, tracking data may be projected onto the video image. The parameters were designed to be used with the OpenCV library's Calib3d module (OpenCV, 2017). The following steps describe how coordinates given in a tools' coordinate frame (TCS) are projected onto the video image:

1. The 6D transformation of a tool as reported by the tracking system describes the transformation of the tool coordinate system (TCS) onto the camera's coordinate system (CCS):

   TCS→CCS

   The static 6D transformation described by the parameters in Param.Lens.6D.* describe the transformation from the camera coordinate system (CCS) onto the video coordinate system (VCS):

   CCS→VCS

   Combining the two transformations results in a new 6D transformation that describes the transformation from the tool coordinate system onto the video coordinate system:

   TCS→VCS

   A 3D coordinate (x,y,z) expressed in the TCS can be transformed into the VCS by sequentially applying the two 6D transformations to it.

2. Project the 3D coordinate (x,y,z) in VCS onto the video image:

   Pass the 3D position, the camera's resolution-specific pinhole model values and the camera's distortion coefficients to the OpenCV Calib3d::projectPoints method to find the tracking point's position relative to the top left corner of the video frame.

3. To highlight the marker positions of all markers on a tool, repeat these steps for each 3D coordinate in the tool definition file or using the 3D tracking option of the track tools command.

The resulting (x,y) coordinates are the location of the tool in image space, from the top-left corner.

To draw a set of axes to represent the tool's coordinate system:

1. Calculate the projection coordinates for each of the axes:

   $$\begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ s \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix}$$

   where s is the desired length of the axis lines.

2. Draw lines from the tool's projected origin to each of the projected axis points.

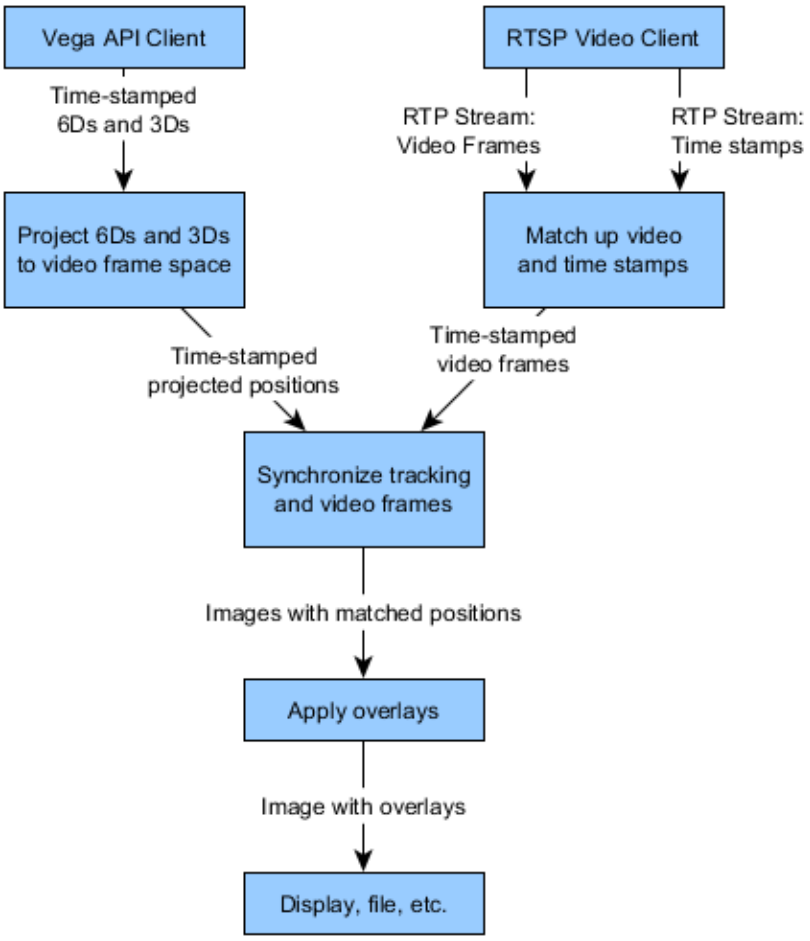
E.4 Time Synchronization

The Vega Position Sensor provides a time stamp in every 6D/3D tracking frame using the BX2 command. The video camera unit's server provides two RTP (Real-time Transport Protocol) streams through the RTSP (Real-Time Streaming Protocol) server implementation. The first RTP stream is the H.264-encoded video frames, and the second RTP stream are packets with key-length-value objects, including the video stream time stamp.

