

Chapter 5. Software Reference

Chapter Objectives

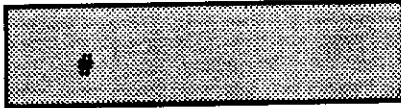
The information in this chapter will enable you to:

- Identify the four types of commands in Compumotor's X-Series Language
- Use this chapter as a reference for the function, range, default, and sample use of each command

Definition & Description of Command Types

Command Syntax

NAME:	The name of the command is shown here.
SYNTAX:	The proper syntax for the command is shown here. The specific parameters that are associated with the command are also shown. Definitions of the parameters are shown below. <ul style="list-style-type: none"> a This indicates that a device address must accompany the command. Only the device specified by this parameter will receive and execute the command. If this parameter , or any other parameter appears within brackets <>, providing a value for the parameter is optional. n An integer (n) may be used to specify a variety of values (acceleration, velocity, etc.) x This represents a letter that you must supply to distinguish which command function you want to apply. x, y This represents a letter that you must supply to distinguish which command functions you want to apply. s This indicates that a sign character, either positive or negative (+ or -) is required.
DEFAULT:	The factory default setting
RANGE:	The range of values for the parameter(s)
VALID:	Revision history. The revision of software when the command was added or modified.
RESPONSE:	In this field, a sample of the command is provided and followed by its response. The response includes the number of characters returned, their format, and any delimiters.
TYPE:	<p><i>Motion:</i> Motion commands affect motor motion. For example, accelerate, velocity, distance, go home, stop, direction, mode, etc.</p> <p><i>Status:</i> Status commands respond (report back) with information.</p> <p><i>Setup:</i> Setup commands define setup conditions for the application. Setup commands include the following types of commands:</p> <ul style="list-style-type: none"> • Homing (Go home acceleration and velocity, etc.) • Input/Output (Limits, scan time, in-position time, etc.) • Tuning (Servo or position tracking) • General (Set switches, return to factory settings, etc.) <p><i>Programming:</i> Programming commands affect programming and program flow. For example, trigger, output, all sequence commands, quote, time delays, pause and continue, enable and front-panel, loop and end loop, line feed, carriage return, and backspace.</p>
ATTRIBUTES:	<p>Immediate or Buffered</p> <ul style="list-style-type: none"> • Device Address Optional, Device Address Required, or Never Saved • Savable in Sequence or Independently Saved
DESCRIPTION:	A description of the command is provided here. The description defines the command's purpose and explains when you might use it.
SEE ALSO:	Commands that are related or similar to the command described are listed here.
EXAMPLES:	An example of how you might use the command is provided.



NAME: Single Step
SYNTAX: <a>#
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Immediate, Device Address Optional, Never Saved
DESCRIPTION: This command controls the execution of a sequence when the **XST** (Single Step Mode) is enabled. Each time you enter the # command followed by a delimiter (carriage return or space), one command in the sequence buffer will be executed. You can run in Single Step mode only if you have RS-232C interface connected to a host. If you issue a Kill (**K**) command, while you are in Single Step mode, the sequence execution will be aborted, but the Single Step mode is retained. When you cycle power, the controller will no longer be in single step mode.

SEE ALSO:
EXAMPLE:

XST	
<u>Command</u>	<u>Description</u>
> XE1	Erases sequence 1
> XD1	Defines sequence 1
> A5	Sets acceleration to 5 rps ²
> V2	Sets velocity to 2 rps
> D10000	Sets distance to 10,000 steps
> G	Executes the move (Go)
> XT	Ends sequence definition
> XST1	Enables Single Step mode
> XR1	Executes Sequence 1
> XTR1	Enables Trace mode
> #	Execute first command

Execute the first command
 *SEQUENCE_001_COMMAND_A5
 Displays the first command executed.

Execute the second command
 *SEQUENCE_001_COMMAND_V2
 Displays the second command executed.

Execute the third command
 *SEQUENCE_001_COMMAND_D10000
 Displays the third command executed.

Execute the fourth command.
 *SEQUENCE_001_COMMAND_G
 Displays the fourth command executed. Motor should have moved 10,000 steps.

Execute the fifth command
 *SEQUENCE_001_COMMAND_XT
 Displays the last command.

A

NAME: Acceleration
SYNTAX: <a>**A**n
DEFAULT: n = 10 rps²
RANGE: n= 0.001 to 9999.999 rps²
VALID:
RESPONSE: a**A** ***A**n[cr]
TYPE: Motion
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: The Acceleration command specifies the acceleration rate to be used upon executing the next Go (**G**) command. The acceleration remains set until you change it and is stored in nonvolatile memory. You do not need to reissue this command for subsequent Go (**G**) commands.

SEE ALSO: **D, V, G**

EXAMPLES:	<u>Command</u>	<u>Description</u>
	> MN	Sets the moves to mode normal (preset moves)
	> A5	Sets Acceleration to 10 rps ²
	> V10	Sets Velocity to 10 rps
	> D10000	Sets Distance to 10,000 steps
	> G	Executes the move (Go)

B

NAME: Buffer Status Report
SYNTAX: a**B**
DEFAULT: n = R
RANGE: n = B, R
VALID:
RESPONSE: a**B** *n[cr]
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: The buffer status command reports the status of the command buffer. If the command buffer is empty or less than 90% full, the controller will respond with a *R[cr]. A *B[cr] response will be issued if less than 10% of the command buffer is free. The command buffer is 2,000 bytes long.

This command is commonly used when a long series of commands will be loaded remotely. If the buffer size is exceeded, the extra commands will never be received by the controller.

*R = More than 10% of the buffer is free
 *B = Less than 10% of the buffer is free

SEE ALSO: **BS**

EXAMPLES:	<u>Command</u>	<u>Response</u>
	> IB	* R [cr] (more than 10% of the Buffer is free)

BCDB

NAME: Buffered Configure Dead Band
SYNTAX: <a>**BCDB**n
DEFAULT: n = 0
RANGE: n = 0 - 32,767
RESPONSE: a**BCDB** *SLIP_FAULT_DEADBAND=n_STEPS[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: The buffered configure dead band command defines the dead band value. If a parameter follows the command, the value will become the new dead band value. An Output may be configured to indicate when the absolute value of the following error is outside of the dead band region (the **OUT** command). When the following error exceeds the dead band, the exceed dead band indicator output will be turned ON. When the following error is within the dead band region set with this command, the exceed dead band indicator output will be OFF.

SEE ALSO:

EXAMPLES:

OUT, CDBCommandDescription

> A10	Sets acceleration to 10 rps ²
> V5	Sets velocity to 5 rps
> D50000	Sets distance to 10,000 steps
> BCDB50	Configure dead to 50 steps
> OUT1G	Configures Output 1 as Dead band Output Output 1 will go on when the Position Error exceeds the deadband Execute the Move (Go)

The Exceed Dead band Indicator output indicates whether or not the following error exceeds 50 motor steps

BCDG

NAME: Buffered Configure Differential Gain
SYNTAX: <a>**BCDG**n
DEFAULT: n = 0
RANGE: n = 0-99
RESPONSE: a**BCDG** *DIFFERENTIAL_GAIN=N_PERCENT [cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the differential gain to be used for tuning. When you tune the drive using the Buffered Differential Gain (**BCDB**) command, you are actually setting a percentage of the Differential Gain Maximum (**CDM**). The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO:

EXAMPLES:

CDG, CDM, BCDM, SV, Tuning SectionCommandDescription

> A5	Sets Acceleration to 5 rps ²
> V10	Sets Velocity to 10 rps
> D32000	Sets Distance to 32,000 steps
> BCDG44	Sets the differential gain to 44 percent of the maximum value
> G	Executes the move (Go)
> BCDG20	Sets the differential gain to 20 percent of the maximum value
> G	Executes the move (Go)

The derivative gain is set to 44 percent of the maximum value before the first 32,000 step move, and to 20 percent of the maximum value before the second 32,000-step move.

