

TA620 STANDALONE SPINDLE CONTROLLER



Software Users MANUAL BETA v1.01

Copyright Information

© 2006 Trust Automation, Inc. All rights reserved.

This document is provided for Trust Automation, Inc. customers, solely for the purpose of assisting our customers in the use and installation of our products. Other uses are unauthorized without the written permission of Trust Automation, Inc. The text and graphics included are for purpose of illustration only and information is subject to change without notice. Trust Automation, Inc. and the Trust Automation, Inc. logo are trademarks of Trust Automation, Inc, a California corporation.

For information regarding re-use of this material or to report errors, omissions, inconsistencies, etc, please contact Technical Support at:

Trust Automation, Inc.

205 Suburban Road

San Luis Obispo, CA. 93401

E-mail Technical Support: support@trustautomation.com

Web: www.trustautomation.com

Phone: (805) 544-0761

FAX: (805) 544-4621

Handling and Safety Information

Trust Automation products contain static sensitive parts that may be damaged if handled improperly. We strongly encourage you to follow proper ESD procedures when handling electronic components. Removing component covers, except where expressly permitted, may expose products to static damage and increase the risk of premature failure.

Some Trust Automation products generate high voltages. Maintenance or repair should only be performed by qualified personnel and only under power down conditions. Maintenance and repair shall be limited to those items described in this operating manual as user approved. All other repair and maintenance shall be performed by Trust Automation, Inc.

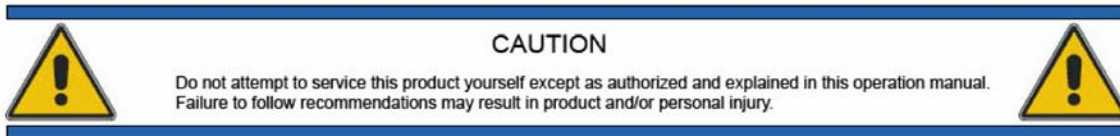


Table of Contents

| | |
|--|----|
| Table of Contents | 3 |
| 1.0 Software Command Structure | 4 |
| 2.0 Command Overview | 8 |
| 2.1 Controller Level Commands | 8 |
| 2.2 Spindle Set/Get Commands | 8 |
| 2.3 Spindle Move Commands | 9 |
| 2.4 Axis Level Commands | 9 |
| 3.0 Spindle Controller Set/Get Commands | 11 |
| 3.1 CSL - Set Hardware Interlocks | 11 |
| 3.2 CGL - Get Hardware Interlocks | 12 |
| 3.3 CSC - Set Chuck | 13 |
| 3.4 CGC - Get Chuck | 14 |
| 3.6 CSD - Set Spindle Direction | 15 |
| 3.7 CGD - Get Spindle Direction | 16 |
| 3.8 CSO - Set Pulse Out | 17 |
| 3.9 CGO - Get Pulse Out | 18 |
| 3.10 CSI - Set External Speed Input | 19 |
| 3.11 CGI - Get External Speed Input | 20 |
| 3.12 CSE - Set Encoder Out Source | 21 |
| 3.13 CGE - Get Encoder Out Source | 22 |
| 3.14 CGP - Get Drive Data | 23 |
| 4.0 Spindle Controller Move Commands | 24 |
| 4.1 CMV - Perform a Trapezoidal Velocity Move | 24 |
| 4.2 CMS - Perform an S-Curve Velocity Move | 25 |
| 4.3 ASX - Stop Accelerating and Maintain Current Speed | 26 |
| 4.4 ASC - Change Commanded Acceleration | 27 |
| 4.5 CME - Perform an Externally Controlled Velocity Move | 28 |
| 5.0 Manual Revision History | 29 |

1.0 Software Command Structure

The software command structure uses a condensed format language designed for fast serial or USB execution.

Command Structure:

Each command is initiated with a three letter identifier, and all command fields are separated by commas. (See individual command descriptions for necessary parameters) All commands and their responses are terminated with carriage returns **<CR>**.