To draw 6D/3D tracking-synchronized overlays onto the video stream in the application, two separate synchronization steps are required (Figure E-3). The first step is to line up the two incoming RTP streams using RTP synchronization. A GStreamer implementation example is shown in Figure 4. The time-stamped video frames may be buffered while the incoming time-stamped 6D/3D data is also buffered, then matched up by RTP packet time stamps.

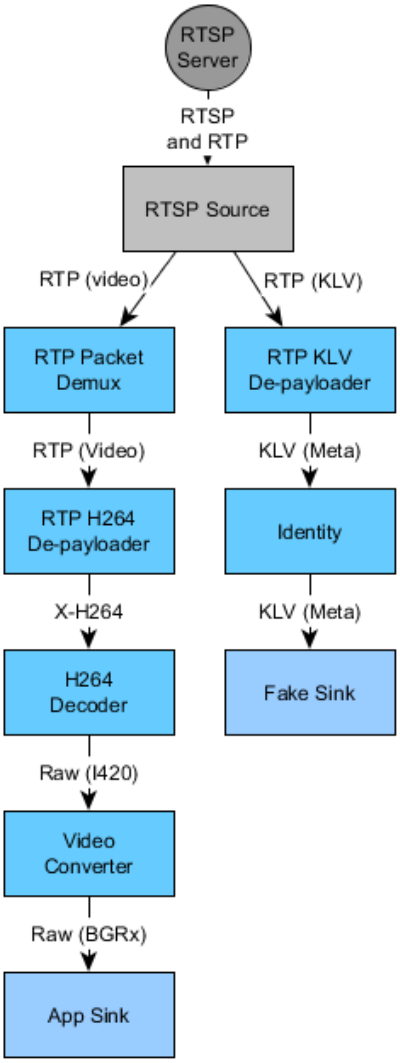
A second synchronizer is used to align the time-stamped video frames with time-stamped 6D/3D data. Because the two streams are arriving at different frequencies, there will not be a one-to-one match-up. The video camera's image stream is not synchronized to the tracking frames, although it uses the same clock.

Figure E-3 Time synchronization block diagram



The GStreamer audio- and video-streaming client supports the processing of RTP KLV packets. A GStreamer pipeline implementation is shown below (Figure E-4) that may be used to stream video and time stamp data from the Video Control Unit's video server.

Figure E-4 Example of a GStreamer video pipeline to decode video and time stamp RTP packets into raw video



Note **To ensure that the RTP GStreamer bin will order the incoming RTP packets from the two streams properly, the "buffer-mode" of the RTPBin is set to 0 (none).**

E.5 Decoding the KLV RTP stream

The RTP packets containing the time stamp follow the SMPTE336M format (SMPTE, 2017). The format of the 25-byte key-length-value time stamp object is described in Table E-1.

Table E-1 Description of Time Stamp KLV Object

Size(bytes)	Description
16	Time stamp key, Table E-2
1	Length of the time stamp in bytes = 8
8	Time stamp, nanoseconds since the epoch (big-endian)

The 16 bytes of the Key are shown in Table E-2:

Table E-2 KLV Time Stamp Key

0x06	0x0E	0x2B	0x34	0x01	0x01	0x01	0x03
0x07	0x02	0x01	0x01	0x01	0x05	0x00	0x00

For example, the time stamp bytes in Table E-3:

Table E-3 Example of KLV Time Stamp Bytes

0x15	0x71	0x81	0xb8	0x4c	0x44	0x68	0xa0

decode to the time stamp in Figure E-5:

Figure E-5 Time Stamp Decoded from the Bytes in Table E-3

2018-12-18 18:46:15, 718,308µs

Use the RTP packet times in the video frame stream and the KLV stream to match up the KLV meta data time stamp with each received video frame.

