Aurora Application Program Interface Guide

Revision 4
October 2011

Revision Status

Revision Number	Date	Description
1	April 2007	Initial release.
2	November 2007	Added GET command, added dome volume and metal resistance to SFLIST command
3	July 2010	Added RESET command
4	October 2011	Added TTFG volume to SFLIST command. Deprecated DSTART and DSTOP commands

Part Number: IL-1070114

Printed in Canada.

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

Copyright 2007, 2011, Northern Digital Inc.

All rights reserved. No part of this document may be reproduced, transcribed, transmitted, distributed, modified, merged or translated into any language in any form by any means - graphic, electronic, or mechanical, including but not limited to photocopying, recording, taping or information storage and retrieval systems - without the prior written consent of Northern Digital Inc. Certain copying of the software included herein is unlawful. Refer to your software license agreement for information respecting permitted copying.

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.

Table of Contents

Read Me First!

This guide describes revision D.001.007 of the Aurora Application Program Interface (API). The APIREV (page 10) command returns the revision of the API that is compatible with the firmware in your Aurora System. If the APIREV command is not supported by your system, contact NDI technical support.

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

Warnings

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

When using reply option 0800 with the BX (page 12) or TX (page 63) command, you must take appropriate action to detect when a tool is out of volume, and determine whether this situation is detrimental to your application. If a tool is out of volume, reply option 0800 enables the system to return data that may lead to inaccurate conclusions and may cause personal injury.

Disclaimers

Due to the nature of the mathematical model that the Aurora System uses to produce transformations, there is a very infrequent occurrence where the system may randomly return a single frame of significantly inaccurate or misleading data. To reduce the impact of this single frame on a measuring task, be aware of the possibility of this occurrence, and take such data into consideration when collecting transformations.

A complete list of the warnings, classifications, and approvals that apply to the Aurora System is included in the user guide shipped with the system.

Contact Information

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

INTERNATIONAL HEADQUARTERS:
NORTHERN DIGITAL INC.
103 Randall Drive
Waterloo, ON, Canada N2V 1C5

Phone: + 1 (519) 884-5142
Toll Free: + 1 (877) 634-6340
Global: + (800) 634-634-00
Fax: + 1 (519) 884-5184

Email: support@ndigital.com
Website: www.ndigital.com

EUROPEAN OFFICE:
NDI EUROPE GmbH
Fritz-Reichle-Ring 2
D-78315 Radolfzell
Germany

Phone: + 49 (77 32) 939 19 00
Global: + (800) 634 634 00
Fax: + 49 (77 32) 939 19 09

Email: support@ndieurope.com
Website: www.ndieurope.com

ASIA PACIFIC OFFICE:
NDI ASIA PACIFIC
Unit 301, 3/F Core Building 1
No. 1 Science Park East Avenue,
Hong Kong Science Park,
Shatin, New Territories,
Hong Kong

Phone: + (852) 2802 2205
Fax: + (852) 2802 0060
Email: APsupport@ndigital.com
Website: www.ndigital.com

Updates

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, check the NDI Support Site regularly for update information:

http://support.ndigital.com.

1 List of Commands

Command	Page	Description
APIREV	10	Returns the API revision number that functions with your system.
BEEP	11	Sounds the system beeper.
BX	12	Returns the latest tool transformations and system status in binary format.
COMM	16	Sets the serial communication settings of the system.
DSTART	18	Deprecated. Use TSTART.
DSTOP	19	Deprecated. Use TSTOP.
ECHO	20	Returns exactly what is sent with the command.
GET	21	Returns the values of user parameters.
INIT	22	Initializes the system.
LED	23	Changes the state of visible LEDs on a tool.
PDIS	25	Disables the reporting of transformations for a particular port handle.
PENA	26	Enables reporting of transformations for a particular port handle.
PHF	27	Releases system resources from an unused port handle.
PHINF	28	Returns information about the tool associated with the port handle.
PHSR	33	Returns the number of assigned port handles and the port status for each one. Assigns a port handle to a tool.
PINIT	36	Initializes a port handle.
PPRD	38	Reads data from the SROM device in a tool.
PPWR	40	Writes data to the SROM device in a tool.
PSEL	42	Selects a tool SROM device as the target for reading or writing with PPRD or PPWR.
PSOUT	43	Sets the status of the GPIO in the Aurora System.
PSRCH	45	Returns a list of valid SROM device IDs for a tool.
PURD	46	Reads data from the user section of the SROM device in a tool.
PUWR	48	Writes data to the user section of the SROM device in a tool.
PVWR	50	Overrides a tool definition file in a tool, and can be used to test a tool definition file before permanently recording the tool definition file onto the SROM device.
RESET	52	Resets the system
Serial break	8	Resets the system.
SFLIST	54	Returns information about the supported features of the system.
TSTART	59	Starts Tracking mode.
TSTOP	61	Stops Tracking mode.
TTCFG	62	Sets up a configuration for a tool, so that you can test the tool without using a tool definition file.
TX	63	Returns the latest tool transformations and system status in text format.
VER	67	Returns the firmware revision number of critical processors installed in the system.
VSEL	69	Selects a characterized measurement volume.

2 Communicating with an Aurora System

This chapter describes various aspects of communicating with an Aurora System. It contains the following sections:

2.1 Communication Overview

From the application perspective, the Aurora 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>`.

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. There may be some delay in the response of certain commands; see "Receiving System Replies" on page 4 for details.

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.

2.2 Changes in Implementation

This chapter describes the changes in implementation from previous versions of the Aurora API.

Note If you have written an application for an NDI Polaris System, see "For Polaris Users" on page 73 for differences between the Polaris API and the Aurora API.

Changes from API Revisions Prior to D.001.004

Read this section if you have written an application for an Aurora System with combined firmware revision 005 or earlier.

1. **New commands**:
   - APIREV (page 10) returns the API revision number.
   - ECHO (page 20) returns exactly what is sent with the command.

2. **Change to BX/TX:** Bit 8 in the <Port Status> section of reply option 0x0001 has been defined as “a sensor coil is broken.” This bit is set when a sensor coil lead wire breaks, either along the lead wire length or at its connection points.

Note Broken sensor coils that result in a short will not be detected.

3. **API revisions**: The API now has a revision number. If you upgrade the firmware, you can check the API revision to determine whether your application software may need to be updated. In the event of a firmware change that does not affect the API, the API revision number will remain the same, and any application software written using that API revision will continue to function with the new firmware revision. To determine the API revision programmed into your system, use the command APIREV (page 10).

2.3 Operating Modes

The Aurora System has two modes of operation: Setup and Tracking. 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. Typical Setup mode tasks may include initializing the system, writing to the SROM device on a tool, or checking the system firmware revisions.

The order of the commands sent while in Setup mode is important. For example, a port handle must be initialized (PINIT) before it can be enabled (PENA). Refer to the command documentation in Chapter 3 to see the prerequisites for each command.

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

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 BX and TX 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.

2.4 General Syntax

Commands must be sent from the host computer to the system in one of the two following formats.

Note To ensure the integrity of data transmission, NDI recommends using format 1, as well as verifying the returned Cyclic Redundancy Check (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. Commands and parameters are not case-sensitive.

This format requires a 16-bit CRC 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 $x^{16} + x^{15} + x^2 + 1$. See “Sample C Routines” on page 82 for sample code to calculate the CRC.

Format 2

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

A <SPACE> 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. Commands and parameters are not case-sensitive.

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 troubleshooting.

To ensure the integrity of data transmission, NDI recommends using format 1, as well as verifying the returned CRC on the host computer.

2.5 Receiving System Replies

Note The Aurora System may take longer to send a response for some commands than for other commands. For example, PINIT and TSTART may take up to five seconds. A reset may take as long as 12 seconds under special circumstances (for example, when a Field Generator is connected for the first time). If a timeout is detected, sending a serial break to the system should return the system to normal operation.

Binary Replies

The BX command returns binary replies. All other commands return ASCII replies.

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

<Reply><CRC16>

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, or ERROR<error code>. The <error code> is a two-digit hexadecimal error number. See “Error Code Definitions” on page 70 for a listing of all the error messages associated with error numbers.

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

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

ASCII Replies

All commands return ASCII replies except BX, which returns 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`, or `ERROR<error code>`. The <error code> is a two-digit hexadecimal error number. See “Error Code Definitions” on page 70 for a listing of all the error messages associated with error numbers.

2.6 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.

- 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 8 at a constant rate of 40 Hz. 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 12) or TX (page 63).
- 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. The shape type and shape parameters are returned with the command SFLIST (page 54).
- 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, use the command VER 5 (page 67).

- 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. See the command APIREV (page 10) for details.