XXX,field1,field2,field3, ... ,fieldN<CR>

XXX,[axis],field1,field2,field3, ... ,fieldN<CR>

Response Structure:

All commands return an immediate response composed of an underscore “_” followed by the three letter command. All return message fields are comma delimited. If the command requires an axis as the first parameter, then the response will include the axis as the first parameter returned.

_XXX,field1,field2,field3, ... ,fieldN<CR>

_XXX,[axis],field1,field2,field3, ... ,fieldN<CR>

Error Structure:

Error responses start with the standard command response followed by ERR, the error code, and the error message. Error code descriptions are located in Table 5.

_XXX,<[axis]>,ERR,[error code],[error description]<CR>

Examples of Error Return Responses:

In this example the home mode parameter value (8) is out of range. Notice that the axis parameter 0 is included in the error response.

```
sent      SHM,0,8<CR>
received  _SHM,0,ERR,00003,Invalid parameter value<CR>
```

In this example the axis parameter value (10) is out of range:

```
sent      GHM,10<CR>
received  _GHM,10,ERR,00029,Axis out of range <CR>
```

In this example the axis parameter value (-2) is negative. Notice that although the command requires an axis parameter, the controller will not return a negative axis parameter in the error response. This is true for all commands that take an axis parameter.

```
sent      GHM,-2<CR>
received  _GHM,ERR,00029,Axis out of range <CR>
```

Command Groups:

The first character specifies the command group and the following two characters identify a command within that group. Below are descriptions of the command groups:

GXX, Get Data Commands

This class of command retrieves parameter data from the controller. Get commands return values such as tuning parameters and controller status.

Example of a get command that requires an axis parameter:

Get Actual Position - used to read the feedback position from the controller

sent **GAP,2<CR>**
received **_GAP,2,23546<CR>**

Example of a get command that does not require an axis parameter:

Get Response Mode - used to determine if the controller will send asynchronous responses

sent **GRM<CR>**
received **_GRM,SYNC<CR>**

SXX, Set Data Commands

This class of command sets parameter data on the controller. These commands set values such as tuning parameters and IO active levels. Unless otherwise specified in the command description, all set commands set both parameter values in memory and EEPROM.

Example of a set command that requires an axis parameter:

Set Actual Position - used to set the feedback position register

sent **SAP,2,23546<CR>**
received **_SAP,2<CR>**

Example of a set command that does not require an axis parameter:

Set Response Mode - used to enable or disable asynchronous responses

sent **SRM,SYNC<CR>**
received **_SRM <CR>**

IXX, Immediate Action Commands (I/O Changes)

This class of command performs actions that conclude before the standard response is issued. These actions include setting digital outputs, enabling drives, and setting the servo loop.

Example of enabling the servo loop:

sent **ILP,0,ON<CR>**
Controller enables the servo loop
received **_ILP,0<CR>**

Example of enabling the spindle drive:

sent **IDR,0,ON<CR>**
Controller enables the spindle drive
received **_IDR,0<CR>**

AXX, Long Term Action Commands (Motion)

This class of command starts motion and other long term commands that conclude sometime after the standard response is issued. Action commands can optionally send an additional response when they complete. The additional response takes the form of the standard response followed by a comma and COMPLETE. This feature is governed by the **SRM** (Set Response Mode) command. In ASYNC mode, the controller sends an additional response when an action completes. In SYNC mode, the controller sends only one response to each command it receives. In this mode, the user must poll the controller with the **GAS** (Get Axis Status) command to determine when a move completes.

Example of successful home move in ASYNC mode:

```
sent      AMH,0<CR>
received  _AMH,0<CR>
Controller performs and completes the home operation
received  _AMH,0,COMPLETE<CR>
```

Example of successful home move in SYNC mode:

```
sent      AMH,0<CR>
received  _AMH,0<CR>
Controller performs and completes the home operation
The controller updates the axis status, but does not send a response
```

If an error occurs before the movement completes, the COMPLETE message will not be sent. Instead, an error message will be sent.