E.6 References

Camera Resectioning. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Camera_resectioning#Zhang's_method

OpenCV. (2017, October 24). *Camera Calibration and 3D Reconstruction*. Retrieved from OpenCV 3.3.1 Documentation: https://docs.opencv.org/3.3.1/d9/d0c/group__calib3d.html

Pinhole camera model. (n.d.). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Pinhole_camera_model

SMPTE. (2017, August 21). *ST 336:2017 - SMPTE Standard - Data Encoding Protocol using Key-Length-Value.* Retrieved from IEEE Xplore Digital Library: https://ieeexplore.ieee.org/document/8019807

Abbreviations and Acronyms

Abbreviation or Acronym	Definition
API	Application Program Interface
CCS	Camera Coordinate System, the coordinate system of the Position Sensor.
CRC	Cyclic Redundancy Check
IEEE	Institute of Electrical and Electronic Engineers
IRED	Infrared light Emitting Diode
LED	Light Emitting Diode
LOS	Line of Sight
OOV	Out of Volume
PSE	Power Sourcing Equipment
PSU	Position Sensor Unit
Rev xx	Combined firmware revision. For example, rev 24 refers to combined firmware revision 024.
RMS	Root Mean Square
SCU	System Control Unit
SROM	Serial Read Only Memory
TCS	Tool Coordinate System, the local coordinate system in the tool's frame of reference.
TIP	Tool-In-Port
UV	Refers to the rows and columns on the Position Sensor. U is the column number and V is the row number
VCU	Video Camera Unit
VCS	Video Coordinate System, the coordinate system of the Vega video camera.

Glossary

6D transformation

Mathematical operation that describes the transformation from one 3D coordinate system to another. A 6D transformation consists of a rotation in 3D, followed by a translation.

6D/3D

6D transformation of a tool and optional relative translation of tool markers.

characterized measurement volume

The characterized measurement volume is the volume within the field of view where accuracy is within specified limits. NDI cannot guarantee measurement accuracy performed outside this region.

faces

Tool faces are separate rigid bodies that make up a tool. Up to eight faces can be defined for one tool.

firmware

Firmware is a computer program stored in Polaris hardware that controls the Polaris system.

frame rate

The frame rate (or base frame rate) is the rate at which the system takes images of the measurement volume.

KLV

Key-Length-Value (KLV) is an optional type of data for an RTP packet defined by SMPTE336M, containing a 16-byte key for the value, a 1-byte length of the value, and the value.

maximum 3D error

Maximum 3D error applies to individual markers. This parameter in the tool definition file specifies the maximum allowable difference between the actual and expected location of a marker on a tool.

maximum marker angle

The maximum marker angle parameter in the tool definition file determines if the Position Sensor can view a specific marker and whether it should be included in the transformation calculated for the tool.

missing

If the system cannot detect a marker, that marker is considered missing. If the system cannot detect enough markers on a tool to determine a transformation, that tool is considered missing.

Quaternion

Quaternions can be used to describe a 3D rotation. Alternatives include Euler Angles or Polar/Spherical notation.

RTP

Real-time Transport Protocol (RTP) is a network protocol for delivering audio and video over a network. The Vega video camera server implements this protocol. In the video camera implementation, individual RTP packets may contain portions of an encoded video image or meta-data with a time stamp.

RTSP

Real-Time Streaming Protocol (RTSP) is the network control protocol the Vega video camera server uses to set up the RTP video stream with a video client.

SCU

The System Control Unit (SCU) is a component of the hybrid Polaris Vega system.

stray marker

A stray marker is a marker that is not part of a tool.

SROM device

A tool definition file can be programmed into the SROM device so that the tool can carry its own information for automatic retrieval by an NDI measurement system.

switch

A switch, when activated, initiates certain actions in the associated software application. A tool may have switches incorporated into its design.

tool definition file

A tool definition file stores information about a tool. This includes information such as the placement of the tool's markers, the location of its origin, and its manufacturing data. A tool definition file is formatted as .rom.

tracking rate

The tracking rate is the rate at which the system reports transformations for all the tools being tracked. The tracking rate cannot exceed the frame rate.

VCU

The Video Camera Unit (VCU) is an optional component of the passive Polaris Vega system.