BCDM

NAME: Buffered Configure Differential Maximum
SYNTAX: <a>**BCDM**n
DEFAULT: n = 100
RANGE: n = 0 - 32,767
VALID:
RESPONSE: a**BCDM*** *DIFFERENTIAL_GAIN_MAXIMUM=N[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the maximum differential gain you may use for tuning purposes. If a valid number is entered, it will become the new maximum differential gain. This command is identical to **CDM** except that it is buffered. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO: **CDM, BCDG, CDG, SV, Tuning Section**

<u>Command</u>	<u>Description</u>
> A5	Sets acceleration to 5 rps ²
> V10	Sets velocity to 10 rps
> D32000	Sets distance to 32,000 steps
> BCDM15000	Sets maximum derivative gain to 15,000
> G	Executes the Move (Go)
> BCDM5000	Sets derivative maximum gain to 5,000
> G	Executes the Move (Go)

The differential gain is set to 15,000 before the first 32,000-step move, and to 5,000 before the second 32,000-step move.

BCIG

NAME: Buffered Configure Integral Gain
SYNTAX: <a>**BCIG**n
DEFAULT: n = 0
RANGE: n = 0-99
VALID:
RESPONSE: a**BCIG** *INTEGRAL_GAIN_=n_PERCENT[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Independently Saved
DESCRIPTION: If you supply a value, the specified value will become the new value for the integral gain. A new value for the **BCIG** will be calculated using the entered value as a percent of maximum Integral value. The **CIG** command differs from the **BCIG** command only in that it can be executed sequentially within a sequence. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO: **CIG, CIM, BCIM, SV, Tuning Section**

<u>Command</u>	<u>Description</u>
> A5	Sets acceleration to 5 rps ²
> V10	Sets velocity to 10 rps
> D32000	Sets distance to 32,000 steps
> G	Executes the move (Go)
> BCIG50	Sets integral gain to 50% of maximum value
> G	Executes the move (Go)

The first 32,000-step move is made, then the integral gain is changed to 50 percent of maximum value before executing the second 32,000-step move.

BCIL

NAME: Buffered Configure Maximum Integral Sum Limit
SYNTAX: <a>**BCIL**n
DEFAULT: n = 65535
VALID: n = 0 - 214748347
RESPONSE: a**BCIL** *INTEGRATOR_LIMIT=n[cr]
TYPE: Setup
ATTRIBUTES: Buffered , Device Address Optional, Independently Saved
DESCRIPTION: This command defines the maximum integral sum limit. The value for the sum of the integral value will not be able to exceed the value set by the **BCIL** command. This command is useful if you want to move into a final position quickly. If you have a small **BCIL** value, the system will not allow the sum of the errors to exceed the n value resulting in the motor moving into position faster. If you make the **BCIL** value very large, the response of the final move will be slow. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO:

CIG, CIL, BCIM, BCIG, SV, CIM, Tuning Section

EXAMPLES:

<u>Command</u>	<u>Description</u>
> BCIL70000	Sets the maximum sum of the integral error to 70,000

BCIM

NAME: Buffered Configure Integral Maximum
SYNTAX: <a>**BCIM**n
DEFAULT: n = 1000
RANGE: n = 0-32,767
VALID:
RESPONSE: a**BCIM** *INTEGRAL_GAIN_MAXIMUM=n[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Independently Saved
DESCRIPTION: This command is identical to the **CIM** command, except that it is buffered. This command defines the maximum integral gain you may use for tuning purposes. If you enter a valid number, it will become the new maximum integral gain. When you tune the drive using the Configure Integral Gain (**CIG**) command, you are actually setting the percentage of the maximum integral gain. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO:

CIM, CIG, BCIM, SV, CIL, BCIL, Tuning Section

EXAMPLE:

<u>Command</u>	<u>Description</u>
> A5	Sets acceleration to 5 rps ²
> V10	Sets velocity to 10 rps
> D32000	Sets distance to 32,000 steps
> BCIM2000	Sets maximum integral gain to 2000
> G	Executes the move (Go)
> BCIM5000	Sets maximum integral gain to 5000
> G	Executes the move (Go)

The maximum integral gain is set to 2000 before the 32,000-step move and to 5000 after the first move.

BCPE

NAME: Buffered Configure Position Error
SYNTAX: <a>**BCPE**n
DEFAULT: n = 4000
RANGE: n = 0 - 999,999,999
RESPONSE: a**BCPE** ***MAXIMUM_POSITION_ERROR=n_STEPS**[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: If you supply a value, the specified value will become the new value for the maximum position (or following) error. If the actual following error exceeds this number (In Encoder Steps), the JSI will shutdown the drive and flash an error code 20 (following error exceeded).
 This command is identical to **CPE** except that it is buffered. It is useful when the position error must be changed within a sequence.

SEE ALSO:
EXAMPLE:

CPE, CDB, Tuning Section

<u>Command</u>	<u>Description</u>
> A5	Sets acceleration to 5 rps ²
> V10	Sets velocity to 10 rps
> BCPE4000	Sets maximum following error to 4,000
> D32000	Sets distance to 32,000
> G	Executes the move (Go)
> BCPE6000	Sets maximum following error to 6,000
> G	Executes the move (Go)

The maximum following error is set to 4,000 before the first 32,000-step move and to 6,000 before the second 32,000-step move.

BCPG

NAME: Buffered Configure Proportional Gain
SYNTAX: <a>**BCPG**n
DEFAULT: n = 10
RANGE: n = 0 - 99
RESPONSE: a**BCPG** ***PROPORTIONAL_GAIN=n_PERCENT**[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the proportional gain to be used for tuning. When you tune the drive using the Buffered Configure Proportional Gain (**BCPG**) command, you are actually setting a percentage of the proportional gain maximum (**CPM**). This command is useful in tuning out the proportional error during a move. This command is identical to **CPG** except that it is buffered. It is useful when the proportional gain needs to be changed in a sequence. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO:
EXAMPLES:

CPG, CPM, Tuning Section, BCPM, SV

<u>Command</u>	<u>Description</u>
> A5	Sets acceleration to 5 rps ²
> V10	Sets velocity to 10 rps
> D32000	Sets distance to 32,000
> BCPG20	Sets proportional gain to 20
> G	Executes the move (Go)
> BCPG50	Sets proportional gain to 50
> G	Executes the move (Go)

The proportional gain is set to 20 percent of the maximum value before the first 32,000 step move and to 50 percent of the maximum value before the second 32,000 step move.

BCPM

NAME: Buffered Configure Proportional Maximum
SYNTAX: <a>**BCPM**n
DEFAULT: n = 500
RANGE: n = 0 - 32,767
VALID:
RESPONSE: a**BCPM** *PROPORTIONAL_GAIN_MAXIMUM=n[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command defines the maximum proportional gain you can use for tuning purposes. If a valid number is entered, it will become the new maximum proportional gain. When you tune the drive using the proportional gain (CPG), you will actually set a percentage of the Maximum Proportional gain. This command is identical to **CPM** except that it is buffered. It is useful when the proportional maximum must be changed within a sequence. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO: **CPM, CPG, BCPG, SV, Tuning Section**

EXAMPLES:	<u>Command</u>	<u>Description</u>
	> A5	Sets acceleration to 5 rps ²
	> V10	Sets velocity to 10 rps
	> D25000	Sets distance to 25,000 steps
	> BCPM5000	Sets maximum proportional gain to 5000
	> G	Executes the move (Go)
	> BCPM6000	Sets maximum proportional gain to 6,000
	> G	Executes the move (Go)

BS

NAME: Buffer Status Report
SYNTAX: a**BS**
DEFAULT: n = 2000
RANGE: n = 0 to 2000
VALID:
RESPONSE: a**BS** *n[cr]
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: Reports the number of characters (including delimiters) that can be loaded into the command buffer. When entering long string commands, check the buffer status to be sure that buffer space is available.

If you are using a computer interface, you can load the program lines to the controller faster than the controller can execute them. Therefore, the commands would get stored in the buffer. This command tells you how many more characters can be downloaded to the controller.

SEE ALSO:

EXAMPLES:	<u>Command</u>	<u>Description</u>
	> BS *100	Space for 100 characters remain in the command buffer

C

NAME: Continue
SYNTAX: <a>**C**
DEFAULT: None
RANGE: None
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Immediate, Device Address Optional, Never Saved
DESCRIPTION: The Continue (**C**) command ends a pause state. It enables your controller to continue executing buffered commands. After you initiate a pause with the Pause (**PS**) command or the Pause and Wait for Continue (**U**) command, you can clear it with a Continue command. This command is useful when you want to transmit a string of commands before you actually need to execute them.