2.7 Port Handles

About Port Handles

The system assigns each tool a port handle. Port handles are two characters in hexadecimal format, 0x0A to 0xFF. 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 33)	Returns the number of assigned port handles and the port status for each one. Assigns a port handle to a tool.
PVWR (page 50)	Overrides a tool definition file in a tool, and can be used to test a tool definition file before permanently recording the tool definition file onto the SROM device.
PINIT (page 36)	Initializes a port handle.
PHINF (page 28)	Returns port handle status, and information about the tool associated with the port handle, including physical port location.
PHF (page 27)	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 a new port handle to the tool. The old handle is reported as disabled and should be freed using PHF.
PENA (page 26)	Enables reporting of transformations for a particular port handle.
PDIS (page 25)	Disables the reporting of transformations for a particular port handle.

The order in which these commands are used is detailed in Figure 2-1 on page 7.

Disabled Transformations

A transformation may be reported as DISABLED if:

- the port handle was not enabled with PENA (page 26),
- the port handle has been disabled with PDIS (page 25), or
- a 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 28).

Flow Chart for Port Handle Usage

Figure 2-1 on page 7 details the logic for using port handles.

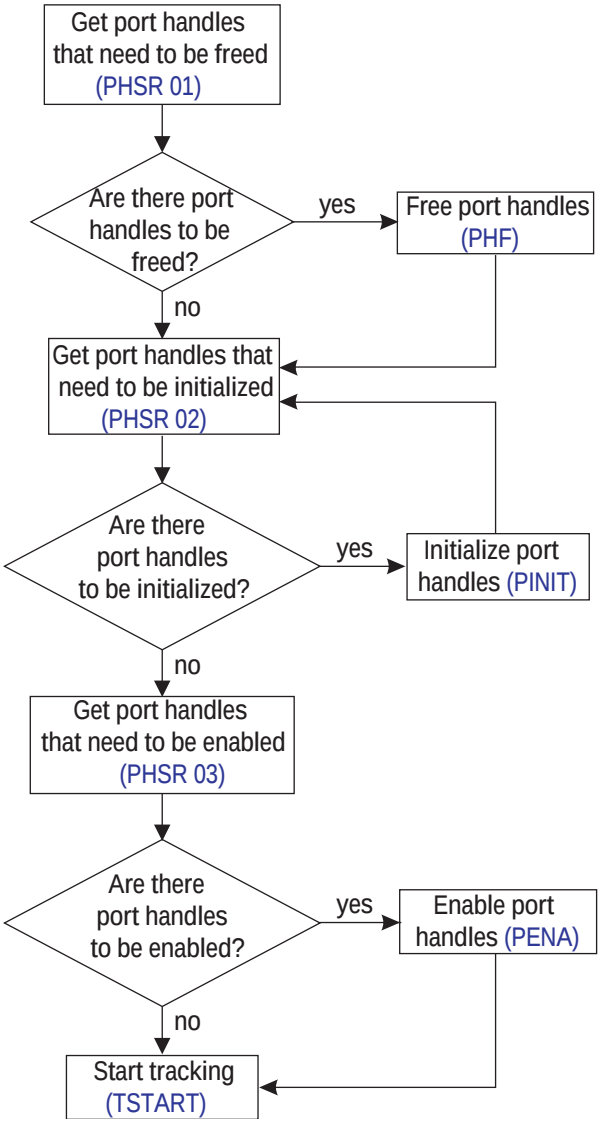


Figure 2-1 Flow Chart for Port Handle Usage

Note For a split port on a dual 5DOF tool, the first PHSR sent will report only one port handle. After the port handle is initialized, it is assigned to channel 0. You must then use PHSR again to assign a port handle to channel 1. The port handle for channel 1 is initialized automatically.

3 Commands

Before sending any commands to the system, read the user guide that accompanied your system to ensure that you have a full understanding of the system functionality. The user guide is available on the NDI Support Site at http://support.ndigital.com.

Resetting the System with a Serial Break

Resets the system.

Operating Mode

All modes

Syntax

The method depends on the host computer.

Reply

`RESET<CRC16><CR>`

Usage Notes

1. The serial break is a special condition, and is specific to the host computer operating system. Refer to your computer manuals to determine how to generate a serial break.
2. The serial break is a good recovery method in the following situations:
   - System warm boot: the system is powered up, but you don't know the communication setup, or which mode the system is in.
   - Synchronization error: while in the Tracking mode, the BX or TX reply status returns that synchronization errors have occurred.
   - Loss of communication: the host computer and the system can no longer communicate.

Note Under normal operation, the Aurora System will return a response to any given command in under 10 seconds. If a timeout is detected, sending a serial break to the system should resolve the problem and return the system to normal operation.

3. After a serial break:
   - All processors receive a hard reset.
   - The system serial communication settings return to the default values: 9600 baud, 8 data bits, no parity, 1 stop bit, and no hardware handshaking.
   - A distinctive 2-beep sequence sounds to indicate that the system is re-initialized.

Example

Command:

N/A

Reply:

RESETBE6F<CR>

APIREV

Returns the API revision number.

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>

See page 70 for error code definitions.

Reply Component	Description
Family	1 ASCII character For the Aurora System, this character is always D. (Other types of NDI measurement systems use other characters.)
Major revision number	3 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	3 ASCII characters The minor revision 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:
D.001.00450D4

BEEP

Sounds the system beeper.

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>`

See page 70 for error code definitions.

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.

Example

Command:

`BEEP 1`

Reply:

`1D4C1`

BX

Returns the latest tool transformations and system status information in binary format.

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. Reply option 0800 is not reported separately from reply option 0001; it simply enables the system to return certain information. **Valid Values:**	
	0001	Transformation data (default)
	0800	Out-of-volume transformations

Replies

Upon Success:

```
<Start Sequence><Reply Length><Header CRC><Number of Handles>
<Handle 1><Handle 1 Status><Reply Option 0001 Data>
...
<Handle n><Handle n Status><Reply Option 0001 Data>
<System Status><CRC16>
```

Note The reply for the BX command is binary data.

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

On Error:

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

See page 70 for error code definitions.

Reply Component	Description	
`Start Sequence`	2 bytes: A5C4 Indicates the start of the BX reply.	
`Reply Length`	2 bytes Indicates the number of bytes in the reply body between the <Header CRC> and the <CRC16>, exclusive.	
`Header CRC`	16 bits CRC of <Start Sequence> and <Reply Length>	
`Number of Handles`	1 byte The number of port handles for which transformations are returned.	
`Handle n`	1 byte The port handle whose transformation follows.	
`Handle Status`	1 byte **Possible Values:**	
	01	Valid
	02	Missing
	04	Disabled
`Reply Option m Data`	The data specific to the requested reply option. See the reply option information below for details:	
	Reply option 0001 (transformation data) (default)	
	Reply option 0800 (reporting all transformations)	
`System Status`	2 bytes **Bit field:**	
	bit 0	System communication synchronization error
	bit 1	Reserved
	bit 2	System CRC error
	bit 3	Recoverable system processing exception
	bit 4	Hardware failure
	bit 5	Hardware change. This bit is set if the Field Generator is disconnected from the System Control Unit.
	bit 6	Some port handle has become occupied
	bit 7	Some port handle has become unoccupied
	bits 8 to 15	Reserved

Reply Option 0001 - Transformation Data

<Reply Option 0001 Data> = <Q_0><Q_x><Q_y><Q_z><T_x><T_y><T_z><Indicator Value>
<Port Status><Frame Number>
or
<Reply Option 0001 Data> = <Port Status><Frame Number>

Reply Component	Description		
Q0, Qx, Qy, Qz	4 bytes each Rotational components of the transformation, quaternion, unitless, reported as IEEE 32 bit, single precision, floating point numbers.		
Tx, Ty, Tz	4 bytes each Translational components of the transformation, in mm, reported as IEEE 32 bit, single precision, floating point numbers.		
Indicator Value	4 bytes An estimate of how well the Aurora System calculated the transformation. Value range is ≥ 0. For 6DOF tools, the indicator value compares sensor coil measurements to the tool's design (as described by its SROM device or tool definition file). For 5DOF tools, the indicator value is always zero. Unitless, reported as IEEE 32-bit, single precision, floating point number		
Port Status	4 bytes **Bit field:**		
	bit 0	Occupied	Bits 0 to 3 are shared for dual, 5DOF tools.
	bit 1	GPIO line 1 closed	
	bit 2	GPIO line 2 closed	
	bit 3	GPIO line 3 closed	
	bit 4	Initialized	
	bit 5	Enabled	
	bit 6	Out of volume	
	bit 7	Partially out of volume. This bit is set only for 6DOF tools, when one sensor coil is inside the measurement volume and the other sensor coil is outside the measurement volume.	
	bit 8	A sensor coil is broken. This bit is set when a sensor coil lead wire breaks, either along the lead wire length or at its connection points.	
	bits 9 to 11	Reserved	
	bit 12	Processing exception	
	bits 13 to 31	Reserved	

Reply Component	Description
`Frame Number`	4 byte unsigned number The frame number is an internal counter related to data acquisition. The counter starts at power up and does not reset until the system is reset, the system is powered up again, or reply option 80 is sent with the TSTART command. The frame rate is 40 Hz. 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.