Example of failed home move in ASYNC mode:

```
sent      AMH,0<CR>
received  _AMH,0<CR>
Controller performs but fails to complete the home operation
received  _AMH,0,ERR,00024,Following Error<CR>
```

Example of failed home move in SYNC mode:

```
sent      AMH,0<CR>
received  _AMH,0<CR>
Controller performs but fails to complete the home operation
The controller notes the error and updates the axis status but sends no response
```

Response Mode:

The controller has two basic modes of communication. In SYNC mode, the controller sends one response to each command it receives. SYNC mode reduces the complexity of the host side interface because communication is always initiated by the host. The down side is that the host must poll the controller for status of axis motion and axis errors. In ASYNC mode, the controller adds responses that are not initiated by the host. ASYNC mode increases the complexity of the host side interface because it allows the controller to send responses as events occur. The two events that generate these additional responses are motion completions and errors.

Event monitoring in SYNC mode:

Error and motion status are not directly reported in SYNC mode. Use the **GAS** (Get Axis Status) command to follow motion progress and check for errors.

Action command responses in ASYNC mode:

For each action command issued in ASYNC mode, there is an immediate response and a final response. The final response is issued when the move completes either with or without error. A successful completion results in the COMPLETE message outlined below. A failed move results in the ERR message outlined below.

_AXX,[axis],COMPLETE<CR>

_AXX,[axis],ERR,[error code],[error description]<CR>

Spindle action command responses in ASYNC mode:

Spindle action commands begin with CM and include **CMH**, **CMV**, **CMS**, and **CME**. For each spindle action command issued in ASYNC mode, there is an immediate response and a final response. The final response is issued when the move completes either with or without error. A successful completion results in the COMPLETE message outlined below. A failed move results in the ERR message outlined below.

_CMX,COMPLETE<CR>

_CMX,ERR,[error code],[error description]<CR>

Error responses in ASYNC mode:

Some errors occur independent of action commands. These errors are returned in the form of an asynchronous error as outlined below. The most common asynchronous errors include drive faults, supply faults, and following errors.

_ASY,ERR,[error code],[error description]<CR>

General note on position related parameter ranges:

All range values for position and position related values assume an encoder ratio of 1. If the encoder ratio is set to a value other than 1, that ratio must be applied to the range values.

2.0 Command Overview

2.1 Controller Level Commands

| Description | Command |
|----------------------------|--------------------------|
| HELLO | HLO,[STRING W/O SPACES] |
| RESET | RST |
| HALT | HLT |
| SET COMMUNICATIONS OPTIONS | SCM,[BAUD] |
| GET COMMUNICATIONS OPTIONS | GCM |
| GET REVISION | GRV |
| SET OPERATION MODE | SMD,[PERAXIS,ONECOMMAND] |
| GET OPERATION MODE | GMD |
| SET RESPONSE MODE | SRM,[SYNC/ASYNC] |
| GET RESPONSE MODE | GRM |
| SET SERVO SAMPLE RATE | SSR,[SAMPLE RATE] |
| GET SERVO SAMPLE RATE | GSR |
| GET MAXIMUM SUPPORTED AXES | GAM |
| SET AN AXIS ACTIVE | SAA,[AXIS],[STATE] |
| GET ACTIVE AXES | GAA,[AXIS] |
| SET AN EEPROM DATA FIELD | SEE,[INDEX],[DATA] |
| GET THE EEPROM DATA FIELD | GEE,[INDEX] |
| GET ERRORS | GER,[0/1/2/3/C/ALL] |
| CLEAR ERRORS | SER,[0/1/2/3/C/ALL] |
| GET SYSTEM STATUS | GST |