If you are using the resume function (**SSL1**), the Continue (**C**) command can also be used resume execution of a move or a sequence after the program is halted by the Stop (**S**) command. Upon entering the C command, the program or the move will be started from the point where it stopped.

SEE ALSO: **PS, U, SSL**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> PS	Pauses execution until the indexer receives a C command
> MC	Sets move to Continuous mode
> A5	Sets acceleration to 5 rps ²
> V5	Sets velocity to 5 rps
> G	Executes the move (Go)
> T10	Waits 10 seconds after the move
> V0	Sets velocity to zero
> G	Decelerates the motor to zero velocity
> C	Starts executing commands in buffer

CDB

NAME: Configure Dead Band
SYNTAX: <a>**CDBn**
DEFAULT: n = 0
RANGE: n = 0-32,767
RESPONSE: a**CDB** *SLIP_FAULT_DEADBAND=nSTEPS[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional, Savable in Sequence
DESCRIPTION: The **CDB** command defines the new dead-band value in motor steps. If dead-band is exceeded the slip fault output (if defined using the **OUT** command) will be active. When the exceeded deadband indicator output is off it indicates that the absolute value of the following error is within the dead band region. This is useful when you need to know if the motor rotor is within a certain tolerance range with respect to the commanded position.

SEE ALSO:
EXAMPLES:

<u>Command</u>	<u>Description</u>
> 1CDB50	Sets the dead-band to 50 steps
> 1CDB	(Response) *SLIP_FAULT_DEADBAND=50 STEPS
> OUT1G	Sets Output 1 as a Exceeded Deadband Indicator Output
> A10	Sets acceleration to 10 rps ²
> V5	Sets velocity to 10 rps
> D25000	Sets distance to 25,000
> G	Executes the move (Go)

If the motor is more than 50 encoder steps off, the JSI will activate Output 1.

CDG

NAME: Configure Differential Gain
SYNTAX: <a>CDGn
DEFAULT: n = 0
RANGE: n = 0 - 99
VALID:
RESPONSE: aCDG *DIFFERENTIAL_GAIN=n_PERCENT[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the differential gain to be used for tuning. When you tune the drive using the Differential Gain (CDG) command, you are actually setting a percentage of the Differential Gain Maximum (CDM). The Save (SV) command must be used to retain the value in nonvolatile memory.

SEE ALSO: BCDG, CDM, BCDM, SV, Tuning Section

EXAMPLES:

<u>Command</u>	<u>Description</u>
> 1CDG30	Sets the differential gain to 30% of the maximum value

CDM

NAME: Configure Differential Maximum
SYNTAX: <a>CDMn
DEFAULT: n = 100
RANGE: n = 0 - 32,767
VALID:
RESPONSE: aCDM *DIFFERENTIAL_GAIN_MAXIMUM=n[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the maximum differential gain you may use for tuning purposes. If you enter a valid number, it will become the new maximum differential gain. When you tune the drive using the Configure Derivative Gain (CDG) command, you will actually set a percentage of the maximum differential gain.(CDM). The Save (SV) command must be used to retain the value in nonvolatile memory.

SEE ALSO: BCDM, CDG, BCDG, SV, Tuning Section

EXAMPLES:

<u>Command</u>	<u>Description</u>
> CDM500	Sets the maximum differential gain to 500
> CDG35	Sets the differential gain to 35% of the maximum value set by CDM.

CFB

NAME: Configure Feedback Device
SYNTAX: <a>CFBn.x
DEFAULT: n = 1
RANGE: n = 1, 2, x = 200-32768
RESPONSE: *DEVICE_NUMBER_n[cr]*DEVICE_RESOLUTION_x[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional
DESCRIPTION: This command configures the JSI Controller to accept position feedback from the feedback device number entered, and also the feedback device resolution. The controller output must be off to use the CFB command.

CFB1: Selects an incremental encoder
CFB2: Selects an AR23 absolute encoder with -4 decoder box (ABS8)
CFB3: Selects an AR-C, AL-C, or Photo-Trak absolute encoder
CFB4: Selects an AR23 absolute encoder with -1 decoder box

Once you define the feedback device and the resolution of the feedback device, the JSI controller will scan either the POSITION FEEDBACK Input for absolute encoder feedback (**CFB2**, **CFB3**, or **CFB4**) or the ENCODER input for incremental encoder feedback (**CFB1**). You must turn off the JSI with the **OFF** command before changing the encoder type or resolution.

In incremental encoder mode (**CFB1**), the JSI can accept TTL Square Wave quadrature pulses in either a differential or single-ended fashion. Compumotor's incremental encoders are differential quadrature encoders (refer to Figure 3-4 in Chapter 3 for a wiring diagram of this encoder). You can also define encoder resolution with the **CFB1** command. The default setting is 4,000 steps per revolution. The maximum frequency of pulses that the JSI can read is 60 KHz before quadrature. Therefore, after quadrature, the maximum frequency the JSI can read is 240 KHz.

In the absolute encoder mode (**CFB2**, **CFB3**, or **CFB4**), the JSI reads the parallel data that the absolute encoder provides. Because it takes longer to process parallel positional information, the servo sample period for the PID loop will be slowed to 1,024 microseconds from 512 microseconds. However, the I/O functions are still performed every 512 microseconds. Refer to Chapter 3, *Installation*, for connection diagrams for the absolute encoders.

Entering the resolution and leaving the device number blank (e.g., **CFB,5000**) only changes the resolution, leaving the feedback device the same. When you change the feedback device, some other parameters are affected.

In **CFB1, 4000**, the Motor Resolution (**CMR**) command and Configure Position Error (**CPE**) both become 4,000. In **CFB2, 16,384**, both **CMR** and **CPE** both default to 16,384. In incremental mode (**CFB1**) the controller sample period time is 512 usec. If you use absolute encoder mode (**CFB2**), the sample period goes up to 1024 usec. You must set the JSI to **CFB2** if you are using an absolute encoder, otherwise the sample period will get the controller and the encoder out of synchronicity.

SEE ALSO:
EXAMPLE:

CMR, CPE

Command

> **CFB1,8000**

> **CFB**

Description

Define the feedback device as incremental encoder with the resolution of 8,000 steps per revolution.

*DEVICE_NUMBER = 1_INCREMENTAL ENCODER

*DEVICE_RESOLUTION = 8000

CIG

NAME: Configure Integral Gain

SYNTAX: <a>**CIG**n

DEFAULT: n = 0

RANGE: n = 0 - 99

VALID:

RESPONSE: a**CIG** *INTEGRAL_GAIN= n_PERCENT[cr]

TYPE: Setup

ATTRIBUTES: Immediate, Device Address Optional, Independently Saved

DESCRIPTION: This command configures the integral gain to be used for tuning. When you tune the drive using Configure Integral Gain (**CIG**), you are actually setting a percentage of the Maximum Integral Gain (**CIM**). The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO:

EXAMPLES:

BCIG, CIM, BCIM, SV, CIL, BCIL, Tuning Section

Command

> **CIG70**

Description

Sets the integral gain to 70% of the maximum integral gain.

CIL

NAME: Configure Integral Sum Maximum Limit
SYNTAX: <a>**CIL**n
DEFAULT: n = 65535
RANGE: n = 0 - 2147483647
VALID:
RESPONSE: a**CIL** *INTEGRATOR_LIMIT=n[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the maximum integral sum limit. The value for the sum of the integral value is set by the **CIL** command. This command is useful if you want to move into a final position quickly. If you have a small **CIL** value, the system will not allow the sum of the errors to exceed the n value. This will result in the motor moving into position faster. If you make the **CIL** value very large, the response of final move will be slow. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO: **CIG, BCIG, CIM, BCIM, SV, BCIL, Tuning Section**
EXAMPLE:

<u>Command</u>	<u>Description</u>
CIL70000	Set the maximum sum of the integral error to 70,000

CIM

NAME: Configure Integral Maximum
SYNTAX: <a>**CIM**n
DEFAULT: n = 1000
RANGE: n = 0 - 32,767
VALID:
RESPONSE: a**CIM** *INTEGRAL_GAIN_MAXIMUM=n[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the maximum integral gain you may use for tuning purposes. If you enter a valid number, it will become the new maximum integral gain. When you tune the drive using the Configure Integral Gain (**CIG**) command, you are actually setting the percentage of the maximum integral gain. The Save (**SV**) command must be used to retain the value in nonvolatile memory.