Reply Option 0800 - Reporting All Transformations

This option enables the reporting of transformations when the tool is out of volume. This reply option must be OR’d with reply option 0001.

Warning! When using reply option 0800 with the BX command, you must take appropriate action to detect when a tool is out of volume, and determine whether this situation is detrimental to your application. If a tool is out of volume, reply option 0800 enables the system to return data that may lead to inaccurate conclusions and may cause personal injury.

Usage Notes

1. 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 63).
2. To use the BX command, the data bits parameter must be set to 0 (8 bits) using COMM (page 16).
3. Replies are returned in little endian format.
4. See the user guide for detailed descriptions of the port status and system status bits.
5. By default, transformations will not be reported if the tool is either partially or wholly out of the characterized measurement volume. To report these transformations, you must use reply option 0800 OR’d with reply option 0001. The accuracy of these transformations is unknown.
6. Bit 8 in the <Port Status> section of reply option 0001 may not always indicate when a lead wire is broken. Broken sensor coils that result in a short will not be detected.

Example

Command:

```
BX 0801
```

Reply:

```
A5C4005723130201013F3AF3CABE5B7209BF1C07713E635592C39E831F43332973C500511
33DA5BD9F00000031000002CC02013EA1B5D03D137D21BD787C673F72394A4286B6CB4360
6EF4C50468C13ED4E74100000031000002CD000059C9
```

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

COMM

Sets the serial communication settings for the system.

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 Aurora System and the host computer, in bits per second. **Valid Values:**	
	0	9600 bps (default)
	1	14 400 bps
	2	19 200 bps
	3	38 400 bps
	4	57 600 bps
	5	115 200 bps
	6	921 600 bps, see usage note 5
	A	230 400 bps, see usage note 6
`Data Bits`	The data bits parameter must be set to 8 bits in order to use the BX command. **Valid Values:**	
	0	8 bits (default)
	1	7 bits
`Parity`	**Valid Values:**	
	0	None (default)
	1	Odd
	2	Even
`Stop Bits`	**Valid Values:**	
	0	1 bit (default)
	1	2 bits
`Hardware Handshaking`	**Valid Values:**	
	0	Off (default)
	1	On

Replies

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

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

See page 70 for error code definitions.

Usage Notes

1. 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).
2. To use the BX command you must set the data bits parameter to 0 (8 bits).
3. 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.
4. NDI strongly recommends using hardware handshaking when using the higher baud rates.
5. 921600 baud is only available via USB and with combined firmware revision 009 (API Revision D.001.006) or later. A USB port is available on Aurora V2 systems.
6. Baud rate parameter "A" is available with combined firmware revision 009 (API Revision D.001.006) or later.

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.

DSTART

This command is deprecated. Use TSTART (page 59) instead.

DSTOP

This command is deprecated. Use TSTOP (page 61) instead.

ECHO

Returns exactly what is sent with the command.

Operating Mode

All modes

Syntax

`ECHO<SPACE><Four or more ASCII characters><CR>`

Replies

Upon Success:

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

On Error:

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

See page 70 for error code definitions.

Example

Command:

```
ECHO Testing!
```

Reply:

```
Testing!A81C
```

GET

The GET command returns the values of user parameters.

(The only user parameter values currently available are the timeouts for the API commands.)

Operating Mode

All modes

Syntax

```
GET<SPACE><User Parameter Name><CR>
(Only Info.Timeout is currently supported.)
```

Parameter	Description
`User Parameter Name`	A string, identifying the name of the user parameter. May include a trailing wild card character (*). Use **GET *** to return all user parameter values.

Replies

Upon success

```
 <User Parameter Name>=<value><LF>(repeated for each user parameter
name)<CRC16><CR>
```

On error

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

See page 70 for error code definitions.

Example

Command

```
GET Info.Timeout.PINIT
```

Reply

```
Info.Timeout.PINIT=5<LF>96A7
```

INIT

Initializes the system.

Operating Mode

All modes

Syntax

`INIT<SPACE><CR>`

Replies

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

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

See page 70 for error code definitions.

Usage Notes

1. During power up or system reset, the system configuration is determined. The configuration includes firmware revisions and the characterized measurement volume for which the Field Generator has been calibrated. The INIT command ensures that the system configuration was determined successfully.
2. The system will automatically return to Setup mode after using the INIT command.

Example

Command:
`INIT`

Reply:
`OKAYA896`

LED

Changes the state of visible LEDs on a tool.

Operating Mode

All modes

Prerequisite Command

PINIT (page 36)

Syntax

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

Parameter	Description	
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF	
`LED Number`	Specifies the LED. The LED number does not correspond to the GPIO line number. For example, LED 1 corresponds to the first LED on the tool, which may not be on GPIO line 1. **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>`

See page 70 for error code definitions.

Usage Notes

The GPIO line must be defined as “visible LED” for the LED command to function.

Example

Command:

```
LED 0A1S
```

Reply:

```
0KAYA896
```

This turns on solid the first LED on the tool associated with port handle 0A.

PDIS

Disables the reporting of transformations for a particular port handle.

Operating Mode

Setup

Prerequisite Command

PENA (page 26)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF

Replies

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

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

See page 70 for error code definitions.

Example

Command:
`PDIS 0A`

Reply:
`OKAYA896`

PENA

Enables the reporting of transformations for a particular port handle.

Operating Mode

Setup

Prerequisite Command

PINIT (page 36)

Syntax

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

Parameter	Description	
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF	
`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 switches and LEDs, but no sensor coils. No transformations are returned for a button box tool, but switch status is returned.

Replies

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

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

See page 70 for error code definitions.

Usage Notes

The tool tracking priority selected does not currently affect tracking behaviour.

Example

Command:
```
PENA 0AD
```

Reply:
```
OKAYA896
```

PHF

Releases system resources from an unused port handle.

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

```
PHF<SPACE><Port Handle><CR>
```

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF

Replies

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

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

See page 70 for error code definitions.

Usage Notes

1. 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.
2. 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 0A
```

Reply:
```
OKAYA896
```

This frees port handle 0A, so it is no longer assigned to a tool.

PHINF

Returns information about the tool associated with the port handle.

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

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

Parameter	Description	
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF	
`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)
	0004	Tool part number
	0008	Switch and visible LED information
	0020	Physical port location
	0040	GPIO line definitions

Replies

Upon Success:

If the port handle has been initialized:

`<Reply Option 0001 Data><Reply Option 0004 Data>...<Reply Option 0040 Data><CRC16><CR>`

If the tool has been disconnected since the port handle was assigned:

`UNOCCUPIED<CRC16><CR>`

On Error:

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

See page 70 for error code definitions.

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	8 characters <Tool Type> = <Main Type><Number of Switches><Number of Visible LEDs><Reserved><Subtype>		
	Main Type	2 hexadecimal characters **Possible Values:**	
		01	Reference
		02	Probe
		03	Button box or foot switch
		04	Software-defined
		05 and 06	Reserved
		07	Calibration device
		08 to 0A	Reserved
		0B	Catheter
		0C to FF	Reserved
	Number of Switches	1 character	
	Number of Visible LEDs	1 character	
	Reserved	2 characters	
	Subtype	2 characters	
Manufacturer's ID	12 characters		
Tool Revision	3 characters		
Serial Number	8 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 2005 is 105)	

Reply Component	Description	
`Port Handle Status`	2 hexadecimal characters (8 bits) **Bit field:**	
	bit 0	Occupied
	bit 1	GPIO line 1 closed
	bit 2	GPIO line 2 closed
	bit 3	GPIO line 3 closed
	bit 4	Port handle initialized
	bit 5	Port handle enabled
	bits 6 and 7	Reserved

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`	2 hexadecimal characters (8 bits) This option reports the information found in the tool definition file. It is not information sensed by the hardware. **Bit field:**	
	bit 0	Reserved
	bit 1	Input supported on GPIO line 1
	bit 2	Input supported on GPIO line 2
	bit 3	Input supported on GPIO line 3
	bit 4	Tool tracking LED supported
	bit 5	Visible LED supported on GPIO line 1
	bit 6	Visible LED supported on GPIO line 2
	bit 7	Visible LED supported on GPIO line 3

Reply Option 0020 - Physical Port Location

```
<Reply Option 0020 Data> = <System Control Unit Serial Number><Reserved>
<Port Number><Channel Number>
```

Reply Component	Description
`System Control Unit Serial Number`	8 characters
`Reserved`	2 characters
`Port Number`	2 ASCII characters **Possible Values:** 01 to 04.
`Channel Number`	2 characters For a dual, 5DOF tool. **Possible Values:** 00 or 01

Reply Option 0040 - GPIO Line Definitions