2.2 Spindle Set/Get Commands

| Description | Command |
|--|----------------|
| Set Hardware Interlocks | CSL,[mask] |
| Get Hardware Interlocks | CGL |
| Set Chuck to Clamp or Open | CSC,[state] |
| Get Chuck Clamp State | CGC |
| Get Spindle Pressure State | CGP |
| Set Spindle Rotation Direction | CSD,[rotation] |
| Get Spindle Rotation Direction | CGD |
| Set Pulse Out to Specific Encoder Signal | CSO,[source] |
| Get Pulse Out Source | CGO |
| Set External Speed Input Source | CSI,[source] |
| Get External Speed Output Source | CGI |
| Set Encoder Out Source | CSE,[source] |
| Get Encoder Out Source | CGE |

2.3 Spindle Move Commands

| Description | Command |
|--|-------------------|
| Move Spindle to RPM using a trapezoidal trajectory | CMV ,[RPM] |
| Move Spindle to RPM using an s-curve trajectory | CMS ,[RPM] |
| Switch spindle into external velocity mode | CME |

2.4 Axis Level Commands

| Description | Command |
|--------------------------------------|--|
| SET MOTORTYPE | SMT ,[AXIS],[TYPE],[COMMUTATION] |
| GET MOTORTYPE | GMT ,[AXIS] |
| SET FEEDBACK TYPE | SFT ,[AXIS],[TYPE] |
| GET FEEDBACK TYPE | GFT ,[AXIS] |
| GET AXIS STATUS | GAS ,[AXIS] |
| SET SINCOM PARAMETERS | SSC ,[AXIS],...[ADDITIONAL PARAMS] |
| GET SINCOM PARAMETERS | GSC ,[AXIS] |
| | |
| SET MOTOR TO ENCODER RATIO | SRT ,[AXIS],[RATIO] |
| GET MOTOR TO ENCODER RATIO | GRT ,[AXIS] |
| SET BRAKE DELAY | SBD ,[AXIS],[MILISECONDS] |
| GET BRAKE DELAY | GBD ,[AXIS] |
| | |
| SET HOMING MODE | SHM ,[AXIS],[MODE] |
| GET HOME MODE | GHM ,[AXIS] |
| SET HOME OFFSET DISTANCE | SHO ,[AXIS],[OFFSET] |
| GET HOME OFFSET DISTANCE | GHO ,[AXIS] |
| SET HOME LIMIT LEVEL | SHL ,[AXIS],[LEVEL] |
| GET HOME LIMIT LEVEL | GHL ,[AXIS] |
| | |
| SET DRIVE FAULT LEVEL | SFL ,[AXIS],[LEVEL] |
| GET DRIVE FAULT LEVEL | GFL ,[AXIS] |
| SET DRIVE ENABLE LEVEL | SEL ,[AXIS],[LEVEL] |
| GET DRIVE ENABLE LEVEL | GEL ,[AXIS] |
| SET ENCODER LEVEL | SLE ,[AXIS],[LEVEL] |
| GET ENCODER LEVEL | GLE ,[AXIS] |
| | |
| SET HALL A LEVEL | SHA ,[AXIS],[LEVEL] |
| GET HALL A LEVEL | GHA ,[AXIS] |
| SET HALL B LEVEL | SHB ,[AXIS],[LEVEL] |
| GET HALL B LEVEL | GHB ,[AXIS] |
| SET HALL C LEVEL | SHC ,[AXIS],[LEVEL] |
| GET HALL C LEVEL | GHC ,[AXIS] |
| | |
| SET POSITIVE SOFTWARE LIMIT POSITION | SPL ,[AXIS],[POSITION] |
| GET POSITIVE SOFTWARE LIMIT POSITION | GPL ,[AXIS] |
| SET NEGATIVE SOFTWARE LIMIT POSITION | SNL ,[AXIS],[POSITION] |
| GET NEGATIVE SOFTWARE LIMIT POSITION | GNL ,[AXIS] |
| SET HARD LIMIT PARAMETERS | SHP ,[AXIS],[ACTION],[POSLEVEL],[NEG LEVEL] |
| GET HARD LIMIT PARAMETERS | GHP ,[AXIS] |
| | |
| SET STATIC FILTER PARAMETERS | SSF ,[AXIS],[KP],... [FE] |
| GET STATIC FILTER PARAMETERS | GSF ,[AXIS] |
| SET MOTION FILTER PARAMETERS | SMF ,[AXIS],[KP],... [FE] |
| GET MOTION FILTER PARAMETERS | GMF ,[AXIS] |
| SET HOME FILTER PARAMETERS | SHF ,[AXIS],[KP],... [FE] |
| GET HOME FILTER PARAMETERS | GHF ,[AXIS] |