SEE ALSO: **BCIM, CIG, SV, CIL, BCIG, BCIL Tuning Section**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> CIM1000	Sets the maximum integral gain to 1,000
> CIG45	Sets the integral gain to 45% of the maximum integral gain

COCH

NAME: Configure Output as Current Source (High)
SYNTAX: <a>**COCH**
DEFAULT: Controller is factory set to ± 10 volt voltage source
RANGE: None
VALID:
RESPONSE: None
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional
DESCRIPTION: This command configures the JSI controller's output stage as a 200 mAmp current source. You may use the **COP** command to limit the maximum current output (Make it lower than 200 mAmp). From the factory setting, the controller is set as a voltage output source (**COV**). You must issue this command to change your output from voltage to current. All Compumotor's Analog Motor/Drive's are voltage input drives. You must not configure the JSI controller as a current source if the drive is voltage controlled. You may also use the **COCL** command to configure the output as a current source output with a maximum current of 60mA.

SEE ALSO: **COV, COCL, COP, DCC**

<u>Command</u>	<u>Description</u>
COCH	Configure the output as a 200 mAmp current source.

COCL

NAME: Configure Output as Current Source (Low)
SYNTAX: <a>**COCL**
DEFAULT: Controller is factory set to ± 10 volt voltage source
RANGE: None
VALID:
RESPONSE: None
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional
DESCRIPTION: This command configures the JSI Controller's output stage as a 60 mAmp current source. You may use the **COP** command to limit the maximum current output (Make it lower than 60 mAmp). The advantage of the **COCL** command over the **COCH** (High output current to 200 mAmp) is that the resolution is higher for **COCL** than for **COCH**. The JSI controller is set as voltage output source (**COV**) from the factory. You must issue the **COCL** command to change your output from a voltage to current source. You must not configure the JSI Controller as a current source if the drive is voltage controlled. You may also use the **COCH** command to configure the output as current source output with a maximum current of 200 mAmps.

SEE ALSO: **COV, COCH, COP, DCC**

<u>Command</u>	<u>Description</u>
COCL	Configure the output of the JSI controller as a 60 mAmp current source.

COP

NAME: Configure Output Peak
SYNTAX: <a>**COP**n
DEFAULT: n = 100
RANGE: n = 0 - 100
VALID:
RESPONSE: See Example
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional
DESCRIPTION: This command sets the maximum value for the output command (**COV**, **COCL**, **COCH**) can be. The command specifies the maximum value of the control output as the percentage of the maximum control output. The JSI Controller uses the 12 bit DAC (Digital to Analog Converter) to determine the output current or output voltage. When the output peak is set at 100% (**COP100**), all of the DAC is used. At 50% (**CO50**), only half of the DAC range is used. If the output is configured as a 200mA current source (**COCH**), at 100% the output will be 200mA, and the full DAC range is used. If the output peak is set at 50% (**COP50**), the output current maximum will be 100 mA. For a 60 mA current source (**COCL**), 100% corresponds to 60mA output, and 50% corresponds to 30mA. The voltage source (**COV**) at 100% will provide 10 volts maximum output, and at 50% it will provide 5 volts maximum output. Use the DCC (Display current configuration command) to see if you are in either current or voltage mode.

SEE ALSO: **COV, COCH, COCL, DCC**
EXAMPLE:

<u>Command</u>	<u>Description</u>
COCH	Configures output as current source (200mA)
COP10	limit the maximum output as 10% of the maximum (+ 20mA)
ICOP	The response should be *CURRENT_OUTPUT_LIMIT = 19.93_MILLIAMPS[cr]

COV

NAME: Configure Output as Voltage Source
SYNTAX: <a>**COV**
DEFAULT: Controller is factory set to ± 10 volt voltage source
RANGE: None
VALID:
RESPONSE: None
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional
DESCRIPTION: This command configures the JSI controller's output stage as a ± 10 volt voltage source. The JSI is set as a voltage output source. You may use the **COP** command to limit the maximum voltage output (Make it lower than 10 Volts). If you need to use the current output, you must use either the **COCH** or **COCL** commands to change the controller from voltage output to current output. The voltage output uses a 12 bit DAC (Digital to Analog Converter). When you use the **COP** command to limit the output voltage, you will not affect the resolution of the output voltage.

SEE ALSO: **COCH, COCL, COP, DCC**
EXAMPLE:

<u>Command</u>	<u>Description</u>
COV	Sets the output as voltage source
COP50	Set the maximum output as 50% of + 10V.

CMR

NAME: Configure Motion Resolution
SYNTAX: <a>**CMR**n
DEFAULT: n = 4,000
RANGE: n = 200 - 32768
VALID:
RESPONSE: a**CMR** ***MOTION_RESOLUTION**=n_STEPS/REV[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Independently Saved
DESCRIPTION: You can define the motion resolution using this command. You must shut down the drive with the **OFF** command before defining a new resolution.

Changing motion resolution with the **CMR** command does not affect the system's dynamic performance. The microprocessor converts the commanded position to the appropriate absolute feedback position mathematically.

Before you define the new resolution, you must issue either the **OFF** or **ST1** command to shut down the drive. After the **CMR** command is executed, you must issue either the **ON** or **ST0** command to re-enable the drive. The new resolution that you issue will not take effect until you issue a distance command after the **CMR** command.

It should be noted that this command changes the values and responses of any other command that uses motion steps. You should configure motion resolution before you use any of the other commands that are defined in motion steps.

SEE ALSO: **OFF, ON, ST1, ST0**
EXAMPLE:

Command	Description
OFF	Turns Amplifier off
CMR4096	Defines a motion resolution of 4,096 steps per revolution
ON	Turns amplifier on

This example demonstrates the procedures required to change the motion's resolution.

CPB

NAME: Configure Push Button
SYNTAX: <a>**CPB**n
DEFAULT: n = 1
RANGE: n = 0, 1
VALID:
RESPONSE: a**CPB** *1_FRONT_PANEL_ENABLED_[cr] or *0_FRONT_PANEL_DISABLED_[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command allows you to control user access to the front panel push buttons. Since the PID tuning parameters can be modified via the Front Control Panel, and changing these values can effect move time and final position accuracy, it may be desirable to disable this interface during normal operations. The **CPB1** command enables the front panel push buttons and **CPB0** will disable the front panel.

SEE ALSO:

EXAMPLES:

<u>Command</u>	<u>Description</u>
> L	Loop infinite times
> A10	Sets acceleration to 10 rps ²
> V1	Sets velocity to 1 rps
> D5000	Sets distance to 5,000 steps
> G	Executes the move (Go)
> CPB1	Enable push buttons
> T10	Wait 100 seconds
> CPB0	Disable push buttons
> N	End loop

This sequence will only allow the user to access the front panel push buttons for 10 seconds after the move is complete. During the move, the front panel push buttons for tuning are disabled.

CPE

NAME:	Buffered Configure Position Error
SYNTAX:	<a>CPEn
DEFAULT:	n = 4000
RANGE:	n = 0 - 999,999,999
VALID:	
RESPONSE:	aCPE *MAXIMUM_POSITION_ERROR=n_STEPS[cr]
TYPE:	Setup
ATTRIBUTES:	Immediate, Device Address Optional
DESCRIPTION:	If a value is supplied, the specified value will become the new value for the maximum position (or following) error.

Depending on your load and the move parameters you will have different following errors: Higher acceleration will result in greater following error. This command defines the acceptable amount of following error for the system.

SEE ALSO:

EXAMPLE:

BCPE, CDB, BCDB, Tuning Section

<u>Command</u>	<u>Description</u>
> A5	Sets Acceleration to 5 rps ²
> V10	Sets Velocity to 10 rps
> CPE4000	Sets maximum following error to 4,000
> D32000	Sets Distance to 32,000
> G	Executes the move (Go)
> CPE6000	Sets maximum following error to 6,000
> G	Executes the move (Go)

The maximum following error is set to 4,000 before the first 32,000 step move and to 6,000 before the second 32,000 step move.