```
<Reply Option 0040 Data> = <GPIO line 1><GPIO line 2><GPIO line 3>
<Reserved>
```

Reply Component	Description	
GPIO `n`	1 hexadecimal character Definition of the n^{th} general purpose input/output line. **Possible Values:**	
	0	Not available, or defined as a tool tracking LED
	1	Input
	2	Output
	3	Visible LED
	4	Always high
`Reserved`	1 character	

Usage Notes

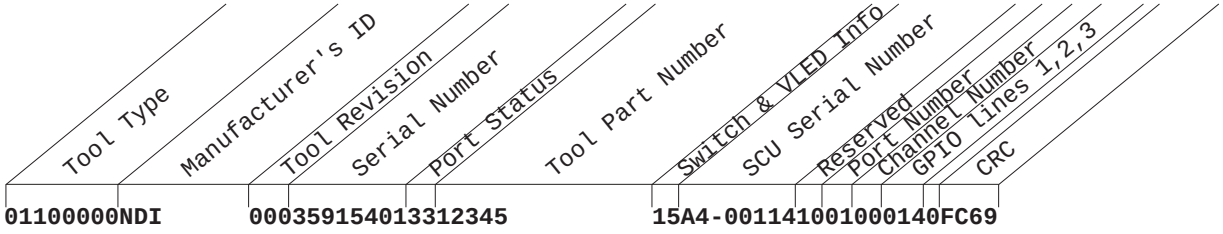
1. With the exception of reply option 0020 and the port status, all of the information returned in with this command is read from the tool's SROM device or from a tool definition file assigned to the port handle with PVWR (page 50).
2. If the port handle has not been initialized, the system will correctly report only the SCU serial number and port number (in reply option 0020), and the port status (in reply option 0020).
3. **Reply Option 0040:** If a GPIO line is defined as a visible LED, it will also be reported in reply option 0008. The tracking LED will be reported as available and is also reported in reply option 0008.

Example

Command:

```
PHINF 0A006D
```

Reply:



PHSR

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

Operating Mode

Setup

Prerequisite Command

INIT (page 22)

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>`

See page 70 for error code definitions.

<table>
<tr><th>Reply Component</th><th colspan="2">Description</th></tr>
<tr><td>Number of Port Handles</td><td colspan="2">2 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.</td></tr>
<tr><td>nth Port Handle</td><td colspan="2">2 hexadecimal characters

Specifies the port handle whose status follows.</td></tr>
<tr><td>nth Port Handle Status</td><td colspan="2">3 hexadecimal characters (12 bits)

Bit field:</td></tr>
<tr><td></td><td>bit 0</td><td>Occupied</td></tr>
<tr><td></td><td>bit 1</td><td>GPIO line 1 closed</td></tr>
<tr><td></td><td>bit 2</td><td>GPIO line 2 closed</td></tr>
<tr><td></td><td>bit 3</td><td>GPIO line 3 closed</td></tr>
<tr><td></td><td>bit 4</td><td>Initialized</td></tr>
<tr><td></td><td>bit 5</td><td>Enabled</td></tr>
<tr><td></td><td>bit 6 to 11</td><td>Reserved</td></tr>
</table>

Usage Notes

1. When you send the PHSR command, the system will detect and assign port handles to any tools that do not already have a port handle assigned (i.e. any tools that were connected after the last PHSR call). It will then return the requested port handle information.
2. For a split port on a dual 5DOF tool, the first PHSR sent will report only one port handle. After the port handle is initialized, it is assigned to channel 0. You must then use PHSR again to assign a port handle to channel 1. The port handle for channel 1 is initialized automatically. See “Flow Chart for Port Handle Usage” on page 7.
3. If you disconnect a 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.
4. If you connect a tool to the system while the system is in Tracking mode (either reconnecting a tool that was just disconnected, or connecting a new tool), 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 2-1 on page 7.
   c) Re-enter Tracking mode (TSTART).

Examples

Command:
```
PHSR
```

Reply:
```
001414
```

In this case, there are no tools connected to the system.

Command:
```
PHSR
```

Reply:
```
010A001C1B5
```

In this case, one tool is connected to the system and it has been assigned port handle 0A. This port handle is not initialized or enabled.

Command:
```
PHSR 03
```

Reply:
```
040A01F0B01F0C01F0D01F2DDB
```

In this case, four tool is connected to the system and have been assigned port handles 0A, 0B, 0C, and 0D. All four port handles are initialized but not enabled.

PINIT

Initializes a port handle.

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF

Replies

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

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

See page 70 for error code definitions.

Usage Notes

1. Use PINIT to initialize a port handle for a tool after you have assigned a port handle using PHSR. If you are using PVWR to override the SROM device in a tool, use PINIT after PVWR.
2. For a split port on a dual 5DOF tool, the first PHSR sent will report only one port handle. After the port handle is initialized using PINIT, it is assigned to channel 0. You must then use PHSR again to assign a port handle to channel 1. The port handle for channel 1 is initialized automatically. See “Flow Chart for Port Handle Usage” on page 7.
3. If the tool description is read from a tool definition file that has been loaded using PVWR (page 50), initialization involves unpacking and verifying the tool definition file. This process is almost instantaneous.
4. If the tool description is read from an SROM device, initialization involves reading, unpacking, and verifying the tool definition file contents. This process takes approximately two seconds if successful, or several seconds longer if a problem is encountered and retries are attempted by the system.
5. A tool will still initialize when the system returns `WARNING`.

Example

Command:

```
PINIT 0A
```

Reply:

```
OKAYA896
```

This initializes port handle 0A.

PPRD

Reads data from the SROM device of a tool.

Operating Mode

Setup

Prerequisite Command

PSEL (page 42)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF
`SROM Device Address`	4 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>`

See page 70 for error code definitions.

Usage Notes

1. 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.
2. PPRD reads 64 bytes of data from the SROM device starting at a specified SROM device address.
3. You must select the SROM device as the reading target with PSEL (page 42) before sending the PPRD command.

Example

Command:

```
PPRD 0A0000
```

Reply:

```
0100410100000002000000000053CDD33020000000200000000000000000000000000000000
00000000000000000000000000000000008000000780B6D3D9E0F73BE7DE9
```

PPWR

Writes data to the SROM device in a tool.

Operating Mode

Setup

Prerequisite Command

PSEL (page 42)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF
`SROM Device Address`	4 hexadecimal characters **Valid Values:** 0x0000 to 0x07C0
`SROM Device Data`	64 bytes (128 hexadecimal characters) of data

Replies

Command:
`OKAY<CRC16><CR>`

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

See page 70 for error code definitions.

Usage Notes

1. PPWR writes 64 bytes of data to the SROM device starting at a specified SROM device address.
2. You must select the SROM device as the reading target with PSEL (page 42) before sending the PPWR command.
3. 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).
4. The tool description section of a tool SROM device is a 1 KB, write-once area that must be written in 64-byte chunks. If the information being written to the system is less than 64 bytes,

then the remainder of the chunk must be padded out with 1s to maintain the 64-byte size before being written to the SROM device.

5. An SROM device is considered blank if its contents are all 0xFFs.

Example

Command:

```
PPWR 0A00C000000000000000000000000000000000000000000000000000000000C8410000C8
4100000000000000000000000000000000000000000000000000000000000002C44
```

Reply:

```
OKAYA896
```

PSEL

Selects an SROM device target for a tool.

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF
`Tool SROM Device ID`	16 characters Use PSRCH (page 45) to determine the IDs of the tool's SROM device.

Replies

Command:
`OKAY<CRC16><CR>`

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

See page 70 for error code definitions.

Example

Command:
`PSEL 0A0B3876530000005B`

Reply:
`OKAYA896`

This selects the SROM device with ID 0B3876530000005B for port handle 0A.

PSOUT

Sets the states of the general purpose input/output (GPIO) lines in a tool.

Operating Mode

All modes

Prerequisite Command

PINIT (page 36)

Syntax

```
PSOUT<SPACE><Port Handle><GPIO line 1 State><GPIO line 2 State>
<GPIO line 3 State><CR>
```

Parameter	Description	
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF	
`GPIO line n State`	State of the nth GPIO line **Valid Values:**	
	N	No change
	S	Solid on
	P	Pulse
	O	Off

Replies

Upon Success:

`OKAY<CRC16><CR>`

On Error:

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

See page 70 for error code definitions.

Usage Notes

You can check the status of an output using PHINF (page 28).

Example

Command:

```
PSOUT 0ANSN
```

Reply:

```
OKAYA896
```

This sets the three GPIO lines associated with port handle 0A to no change, solid on, and no change.

PSRCH

Returns a list of valid SROM device IDs for a tool.

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

```
PSRCH<SPACE><Port Handle><CR>
```

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF

Replies

Upon Success:

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

On Error:

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

See page 70 for error code definitions.

Reply Component	Description
`Number of SROM devices`	1 character
`SROM device n ID`	16 characters

Usage Notes

The tool 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 42).

Example

Command:

```
PSRCH 0A
```

Reply:

```
10B3876530000005B7FFF
```

In this case, there is one SROM device, with ID 0B3876530000005B.

PURD

Returns the tool information in the user section of the SROM device in a tool.

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF
`User SROM Device Address`	4 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>`

See page 70 for error code definitions.

Usage Notes

1. 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.
2. The PURD command returns 64 bytes of data at a time.

Example

Command:

```
PURD 0A0000
```

Reply:

```
0022446688AACCEE0022446688AACCEE0022446688AACCEE0022446688AACCEE002244668
8AACCEE0022446688AACCEE0022446688AACCEE0022446688AACCEE3CC0
```

PUWR

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

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

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

Parameters	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF
`User SROM device address`	4 hexadecimal characters **Valid Values:** 0x0000 to 0x03C0
`User SROM device data`	64 bytes of data to write (128 hexadecimal characters)

Replies

Command:
`OKAY<CRC16><CR>`

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

See page 70 for error code definitions.

Usage Notes

1. 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.
2. 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).
3. The user section of a tool SROM devices is a 1-KB, write-once area that must be written in 64-byte chunks. If the information being written to the system is less than 64 bytes, then the

remainder of the chunk must be padded out with 1s to maintain the 64-byte size before being written to the system.

4. Unused portions of the SROM device can be written to by setting the SROM device address appropriately. To determine which portions of the SROM device are unused, read the contents of the SROM device using PURD (page 46).

Example

Command:

```
PUWR 0A00800000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000A927
```

Reply:

```
OKAYA896
```

PVWR

Allows you to upload a tool definition file for a tool. The Aurora System will use the data in the uploaded tool definition file instead of the data in the tool's SROM device.

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF
`Start Address`	4 hexadecimal characters Increment the start address by 64 bytes with each chunk of data sent for a particular port handle. **Valid Values:** 0x0000 to 0x03C0
`Tool Definition Data`	128 hexadecimal characters

Replies

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

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

See page 70 for error code definitions.

Usage Notes

1. Use PVWR in the following cases:
   - To assign a tool definition file to a tool, to override the SROM device in the tool.
   - To assign a tool definition file to a tool, to test the tool definition file before permanently recording the tool definition file onto the SROM device.

2. 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).
3. 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.
4. After using PVWR, initialize (PINIT) and enable (PENA) the port handle in order to track the tool.
5. To permanently write a tool definition file to an SROM device, use PPWR (page 40).

Example

Command:

```
PVWR 0B00004E444900551C000001000000000000010100000001A419335A000000030000
0003000000000000400000000000000000000000000000000000000000000000000000000
```

Reply:

```
OKAYA896
```

RESET

Resets the system

Operating Mode

All modes

Syntax

`RESET<SPACE><Reply Option><CR>`

Parameter	Description	
`Reset Option`	Optional. Specifies the type of reset. If no reset option is specified, the system performs a RESET 1. **Valid Values:**	
	0	Generates a “soft” reset and resets the baud rate to 9600. System will not beep
	1	Same behaviour as resetting the system with a serial break. See page 8

Replies

Upon Success:
RESET 0
`OKAY<CRC16><CR>`

RESET 1
`RESET<CRC16><CR>`

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

See page 70 for error code definitions.

Usage Notes

1. RESET can only be used when the host computer and the Aurora System are at the same baud rate. To reset the system when the baud rates do not match or when the system is in an unknown state use a serial break, see page 8.
2. Reply option 1 (hard reset) is only valid with the Aurora V2 System. It is not supported on older Aurora Systems.
3. If the command is successful, the reply will be sent at the default communications setting; 9600 baud, 8 data bits, not parity, 1 stop bit, hardware handshaking off.

Example

Command:

```
RESET 1
```

Reply:

```
RESETBE6F
```

SFLIST

Returns information about the supported features of the system.

Operating Mode

Setup

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
	03	Number of volumes and volume shapes
	10	Number of ports

Replies

Upon Success:
`<Reply Option n Data>`

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

See page 70 for error code definitions.

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 03 (Number of volumes and volume shapes) Reply option 10 (Number of ports)

Reply Option 00 - Supported Features Summary

Reply Component	Description	
`Reply Option 00 Data`	8 hexadecimal characters (32 bits) **Bit field:**	
	bits 0 and 1	Reserved
	bit 2	Multiple volume characterization parameters supported
	bits 3 to 15	Reserved
	bit 16	Magnetic ports available
	bits 17 and 18	Reserved
	bit 19	Field Generator available
	bits 20 to 31	Reserved

Reply Option 03 - Volumes

`<Reply Option 03 Data> =`

`<Number of Volumes><1st Shape Type><1st Shape Parameters><Reserved><Metal Resistant><LF>`

`...`

`<nth Shape Type><nth Shape Parameters><Reserved><Metal Resistant><LF><CRC16><CR>`

Reply Component	Description	
`Number of Volumes`	1 hexadecimal character	
`nth Shape Type`	1 hexadecimal character **Possible Values:**	
	9	Cube volume
	A	Dome volume
`nth Shape Parameters`	10 parameters, 7 characters each (a sign, and six digits with an implied decimal in the position XXXX.XX) See the shape parameters below.	
`Reserved`	1 hexadecimal character	
`Metal Resistant`	1 hexadecimal character **Possible Values:**	
	0	no information
	1	metal resistant
	2	not metal resistant
	3-F	reserved

Shape Parameters

<n^{th} Shape Parameters> in reply option 03 returns the following values (illustrated in Figure 3-1 and Figure 3-2)

Cube Volume:

Shape Parameter	Value	Description
D1	-250 mm	Minimum *x* value
D2	+250 mm	Maximum *x* value
D3	-250 mm	Minimum *y* value
D4	+250 mm	Maximum *y* value
D5	-550 mm	Minimum *z* value
D6	-50 mm	Maximum *z* value
D7 to D10		Reserved

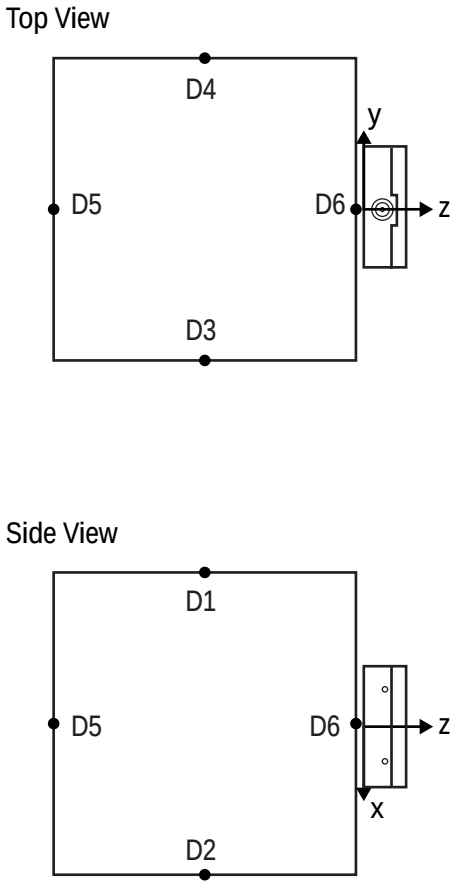


Figure 3-1 Cube Volume Parameters

Dome Volume:

There are currently two Field Generators that support the dome volume, the Planar Field Generator (PFG) and the Tabletop Field Generator (TTFG). The following table details values for both FGs.

Shape Parameter	Description	Value PFG (mm)	Value TTFG (mm)
D1	Offset from Field Generator	+50	+120
D2	Cylinder radius along x-axis	+480	+210
D3	Minimum dome radius	+50	+120
D4	Maximum dome radius	+660	+600
D5	Cylinder radius along y-axis **Note: If D5 is equal to 0, the y-axis radius is equal to the x-axis radius (circular cylinder)**	0(+480)	+300
D6	Maximum offset from Field Generator **Note: If D6 is equal to 0, the maximum offset is equal to D4 (the maximum dome radius)**	0(+660)	+600
D7 to D10	Reserved	—	—

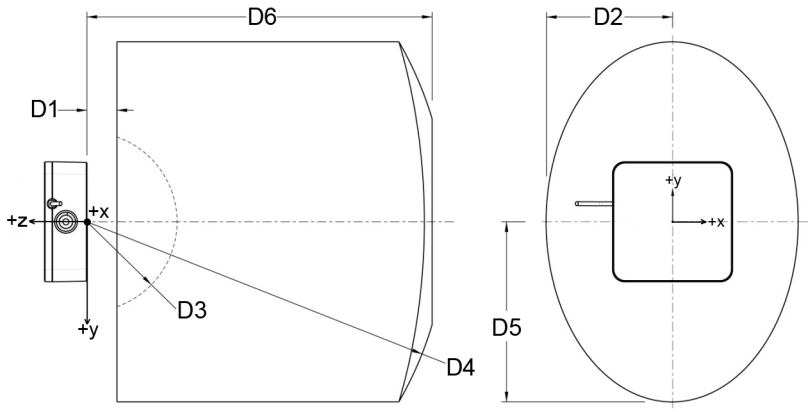


Figure 3-2 Dome Volume Parameters

Reply Option 10 - Number of Ports Available

Reply Component	Description
`Reply Option 10 Data`	2 hexadecimal characters

Usage Notes

Reply Option 03 Use both the shape type and the shape parameters to graphically represent the characterized measurement volume. There may be multiple volumes with the same shape type. All volumes of the same shape type use the shape parameters the same way.

Example

Command:

```
SFLIST 03
```

Reply:

```
29-025000+025000-025000+025000-055000-005000+000000+000000+000000
+00000011
A+005000+048000+005000+066000+000000+000000+000000+000000+000000+00000011
D3BB
```

TSTART

Starts Tracking mode.

Operating Mode

Setup

Prerequisite Command

INIT (page 22)

Syntax

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

Parameter	Description
`Reply Option`	40 (Optional, starts tracking in faster acquisition mode) 80 (Optional, resets the frame counter to zero) The reply options are hexadecimal numbers that can be OR'd (C0).

Replies

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

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

See page 70 for error code definitions.

Usage Notes

If the reply option 80 is not used, only resetting the system resets the counter for the frame number. The frame number is reported in reply option 0001 of the TX (page 63) and BX (page 12) commands.

Using reply option 40 will result in the following:

- A data acquisition rate of 66 Hz. (It may be lower when serial interface limits the speed.)
- A shorter frame length of 15 ms.
- Precision uncertainty increases by a factor of 2.

Example

Command:
TSTART

Reply:
0KAYA896

TSTOP

Stops Tracking mode.

Operating Mode

Tracking

Prerequisite Command

TSTART (page 59)

Syntax

`TSTOP<SPACE><CR>`

Replies

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

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

See page 70 for error code definitions.

Example

Command:
`TSTOP`

Reply:
`OKAYA896`

TTCFG

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

Operating Mode

Setup

Prerequisite Command

PHSR (page 33)

Syntax

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

Parameter	Description
`Port Handle`	2 hexadecimal characters **Valid Values:** 0A to FF

Replies

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

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

See page 70 for error code definitions.

Usage Notes

1. TTCFG internally sets up a test configuration for a 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.
2. After sending the TTCFG command, you will need to initialize (PINIT) and enable (PENA) the port handle before using any other commands that list these as prerequisites.
3. TTCFG configures the tool as dual, 5DOF. After initializing the port handle, a second port handle will be automatically generated. This second port handle will be already initialized.

Example

Command:
`TTCFG 0A`

Reply:
`OKAYA896`

TX

Returns the latest tool transformations and system status in text format.

Operating Mode

Tracking

Syntax

```
TX<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. Reply option 0800 is not reported separately from reply option 0001; it simply enables the system to return certain information. **Valid Values:**	
	0001	Transformation data (default)
	0800	Transformations not normally reported

Replies

Upon Success:

```
<Number of Handles><Handle 1><Reply Option 0001 Data><LF>
...
<Handle n><Reply Option 0001 Data><LF>
<System Status><CRC16><CR>
```

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

On Error:

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

See page 70 for error code definitions.

Reply Component	Description
`Number of Handles`	2 hexadecimal characters The number of port handles for which transformations are returned.
`Handle n`	2 hexadecimal characters The port handle whose transformation follows.

Reply Component	Description	
Reply Option m Data	The data specific to the requested reply option. See the reply option information below for details:	
	Reply option 0001 (transformation data) (default)	
	Reply option 0800 (reporting all transformations)	
System Status	4 hexadecimal characters (16 bits) The status of the system. **Bit field:**	
	bit 0	System communication synchronization error
	bit 1	Reserved
	bit 2	System CRC error
	bit 3	Recoverable system processing exception
	bit 4	Hardware failure
	bit 5	Hardware change. This bit is set if the Field Generator is disconnected from the System Control Unit.
	bit 6	Some port handle has become occupied
	bit 7	Some port handle has become unoccupied
	bits 8 to 15	Reserved

Reply Option 0001 - Transformation Data

<Reply Option 0001 Data> = <Q_0><Q_x><Q_y><Q_z><T_x><T_y><T_z><Indicator Value><Port Status><Frame Number>
or
<Reply Option 0001 Data> = MISSING<Port Status><Frame Number>

Reply Component	Description
Q0, Qx, Qy, Qz	6 characters each (a sign, and 5 decimal digits with an implied decimal in the position X . XXXX) Rotational component of the transformation, quaternion, unitless.
Tx, Ty, Tz	7 characters each (a sign, and 6 decimal digits with an implied decimal in the position XXXX . XX) Translational components of the transformation, in mm.
Indicator Value	6 characters (a sign, and 5 decimal digits with an implied decimal in the position X . XXXX) An estimate of how well the Aurora System calculated the transformation. Values range from 0 to 9.9. A higher value indicates a higher error. For 6DOF tools, the indicator value compares sensor coil measurements to the tool's design (as described by its SROM device or tool definition file). For 5DOF tools, the indicator value is always zero.

<table>
<tr><th>Reply Component</th><th colspan="3">Description</th></tr>
<tr><td><code>Port Status</code></td><td colspan="3">8 hexadecimal characters (32 bits)

Bit field:</td></tr>
<tr><td></td><td>bit 0</td><td>Occupied</td><td rowspan="4">Bits 0 to 3 are shared for dual, 5DOF tools.</td></tr>
<tr><td></td><td>bit 1</td><td>GPIO line 1 closed</td></tr>
<tr><td></td><td>bit 2</td><td>GPIO line 2 closed</td></tr>
<tr><td></td><td>bit 3</td><td>GPIO line 3 closed</td></tr>
<tr><td></td><td>bit 4</td><td colspan="2">Initialized</td></tr>
<tr><td></td><td>bit 5</td><td colspan="2">Enabled</td></tr>
<tr><td></td><td>bit 6</td><td colspan="2">Out of volume</td></tr>
<tr><td></td><td>bit 7</td><td colspan="2">Partially out of volume. This bit is set only for 6DOF tools, when one sensor coil is inside the measurement volume and the other sensor coil is outside the measurement volume.</td></tr>
<tr><td></td><td>bit 8</td><td colspan="2">A sensor coil is broken.
This bit is set when a sensor coil lead wire breaks, either along the lead wire length or at its connection points.</td></tr>
<tr><td></td><td>bits 9 to 11</td><td colspan="2">Reserved</td></tr>
<tr><td></td><td>bit 12</td><td colspan="2">Processing exception</td></tr>
<tr><td></td><td>bits 13 to 31</td><td colspan="2">Reserved</td></tr>
<tr><td><code>Frame Number</code></td><td colspan="3">8 hexadecimal characters

The frame number is an internal counter related to data acquisition. The counter starts at power up and does not reset until the system is reset, the system is powered up again, or reply option 80 is sent with the TSTART command. The frame number corresponds to the frame in which the raw data, used to calculate the accompanying transformation, was collected.</td></tr>
</table>

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

Reply Option 0800 - Reporting All Transformations

This option enables the reporting of transformations or translations when the tool is out of volume. This reply option must be OR'd with reply option 0001.

When using reply option 0800 with the TX command, you must take appropriate action to detect when a tool is out of volume, and determine whether this situation is detrimental to your application. If a tool is out of volume, reply option 0800 enables the system to return data that may lead to inaccurate conclusions and may cause personal injury.

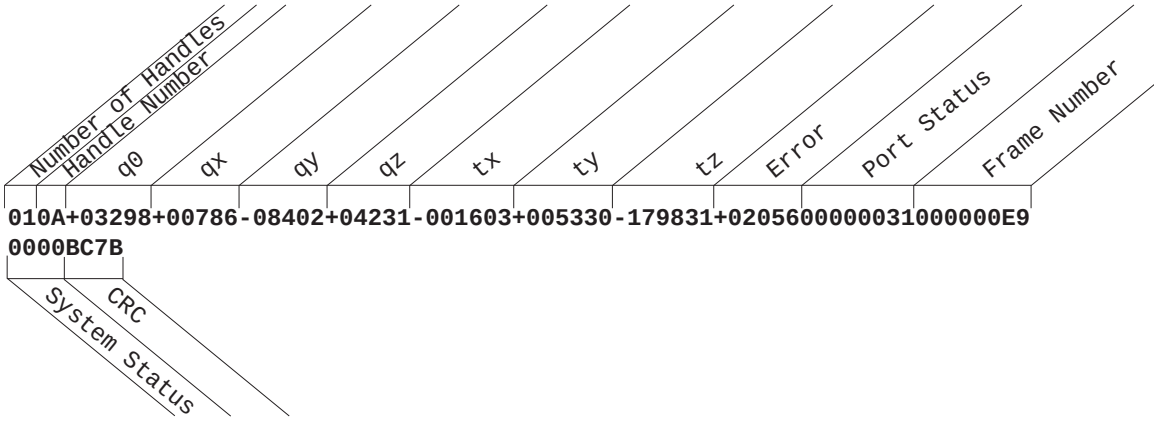
Usage Notes

1. 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 BX (page 12).
2. See the user guide for detailed descriptions of the port status and system status bits.
3. By default, transformations will not be reported if the tool is either partially or wholly out of the characterized measurement volume. To report these transformations, you must use reply option 0800 OR'd with reply option 0001. The accuracy of these transformations is unknown.
4. Bit 8 in the <Port Status> section of reply option 0001 may not always indicate when a lead wire is broken. Broken sensor coils that result in a short will not be detected.