Axis level commands continued:

| Description | Command |
|----------------------------------|---|
| SET BIQUAD FILTER COEFFICIENTS | SBC ,[AXIS],[FILTER],[B0],...[K] |
| GET BIQUAD FILTER COEFFICIENTS | GBC ,[AXIS],[FILTER] |
| SET BIQUAD FILTER ENABLE | SBE ,[AXIS],[FILTER],[ON/OFF] |
| GET BIQUAD FILTER ENABLE | GBE ,[AXIS],[FILTER] |
| | |
| SET AXIS POSITION | SAP ,[AXIS],[POSITION] |
| SET PROGRAMMED VELOCITY | SVL ,[AXIS],[TYPE],[VEL] |
| GET PROGRAMMED VELOCITY | GVL ,[AXIS],[TYPE] |
| | |
| SET PROGRAMMED ACCELERATION | SAC ,[AXIS],[TYPE],[ACC] |
| GET PROGRAMMED ACCELERATION | GAC ,[AXIS],[TYPE] |
| SET PROGRAMMED JERK | SJK ,[AXIS],[TYPE],[JRK] |
| GET PROGRAMMED JERK | GJK ,[AXIS],[TYPE] |
| | |
| SET GEARING AXES | SGR ,[SLAVE],[MASTER],[RATIO],[TYPE] |
| GET GEARING AXES | GGR ,[SLAVE] |
| SET PHASE OFFSET | SPH ,[AXIS],[OFFSET] |
| GET PHASE OFFSET | GPH ,[AXIS] |
| SET PHASE ANGLE | SPA ,[AXIS],[ANGLE] |
| GET PHASE ANGLE | GPA ,[AXIS] |
| | |
| SET HARD STOP PARAMETERS | SHS ,[AXIS],[PERIOD],[THRESHOLD] |
| GET HARD STOP PARAMETERS | GHS ,[AXIS] |
| SET MOTION TIMEOUT | STM ,[AXIS],[PERIOD] |
| GET MOTION TIMEOUT | GTM ,[AXIS] |
| | |
| Set External Profile Parameters | SXM ,[axis],[profile],[phase],[pos],[vel],[acc],[jrk],[time] |
| Get External Profile Parameters | GXM ,[axis],[profile],[phase] |
| Set External Profile Parameters | SXP ,[axis],[profile],[param],[mode] |
| Get External Profile Parameters | GXP ,[axis],[profile],[phase] |
| Set External Profile Phase Count | SXF ,[axis],[profile] |
| Get External Profile Phase Count | GXF ,[axis],[profile] |
| Set External Profile Count | SXC ,[axis],[count] |
| Get External Profile Count | GXC ,[axis],[count] |

3.0 Spindle Controller Set/Get Commands

3.1 CSL - Set Hardware Interlocks

Purpose:

This command returns a mask that determines which interlocks are used. A “1” in the appropriate bit indicates that the interlock is used. If an interlock is used, then its corresponding input must be active before sending spindle move commands. If the interlock is not met, then an error will be returned upon attempting any spindle move commands other than CMV,0 and CMS,0.