CPG

NAME: Configure Proportional Gain
SYNTAX: <a>CPGn
DEFAULT: n = 10
RANGE: n = 0 - 99
VALID:
RESPONSE: aCPG *PROPORTIONAL_GAIN=n_PERCENT[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the proportional gain to be used for tuning. When you tune the drive using the Configure Proportional Gain (CPG) command, you are actually setting a percentage of the Proportional Gain Maximum (CPM). The Save (SV) command must be used to retain the value in nonvolatile memory.

This command is useful in tuning out the positional error during moves.

SEE ALSO: BCPG, CPM, BCPM, Tuning Section, SV

<u>Command</u>	<u>Description</u>
> 1CPG4	Configure the proportional gain to four percent of the maximum value.

CPM

NAME: Configure Proportional Maximum
SYNTAX: <a>CPMn
DEFAULT: n = 500
RANGE: n = 0 - 32,767
VALID:
RESPONSE: aCPM *PROPORTIONAL_GAIN_MAXIMUM = n[cr]
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional, Independently Saved
DESCRIPTION: This command defines the maximum Proportional Gain you can use for tuning purposes. If a valid number is entered, it will become the new maximum proportional gain. When you tune the drive using the proportional gain (CPG), you will actually set a percentage of the Maximum Proportional Gain. The Save (SV) command must be used to retain the value in nonvolatile memory.

SEE ALSO: BCPM, CPG, BCPG, SV, Tuning Section

<u>Command</u>	<u>Description</u>
> A5	Sets acceleration to 5 rps ²
> V10	Sets velocity to 10 rps
> D25000	Sets Distance to 25,000 steps
> CPM5000	Sets maximum proportional gain to 5000
> G	Executes the move (Go)
> CPM6000	Sets maximum proportional gain to 6,000
> G	Executes the move (Go)

CR

NAME: Carriage Return
SYNTAX: a**CR**
DEFAULT: None
RANGE: None
VALID: None
RESPONSE: a**CR** [cr]
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Required, Savable in Sequence
DESCRIPTION: The Carriage Return (**CR**) command can determine when the indexer has reached a particular point in the execution buffer. When the indexer reaches this command in the buffer, it responds by issuing a carriage return (ASCII 13) over its interface back to the host computer. If you place the **CR** command after a Go (**G**) command, it will indicate when a move is complete. If you place the **CR** command after a Trigger (**TR**) command, it will indicate when the trigger condition is met.

SEE ALSO:
EXAMPLE:

<u>Command</u>	<u>Description</u>
> MPA	Sets mode for absolute position
> A50	Sets acceleration to 50 rps ²
> V5	Sets Velocity to 5 rps
> D5000	Sets distance to 5,000 steps
> G	Executes the move (Go)
> ICR	Sends a carriage return

The motor moves 5,000 steps. When the motor stops, the indexer sends a carriage return over its interface.

CVS

NAME: Configure Velocity Scale mode
SYNTAX: <a>**CVS**n, x
DEFAULT: n = 0, x=100
RANGE: n = 0 -2, x=0-5000
VALID:
RESPONSE: a**CVS** *n_ANALOG_MODE_DISABLED *ANALOG_GAIN_x if n=1, response is *n_ANALOG_FOLLOWING_MODE *ANALOG_GAIN_x if n=2, response is *n_ANALOG_SCALING_MODE *ANALOG_GAIN_x
TYPE: Setup
ATTRIBUTES: Immediate, Device Address Optional
DESCRIPTION: This command configures the JSI for using the analog input interface in one of two possible modes. The two modes are the velocity following mode and the velocity scaling mode. The modes are selectable by choosing a number from 0 to 2 for n in the command above. To set up the JSI for one of the possible modes the unit must have the output command disabled by issuing an OFF command. The the command is entered as follows; type in CVS followed by a number to select the desired mode, then a comma (,) followed by the analog gain.

VELOCITY FOLLOWING mode n=1
 Type in:
CVS 1,100

$n=1$ chooses the velocity following mode. In the velocity following mode the JSI will command a velocity profile which is proportional to the velocity signal applied at the analog input connector. The proportion factor is determined by the analog gain which is entered after the mode number. In this example a factor of 100 is entered. The velocity that the JSI will command is determined by the following equation.

$$\frac{\text{Analog input}}{10 \text{ volts}} \cdot \frac{\text{analog gain}}{100} \cdot V = \text{velocity commanded}$$

Analog input: This is the voltage level present at the analog input. A 10 bit analog to digital converter converts the analog signal into binary information. Typically in the velocity following mode the analog input signal will be the analog signal from a velocity tachometer attached to the servo whose velocity is to be followed.

10 volts: This is the maximum absolute value that the analog to digital converter can accept.

analog gain: This is the gain factor that is entered with the command after the mode number. In this case the analog gain is 100.

100: The analog gain is entered as a scale factor. Thus if the analog gain were 100 then the scale factor would be 1, and if the analog input were 10 volts then the factor which the velocity is multiplied by will be 1.

V: This is the velocity that the user enters using the V command.

In this mode a value for the velocity is chosen, then the velocity following mode is chosen using the number 1, and the analog gain is chosen to be between 0 and 5000. It should be noted that this velocity is the velocity of the velocity command profile. From this velocity profile a position command is determined for each sample period. The actual position and the commanded position create an error which goes through the PID algorithm and forms the analog velocity output command at the command output interface. This velocity is not the same as the velocity produced by the above equation. It is the velocity after the commanded velocity has gone through the PID filter.

VELOCITY SCALING mode $n=2$

Selects the velocity scaling mode. In this mode the voltage at the analog input is used to scale the commanded velocity. The JSI must be used in the continuous mode (MC command) to use the velocity scaling mode. The velocity is scaled according to the following equation:

$$\frac{\text{Analog input}}{10 \text{ volts}} \cdot \frac{\text{analog gain}}{100} \cdot V + V = \text{velocity commanded}$$

Analog input: This is the voltage level present at the analog input. A 10 bit analog to digital converter converts the analog signal into binary information. Typically in the velocity following mode the analog input signal will be the analog signal from a velocity tachometer attached to the servo whose velocity is to be followed.

10 volts: This is the maximum absolute value that the analog to digital converter can accept.

analog gain: This is the gain factor that is entered with the command after the mode number. In this case the analog gain is 100.

- 100:** The analog gain is entered as a scale factor. Thus if the analog gain were 100 then the scale factor would be 1, and if the analog input were 10 volts then the factor which the velocity is multiplied by will be 1.
- V:** This is the velocity that the user enters using the V command.

The JSI is placed in the continuous mode using the MC command. The CVS command is then used with n=2 for the scaling mode followed by a comma and the analog gain. The analog gain can be from 0 to 5000. The voltage at the analog input is divided by the full scale voltage of 10 volts then multiplied by the analog gain divided by 100. This is then multiplied by the velocity and added to the velocity. Multiplying the number times the velocity maintains the units of revs/second. Typically the analog input will be an external error signal which, based on its value will slow down or speed up the servo. The continuous velocity is adjusted by the analog input according to the above equation.

To disable the analog input mode which is selected by either a 1 or a 2 type in n=0. n=0 will disable the analog input signal from having any affect.

D

NAME: Distance
SYNTAX: <a>Dn
DEFAULT: n = 4000
RANGE: n = ± 838,860,800
VALID:
RESPONSE: aD *Dn[cr]
TYPE: Motion
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: The Distance (D) command defines motion in either the number of steps it will move or the absolute position it will seek after a Go (G) command is entered. In incremental mode (MPI), the value set with the Distance (D) command will be the distance motion will travel on all subsequent Go (G) commands. In absolute mode (MPA), the distance moved by the motor will be the difference between the current motor position and the position (referenced to the zero position) set with the D command. A valid distance must be defined with the D command before a preset move can be executed. The Distance (D) command has no effect on continuous moves (MC). The Distance (D) value is stored in nonvolatile memory.