Example

Command:
TX

Reply:



The system returned transformation data for one tool.

VER

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

Operating Mode

Setup

Syntax

`VER<SPACE><Reply Option><CR>`

Parameter	Description	
Reply Option	Specifies which information will be returned. The reply options cannot be OR'd. **Valid Values:**	
	0	System Control Processor
	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.
	7	Field Generator information
	8	Sensor Interface Unit information

Replies

Upon Success:

`<Reply Option Data><CRC16><CR>`

On Error:

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

See page 70 for error code definitions.

Reply Option 0 - System Control Processor

```
<Reply Option 0 Data> =
<Type of Firmware><LF>
<NDI Serial Number><LF>
<Characterization Date><LF>
<Freeze Tag><LF>
<Freeze Date><LF>
<Copyright Information><LF>
```

Reply Option 4 - System Control Processor with Enhanced Revision Numbering

```
<Reply Option 4 Data> =
<Type of Firmware><LF>
<NDI Serial Number><LF>
<Characterization Date><LF>
<Freeze Tag><LF>
<Freeze Date><LF>
<Copyright Information><LF>
```

Reply Option 5 - Combined Firmware Revision Number

```
<Reply Option 5 Data> = <Combined Firmware Revision (three characters)>
```

Reply Option 7 - Field Generator Information

```
<Reply Option 7 Data> =
<Field Generator Serial Number><LF>
<Field Generator Model><LF>
<Characterization Date><LF>
```

Reply Option 8 - Sensor Interface Unit Information

```
<Reply Option 8 Data> =
<Port 1>
<Type of Firmware><LF>
<Freeze Tag><LF>
<Freeze Date><LF>
...
<Port 4>
<Type of Firmware><LF>
<Freeze Tag><LF>
<Freeze Date><LF>
```

Note — If no SIU is connected to the port, the system will report “Version Not Available” instead of the <Type of Firmware>, <Freeze Tag>, and <Freeze Date> information.

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.

Examples

Command:

```
VER 5
```

Reply:

```
0061994
```

VSEL

Selects a characterized measurement volume.

Operating Mode

Setup

Prerequisite Command

INIT (page 22)

Syntax

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

Parameter	Description
`Volume Number`	1 hexadecimal character **Valid Values**: 1 to the maximum returned by SFLIST (page 54)

Replies

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

On Error:
`ERROR<Error Code><CRC16><CR>`
`See page 70 for error code definitions.`

Usage Notes

Use SFLIST (page 54) to determine which measurement volumes are available.

Example

Command:
`VSEL 1`

Reply:
`OKAYA896`

4 Error Code Definitions

If the system receives an invalid command, it responds to the host with the message `ERROR<Error Code><CRC>`. Table 4-1 identifies the error codes and their definitions.

Table 4-1 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 tool tracking priority is out of range, or the tool has sensor coils 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 operating 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	Unable to initialize the port handle.
14	Invalid Field Generator characterization parameters or incompatible hardware
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.
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.

Table 4-1 Error Code Definitions (Continued)

Error Code	Definition
19	Unable to read device's firmware revision information. This occurs if: • the processor selected is out of range • the system is unable to inquire firmware revision information from a processor
1A	Internal system error. This occurs when the system is unable to recover after a system processing exception.
1B - 1C	Reserved.
1D	Unable to search for SROM device IDs.
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 SROM device starting address is out of range • the system is unable to auto-select the first SROM device on the given port handle as a target for writing to • an SROM device on the given port handle has not previously been selected with the PSEL command as a target to write to • the system is unable to write a page of SROM device data successfully
20	Unable to select SROM device for given port handle and SROM device ID.
21	Reserved.
22	Enabled tools are not supported by selected volume parameters.
23	Command parameter is out of range.
24	Unable to select parameters by 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 - 28	Reserved.
29	Main processor firmware is corrupt.
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.

Table 4-1 Error Code Definitions (Continued)

Error Code	Definition
2F - 30	Reserved.
31	Invalid input or output state.
32	Invalid operation for the device associated with the specified port handle.
33	Feature not available.
34 - C1	Reserved.
C2	Command not supported by proxy.
C3	Offline fit not possible, eg no or incompatible DLL.
C4	Incompatible replies from two SCUs, eg different firmware revisions.
C5-F3	Reserved.
F4	Unable to erase Flash SROM device.
F5	Unable to write Flash SROM device.
F6	Unable to read Flash SROM device.
F7 - FF	Reserved.

Appendix A For Polaris Users

The Aurora System shares a large part of its API with the NDI Polaris System. If you have written an application for use with the Polaris System, you can adapt it to function with the Aurora System. The following sections list the differences between the Aurora API and various versions of the Polaris API.

Read the section that corresponds to the type of Polaris System and version of Polaris firmware for which your application is designed:

- “Differences Between the Aurora System and the Polaris System (Rev 18)” on page 74
  Read this section if your application is written for use with a Polaris System with combined firmware revision 018.
- “Differences Between the Aurora System and the Polaris System (Rev 24)” on page 76
  Read this section if your application is written for use with a Polaris System with combined firmware revision 022 to 024.
- “Differences Between the Aurora System and the Polaris Vicra or Polaris Spectra System” on page 79
  Read this section if your application is written for use with a Polaris Vicra System or a Polaris Spectra System.

To determine the combined firmware revision of your Polaris System, use the command VER 5. If your application is written for a Polaris System that uses a combined firmware revision other than Rev 22-24 or Rev 18, contact NDI technical support for more information.

A.1 Differences Between the Aurora System and the Polaris System (Rev 18)

Read this section if your application is written for use with a Polaris System with combined firmware revision 018.

Deprecated Commands

- GX, replaced by BX (page 12) and TX (page 63)
- PSTAT, replaced by PHSR (page 33) and PHINF (page 28)
- PVCLR, replaced by PHF (page 27)
- PVTIP, no replacement for the Aurora System

New Commands

- APIREV (page 10) returns the API revision number.
- ECHO (page 20) returns exactly what is sent with the command.

Changed Command

- **PHINF (page 28):** In reply option 0x0001, the size of the <tool type> field increases from 7 to 8 characters.

Polaris-Only Commands

These commands function only with the Polaris System. Sending these commands to the Aurora System will generate an error:

3D	IRINIT
IRATE	PFSEL
IRCHK	PHRQ
IRED	TCTST

Aurora-Only Command

PSOUT (page 43) functions only with the Aurora System.

Commands with Differences Between Polaris and Aurora

These commands function differently with the Aurora System than with the Polaris System:

SFLIST (page 54)

- **Polaris-only options:** The following reply options are valid only for the Polaris System:
  reply option 0x01 (number of wired tool ports)
  reply option 0x02 (number of wireless tool ports)
  reply option 0x04 (number of wired tool ports available which support tool-in-port detection from current sensing)

- **Aurora-only options:** The following reply option is valid only with the Aurora System:
  reply option 0x10 (number of ports)

SSTAT

- This command will work with the Aurora System, but won't return any useful data. This command will return the following data:

  SSTAT 0001 returns 00
  SSTAT 0002 returns 00
  SSTAT 0004 returns 00
  SSTAT 0100 returns 00000000
  SSTAT 0200 returns 00000000

VER (page 67)

- **Polaris-only options:** The following reply options are valid only with the Polaris System:
  reply option 1 (left sensor processor)
  reply option 2 (right sensor processor)
  reply option 3 (Tool Interface Unit processor)
  reply option 6 (Tool Docking Station)
- **Aurora-only options:** The following reply options are valid only with the Aurora System:
  reply option 7 (Field Generator information)
  reply option 8 (Sensor Interface Unit information)

Differences in Concepts

Port handles: Previous versions of the Polaris API distinguished between wired tool ports (corresponding to wired tools that were physically connected to the system) and wireless tool ports (corresponding to passive tools and active wireless tools). Wired tool ports were assigned a port number, and wireless tool ports were assigned a port letter.

The API no longer supports the concept of tool ports, and instead assigns each tool a port handle. Port handles are independent of the physical port numbering. Port handles provide greater flexibility than port numbers and replace port numbers and port letters in most commands. Port handles range from 0x0A to 0xFF with the Aurora System. (With the Polaris System, port handles range from 0x01 to 0xFF.)

For a split port on a dual 5DOF tool, the first PHSR (page 33) sent will report only one port handle. After the port handle is initialized, it is assigned to channel 0. You must then use PHSR again to assign a port handle to channel 1, and then initialize the port handle using INIT (page 22). See "Flow Chart for Port Handle Usage" on page 7.

GPIO: General purpose input/outputs can be used as either an input or an output with the Aurora System. With the Polaris System, GPIOs can only be used as inputs.

API revisions: The API now has a revision number. If you upgrade the firmware, you can check the API revision to determine whether your application software may need to be updated. In the event of a firmware change that does not affect the API, the API revision number will remain the same, and any application software written using that API revision will continue to function with the new firmware revision. To determine the API revision programmed into your system, use the command APIREV (page 10).