Format:

CSL,[interlock number]

Fields:

- [interlock number]*
- 0x1 identifies the Main Air Interlock (pin 2 of User I/O).
 - 0x2 identifies the Bearing Air Interlock (pin 3 of User I/O).
 - 0x4 identifies the Clamp Air Interlock (pin 4 of User I/O).
 - 0x8 identifies the Aux Interlock (pin 5 of User I/O).

Returns

_CSL

Examples

This will enable the Main Air and Bearing Air interlocks.

```
sent      CSL,0x3
received  _CSL
```

3.2 CGL - Get Hardware Interlocks

Purpose:

This command returns a mask that determines which interlocks are used. A “1” in the appropriate bit indicates that the interlock is used. If an interlock is used, then its corresponding input must be active before sending spindle move commands. If the interlock is not met, then an error will be returned upon attempting any spindle move commands other than CMV,0 and CMS,0.

Format:

CGL

Fields:

- [interlock mask]*
- 0x1 identifies the Main Air Interlock (pin 2 of User I/O).
 - 0x2 identifies the Bearing Air Interlock (pin 3 of User I/O).
 - 0x4 identifies the Clamp Air Interlock (pin 4 of User I/O).
 - 0x8 identifies the Aux Interlock (pin 5 of User I/O).

Returns

CGL,[interlock mask]

Examples

This call to CGL indicates that the Main Air and Bearing Air interlocks are active.

```
sent      CGL
received  _CGL,0x03
```

3.3 CSC - Set Chuck

Purpose:

This command enables and disables the I/O line that controls the chuck.

Format:

CSC,[action]

Fields:

- [action]*
- OPEN. This option opens the chuck.
 - CLAMP. This option clamps the chuck.

Returns

_CSC

Examples

This will set the chuck I/O line to close the chuck.

sent **CSC,CLAMP**
received **_CSC**

3.4 CGC - Get Chuck

Purpose:

This command returns the state of the I/O line that controls the chuck.

Format:

CGC

Fields:

- [state]*
- OPEN. This option opens the chuck.
 - CLAMP. This option clamps the chuck.

Returns

_CGC,[state]

Examples

This indicates that the state of the chuck I/O line is in the state to clamp the chuck.

sent **CGC**
received **_CGC,CLAMP**

3.6 CSD - Set Spindle Direction

Purpose:

This command sets the direction of rotation for the spindle.

Format:

CSD,[direction]

Fields:

[direction] – CW. This option sets the rotation to clockwise.
 – CCW. This option sets the rotation to counter clockwise.

Returns

_CSD

Examples

This will set the direction of spindle rotation to clockwise.

sent **CSD,CW**
received **_CSD**

3.7 CGD - Get Spindle Direction

Purpose:

This command returns the direction of rotation for the spindle.

Format:

CGD

Fields:

- [direction]*
- CW. This option sets the rotation to clockwise.
 - CCW. This option sets the rotation to counter clockwise.

Returns

_CGD,[direction]

Examples

This indicates that spindle rotation is clockwise.

| | |
|-----------------|----------------|
| <i>sent</i> | CGD |
| <i>received</i> | _CGD,CW |

3.8 CSO - Set Pulse Out

Purpose:

This command sets encoder signal that is routed to the Encoder Output BNC connector. Use A or B to select a sector pulse. Use Z to select an Index pulse. If both Sector and Index signals are needed, both are also available from the AUX ENCODER connector when CSE (Set Encoder Out Source) is set to ENC.

Format:

CSO,[type]

Fields:

- [type]*
- A. This option sets the pulse out signal to encoder A.
 - B. This option sets the pulse out signal to encoder B.
 - Z. This option sets the pulse out signal to encoder Index.

Returns

_CSO

Examples

This will set the pulse out signal to the encoder index.

sent **CSO,Z**
received **_CSO**

3.9 CGO - Get Pulse Out

Purpose:

This command returns the encoder signal that is routed to the pulse out BNC connector. Options A and B indicate a sector pulse. Option Z indicates an Index pulse.