SEE ALSO: A, G, V, MN, MPA, MPI

<u>Command</u>	<u>Description</u>
> MN	Set mode to normal (preset)
> A5	Set Acceleration to 5 rps ²
> V10	Set velocity to 10 rps
> D50000	Set Distance to 50,000 steps
> G	Executes the move (Go)

Motor will travel 50,000 steps in the clockwise direction after the G command is issued.

DAI

NAME: Display Analog Input
SYNTAX: a**DAI**
DEFAULT: None
RANGE: n = ± 10.00
RESPONSE: a**DAI** *ANALOG_INPUT=n_VOLTS
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: This command reports the analog voltage input to the JSI Controller. It can report the value either continuously or one time. This input can be used in conjunction with the Velocity Scaling (**CVS**) command to perform velocity tracking.

a**DAI** Continually reports the analog input value until any key is pressed
 a**DAI**1 Report analog input value 1 time only without descriptive text.

SEE ALSO: **CVS**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> 1 DAI 1	Report analog input 1 time only without descriptive text
> 1 DAI	Continually report the analog input value

DCC

NAME: Display Current Control Configuration
SYNTAX: a**DCC**
DEFAULT: None
RANGE: None
RESPONSE: See Example
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required
DESCRIPTION: This command reports the current configuration of the JSI Controller. It reports the following information:

- Feedback Device Number (Defined by **CFB**)
- Sample Period Time (Defined by **CFB** either as 512 or 1024 usec)
- Output Amplifier Type (Defined by **COV**, **COCL**, **COCH** to determine if the controller output should be a voltage or current source)
- Output Limit (Defined by **COP** to limit the maximum output voltage or current)
- Feedback Device Resolution (Defined by **CFB** command)

SEE ALSO: **CFB**, **COV**, **COCL**, **COCH**, **COP**
EXAMPLE:

```

>1DCC          *CONTROLLER_CONFIGURATION:

*FEEDBACK_DEVICE_NUMBER_=1_INCREMENTAL_ENCODER
  The feedback device is an incremental encoder

*SERVO_SAMPLE_PERIOD_=512_MICROSECONDS
  Controller sampling period is 512 microseconds

*OUTPUT_AMPLIFIER_IS_A_10_VOLT_VOLTAGE_SOURCE
  The JSI is using 10 volts voltage source
*VOLTAGE_OUTPUT_LIMIT=100.00_VOLTS
  The voltage output limit is set to 100% of the maximum

*DEVICE_RESOLUTION_=4000
  Feedback device resolution is 4000 steps/rev after quadrature.

```

DFS

NAME: Display Flags for Controller Status
SYNTAX: a**DFX**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: a**DFX** *bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb*[cr]
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: The display flags for Controller Command returns the Controller status flag as a 32 bit response. Each bit refers to a status flag as shown on the chart below. The associated number for each flag corresponds to each binary (1 or 0) response in the following order:
 31,30,29,28,....3,2,1,0

Bit

31 27 23 19 15 these bits are all reserved for
 30 26 22 18 14 future use they all return zero's
 29 25 20 17 13
 29 24 21 16 12
 10 9 7 6 4 2 1

Bit

14 User Fault no = 0 yes = 1
 11 Enable Circuit 0 = enable 1 = disabled
 8 Failed nonvolatile memory checksum no= 0 yes = 1
 5 max position error exceeded no = 0 yes = 1
 3 command out disabled 0 = no 1 = yes
 0 commanded shutdown 0 = no 1 = yes

DFX

NAME: Display Flags for Controller Status
SYNTAX: a**DFX**
DEFAULT: None
RANGE: b = 0, 1
VALID:
RESPONSE: a**DFX** *bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb_bbbb*[cr]
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: The display flags of Controller Command returns the controller status flag as a 32 bit response. Each bit refers to a status flag as shown on the chart below. The associated number for each flag corresponds to each binary (1 or 0) response is in the following order:
 31,30,29,28,....3,2,1,0

Bit

31 27 23 these bits are all reserved for
 30 26 22 future use they all return zero's
 29 25 13
 29 24

Bit	
21	hit a software CCW limit no = 0, yes = 1
20	hit a software CW limit no = 0, yes = 1
19	home limit not found = 0, found = 1
18	jog disable = 0, enable = 1
17	queued for RM mode no = 0, yes = 1
16	run sequence on power up no = 0 yes = 1
15	U command 0 = not waiting, 1 = waiting for continue
14	waiting for a trigger no = 0 yes = 1
13	home to resolver position no = 0 yes = 1
12	back up to home limit 0 = no yes = 1
11	high speed portion of home move no = 0 1 = in process
10	execute a sequence no = - yes = 1
9	wait on a timer no = 0 yes = 1
8	hit a CCW limit no = 0 yes = 1
7	hit a CW limit no = 0 yes = 1
6	PS command 0 = not waiting 1 = waiting for continue
5	absolute move direction no = CW yes = CCW
4	Incremental/absolute 0 = MPI 1 = MPA
3	Mode preset = 0 continuous = 1
2	Commanded move direction 0 = CW 1 = CCW
1	preset move in progress 0 = not moving 1 = moving
0	continuous move during acceleration/deceleration 0 = not moving 1 = moving

DMO

NAME:	Display Maximum Overshoot
SYNTAX:	aDMO
DEFAULT:	None
RANGE:	
VALID:	
RESPONSE:	aDMO *MAXIMUM_OVERSHOOT_ = n_STEPS
TYPE:	Status
ATTRIBUTES:	Immediate, Device Address Required, Never Saved
DESCRIPTION:	This command reports the number of steps the motor moves past the designated distance. This command is valid in preset moves only.

aDMO1 Gives overshoot report once, without descriptive text
aDMO Continually reports the position overshoot until any key is pressed.

This command is useful for tuning out overshoots in the system. When you tell the motor to execute a move, the overshoot value is set to zero and the overshoot is measured while the move is being made. When a new move is started the error is set to zero again.

SEE ALSO:	None																						
EXAMPLE:	<table> <thead> <tr> <th>Command</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>> L</td> <td>Loop forever until stopped</td> </tr> <tr> <td>> A10</td> <td>Sets acceleration to 10 rps²</td> </tr> <tr> <td>> V2</td> <td>Sets velocity to 2 rps</td> </tr> <tr> <td>> D25000</td> <td>Sets distance to 25,000 steps</td> </tr> <tr> <td>> G</td> <td>Executes the move (Go)</td> </tr> <tr> <td>> T2</td> <td>Waits 2 seconds after the move</td> </tr> <tr> <td>> 1DMO1</td> <td>Reports maximum overshoot once without descriptive text</td> </tr> <tr> <td>> H</td> <td>Changes direction of the move.</td> </tr> <tr> <td>> G</td> <td>Executes the move (Go)</td> </tr> <tr> <td>> N</td> <td>End Loop</td> </tr> </tbody> </table>	Command	Description	> L	Loop forever until stopped	> A10	Sets acceleration to 10 rps ²	> V2	Sets velocity to 2 rps	> D25000	Sets distance to 25,000 steps	> G	Executes the move (Go)	> T2	Waits 2 seconds after the move	> 1DMO1	Reports maximum overshoot once without descriptive text	> H	Changes direction of the move.	> G	Executes the move (Go)	> N	End Loop
Command	Description																						
> L	Loop forever until stopped																						
> A10	Sets acceleration to 10 rps ²																						
> V2	Sets velocity to 2 rps																						
> D25000	Sets distance to 25,000 steps																						
> G	Executes the move (Go)																						
> T2	Waits 2 seconds after the move																						
> 1DMO1	Reports maximum overshoot once without descriptive text																						
> H	Changes direction of the move.																						
> G	Executes the move (Go)																						
> N	End Loop																						

DOI

NAME: Display Output Instantaneous
SYNTAX: aDOI
DEFAULT: None
RANGE: n = ± 10.00VOLTS, ± 200 MILLIAMPS, ± 60 MILLIAMPS
VALID:
RESPONSE: aDOI *COMMAND_OUT=n
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: This command displays the Analog output being applied to the drive. It will constantly update and display the output to the drive. If you are in Voltage Output (**COV**) mode, the information range will be ± 10 volts. In you are in Lower Current Output (**COCL**) mode, the information range will be ± 60 milliamps. If you are in Higher Current Output (**COCH**) mode, the information range will be ± 200 milliamps. Analog output is continuously reported until any key is pressed.

aDOI Continually reports the analog output value until any key is pressed
 aDOI1 Report analog input value 1 time only without descriptive text.