A.2 Differences Between the Aurora System and the Polaris System (Rev 24)

Read this section if your application is written for use with a Polaris System with combined firmware revision 022 to 024.

New Commands

- APIREV (page 10) returns the API revision number.
- ECHO (page 20) returns exactly what is sent with the command.

Polaris-Only Commands

These commands function only with the Polaris System. Sending these commands to the Aurora System will generate an error:

3D	IRINIT
GETIO	PFSEL
IRATE	PHRQ
IRCHK	SETIO
IRED	TCTST

Aurora-Only Command

PSOUT (page 43) functions only with the Aurora System.

Commands with Differences Between Polaris and Aurora

These commands function differently with the Aurora System than with the Polaris System:

BX (page 12) and TX (page 63)

- **Polaris-only options:** The following reply options are valid only with the Polaris System:
  reply option 0x0002 (tool and marker information)
  reply option 0x0004 (3D position of a single stray active marker)
  reply option 0x1000 (3D positions of stray passive markers)
- **Polaris-only bit:** The following bit is valid only with the Polaris System:
  <system status> bit 1 (too much external IR)
- **Aurora-only bits:** The following bits are valid only with the Aurora System:
  <system status> bit 4 (hardware failure)
  <system status> bit 5 (hardware change)
  <port status> bit 8 (broken sensor coil)
  <port status> bit 10 (signal too small)
  <port status> bit 11 (signal too big)
  <port status> bit 13 (hardware failure)

PHINF (page 28)

- **Polaris-only options:** The following reply options are valid only with the Polaris System:
  reply option 0x0002 (wired tool electrical information)
  reply option 0x0010 (tool marker type and wavelength)
- **Aurora-only option:** The following reply option is valid only with the Aurora System:
  reply option 0x0040 (GPIO status)

PHSR (page 33)

- **Polaris-only bit:** The following bit is valid only with the Polaris System:
  <port handle status> bit 7 (tool detected from current sensing)

SFLIST (page 54)

- **Polaris-only options:** The following reply options are valid only for the Polaris System:
  reply option 0x01 (number of wired tool ports)
  reply option 0x02 (number of wireless tool ports)
  reply option 0x04 (number of wired tool ports available which support tool-in-port detection from current sensing)
- **Aurora-only options:** The following reply option is valid only with the Aurora System:
  reply option 0x10 (number of ports)

SSTAT

- This command will work with the Aurora System, but won't return any useful data. This command will return the following data:

  SSTAT 0001 returns 00
  SSTAT 0002 returns 00
  SSTAT 0004 returns 00
  SSTAT 0100 returns 00000000
  SSTAT 0200 returns 00000000

VER (page 67)

- **Polaris-only options:** The following reply options are valid only with the Polaris System:
  reply option 1 (left sensor processor)
  reply option 2 (right sensor processor)
  reply option 3 (Tool Interface Unit processor)
  reply option 6 (Tool Docking Station)
- **Aurora-only options:** The following reply options are valid only with the Aurora System:
  reply option 7 (Field Generator information)
  reply option 8 (Sensor Interface Unit information)

Differences in Concepts

Port handles: Port handles range from 0x0A to 0xFF with the Aurora System. With the Polaris System, port handles range from 0x01 to 0xFF.

For a split port on a dual 5DOF tool, the first PHSR (page 33) sent will report only one port handle. After the port handle is initialized, it is assigned to channel 0. You must then use PHSR again to assign a port handle to channel 1, and then initialize the port handle using INIT (page 22). See “Flow Chart for Port Handle Usage” on page 7.

GPIO: General purpose input/outputs can be used as either an input or an output with the Aurora System. With the Polaris System, GPIOs can only be used as inputs.

API revisions: The API now has a revision number. If you upgrade the firmware, you can check the API revision to determine whether your application software may need to be updated. In the event of a firmware change that does not affect the API, the API revision number will remain the same, and any application software written using that API revision will continue to function with the new firmware revision. To determine the API revision programmed into your system, use the command APIREV (page 10).

A.3 Differences Between the Aurora System and the Polaris Vicra or Polaris Spectra System

Read this section if your application is written for use with a Polaris Vicra System or a Polaris Spectra System.

Polaris-Only Commands

These commands function only with the Polaris Vicra and Polaris Spectra Systems. Sending these commands to the Aurora System will generate an error:

3D	PFSEL
DFLT	PHRQ
GET	RESET
GETINFO	SAVE
GETIO	SET
GETLOG	SETIO
IRATE	SYSLOG
IRCHK	TCTST
IRED	VGET
IRINIT	VSNAP

Commands with Differences Between Polaris and Aurora

These commands function differently with the Aurora System than with the Polaris Vicra and Spectra Systems:

BX (page 12) and TX (page 63)

- **Polaris-only options:** The following reply options are valid only with the Polaris Vicra and Spectra Systems:
  reply option 0x0002 (tool and marker information)
  reply option 0x0004 (3D position of a single stray active marker)
  reply option 0x0008 (3D positions of markers on tools)
  reply option 0x1000 (3D positions of stray passive markers)
- **Polaris-only bits:** The following bits are valid only with the Polaris Vicra and Spectra Systems:
  <system status> bit 8 (diagnostic pending)
  <system status> bit 9 (temperature)
  <port status> bit 8 (algorithm limitation)
  <port status> bit 9 (IR interference)
  <port status> bit 12 (processing exception)
  <port status> bit 14 (fell behind while processing)
  <port status> bit 15 (data buffer limitation)
- **Aurora-only bits:** In the <system status> field, the following bits are valid only with the Aurora System:
  <system status> bit 4 (hardware failure)
  <system status> bit 5 (hardware change)

<port status> bit 8 (broken sensor coil)
<port status> bit 10 (signal too small)
<port status> bit 11 (signal too big)
<port status> bit 13 (hardware failure)

PHINF (page 28)

- **Polaris-only options:** The following reply options are valid only with the Polaris System:
  reply option 0x0002 (wired tool electrical information)
  reply option 0x0010 (tool marker type and wavelength)
- **Aurora-only option:** The following reply option is valid only with the Aurora System:
  reply option 0x0040 (GPIO status)

PHSR (page 33)

- **Polaris-only bit:** The following bit is valid only with the Polaris System:
  <port handle status> bit 7 (tool detected from current sensing)

PSOUT (page 43)

- **Aurora-only option:** The GPIO line state “P” (pulse) is only with the Aurora System.

SFLIST (page 54)

- **Polaris-only options:** The following reply options are valid only for the Polaris System:
  reply option 0x01 (number of wired tool ports)
  reply option 0x02 (number of wireless tool ports)
  reply option 0x04 (number of wired tool ports available which support tool-in-port detection from current sensing)
- **Aurora-only options:** The following reply option is valid only with the Aurora System:
  reply option 0x10 (number of ports)

SSTAT

- This command will work with the Aurora System, but won’t return any useful data. This command will return the following data:

  SSTAT 0001 returns 00
  SSTAT 0002 returns 00
  SSTAT 0004 returns 00
  SSTAT 0100 returns 00000000
  SSTAT 0200 returns 00000000

VER (page 67)

- **Polaris-only options:** The following reply options are valid only with the Polaris System:
  reply option 1 (left sensor processor)
  reply option 2 (right sensor processor)

reply option 3 (Tool Interface Unit processor)
reply option 6 (Tool Docking Station)

- **Aurora-only options:** The following reply options are valid only with the Aurora System:
  reply option 7 (Field Generator information)
  reply option 8 (Sensor Interface Unit information)

Differences in Concepts

User parameters The Polaris Vicra and Polaris Spectra Systems use user parameters to store values for different aspects of the system such as hardware details, system features, and system status. User parameters are not supported by the Aurora System.

Port handles Port handles range from 0x0A to 0xFF with the Aurora System. With the Polaris System, port handles range from 0x01 to 0xFF.

For a split port on a dual 5DOF tool, the first PHSR (page 33) sent will report only one port handle. After the port handle is initialized, it is assigned to channel 0. You must then use PHSR again to assign a port handle to channel 1, and then initialize the port handle using INIT (page 22). See “Flow Chart for Port Handle Usage” on page 7.

GPIO General purpose input/outputs can be used as either an input or an output with the Aurora System. With the Polaris System, GPIOs can only be used as inputs.

Appendix B Sample C Routines

The following sample C routines are included for reference:

Table 4-2 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 Aurora 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 */
```

Abbreviations and Acronyms

Abbreviation or Acronym	Definition
API	Application Program Interface
CRC	Cyclic Redundancy Check
FG	Field Generator
GPIO	General Purpose Input/Output
IEEE	Institute of Electrical and Electronic Engineers
LED	Light Emitting Diode
PFG	Planar Field Generator
RMS	Root Mean Square
SROM	Serial Read Only Memory
TTFG	Tabletop Field Generator

Index

Numerics

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W