Format:

CGO

Fields:

- [type]*
- A. This option sets the pulse out signal to encoder A.
 - B. This option sets the pulse out signal to encoder B.
 - Z. This option sets the pulse out signal to encoder Index.

Returns

_CGO,[type]

Examples

This indicates that the pulse out signal is set to encoder index.

sent **CGO**
received **_CGO,Z**

3.10 CSI - Set External Speed Input

Purpose:

This command switches the external speed input between Pulse In and Aux Encoder. Both options allow for external control of the spindle speed. In the case of the Pulse In, a frequency is used to control spindle speed. In the case of the Aux Encoder input, an encoder stream is used to control the spindle speed.

Format:

CSI,[type]

Fields:

[type]

- PULSE. This option uses the signal at the Frequency Input BNC connector to govern spindle speed.
- AUXENC. This option uses the Aux Encoder input to govern spindle speed.

Returns

_CSI

Examples

This will set the external speed input to Pulse In.

| | |
|-----------------|------------------|
| <i>sent</i> | CSI,PULSE |
| <i>received</i> | _CSI |

3.11 CGI - Get External Speed Input

Purpose:

This command returns the currently used external speed input.

Format:

CGI

Fields:

[type] – PULSE. This option uses the signal at the Frequency Input BNC connector to govern spindle speed.
 – AUXENC. This option uses the Aux Encoder input to govern spindle speed.

Returns

_CGI,[type]

Examples

This indicates that the external speed input is set to Pulse In.

sent **CGI,PULSE**
received **_CSI**

3.12 CSE - Set Encoder Out Source

Purpose:

This command switches the Aux Encoder Output signals present at the AUX ENCODER connector between those from the spindle ENCODER connector and Aux Encoder Input. Using the ENC option drives Sector and Index signals to the AUX ENCODER connector.

Format:

CSE,[type]

Fields:

[type]

- ENC. This option sets the encoder out to the spindle encoder.
- AUXENC. This option sets the encoder out to the aux encoder.

Returns

_CSE

Examples

This will set the encoder out signals to those from the spindle encoder.

```
sent      CSE,ENC
received  _CSE
```

3.13 CGE - Get Encoder Out Source

Purpose:

This command returns whether the signals present at the encoder output are from the spindle encoder or Aux Encoder.

Format:

CGE

Fields:

- [type]*
- ENC. The option sets the encoder out to the spindle encoder.
 - AUXENC. This option sets the encoder out to the aux encoder.

Returns

_CGE,[type]

Examples

This indicates that the encoder out signals come from the spindle encoder.

sent **CGE**
received **_CGE,ENC**

3.14 CGP - Get Drive Data

Purpose:

This command returns drive monitoring data. It is useful for diagnosing controller issues related to drive temperature and performance.

Format:

CGP

Fields:

| | |
|-------------------------|--|
| <i>[temp]</i> | – Heat sink temperature in degrees C. |
| <i>[phase V]</i> | – Highest voltage across all three phases. |
| <i>[supply V]</i> | – Half supply voltage. |
| <i>[supply current]</i> | – Current drawn from the supply. |
| <i>[wattage]</i> | – Wattage dissipated by one half of one drive phase. |
| <i>[integrand]</i> | – Value related to wattage limit. |
| <i>[static]</i> | – Value related to wattage limit. |

Returns

`_CGP,[temp],[phase V],[supply V],[supply current],[wattage],[integrand],[static]`

Examples

This indicates that the encoder out signals come from the spindle encoder.

```
sent      CGP
received  _CGP,22,1,42,0,0,0,S
```

4.0 Spindle Controller Move Commands

4.1 CMV - Perform a Trapezoidal Velocity Move

Purpose:

The purpose of this command is to move an axis at a specific velocity using a trapezoidal trajectory. The RPM passed to the command is the velocity the axis will accelerate or decelerate to and stay at indefinitely. A commanded velocity of 0 will cause the system to decelerate to a stop. Once the axis has reached the commanded velocity, other velocity commands may be sent, but position move commands should be executed only after returning the axis to rest. The acceleration can be changed while this command is executing by using the ASC command. This command can be aborted with ASX command.