SEE ALSO: **COV, COCL, COCH**
EXAMPLES:

<u>Command</u>	<u>Description</u>
> COV	Change JSI to + 10 Volts Output Mode
> DOI1	Report output voltage once
> DOI	Reports the output voltage continuously

DPA

NAME: Display Position Actual
SYNTAX: aDPA
DEFAULT: None
RANGE: None
VALID:
RESPONSE: aDPA *ACTUAL_POSITION=n_STEPS
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: The display position actual command displays the actual position in feedback steps. The function is canceled by sending any single ASCII character to the JSI.

aDPA1 reports the actual position 1 time without descriptive text
 aDPA reports the actual position continuously until a character is sent to the JSI

DPE

NAME: Display Position Error
SYNTAX: aDPE
DEFAULT:
RANGE: None
VALID:
RESPONSE: aDPE *POSITION_ERROR_=n_STEPS
TYPE: Status
ATTRIBUTES: Buffered, Device Address Required, Never Saved
DESCRIPTION: This command reports the difference between the position setpoint and the current position. The **PID** algorithm uses this number to determine the sign and magnitude of the command from the JSI. The larger the position error, the larger the control signal output to correct the error. This value is also used by the controller to see if the motor is within the deadband specified by the **CDB** command.

aDPE Continually reports the position error value until any key is pressed
 aDPE1 Report Position error value 1 time only without descriptive text.

SEE ALSO:

CDB

EXAMPLES:

Command

Description

aDPE

Continually reports the position error value until any key is pressed

aDPE1

Report position error value 1 time only without descriptive text.

DPS

NAME: Display Setpoint Position
SYNTAX: aDPS
DEFAULT: None
RANGE: None
VALID:
RESPONSE: aDPS *POSITION_SETPOINT=n_STEPSn[cr]
TYPE: Status
ATTRIBUTES: Buffered, Device Address Required, Never Saved
DESCRIPTION: This command reports the absolute position setpoint. This reporting is continued until any character is sent to the indexer through an RS-232C interface.

aDPS Continually reports the setpoint position value until any key is pressed

aDPS1 Report[position setpoint value 1 time only without descriptive text.

The position setpoint indicates the JSI's commanded position for **PID** computation.

SEE ALSO:

None

EXAMPLES:

Command

Response

> DPS

Reports the setpoint position continuously

> DPS1

Report setpoint position 1 time only without descriptive text.

DRD

NAME: Read Distance via Parallel Input/Output
SYNTAX: <a>**DRD**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command initiates the controller to read distance value from the parallel inputs. You must set up the inputs as Data input using the **IN** command. You must also set up outputs as strobe outputs using the **OUT** command. Typically, you would set up 8 inputs as DATA Inputs and 5 Outputs as strobe outputs using the **OUT** command. If an input is designated as a Data Sign, then if it is active during any byte read, the sign becomes negative, otherwise the inactive condition (default) is a positive sign.

You must also set up the strobe output delay time (typically greater than a **PLC** Scan Time, if using a **PLC**, or a minimal debounce time if using thumbwheel switches) so that the device you are getting data from will know when the **JSI** is ready for data. Sequence of events take place when we enter the **DRD** command:

- Step 1** The lowest Output bit designated as strobe output will go on and stay on for strobe output delay time defined by **STR** command.
- Step 1A** If an input is designated as a data valid input, then the strobe output will stay active until the data valid input is active.
- Step 2** The **JSI** will read the parallel inputs designated as **DATA** inputs. The lowest inputs will be the least significant bit. The **JSI** reads 8 bits (2 BCD digits) and stores them into two digits on the distance register.
- Step 3** Next Strobe output bit designated as strobe output will go on and stay on for delay time defined by **STR** command.
- Step 3A** If an input is designated as a data valid input, then the strobe output will stay active until the data valid input is active.
- Step 4** The **JSI** will read the parallel inputs designated as **DATA** inputs are placed in the lowest digits on the position register. Previously entered values are shifted two positions to the left.
- Step 5** Steps 3 through 4 are repeated as many times as the strobe outputs are available. You can use up to 5 different outputs as strobe outputs. The last two digits read are the least significant digit of the distance value.

SEE ALSO: **STR, IN, OUT**
EXAMPLE: The following example shows how the **JSI** reads the distance value of 3589721 steps. We will use inputs 1 through 8 for **DATA** inputs and outputs 1 through 4 for Strobe Outputs.

Step 1 Type the following:

<u>Command</u>	<u>Description</u>
OUT1J	Set Output 1 as Strobe Output
OUT2J	Set Output 2 as Strobe Output
OUT3J	Set Output 3 as Strobe Output
OUT4J	Set Output 4 as Strobe Output
IN1N	Set Input 1 as a Least Significant DATA input
IN2N	Set input 2 as a DATA input
IN3N	Set input 3 as a DATA input
IN4N	Set input 4 as a DATA input
IN5N	Set input 5 as a DATA input
IN6N	Set input 6 as a DATA input
IN7N	Set input 7 as a DATA input
IN8N	Set input 8 as a the Most significant DATA input
STR500	Set strobe output delay time to 500 msec.
A10	Sets acceleration to 10 rps ²
V5	Set velocity to 5 rps

DTP

NAME: Display Tuning Parameters
 SYNTAX: aDTP
 DEFAULT: None
 RANGE: None
 VALID:
 RESPONSE: See Description
 TYPE: Status
 ATTRIBUTES: Immediate, Device Address Required
 DESCRIPTION: This command displays the servo tuning parameters in block format.

P Proportional Gain
 I Integral Gain
 D Derivative Gain

The first row displays the percentage of the maximum gain being used by the JSI, and the second row displays the maximum gain set for each parameter.

This command also reports

- The sum of integrator Limit set by **CIL** command
- The maximum position error set by **CPE** command
- The motion resolution set by **CMR** command
- Status of the Hardware limit enabled/not enabled

SEE ALSO: **CIL, CPE, CMR, LD, CIG, CPG, CDG, CIM, CPM, CDM**
 EXAMPLE: **1DTP**

	P	I	D	
PERCENT	10	0	0	Proportional gain is set to 10% of the maximum (500)
MAXIMUM	500	1000	1000	Integral and Derivative gains are turned off (0%)

*INTEGRATOR_LIMIT=65535
CIL is set to 65535
 *MAXIMUM_POSITION_ERROR=4000_STEPS
CPE is set to 4000
 *MOTION_RESOLUTION=4000_STEPS/REV
CFB resolution is set to 4000 steps
 *LIMITS_ENABLED
 Limits are enabled by **LD0**

DVA

NAME: Display Velocity Actual
SYNTAX: a**DVA**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: a**DVA** *ACTUAL_VELOCITY=n_RPM
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: This number is reported in revolutions per minute. This value is the actual velocity being read from the feedback device.

a**DVA** Continually reports the actual velocity continuously until any key is pressed
 a**DVA1** Report the actual velocity 1 time only without descriptive text.

DVS

NAME: Display Velocity Setpoint
SYNTAX: a**DVS**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: a**DVS** *VELOCITY_SETPOINT=n
TYPE: Status
ATTRIBUTES: Immediate, Device Address Required, Never Saved
DESCRIPTION: This number is reported in steps per second and can be reported either once or continually. This value is the velocity being calculated by the PID loop.

a**DVS** Continually reports velocity setpoint until any key is pressed
 a**DVS1** Report the velocity setpoint 1 time only without descriptive text.

SEE ALSO: None
EXAMPLES: Command Description
 > **IDVS** Velocity setpoint is reported until a number key is pressed.

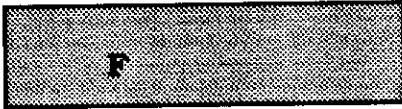
E

NAME: Enable RS-232C Communication Interface
SYNTAX: <a>**E**
DEFAULT: Communications Enabled
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Immediate, Device Address Optional, Never Saved
DESCRIPTION: The Enable Communications Interface (**E**) command allows the controller to accept commands over the RS-232C interface. You can re-enable the RS-232C interface with this command if you had previously disabled the interface with the Disable Communications Interface (**F**) command. If several units are using the same communications interface, the **E** and **F** commands can help streamline programming.

SEE ALSO:
EXAMPLES:

F	<u>Command</u>	<u>Description</u>
> F		Disables all units (axes) on the communications interface
> 1E		Enables Device 1
> 4E		Enables Device 4
> G		Executes the move (only axes 1 and 4 will move)
> E		Enable TS-232C communications on all devices.

Only Axes 1 and 4 will make the preset move. Then all the others will be enabled.



NAME: Disable RS-232C Communications Interface
SYNTAX: <a>**F**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Programming
ATTRIBUTES: Immediate, Device Address Optional, Never Saved
DESCRIPTION: The Disable Communications Interface (**F**) command is useful when you are programming multiple units on a single interface. Axes that are not intended to process global commands receive device specific **F** commands. This allows you to program other units without specifying a device identifier on every command. I

SEE ALSO:
EXAMPLES:

E	<u>Command</u>	<u>Description</u>
> 1F		Disables the communications interface on units with device address 1
> 3F		Disables the communications interface on units with device address 3
> G		All of the indexers (except devices 1 and 3) will execute the move (Go)



NAME: Go
SYNTAX: <a>**G**
DEFAULT: None
RANGE: None
VALID:
RESPONSE: None
TYPE: Motion
ATTRIBUTES: Buffered, Device Address Optional, Saveable in Sequence
DESCRIPTION: The Go (**G**) command instructs motion to occur using motion parameters that you have previously entered. You do not have to re-enter Acceleration (**A**), Velocity (**V**), Distance (**D**), or the current mode (**MN** or **MC**) commands with each Go (**G**) command. The acceleration (**A**), Velocity (**V**), and Distance (**D**) values are saved in nonvolatile memory. In the Incremental Preset mode (**MPI**), a Go (**G**) command will move the steps you specified with the Distance (**D**) command.

A Go (**G**) command in the Absolute Preset mode (**MPA**) will not cause motion unless you enter a change in Distance (**D**) command first, from where you are.

In the Continuous mode (**MC**), you only need to enter the Acceleration (**A**) and Velocity (**V**) commands prior to the Go (**G**) command. The system ignores the Distance (**D**) command in this mode.

No motion will occur until you enter the Go (**G**) command in both the Normal (**MN**) and Continuous (**MC**) modes.

If motion does not occur with the Go (**G**) command, an activated software travel limit or end-of-travel limit switch may be on. Check the limit switches.

SEE ALSO:
EXAMPLES:

S, MN, MC, A, V, D, MPI, MPA	
<u>Command</u>	<u>Description</u>
> MN	Sets mode to normal (preset)
> A5	Sets acceleration to 5 rps ²
> V10	Sets velocity to 10 rps
> D25000	Sets distance to 25,000 steps
> G	Executes the move (Go)
> A1	Sets acceleration to 1 rps ²
> G	Executes the move (Go)

Assuming the indexer is in Incremental Preset mode, a move of 25,000 steps executes and then repeats the 25,000-step move using the new acceleration value of 1 rps² (Total distance moved = 50,000 steps).

GH

NAME: Go Home
SYNTAX: <a>**GH** <n>
DEFAULT: n = 1
RANGE: n = ± .01 - 99.99 rps
VALID:
RESPONSE: None
TYPE: Motion
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: The Go Home (**GH**) command instructs the JSI to search for an the home limit switch in the positive or negative direction. It will define Home as the position where the Home Limit signal changed states nearest the edge selected with the OS command. As soon as the selected edge is detected (usually via a load activated switch) the motor decelerates to 0 velocity.

The controller will then position the motor on the outside of the selected edge. Finally, the motor will creep at Final Go Home velocity defined by **GHF** command in the direction of the home active region.

The process is the same in encoder step mode, except the controller also looks for the Z Channel input as well as the Home Limit input to be active. This means that the Z Channel pulse should be *enveloped* by the active region of the Home Limit input.

The controller will reverse direction if an End-Of-Travel limit is activated while searching for Home; however, if a second End-Of-Travel limit is encountered in the new direction the Go Home procedure will stop and the operation will be aborted. The **RG** command will indicate if the operation

was successful. If the Backup To Home Switch (**OSB**) function is disabled by the **OS** command, the the motor will be considered at Home if the Home limit input is still active at the end of the deceleration, following the encounter of the selected edge of the home switch. See the Application Design Section for a detailed description of the homing function.

If you set the Go Home velocity using the **GH** command, the motor will move in the direction and velocity specified by the **GH** command and the value is stored in the **GHV** command. If you do not specify the velocity, the homing will be executed using the velocity and direction defined by the **GHV** command.

SEE ALSO:
EXAMPLE:

GHA, GHV, IN, INL, GHF, OSB, OSC, OSD, OSG, OSH

<u>Command</u>	<u>Description</u>
> GHA10	Sets go home acceleration to 10 rps ²
> GHV2	Sets go home velocity to 2 rps
> IN1S	Sets input 1 as the remote go home input
> INL0	The remote go home will be executed when the Go Home input is low (Closed)
> GH	Go Home in the direction and velocity specified by GHV command using the GHA acceleration.

GHA

NAME: Go Home Acceleration
 SYNTAX: <a>**GHA**n
 DEFAULT: n = 99.000 rps²
 RANGE: n = 0.0001 - 9999.999 rps²
 VALID:
 RESPONSE: a**GHA** ***GHA**n[cr]
 TYPE: Motion
 ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
 DESCRIPTION: The Go Home Acceleration (**GHA**) command sets the linear acceleration value that will be used during any subsequent Go Home (**GH**) moves. The value is stored in nonvolatile memory.

The Go Home velocity is set using the Go Home Velocity (**GHV**) command. You must use the **IN** command to specify the remote Go Home input.

SEE ALSO:
EXAMPLE:

GH, GHV, IN, INL

<u>Command</u>	<u>Description</u>
> GHA100	Sets Go Home acceleration to 100 rps ²
> GHV2	Sets Go Home velocity to 2 rps
> IN1S	Sets input 1 as the remote go home input
> INL0	The remote Go Home will be executed when Go Home is low (closed)
> GH	Go Home in the direction and velocity specified by GHV command, using the GHA acceleration

GHF

NAME: Go Home Final Velocity
SYNTAX: <a>**GHF**n
DEFAULT: n = 0.1 rps
RANGE: n = 0-9.99 rps
VALID:
RESPONSE: a**GHF** ***GFH**n[cr]
TYPE: Motion
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command set the velocity for the final approach in the go home sequence. The value is stored in nonvolatile memory.

SEE ALSO: **GH, OS, GA, GHA, GHV**
EXAMPLE:

<u>Command</u>	<u>Description</u>
> GHF.1	The velocity of the final approach of the next Go Home move will be .1 rps
> GH2	Execute Go Home at 2 rps in CW direction.

GHV

NAME: Go Home Velocity
SYNTAX: <a>**GHV**n
DEFAULT: n = ± 10
RANGE: n = ± .01 - 99.99
VALID:
RESPONSE: a**GHV** ***GHV**n[cr]
TYPE: Setup
ATTRIBUTES: Buffered, Device Address Optional, Savable in Sequence
DESCRIPTION: This command sets the velocity and direction to be used for remote homing or for subsequent Go Home (**GH**) commands. This command is stored in nonvolatile memory. When you turn on the remote Go Home input, or issue a Go Home (**GH**) command over RS-232C interface, the motor will start moving in the direction and velocity specified by **GHV** command.

The acceleration of homing process can be defined using the **GHA** command. The remote Go Home input must be specified using the **IN** command.

SEE ALSO: **GH, GHF, GHA, IN, INL, OS**
EXAMPLE:

<u>Command</u>	<u>Description</u>
> GHA10	Sets Go Home acceleration to 10 rps ²
> GHV2	Sets Go Home velocity to 2 rps
> IN1S	Sets input 1 as the remote Go Home input
> INL0	The remote go home will be executed when the Go Home input is low (closed)
> GH	Go home in the direction and velocity specified by the GHV command, using the GHA Acceleration