Format:

CMV,[RPM],[<ACCEL>]

Fields:

- [RPM]* - Final velocity to achieve in RPM.
Range: 0 – 20000 RPM
- [<ACCEL>]* - (optional) Acceleration in RPM/S. The move will use the default acceleration set with SAC if this term is omitted.
Range: 10 – 10000 RPM/s

Returns

None

Examples

This example accelerates axis 0 to 1050 RPM.

```
sent      CMV,1050
received  _CMV
received  _CMV,COMPLETE
```

4.2 CMS - Perform an S-Curve Velocity Move

Purpose:

The purpose of this command is to move an axis at a specific velocity using an s-curve trajectory. The RPM passed to the command is the velocity the axis will accelerate or decelerate to and stay at indefinitely. A commanded velocity of 0 will cause the system to decelerate to a stop. Once the axis has reached the commanded velocity, other velocity commands may be sent, but position move commands should be executed only after returning the axis to rest. This command can be aborted with ASX command.

Format:

CMS,[RPM],[<ACCEL>]

Fields:

- [RPM]* - *Final velocity to achieve in RPM.*
Min: 0RPM *Max: Bearing, Winding, and Supply Dependent*
- [<ACCEL>]* - *(optional) Acceleration in RPM/S. The move will use the default acceleration set with SAC if this term is omitted.*
Min: 10RPM/s *Max: Winding and Supply Dependent*

Returns

None

Examples

This example accelerates axis 0 to 1050 RPM.

```
sent      CMS,1050
received  _CMS
received  _CMS,COMPLETE
```

4.3 ASX - Stop Accelerating and Maintain Current Speed

Purpose:

The purpose of this command is to interrupt a CMV or CMS command and maintain current spindle speed. Note: the spindle is axis 0.

Format:

ASX,[AXIS]

Fields:

None

Returns

None

Examples

This example interrupts a CMV command and maintains the spindle at the speed present when the ASX command was issued.

```
sent      CMV,20000  
sent      _CMV  
sent      ASX,0  
received  _ASX,0  
received  _CMV,COMPLETE
```

4.4 ASC - Change Commanded Acceleration

Purpose:

The purpose of this command is to modify the acceleration of a move initiated with a CMV command while the command is still processing. It allows the user to change the acceleration after issuing CMV,[RPM] but before the _CMV,COMPLETE message is received. Note: the spindle is axis 0.

Format:

ASC,[AXIS],[ACCEL]

Fields:

[ACCEL] - Acceleration in RPM/s.

Returns

None

Examples

This example starts a move to 20000 RPM using the standard acceleration. The acceleration is then changed to 10000 RPM/s upon receipt of the ASC command.

```
sent      CMV,20000
sent      ASC,0,10000
received  _ASC,0
received  _CMV,COMPLETE
```

4.5 CME - Perform an Externally Controlled Velocity Move

Purpose:

The purpose of this command is to start controlling the spindle velocity with an external source. Select the source using the CSI command. Care must be taken to avoid unreachable accelerations. The input frequency should be zero before starting a CME move. The frequency should then be ramped so as not to exceed the controller's maximum acceleration of 7.5K RPM/sec. Failure to do so will likely cause following errors.

Format:

CME

Fields:

None

Returns

None

Examples

This example switches the controller into an externally controlled velocity mode.

```
sent      CME
received  _CME
```

5.0 Manual Revision History

| Revision | Date | Description |
|----------|------------------|---|
| v1.00 | 20 November 2006 | Initial Release - BETA |
| v1.01 | 28 February 2007 | Added functions and command structure section |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |