



## L-Series

# Information and User Manual



## Important Information When Using Our Products

Please note that Ultra Motion's commercial off-the-shelf (COTS) products are not intended for use in critical applications where failure of the product may cause bodily harm or death. Please consider the following information when designing our products into your system.

### Performance

All commercial off-the-shelf products manufactured by Ultra Motion are designed to meet the performance specifications we publish in the product's manual. All life related data is provided as reference only and does not take into account application specific factors that can have significant impacts to the overall life of the product. Application specific factors can include: design loads, transient loads (shock, vibration, inertia), speed, environmental stresses (temperature, contamination), etc. Due to the fact that application specific factors can greatly affect the product's life, it is not possible to provide a generalized Mean Time Before Failure (MTBF). It is the customer's responsibility to determine the suitability of the product for their particular application.

### Software

L-Series actuators have a built-in controller and are shipped with the latest release of controller firmware. The controller firmware is changed from time to time to add features, fix undesired behavior, or change how the controller operates. We do our best to thoroughly test each firmware release, but we do not guarantee that the controller firmware will be free of software problems that may cause undesirable or unpredictable behavior. It is extremely important that you test the actuator for your application and do not use the actuator in applications where the failure or unpredictable operation of the actuator may result in injury or death.

### Change Control

Commercial off the shelf products are subject to changes that do not affect form, fit, or function. These changes can include the use of different PCB components, internal part revisions, suppliers, firmware, coloration, etc. Ultra Motion has the ability to track and manufacture version locked designs if your project has specific change control requirements. In a version locked design, the customer will be notified before any changes are made to their product.

### Quality Control

Ultra Motion actuators are manufactured under our internal quality management system. 100% of the product we manufacture goes through a complete performance QC inspection before leaving our facility. Documented results of QC records are available to all customers.

## Safety Information

**IMPORTANT:** Read this manual before installing and operating the Ultra Motion L-Series Servos. Failure to read this section can result in personal harm or damage to the product.

### Safety Disclaimer

The L-Series is intended to be a subcomponent of a larger piece of machinery or automated system. This section is not intended to provide the safety guidelines for the entire machine or system that the servo is installed into. It is the responsibility of the purchaser or system designer to assess the risks and safety requirements of the end application they are designing.

### Safety Warnings

- Once powered, the servo is capable of rapid motion and can produce large amounts of force. Always ensure that safe clearances from people and equipment are maintained before applying power.
- While the servo operates on low voltage (10 to 36 VDC recommended), you must still use caution when handling and working around the actuator to avoid electrical shock.
- The motor of the actuator can become very hot, especially at high current draws. Take adequate time to cool before handling, and provide adequate ventilation for cooling of this device.

### Safety Notifications

As you read through the manual, you will notice certain safety notifications that indicate other important safety related information.



# Contents

<b>IMPORTANT INFORMATION WHEN USING OUR PRODUCTS</b> .....	<b>2</b>
<b>CONTENTS</b> .....	<b>3</b>
<b>LIST OF TABLES</b> .....	<b>4</b>
<b>LIST OF FIGURES</b> .....	<b>4</b>
<b>REVISION HISTORY</b> .....	<b>5</b>
<b>L-SERIES OVERVIEW</b> .....	<b>6</b>
<i>Typical Applications</i> .....	6
<i>Features</i> .....	6
<i>Product Number Structure</i> .....	7
<i>L2 Specifications</i> .....	9
<i>L2 Performance Graphs</i> .....	11
<i>L2-Series Electrical Interface</i> .....	13
<i>L2-Series Dual Redundant Electrical Interface – Secondary Connector</i> .....	14
<i>L2 Dimensions</i> .....	15
L-SERIES ENABLE INPUT.....	16
TEMPERATURE DERATING.....	17
<b>L-SERIES CONTROLLER OVERVIEW</b> .....	<b>18</b>
<i>Configuration Overview</i> .....	18
<i>L-Series Control Overview</i> .....	19
<b>L-SERIES CAN 2.0B PROTOCOL</b> .....	<b>22</b>
<i>L-Series CAN Behavior/Configuration</i> .....	22
<i>L-Series CAN Telemetry</i> .....	24
<i>L-Series Event Based Asynchronous CAN Messages</i> .....	27
<b>L-SERIES BINARY SERIAL CONTROL PROTOCOL</b> .....	<b>29</b>
<i>L-Series BSC Behavior/Configuration</i> .....	29
<b>L-SERIES RS-485 SERIAL COMMAND LINE INTERFACE</b> .....	<b>34</b>
<i>L-Series Serial CLI Command Details</i> .....	36
<i>L-Series Configuration Variables</i> .....	41
<b>TERTIARY ENCODER OVERVIEW</b> .....	<b>53</b>
<b>REDUNDANT OPERATION</b> .....	<b>58</b>
<i>Methods of Operation</i> .....	58
<i>Operating Limits and Thermal Management</i> .....	59
<i>Tertiary Sensor Voting</i> .....	59
<i>Dynamic Performance</i> .....	59
<i>Dual Redundant Controllers - Default Configurations</i> .....	60
<b>GENERAL ACTUATOR DESIGN GUIDELINES</b> .....	<b>61</b>
<b>L-SERIES FIRMWARE UPDATE</b> .....	<b>63</b>
<b>UNIT CONVERSIONS</b> .....	<b>64</b>
DEFINITIONS.....	64
EQUATIONS.....	64
<b>CONTACT INFORMATION</b> .....	<b>65</b>

## List of Tables

TABLE 1: L2M SPECIFICATIONS .....	9
TABLE 2: L2D - DUAL REDUNDANT SPECIFICATION.....	9
TABLE 3: L2D – DUAL REDUNDANT TERTIARY SENSOR SPECIFICATION .....	9
TABLE 4: L2 APPROXIMATE MASS.....	9
TABLE 5: L2 MECHANICAL LOAD LIMITATIONS .....	10
TABLE 6: DYNAMIC PERFORMANCE CAPABILITIES .....	10
TABLE 7: L2-SERIES STANDARD PINOUT.....	13
TABLE 8: L2-SERIES DUAL REDUNDANT – SECONDARY STANDARD PINOUT.....	14
TABLE 9: COMMAND OPTIONS FOR L-SERIES ACTUATOR CONTROL.....	19
TABLE 10: CONTROL WORD .....	19
TABLE 11: SYSTEM ERRORS REGISTER BIT DESCRIPTIONS.....	20
TABLE 12: SYSTEM WARNINGS REGISTER BIT DESCRIPTIONS.....	20
TABLE 13: STATUS REGISTER BIT DESCRIPTIONS .....	21
TABLE 14: TELEMETRY OPTIONS FOR THE CAN 2.0B PROTOCOL USED IN TXNDATA.....	26
TABLE 15: ASYNCHRONOUS MESSAGE FORMAT .....	27
TABLE 16: WARNING EVENT CODES, CAUSES, AND DATA FIELDS .....	28
TABLE 17: MISCELLANEOUS EVENT CODES, CAUSES, AND DATA FIELDS.....	28
TABLE 18: BSC COMMAND PACKET FORMAT.....	29
TABLE 19: BSC COMMAND CODE LIST .....	29
TABLE 20: BSC RESPONSE PACKET FORMAT .....	30
TABLE 21: BSC RESPONSE PACKET ERROR CODE LIST .....	30
TABLE 22: SERIAL CLI COMMANDS LIST .....	35
TABLE 23: CONFIGURATION VARIABLE REFERENCE LIST .....	42
TABLE 24: TERTIARY SENSOR - SYSTEM ERRORS REGISTER BIT DESCRIPTIONS.....	53
TABLE 25: TERTIARY SENSOR - SYSTEM WARNINGS REGISTER BIT DESCRIPTIONS .....	53
TABLE 26: TERTIARY SENSOR - STATUS REGISTER BIT DESCRIPTIONS.....	54
TABLE 27: TERTIARY SENSOR - TELEMETRY OPTIONS FOR THE CAN 2.0B PROTOCOL USED IN TXNDATA .....	55
TABLE 28: TERTIARY SENSOR - SERIAL CLI COMMANDS LIST .....	56
TABLE 29: TERTIARY SENSOR - CONFIGURATION VARIABLE REFERENCE LIST .....	57
TABLE 30: DUAL REDUNDANT – PRIMARY AND SECONDARY DEFAULT CONFIGURATION VARIABLES .....	60
TABLE 31: FIRMWARE UPDATE COMMAND LINE OPTIONS .....	63
TABLE 32: UNIT CONVERSION DEFINITIONS .....	64
TABLE 33: UNIT CONVERSION EQUATIONS .....	64

## List of Figures

FIGURE 1: L2 PERFORMANCE GRAPHS FOR VARIOUS CONFIGURATIONS .....	12
FIGURE 2: S500DZ-12-35PN SERIES-5 D38999 RECEPTACLES.....	14
FIGURE 3: L2M-SERIES DIMENSIONAL DRAWING, 3.125” STROKE, -B6M0E6*.....	15
FIGURE 4: L2D-SERIES DIMENSIONAL DRAWING, 3.125 INCH STROKE, -B0M0E5.....	15
FIGURE 5: L-SERIES OPTICALLY ISOLATED DIGITAL INPUT SCHEMATIC .....	16
FIGURE 6: DERATING FACTOR FOR MAXIMUM CONTINUOUS LOAD $CT$ AS A FUNCTION OF TEMPERATURE .....	17

## Revision History

Revision	Date	Details
A.01	8/19/2024	Initial Release
A.02	8/30/2024	Details added throughout
B.01	1/23/2025	Added Binary Serial Control Protocol Details
C.01	2/27/2025	Dual Redundant L-Series Information Added
C.02	7/16/2025	Added Firmware update instructions, p/n modification options

## L-Series Overview

The L-Series Linear Servo Actuators are advanced electromechanical actuators equipped with integrated brushless DC control electronics, CAN 2.0B and RS-485 serial communications, and our Phase Index® contactless absolute position feedback. The L-Series design focuses on dynamic operation in the most demanding environments, utilizing durable mechanical elements, top-tier materials, and best-in-class components.

The L-Series Linear Servo Actuators are powered by a high power-density Brushless DC (BLDC) motor, which is paired with a planetary gearhead to generate high torques in a compact package. This motor and gearhead assembly transmits power to a precision ground and preloaded ballscrew. Duplexed angular contact bearings handle all thrust loads and provide zero axial backlash for increased mechanical reliability and control authority.

The L-Series' design lends itself well to a dual redundant implementation with redundant motors, control electronics, and a triplicated absolute position sensor for majority voting. Our redundancy concept provides complete control over the redundant operation to the user with no arbitration, voting, or redundancy decisions made by the actuator. There is no internal communication between the redundant controllers or the triplicated sensor, they are 100% electrically isolated. The user's host can choose to use the redundant actuator in a cold-standby or load sharing arrangement. CAN 2.0B and RS-485 serial communication allow for complete control over the actuator's position and phase currents while also providing detailed telemetry regarding the actuator's health and state. An optically isolated enable input is provided as a hardware disable that removes power from the motor bridge when inactive, bypassing the onboard control electronics completely.

### Typical Applications

---

Aerospace, defense, industrial, and maritime applications requiring:

- Robust, high performance linear servo actuation in harsh environments
- High power density for small space envelopes
- Zero axial backlash operation for increased control authority, stiffness, and reliability
- Integrated BLDC control with CAN and serial command protocols
- Full redundancy for single electrical fault tolerance

### Features

---

- Peak forces up to 1,200 lbf
- Peak speeds up to 5 in/s
- 1.75", 3.125", and 7.5" strokes standard
- Preloaded axial load path for zero backlash
- Contactless, high resolution absolute position feedback
- Built-in brushless DC control electronics with CAN 2.0B and RS-485 serial control modes
- 10 to 36 VDC operating voltage range (62 VDC absolute maximum)
- -40°C to +100°C operating temperature range
- IP67 Sealing, all joints O-ring sealed, spring energized PTFE shaft seal
- All faying surfaces chem-filmed for continuous electrical enclosure and EMI/EMC protection
  - PCBA per IPC-A-610 Class 3, conformally coated with Parylene-C
- Series-5 D38999 Connectors [Series-3 D38999 connectors optional]
- Dual Redundant versions available for single electrical fault tolerance

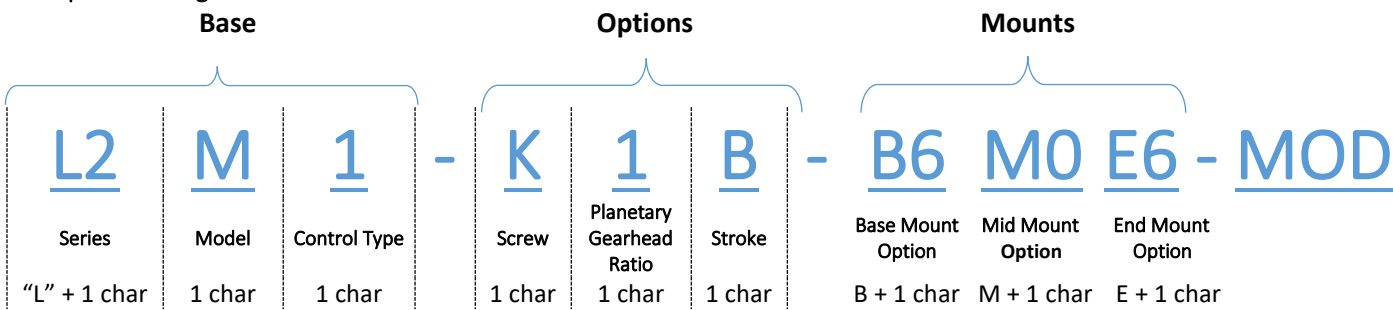
## Product Number Structure

The ordering code of the L-Series Linear Servo, also known as a model number, represents all options selected for that specific model. You may determine the ordering code of a Linear Servo model based on the information in this section. The ordering code utilizes characters which represent the series, model, control type, and options for the actuator you are specifying. The characters and options available are elaborated further in this section.

Use the following diagram to help determine the individual features called out within the ordering code. Further in this section, a full description of all available options can be found further in this section.

This example ordering code given here describes an L2-Series “Linear Servo”

Example Ordering Code L2M1 – K1B – B6M0E6 – MOD



### Series

Code	Characteristics
<b>L1</b>	1.02” [26 mm] width, 200 lbf peak
<b>L2</b>	1.44” [36.6 mm] width, 1,250 lbf peak

### Models

Code	Characteristics
<b>M</b>	Standard model
<b>D</b>	Dual redundant – 2x motors, 2x controllers, 3x absolute position sensors

### Controller Type

Code	Control	Notes
<b>1</b>	CAN 2.0B and RS-485 serial with enable input	-

### Screw Options

Code	Description	Notes
<b>H</b>	2 mm lead ballscrew, ground, light preload	-
<b>J</b>	3 mm lead ballscrew, ground, light preload	L2 only
<b>K</b>	4 mm lead ballscrew, ground, light preload	L2 only

### Planetary Gearhead Ratio

Code	Description	Notes
<b>1</b>	11:1 (absolute reduction 87/8)	-
<b>2</b>	16:1 (absolute reduction 261/16)	-
<b>3</b>	24:1 (absolute reduction 957/40)	-

### Stroke Options

Code	Description	Notes
<b>A</b>	1.75 Inch Stroke	-
<b>B</b>	3.125 Inch Stroke	-
<b>D</b>	7.5 Inch Stroke	4 mm lead ballscrew L2 only

### Base Mounting Options

Code	Description	Notes
<b>B0</b>	No mount	See "L2 Dimensions" for interface overview
<b>B6</b>	5/16" spherical bearing base mount 17-4PH housing/race, PH13-8Mo ball, PTFE liner	Ships attached in the orientation shown in "Figure 3: L2M-Series dimensional drawing, 3.125" stroke, -B6M0E6*"
<b>B6R</b>	B6 Mount installed with 90° rotation to standard	5/16" spherical bearing axis crosses connector

### Mid-Mounting Options

Code	Description	Notes
<b>M0</b>	No Mid Mount option	-

### Shaft Mounting Options

Code	Description	Notes
<b>E5</b>	5/16-24 UNJF threaded hole, 0.625" depth, 2x MS14198 keyslots	-
<b>E6</b>	5/16" Spherical Rod End, 17-4PH housing/race, 440C ball, PTFE liner NAS1193K5C locking device and NAS509-5C jam nut	Rod end ships unattached

### Modifications

Code	Description	Notes
<b>VAC</b>	Modified for operation in vacuum, non-hermetic sealing Contact Ultra Motion Engineering for additional information or vented options	L2 only Connectors switched to D38999/20FB35PN and D38999/20FC35PN by default.
<b>3F</b>	D38999/20FB35PN and D38999/20FC35PN [L2D] connectors	Electroless Nickel Series-3 D38999's
<b>3Z</b>	D38999/20ZB35PN and D38999/20ZC35PN [L2D] connectors	Black Zinc-Nickel Series-3 D38999's
<b>3T</b>	D38999/20TB35PN and D38999/20TC35PN [L2D] connectors	Nickel-PTFE (Duralon) Series-3 D38999's

## L2 Specifications

### L2M Specifications

Operating Voltage	10 to 36 VDC
Absolute Maximum Voltage	62 VDC
Operating Temperature Range	-40°C to +100°C
Standard Stroke Lengths	1.75", 3.125", 7.5"
Nominal Backlash	0 (Preloaded)
Communication Protocols	CAN 2.0B, RS-485 Serial
Environmental Sealing	IP67

Table 1: L2M Specifications

### L2D Specifications

Operating Voltage (Primary and Secondary Controllers)	10 to 36 VDC
Absolute Maximum Voltage	62 VDC
Operating Temperature Range	-40°C to +100°C
Standard Stroke Lengths	1.75", 3.125", 7.5"
Nominal Backlash	0 (Preloaded)
Communication Protocols	CAN 2.0B, RS-485 Serial
Environmental Sealing	IP67

Table 2: L2D - Dual Redundant Specification

### L2D - Tertiary Phase Index Absolute Sensor Specifications

Operating Voltage	10 to 36 VDC
Absolute Maximum Voltage	62 VDC
Operating Temperature Range	-40°C to +100°C
Communication Protocols	CAN 2.0B, RS-485 Serial
Nominal Power Consumption	1.0 Watts

Table 3: L2D – Dual Redundant Tertiary Sensor Specification

### L2 - Mass

Stroke Length	L2M	L2D
1.75 Inch Stroke	2.8 lbm	3.8 lbm
3.125 Inch Stroke	3.1 lbm	4.1 lbm
7.5 Inch Stroke	3.6 lbm	4.6 lbm

Table 4: L2 Approximate Mass

## L2 Mechanical Load Limits

Actuator mechanical load limitations are set by the Brinelling load of the ballnut, or by the peak acceptable torque at the planetary gearhead. The L10 dynamic load ratings of the ballscrew and duplexed angular contact bearings are provided for MTBF estimation based on equivalent loads and inches travelled per hour of operation. Contact Ultra Motion engineering for additional information.

<b>Ballscrew lead</b>	<b>L2 Mechanical Load Limitation</b>	<b>Ballscrew Dynamic Load Rating L10 – 1,000,000 Inches</b>	<b>Thrust Bearing Dynamic Load Rating L10 – 1,000,000 Inches</b>
2 mm lead ballscrew	1,250 lbf	380 lbf	640 lbf
3 mm lead ballscrew	1,050 lbf	890 lbf	735 lbf
4 mm lead ballscrew	785 lbf	675 lbf	810 lbf

Table 5: L2 Mechanical load limitations

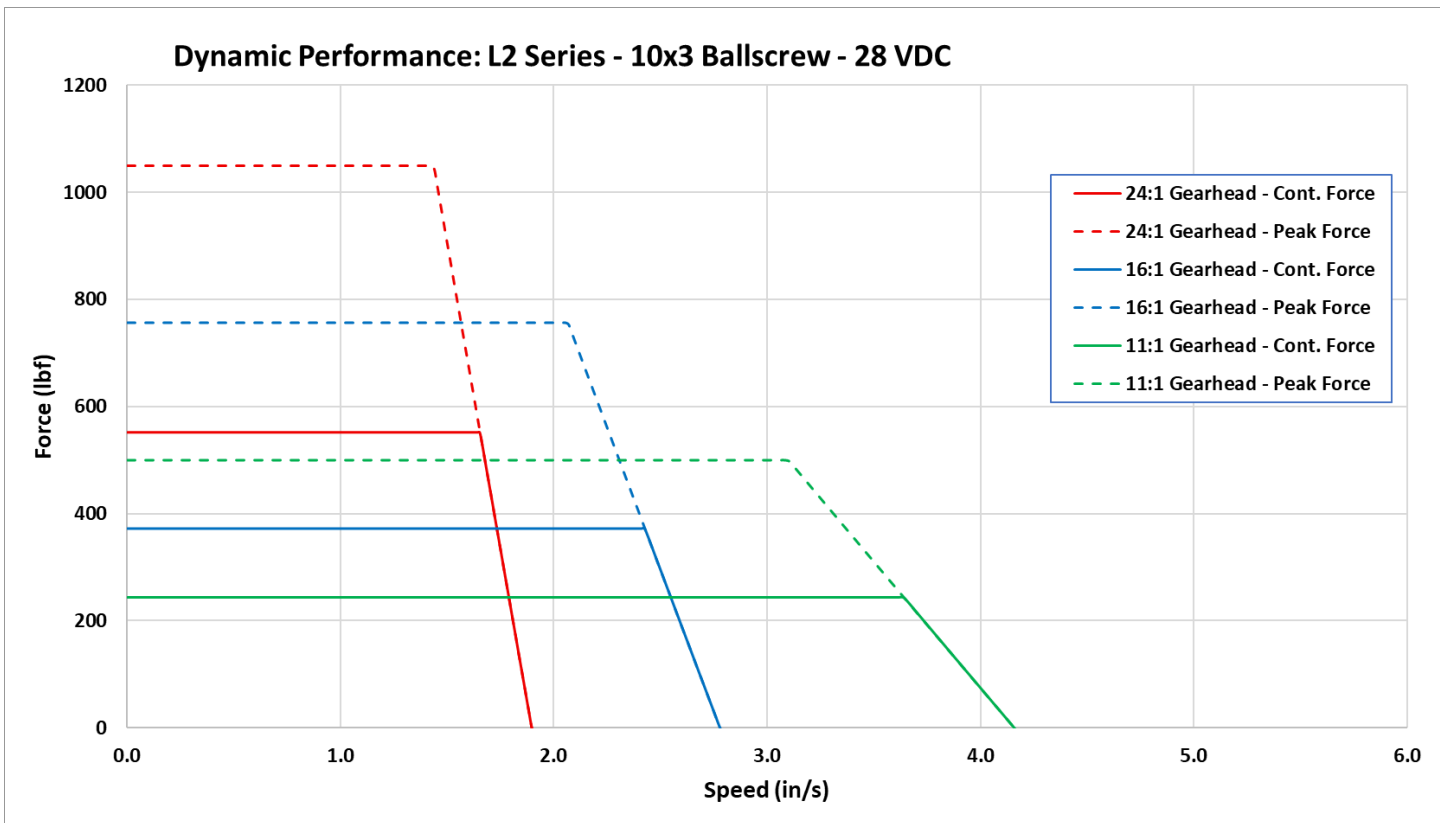
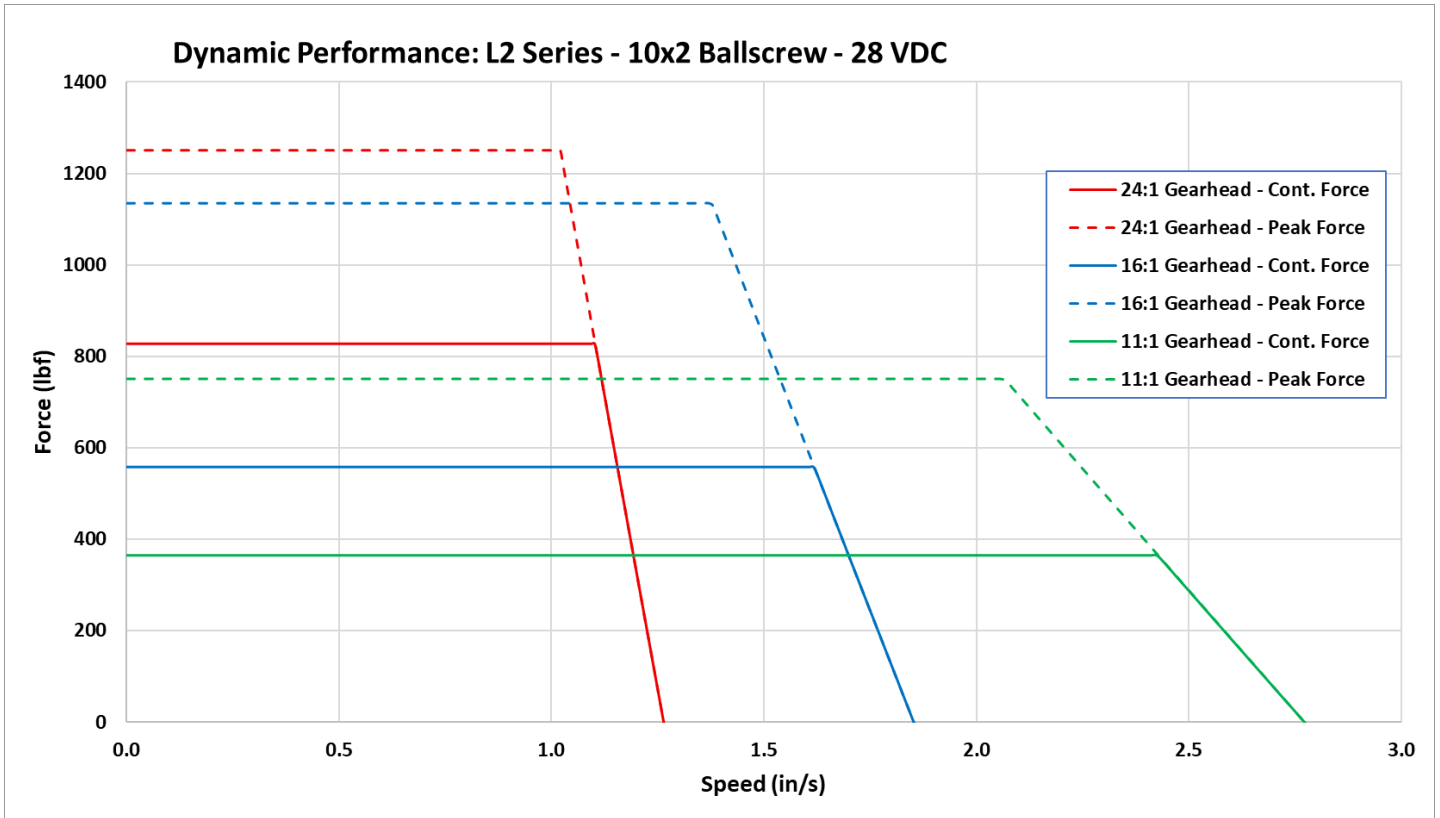
## L2 Dynamic Performance

The following table defines the thermally limited continuous force capability of the actuator with one motor operating assuming a 22°C ambient environment and natural convection. Improved heat transfer environments will lead to increased continuous force capability. The peak load duration will depend on magnitude, and heat transfer environment, with peak capability limited to 2x continuous or the mechanical load limit.

The no-load speed provided is for a 28 VDC bus voltage.

	<b>11:1 Planetary Gearhead</b>	<b>16:1 Planetary Gearhead</b>	<b>24:1 Planetary Gearhead</b>
	<b>Cont.   Peak   No Load Speed</b>	<b>Cont.   Peak   No Load Speed</b>	<b>Cont.   Peak   No Load Speed</b>
<b>2 mm lead ballscrew</b>	365 lbf   750 lbf   2.70 in/s	550 lbf   1,130 lbf   1.75 in/s	830 lbf   1,250 lbf   1.20 in/s
<b>3 mm lead ballscrew</b>	245 lbf   500 lbf   4.00 in/s	370 lbf   750 lbf   2.75 in/s	550 lbf   1,050 lbf   1.75 in/s
<b>4 mm lead ballscrew</b>	185 lbf   375 lbf   5.25 in/s	275 lbf   565 lbf   3.50 in/s	415 lbf   785 lbf   2.40 in/s

Table 6: Dynamic Performance Capabilities



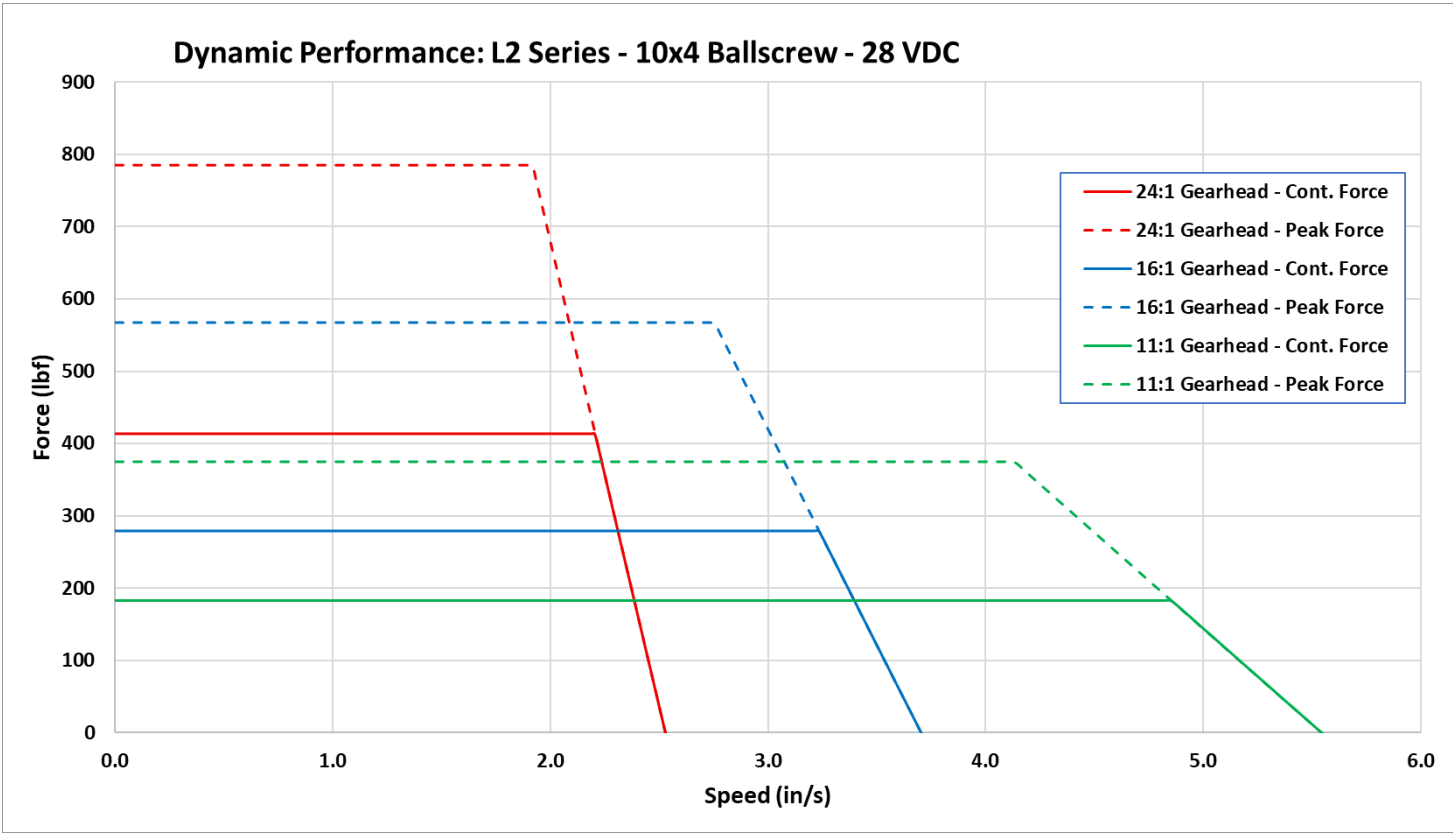
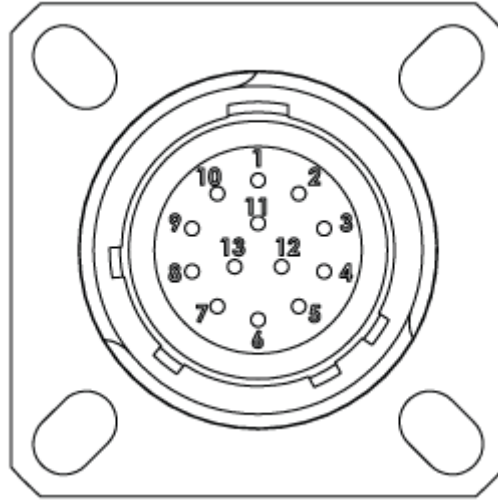


Figure 1: L2 performance graphs for various configurations

## L2-Series Electrical Interface



**WARNING:** The electrical interface and pinouts vary between the actuator controller models. Using an improperly wired cable **will** damage the actuator. Always be aware of your controller and its electrical interface before using the actuator.



S500DZ-10-35PN Series-5 D38999 Receptacle

	Pin No.	Function
Power and Signal	1	GND
	2	EN+
	3	EN-
	4	RS-485B
	5	CAN_H
	6	CAN_L
	7	GND
	8	V+
	9	V+
	10	GND
	11	GND
	12	RS-485A
	13	GND
	SHELL	CHASSIS

Table 7: L2-Series standard pinout

\*The Series-5 D38999 connectors are not interchangeable with Series-III D38999 connectors. Series-III connectors are available as an option

\*\*Contact Ultra Motion engineering if alternative pinouts, connectors, parallelized CAN signals, or other electrical modifications are required in your application.

## L2-Series Dual Redundant Electrical Interface – Secondary Connector



**WARNING:** The electrical interface and pinouts vary between the actuator controller models. Using an improperly wired cable **will** damage the actuator. Always be aware of your controller and its electrical interface before using the actuator.

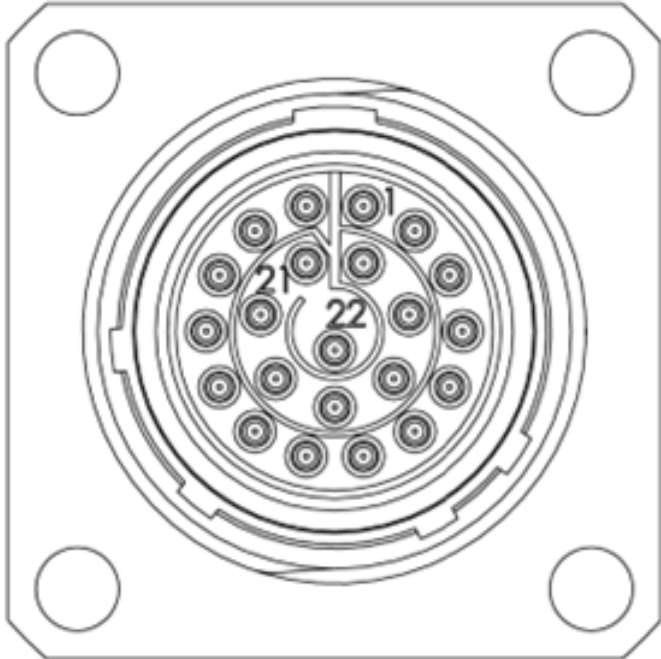


Figure 2: S500DZ-12-35PN Series-5 D38999 Receptacles

	Pin No.	Function
S500DZ-12-35PN Secondary and Tertiary Power and Signal	1	ENC_V+
	2	ENC_CAN_L
	3	ENC_CAN_H
	4	SEC_EN+
	5	SEC_EN-
	6	SEC_CAN_L
	7	SEC_CAN_H
	8	SEC_485A
	9	SEC_GND
	10	SEC_V+
	11	SEC_V+
	12	SEC_GND
	13	ENC_485B
	14	ENC_485A
	15	ENC_GND
	16	ENC_GND
	17	SEC_GND
	18	SEC_485B
	19	SEC_GND
	20	SEC_GND
	21	ENC_GND
	22	SEC_GND
	SHELL	CHASSIS

Table 8: L2-Series Dual Redundant – Secondary standard pinout

\*The Series-5 D38999 connectors are not interchangeable with Series-III D38999 connectors. Series-III connectors are available as an option

\*\*Contact Ultra Motion engineering if alternative pinouts, connectors, or other electrical modifications are required

\*\*\* SEC\_ represents the secondary controller connections

ENC\_ represents the tertiary absolute position sensor connections

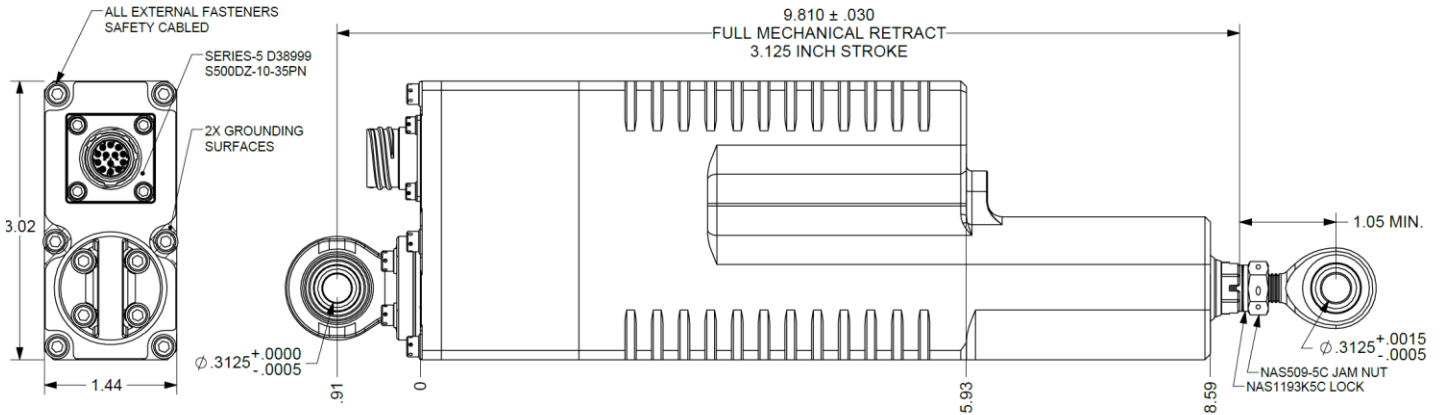


Figure 3: L2M-Series dimensional drawing, 3.125" stroke, -B6M0E6\*

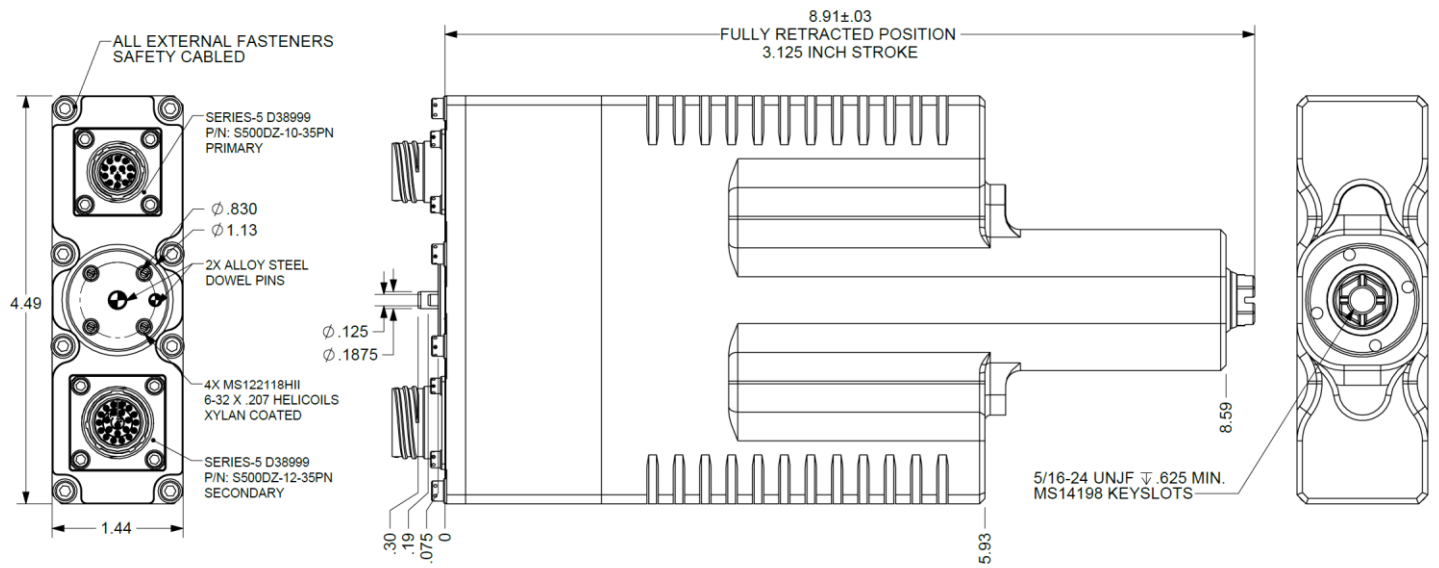


Figure 4: L2D-Series dimensional drawing, 3.125 Inch Stroke, -B0M0E5

\*E6 rod end ships unattached

Contact Ultra Motion for 3D models and detailed drawings, or for information regarding custom mechanical housings, mounts, series-3 D38999 connectors, or any other modifications required to suit your application's requirements.

## L-Series Enable Input

The L-Series [Controller Option 1] has an optically isolated digital input that is used to directly enable/disable the servo's bridge driver IC. The enable input bypasses the actuator's microcontroller, providing the user the ability to directly disable the motor and easily backdrive the actuator when equipped with a ballscrew.

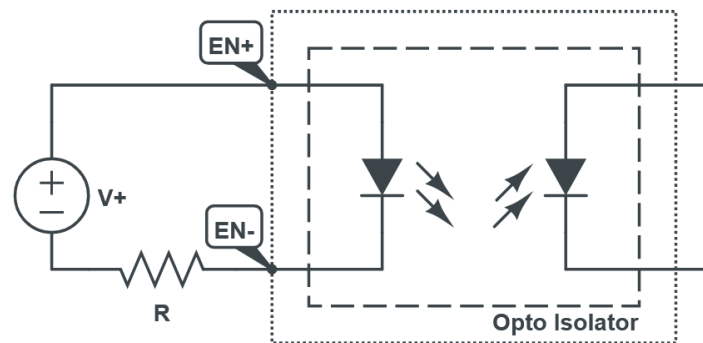


Figure 5: L-Series optically isolated digital input schematic

The controller includes a 2.7 k $\Omega$  resistor in series with the enable input. The digital inputs need to be energized with a forward current between 4 mA and 15 mA to be activated. This can be achieved with a voltage between 12 VDC and 40 VDC with a recommended nominal energizing voltage of 24 VDC.

### Behavior Notes:

- An active enable input allows power to the motor, an inactive enable input removes all power from the motor bridge via a dedicated hardware switch.
- The actuator will set the position setpoint to the current absolute position on the transition from disabled (inactive) to enabled (active).
- An inactive enable input disables the overvoltage dynamic braking functionality in hardware because the bridge driver is electrically disabled. There will be no dynamic braking on a sensed bridge voltage > 55 VDC even with `ovbOn` = 1. Consider adding shunt electronics to protect the bus against high voltage spikes due to energetic backdriving events.

## Temperature Derating

The maximum continuous load rating of the L-Series is related to the rate of heat generation by the motor and the maximum operating temperature of the actuator components. As such, when environmental temperature increases, the maximum continuous load rating of the actuator must decrease accordingly. A derating factor  $C_T$  is applied to the value for the nominal temperature maximum continuous load rating at ( $F_{nom}$ ) to determine the maximum continuous load rating ( $F_T$ ) for the actual operational ambient temperature:

$$F_T = F_{nom} \times C_T$$

The derating factor  $C_T$  is a function of the nominal temperature at which the maximum continuous load rating actuator is defined ( $T_{nom} = 22^\circ\text{C}$  for the L-Series), the maximum permissible winding temperature of the actuator ( $T_{max} = 150^\circ\text{C}$ ), and the ambient temperature of the application to which you are derating ( $T_{ambient}$ ).

$$C_T = \sqrt{\frac{T_{max} - T_{ambient}}{T_{max} - T_{nom}}} = \sqrt{\frac{150^\circ\text{C} - T_{ambient} (^\circ\text{C})}{150^\circ\text{C} - 22^\circ\text{C}}} = \sqrt{\frac{302^\circ\text{F} - T_{ambient} (^\circ\text{F})}{302^\circ\text{F} - 71.6^\circ\text{F}}}$$

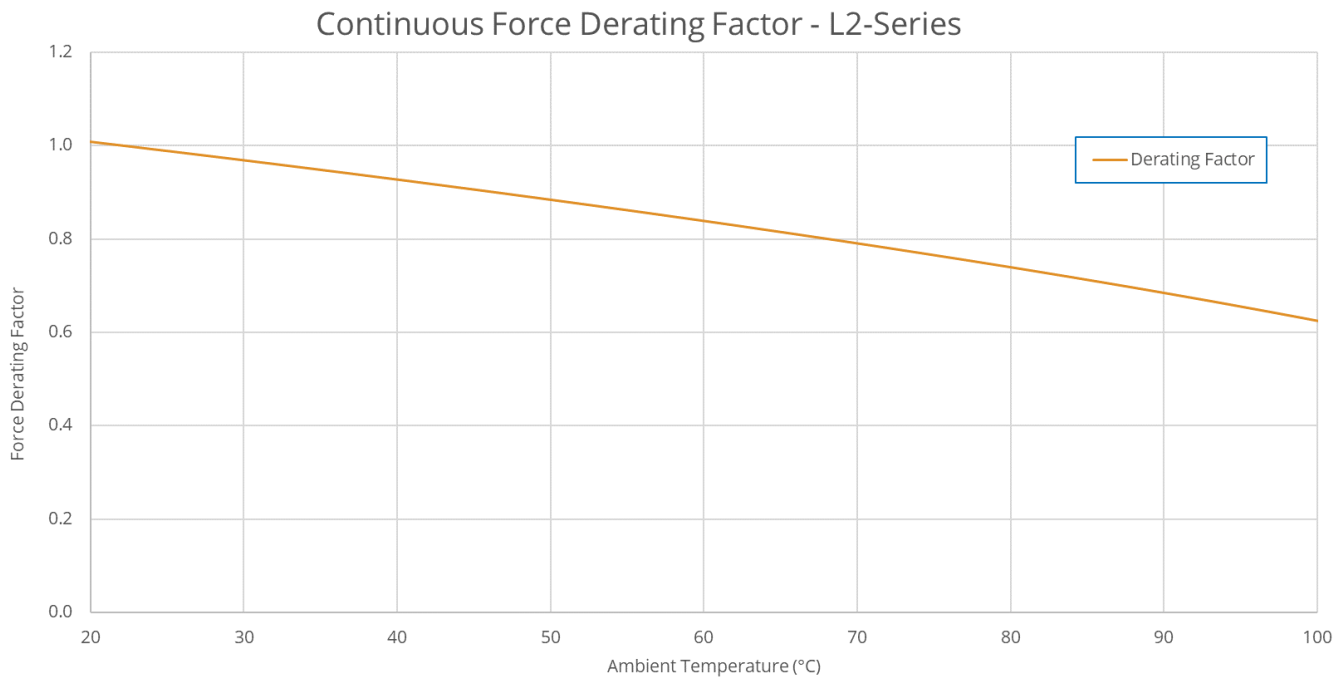


Figure 6: Derating factor for maximum continuous load  $C_T$  as a function of temperature.

## L-Series Controller Overview

This section provides an overview of how to configure the L-Series controller. All configuration variables can be read or written via the RS-485 interface and saved to non-volatile memory.

The RS-485 command line interface is half-duplex and non-addressable. The Binary Serial Control protocol is addressable.

### Configuration Overview

---

#### *Read Variable*

The current value in RAM for all configuration variables can be read via the RS-485 interface with the 'RV' command followed by the case-sensitive configuration variable name. E.g. reading the operating mode can be accomplished with the command "RV `opMode`". Issuing the "RV" command with no argument will show all the available configuration variables.

#### *Write Variable*

All configuration variables can be overwritten via the RS-485 interface with the 'WV' command followed by the case-sensitive configuration variable and the desired setting. E.g., setting the operating mode to 1 can be accomplished with the command "WV `opMode` 1"

#### *Save Configuration*

---

The WV command will only change the variable in RAM, the configuration must be saved to non-volatile memory with the "CW321" command to maintain the changes after power-cycle or reset.

## L-Series Control Overview

This section provides information on the configuration and structure of commands used to control the actuators motion in either CAN or BSC control modes.

### Operating Mode

The operating modes for this firmware include command line interface control via RS-485 serial, and CAN control. The operating mode can be set via the `opMode` configuration variable.

`opMode` = 0 enables motion control through the RS-485 serial Command Line Interface

`opMode` = 1 enables motion control through CAN 2.0B position command messages

`opMode` = 2 enables motion control through the RS-485 Binary Serial Control protocol

### Command Message Data Format

The L-Series can receive up to 8 data bytes in each position command message from the master controller. The `rxData` variable is a string with a maximum length of 8 characters where each character designates a byte of data in the command message. The actuator must be configured to receive a position command.

The data format for received position command messages is configured with the `rxData` configuration variable.

**Command Message Data Format**

Char	Description
<	Position command LSB
>	Position command MSB
(	Max motor current LSB
)	Max motor current MSB
*	Control Word
X	Ignore byte
x	Ignore byte

Table 9: Command options for L-Series Actuator Control

The default value of `rxData` = "<>" requires the user to send a message with 2 data bytes containing the LSB of the position command as the first data byte in the message, and the MSB of the position command as the second data byte. Note that setting the max motor current to a low value does not lead to a freely backdriveable actuator with the L-Series and will instead cause a dynamic braking behavior. The actuator's enable input must be de-energized, Control Word bit 0 set high, or the startup/timeout action set to COAST the motor, to allow the actuator to backdrive freely.

### Control Word

The user has control over several functions through the Control Word if `rxData` includes the "\*" designator.

**Control Word**

Bit	Bit Description
0	COAST the motor when active (0 = normal, 1 = coast) *overrides dynamic brake function
1	Dynamic Brake the motor (0 = normal, 1 = brake)
2	Gain Select (0 = default PID gains, 1 = alternate PID gains)
3 to 7	Reserved

Table 10: Control Word

## Position Interpolation

A position interpolation function provides smooth motion between each position update. The smoother motion provided by the interpolator increases mechanical reliability and the actuator response. Interpolation can be enabled/disabled with the `inEna` configuration variable.

`inEna` = 0 disables interpolation

`inEna` = 1 enables interpolation

## Default Start Position

The L-Series can be configured to drive to a default position on startup before receiving a valid command message, or on failure of the communication bus leading to a timeout event. These events can execute a trajectory move to the position defined by `defPos` with trajectory parameters defined by the `maxSpeed` and `accel` configuration variables. The default position can be set with the `defPos` configuration variable with units are in Phase Index absolute position encoder counts.

## System Errors Register

The System Errors Register contains error status bits that result in automatic entry into safe-mode (motor OFF) when set.

**System Error Register**

Error Bit	Bit Description
0	Configuration not loaded
1	Invalid Configuration in NVM
2	Bridge driver IC initialization failure
3	Phase Index encoder initialization failure
4 to 7	Reserved

Table 11: System Errors Register bit descriptions

## System Warning Register

The System Warning Register contains error status bits that do NOT result in automatic safe-mode (motor OFF).

**System Warning Register**

Warning Bit	Bit Description
0	Bad configuration block in memory, majority of redundant copies are valid
1	Invalid configuration
2	Erroneous reset
3	Bridge enable switch fault
4	Humidity/temperature sensor error
5	FRAM error
6	UART error
7	Phase Index sensor communication error
8	Bridge driver IC fault
9	Hall sensor error
10	Phase error above threshold
11	Phase Index QEI error
12	Phase Index encoder diagnostic error
13	Voltage high (> 55 VDC)
14	CAN initialization error
15	CAN transmit error

Table 12: System Warnings Register bit descriptions

## System Status Register

The status register bytes contain bits that represent different aspects of the L-Series health and state. The configurable telemetry options include latched and non-latched versions of the individual status bytes.

Status Byte	Bit	Status Bit #	Description
0	0	0	Nominal 3.3 V bus
0	1	1	Nominal 5.0 V bus
0	2	2	Nominal 7.5 V bus
0	3	3	Bridge voltage > Input voltage (regen occurring)
0	4	4	Bridge voltage less than minimum (9 VDC)
0	5	5	Bridge voltage greater than maximum (55 VDC)
0	6	6	Input voltage less than minimum (9 VDC)
0	7	7	Input voltage greater than maximum (55 VDC)
1	0	8	Safe Mode, MOTOR OFF
1	1	9	State of the optically isolated enable input
1	2	10	Bridge is active
1	3	11	Bridge driver IC fault
1	4	12	Bridge enable switch fault
1	5	13	Motor current demand capped at maxCurr
1	6	14	Motor current feedback greater than ovCurr
1	7	15	Trajectory move is active
2	0	16	System error register is non-zero
2	1	17	System warning register is non-zero
2	2	18	Temperature at PCB is greater than ovTemp value
2	3	19	Temperature at PCB is less than unTemp value
2	4	20	Humidity at PCB greater than ovHumd value
2	5	21	Phase Index health status (0 = healthy)
2	6	22	Encoder phase error warning limit exceeded
2	7	23	Phase Index to QEI difference exceeded
3	0	24	Absolute position less than or equal to rPos
3	1	25	Absolute position greater than or equal to ePos
3	2	26	Absolute position less than spMin
3	3	27	Absolute position greater than spMax
3	4	28	Absolute position less than posLe
3	5	29	Absolute position greater than posGr
3	6	30	Actuator stopped
3	7	31	Direction of travel is extend
4	0	32	CAN RX error counter > 0 and < 127
4	1	33	CAN TX error counter > 0 and < 127
4	2	34	CAN RX Warning
4	3	35	CAN TX Warning
4	4	36	CAN RX error passive state
4	5	37	CAN TX error passive state
4	6	38	CAN TX Bus Off
4	7	39	CAN TX error flag
5	0	40	CAN RX Timeout Flag
5	1	41	CAN position command capped low at pMin
5	2	42	CAN position command capped high at pMax
5	3	43	Overvolt braking is active
5	4	44	Dynamic braking off (0 = brake on, 1 = brake off)
5	5-7	45-47	Reserved

Table 13: Status Register bit descriptions

## L-Series CAN 2.0B Protocol

The following reference document details Ultra Motion's controller and firmware for the L-Series Linear Servo Actuators

### L-Series CAN Behavior/Configuration

---

This section provides an overview of the L-Series CAN protocol configuration options and behavior. Note that there are no built-in terminating resistors.

#### CAN Baud Rate

---

The L-Series' CAN baud rate is definable with the `CANspd` variable. Values of 0 to 7 are acceptable and defined as follows:

<code>CANspd</code> =	
0:	1 Mbps
1:	500 Kbps
2:	250 Kbps
3:	125 Kbps
4:	100 Kbps
5:	50 Kbps
6:	20 Kbps
7:	10 Kbps

#### CAN Message Type

---

The firmware will support either 11-bit (standard) or 29-bit (extended) identifiers, settable with the `CANext` variable.

`CANext` = 0: Standard message type (11-bit identifier)

`CANext` = 1: Extended message type (29-bit identifier)

#### rxID

---

`rxID` is the actuator's network address and is used along with `rxMask` to filter incoming CAN command messages. For acceptance, the message's identifier must match `rxID` where each corresponding bit in `rxMask` is set to '1'. If `CANext` = 0, only the lower 11 bits of `rxID` and `rxMask` are examined (standard message type). If `CANext` is 1, the lower 29 bits of `rxID` and `rxMask` are examined (extended message type). Configuration is accomplished with the `rxID` variable. `rxID` can be any value from 0x00000000 to 0x1FFFFFFF.

#### rxMask

---

`rxMask` is used along with `rxID` to filter incoming CAN command messages. For acceptance, the message's identifier must match `rxID` where each corresponding bit in `rxMask` is set to '1'. If a bit in `rxMask` is set to '0', the corresponding bit of an incoming can message is matched for acceptance regardless of its value. If `rxMask` is set to 0x00000000, CAN messages with any identifier will be accepted. If `rxMask` is set to 0x1FFFFFFF, only CAN messages with an identifier that exactly matches the relevant bits of `rxID` will be accepted. If `CANext` = 0, only the lower 11 bits of `rxID` and `rxMask` are examined (standard message type). If `CANext` is 1, the lower 29 bits of `rxID` and `rxMask` are examined (extended message type). Configuration is accomplished with the `rxMask` variable. `rxMask` can be any value from 0x00000000 to 0x1FFFFFFF.

### *CAN Interpolation Period*

---

The interpolation period can be set with the `canIvl` variable. This variable is set in 1 ms increments and should be set equal to the expected position update interval from the master controller. If the update rate cannot be matched exactly, setting the interpolation interval slightly larger than the expected update interval is recommended. The default value of `canIvl` is 50. This value is equal to a 50 ms update interval (20 Hz update frequency).

### *Received CAN Message Timeout*

---

The received message timeout period can be set with the `canTO` configuration variable. This variable is set in 1 ms increments. The default value is 1250 (1.25 seconds). When a timeout occurs, the appropriate status register bit (bit 40) will become active and the actuator's timeout behavior (set by `cTOact`) will initiate. The actuator will not execute the timeout action if no valid CAN command frames have been received after powerup.

### *CAN Timeout Action*

---

The L-Series can be configured for different behaviors after a CAN timeout event

- `cTOact` = 0 Holds the last valid position with the last valid max motor current command
- `cTOact` = 1 Executes a trajectory move to the default position `defPos` with the last valid max motor current
- `cTOact` = 2 COAST the motor
- `cTOact` = 3 Dynamic brake the motor
- `cTOact` = 4 Holds the last valid position with the configured `maxCurr` setting
- `cTOact` = 5 Executes a trajectory move to the default position `defPos` with the configured `maxCurr` setting

The timeout action can be set with the `cTOact` configuration variable.

The timeout action will not occur if no valid CAN command frames have been received after powerup. At least one valid CAN command frame must be received for the timeout action to take place.

### *CAN Startup Action*

---

The L-Series can be configured for different behaviors at startup before receiving the first valid CAN command frame.

- `cSUact` = 0 holds the current position
- `cSUact` = 1 executes a trajectory move to the default position `defPos`
- `cSUact` = 2 COAST the motor
- `cSUact` = 3 Dynamic brake the motor

The startup action can be set with the `cSUact` configuration variable.

## L-Series CAN Telemetry

The L-Series can send user-configurable telemetry messages over the CAN bus with a configurable ID. There are three independent telemetry messages that can be configured with separate identifiers, data, and update rates.

### Telemetry Enable

The `txEna` configuration variable is used to turn on/off the L-Series CAN telemetry transmit functionality. The least significant 3 bits correspond to the 3 telemetry messages with bit0 as tx1Ena, bit1 as tx2Ena, and bit2 as tx3Ena.

Example settings for `txEna`:

`txEna` = 0 disables telemetry

`txEna` = 6 (0b110) activates telemetry messages 2 and 3

`txEna` = 7 (0b111) activates all 3 telemetry messages

### Telemetry Interval

The rate at which the L-Series transmits the telemetry CAN message is configurable and set by editing the `txNlvl` variable. The setting is in milliseconds and has a valid range of 2 to 10,000. Care must be taken to not overload the CAN bus by setting the interval value too low. The default values are `tx1lvl` = 1000, `tx2lvl` = 2500, and `tx3lvl` = 5000.

### Telemetry Message ID

The `txNID` variable sets the CAN identifier for the transmitted CAN telemetry message. This represents the destination address of the telemetry message. Either the lower 11 bits (standard) or the lower 29 bits (extended) of `txNID` are used depending on CAN bus message type setting (`CANext`). The user can specify the ID in decimal or hex if preceded with "0x". `txNID` can be any value from 0x00000000 to 0x1FFFFFFF, the default values are `tx1ID` = 0x0000007F, `tx2ID` = 0x0000027F, `tx3ID` = 0x0000037F.

### Telemetry Message Data Format

The data bytes of each telemetry message are configurable with the `txNData` configuration variables. The `txNData` variables are strings where each character designates a variable in the telemetry message. `txNData` may be 1 to 8 characters in length, but the total number of resulting data bytes cannot exceed 8 for a single message.

**Telemetry Message Data Format**

Char	Description	Type	Num Bytes	Available via RS485
A	ID Byte	UINT8	1	Yes
B	Critical Errors Byte	UINT8	1	Yes
C	Warning Byte 0	UINT8	1	Yes
D	Warning Byte 1	UINT8	1	Yes
E	Warning Register – Clear on Broadcast	UINT16	2	Yes
F	CAN Position Command	UINT16	2	Yes
G	Position Demand	UINT16	2	Yes
H	Motor Current Demand	INT16	2	Yes
I	Motor Current Demand Limit	UINT16	2	Yes
J	Duty Cycle Out	INT16	2	Yes
K	Encoder Position Feedback	UINT16	2	Yes
L	Hall Position Feedback	UINT16	2	Yes

M	Encoder Velocity Count	INT16	2	Yes
N	Encoder Velocity Interval	UINT32	4	Yes
O	Motor Current – Instantaneous	INT16	2	Yes
P	Motor Current – Average over interval	INT16	2	No
Q	Motor Current – Minimum over interval	INT16	2	No
R	Motor Current – Maximum over interval	INT16	2	No
S	Bridge Current – Instantaneous	UINT16	2	Yes
T	Bridge Current – Average over interval	UINT16	2	No
U	Bridge Current – Minimum over interval	UINT16	2	No
V	Bridge Current – Maximum over interval	UINT16	2	No
W	Bridge Voltage – Instantaneous	UINT16	2	Yes
X	Bridge Voltage – Average over interval	UINT16	2	No
Y	Bridge Voltage – Minimum over interval	UINT16	2	No
Z	Bridge Voltage – Maximum over interval	UINT16	2	No
a	Supply Voltage – Instantaneous	UINT16	2	Yes
b	Supply Voltage – Average over interval	UINT16	2	No
c	Supply Voltage – Minimum over interval	UINT16	2	No
d	Supply Voltage – Maximum over interval	UINT16	2	No
e	Status Register Byte 0	UINT8	1	Yes
f	Status Register Byte 1	UINT8	1	Yes
g	Status Register Byte 2	UINT8	1	Yes
h	Status Register Byte 3	UINT8	1	Yes
i	Status Register Byte 4	UINT8	1	Yes
j	Status Register Byte 5	UINT8	1	Yes
k	Status Register Byte 0 – Latched High	UINT8	1	No
l	Status Register Byte 1 – Latched High	UINT8	1	No
m	Status Register Byte 2 – Latched High	UINT8	1	No
n	Status Register Byte 3 – Latched High	UINT8	1	No
o	Status Register Byte 4 – Latched High	UINT8	1	No
p	Status Register Byte 5 – Latched High	UINT8	1	No
q	Status Register Byte 0 – Latched Low	UINT8	1	No
r	Status Register Byte 1 – Latched Low	UINT8	1	No
s	Status Register Byte 2 – Latched Low	UINT8	1	No
t	Status Register Byte 3 – Latched Low	UINT8	1	No
u	Status Register Byte 4 – Latched Low	UINT8	1	No
v	Status Register Byte 5 – Latched Low	UINT8	1	No
w	PCB Temperature (2 Hz update)	UINT8	1	Yes
x	PCB Relative Humidity (0.25 Hz update)	UINT8	1	Yes
y	PCB Temp2 (0.25 Hz update)	UINT8	1	Yes
z	PCB Temperature – 32 bits (2 Hz update)	FLOAT32	4	Yes
3	PCB Humidity – 32 bits (0.25 Hz update)	FLOAT32	4	Yes
4	PCB Temp2 – 32 bits (0.25 Hz update)	FLOAT32	4	Yes
6	Position PID I-Term	INT16	2	Yes
7	Serial Number	UINT32	4	Yes
8	UART Status Register	UINT16	2	Yes
9	CAN Errors (MSB = TX errors, LSB = RX errors)	UINT16	2	Yes
+	BSC Position Command	UINT16	2	Yes

^	BSC Position Input	UINT16	2	Yes
&	CAN Position Input	UINT16	2	Yes
#	BSC Control Word	UINT8	1	Yes
~	CAN Control Word	UINT8	1	Yes
@	BSC CRC Errors	UINT16	2	Yes
\$	BSC Timeouts	UINT16	2	Yes
%	Serial TX Dropped	UINT16	2	Yes
!	opMode	UINT8	1	Yes

Table 14: Telemetry options for the CAN 2.0B protocol used in txNData

The telemetry message may include anywhere from 1 to 8 bytes of data and they can be any combination of the above options. Latched status bytes are reset after they are transmitted. Latched low bytes are reset high, and latched high bytes are reset low. Values that are average or min/max over the telemetry interval are also reset after they are transmitted. Values that are reset after they are transmitted should only be accessed from one telemetry message; either [tx1Data](#), [tx2Data](#), or [tx3Data](#), but never more than one at the same time.

Since status bits are tested one time every millisecond, a status bit event lasting less than one millisecond may not be latched.

The default values are: [tx1Data](#) = "GKPCD", [tx2Data](#) = "klmnpb", and [tx3Data](#) = "wxy".

## L-Series Event Based Asynchronous CAN Messages

---

The L-Series can be configured to transmit CAN messages on the occurrence of asynchronous events.

The asynchronous event message frames will all have 8 data bytes with the following format:

CAN Frame Data Byte	Byte Description
0	IDbyte (configuration variable, should be set to a unique value for each actuator on the bus)
1	Event Code
2-7	Event Data Bytes 0-5

Table 15: Asynchronous Message Format

### Configuring Event Messages

---

The actuator will broadcast event messages on the CAN bus with an address defined by `evntID`

There are two separate configuration variables used to enable event message broadcasts, one for system warning events, and one for non-critical “miscellaneous” events.

Warning events occur when a warning bit transitions from low to high. After a warning message is sent for a particular warning bit, a timer with the value of `evntIvl` milliseconds must expire before another message for the same bit can be re-broadcast.

Warning event messages are enabled with the `evntWrn` configuration variable. Setting a bit high in `evntWrn` enables event messages for the corresponding bit in the system warning register. The warning bit will clear after the warning event message is broadcast if the corresponding “clear” bit in the `evntWrnC` mask is high. If the clear bit is low, the warning bit will remain unchanged by the event message broadcast. If the warning bit is not cleared, the warning event will not reoccur.

The `evntMsc` configuration variable works the same way as `evntWrn` with each bit in the mask enabling a pre-defined event message. The events in `evntMsc` are not constrained by a minimum rebroadcast time as the warning events are.

Table 16 and Table 17 detail the pre-defined event messages that can be enabled with `evntWrn` and `evntMsc`:

## System Event Details

The following tables detail the possible asynchronous event messages. Please contact Ultra Motion engineering for details regarding the various event data.

Event Code	evntWrn bit	Warning Event Cause	Warning Event Data Bytes (6 bytes)
0	0	Bad configuration block in NVM	byte 0-5: unused
1	1	Configuration error	byte 0-1: 16bit Config Content Error Code
2	2	Erroneous reset	byte 0-1: 16bit Reset Cause byte 2-5: 32bit Error Address
3	3	Bridge enable switch fault	byte 0-1: 16bit Input Voltage Sense byte 2-3: 16bit Bridge Voltage Sense
4	4	Humidity sensor comm error	byte 0-1: 16bit Sensor Error Code
5	5	FRAM error	byte 0-5: unused
6	6	UART error	byte 0-1: 16bit UART Status byte 2-3: 16bit UART Error Count
7	7	Phase Index sensor comm error	byte 0-1: 16bit Primary Parity Error Count byte 2-3: 16bit Secondary Parity Error Count byte 4-5: 16bit Primary Error Bit Count
8	8	Bridge driver fault	byte 0-1: 16bit Bridge Fault Count byte 2-3: 16bit Bridge Fault Register [1] byte 4-5: 16bit Bridge Fault Register [2]
9	9	Hall sensor error	byte 1-2: 16bit Invalid Hall State Count byte 3-4: 16bit Invalid Hall Sequence Count
10	10	Phase error above threshold	byte 0-1: 16bit Phase Index Phase Error byte 2-3: 16bit Phase Index to QEI Error byte 4-5: 16bit Encoder Error Position
11	11	Phase Index encoder error	byte 0-1: 16bit Phase Index Phase Error byte 2-3: 16bit Phase Index to QEI Error byte 4-5: 16bit Encoder Error Position
12	12	Encoder diagnostic error	byte 0-1: 16bit Primary Diagnostic Register byte 2-3: 16bit Secondary Diagnostic Register byte 4-5: 16bit Encoder Error Position
13	13	Voltage high (> 55 VDC)	byte 0-1: 16bit Input Voltage Sense byte 2-3: 16bit Bridge Voltage Sense
14	14	CAN initialization error	N/A
15	15	CAN transmit error	N/A

Table 16: Warning event codes, causes, and data fields

Event Code	evntMsc bit	Miscellaneous Event Cause	Miscellaneous Event Data Bytes (6 bytes)
16	0	Normal Reset (power-on or user commanded)	byte 0: 8bit System Error Register byte 1: 8bit System Status Byte1 byte 2-5: 32bit Actuator Serial Number
17	1	System Error (motor in safe mode)	byte 0: 8bit System Error Register byte 1: 8bit System Status Byte1 byte 2-3: 16bit System Warning Register byte 4-5: 16bit Config Content Error Code
18	2	Optically Isolated Enable Input Active	byte 0-5: 8bit System Status Byte0-5
19	3	Optically Isolated Enable Input Inactive	byte 0-5: 8bit System Status Byte0-5

Table 17: Miscellaneous event codes, causes, and data fields

# L-Series Binary Serial Control Protocol

The L-Series' Binary Serial Control protocol (BSC) is a CRC protected, addressable RS-485 serial protocol with similar configuration and functionality as the CAN 2.0B protocol. The following section will detail the BSC's usage and behavior. The UART configuration is 8 data bits, no parity bit, 1 stop bit (8N1). There are no built-in terminating resistors.

## L-Series BSC Behavior/Configuration

### *bscAddr*

*bscAddr* is the actuator's network address on the RS-485 bus. For acceptance, the address byte in the RS-485 serial packet must match *bscAddr* exactly. Configuration is accomplished with the *bscAddr* variable. *bscAddr* can be any value from 1 to 128.

### *Command Packet Format*

Each command packet sent to an actuator must be configured as follows:

**BSC Command Packet Format**

Byte	Description
0	Start Byte [0xAA]
1	Actuator Address
2	Command Code
3	Data Length Byte
4...N	Data
N+1	CRC LSB
N+2	CRC MSB

Table 18: BSC Command Packet Format

### *Command Code*

The BSC protocol has four possible command types

**BSC Command Codes**

Command Code	Description
0x01	Command Line Interface Passthrough
0x02	BSC Control Command
0x03	Set Operating Mode
0x04	Read Runtime Variable

Table 19: BSC Command Code List

#### *Command Line Interface Passthrough:*

The CLI passthrough will act the same as sending commands through the command line interface when *opMode* = 0. For example, the data being sent with command code 0x01 can be used to write configuration variables "wv *maxCurr* 8000" or issue a reset command "zc321". Trajectory motion commands are not valid in this mode. Data is sent as ASCII

#### *BSC Control Command:*

The BSC control command is used to send position, motor phase current, and Control Word updates to the actuator based on the value of *rxData* (see Table 9: Command options for L-Series Actuator Control). These commands should be sent to each actuator at a regular rate, defined by the BSC protocol's configured interpolation interval *bscivl*.

### Set Operating Mode:

Command code 0x03 is used to adjust the operating mode while actively in BSC mode (`opMode = 2`)

### Read Runtime Variable:

Command code 0x04 mimics the “RR x” functionality found in the CLI and is used for querying actuators for various telemetry and health information. The response data will be in little endian format. The 0x04 command can be sent with zero data which will return zero data. The maximum command length is 64 bytes and you cannot request more than 255 bytes of response data in a single command.

### Response Packet Format

The actuator will respond to each valid command packet as follows:

Byte	Description
0	Start Byte [0x55]
1	Actuator Address
2	Response Code
3	Data Length Byte
4...N	Data
N+1	CRC LSB
N+2	CRC MSB

Table 20: BSC Response Packet Format

### Response Code

The actuator’s response code will include the command packet’s command code in the upper 4 bits along with an error code in the lower 4 bits. As an example, an invalid argument sent with the CLI passthrough code (0x01) would have the response code 0x16

### BSC Response Packet Error Codes

Error Code	Error Description
0	CMD_OK
1	CMD_ERROR_INVALID_CMD
2	CMD_ERROR_LEN_ZRO
3	CMD_ERROR_INTERNAL
4	CMD_ERROR_ARG_TOOMANY
5	CMD_ERROR_ARG_TOOFEW
6	CMD_ERROR_ARG_INVALID
7	CMD_ERROR_ARG_RANGE
8	CMD_ERROR_STRING_LONG
9	CMD_ERROR_PERMISSION_DENIED
10	CMD_ERROR_NOT_ALLOWED
11	CMD_ERROR_NOT_FOUND
12	CMD_ERROR_COND_STATUS
13	CMD_ERROR_COND_CONFIG
14	CMD_ERROR_CLI_LOCKED
15	CMD_ERROR_BUFFER_FULL

Table 21: BSC Response Packet Error Code List

### *BSC Interpolation Period*

---

The interpolation period can be set with the `bscIvl` variable. This variable is set in 1 ms increments and should be set equal to the expected position update interval from the master controller. If the update rate cannot be matched exactly, setting the interpolation interval slightly larger than the expected update interval is recommended. The default value of `bscIvl` is 50. This value is equal to a 50 ms update interval (20 Hz update frequency).

### *Received BSC Message Timeout*

---

The received message timeout period can be set with the `bscTO` configuration variable. This variable is set in 1 ms increments. The default value is 1250 (1.25 seconds). When a timeout occurs, the appropriate status register bit (bit 40) will become active and the actuator's timeout behavior (set by `bTOact`) will initiate. The actuator will not execute the timeout action if no valid BSC command frames have been received after powerup.

### *BSC Timeout Action*

---

The L-Series can be configured for different behaviors after a CAN timeout event

- `bTOact` = 0 Holds the last valid position with the last valid max motor current command
- `bTOact` = 1 Executes a trajectory move to the default position `defPos` with the last valid max motor current
- `bTOact` = 2 COAST the motor
- `bTOact` = 3 Dynamic brake the motor
- `bTOact` = 4 Holds the last valid position with the configured `maxCurr` setting
- `bTOact` = 5 Executes a trajectory move to the default position `defPos` with the configured `maxCurr` setting

The timeout action can be set with the `bTOact` configuration variable.

The timeout action will not occur if no valid BSC command frames have been received after powerup. At least one valid BSC command frame must be received for the timeout action to take place.

### *BSC Startup Action*

---

The L-Series can be configured for different behaviors at startup before receiving the first valid BSC command frame.

- `bSUact` = 0 holds the current position
- `bSUact` = 1 executes a trajectory move to the default position `defPos`
- `bSUact` = 2 COAST the motor
- `bSUact` = 3 Dynamic brake the motor

The startup action can be set with the `cbUact` configuration variable.

### *Revert to Command Line Interface*

---

A special command is provided to allow a user to manually revert back to `opMode` = 0 through a terminal program when the actuator is in BSC mode. The special command “[opmodezero]” (including the brackets) followed by a carriage return will change the operating mode to zero.

### Group Messages:

A value of 0x00 for the actuator address in a command packet will be accepted by all actuators on the serial bus.

### Receive Buffer Timeout:

The RS-485 peripheral will automatically clear the input buffer if no new bytes are received within 30 bit-times of the previously read byte. This timeout may result in dropped messages, particularly if the host is multitasking and interrupts serial communication mid-transmission.

### Cyclic Redundancy Check:

The CRC algorithm ensures data integrity by calculating a checksum based on a specific polynomial. The following steps outline the CRC-16-CCITT algorithm with the polynomial 0x1021 and an initial value of 0xFFFF used by the L-Series BSC protocol.

1. Exclude the Start Byte (0xAA) and CRC Bytes.
2. Combine the Actuator Address, Command Code, Data Length, and Data bytes.
3. Calculate the CRC on the resulting byte sequence.

### CRC16 Pseudocode:

```
function CRC16_CCITT(data[]):
    result = 0xFFFF          # Initialize result with initial value
    polynomial = 0x1021     # Define CRC-16-CCITT polynomial

    for each byte in data[]:
        result = result XOR (byte << 8) # XOR the byte shifted left by 8 bits

        for bit in 0 to 7:          # Process each bit in the byte
            if result AND 0x8000:    # Check if the MSB is 1
                result = (result << 1) XOR polynomial
            else:
                result = result << 1  # Shift result left by 1

        result = result AND 0xFFFF  # Ensure result stays within 16 bits

    return result
```

### *Example Command and Response Packets:*

---

#### **Read Runtime Variable K - Encoder Position Feedback (2 bytes):**

Command Packet: 0xAA8004014BA64F

Start Byte: 0xAA, Address: 0x80, Command Code: 0x04, Command Length: 0x01, Command Data : 0x4B [b'K'], Checksum: 0xA64F

Received Response: 2048d

Response Packet: 0x55804002000828B2

Start Byte: 0x55, Address: 0x80, Response Code: 0x40, Response Length: 0x02, Response Data: 0x0008, Checksum: 0x28B2

#### **CLI Passthrough "wv ovTemp 40.0"**

Command Packet: 0xAA80010E7776206FF7654656D702034302E30FB56

Start Byte: 0xAA, Address: 0x80, Command Code: 0x01, Command Length: 0x0E, Command Data: 0x7776206F7654656D702034302E30 [b'wv ovTemp 40.0'], Checksum: 0xFB56

Received Response: 40.0

Response Packet: 0x5580100434302E30B2F9

Start Bytes: 0x55, Address: 0x80, Response Code: 0x10, Response Length: 0x04, Response Data: 0x34302E30, Checksum: 0xB2F9

#### **Position Command to 3210d:**

Command Packet: 0xAA8002028A0C0B85

Start Byte: 0xAA, Address: 0x80, Command Code: 0x02, Command Length: 0x02, Command Data: 0x8A0C, Checksum: 0x0B85

Response Packet: 0x5580200020F1

Start Byte: 0x55, Address: 0x80, Response Code: 0x20, Response Length: 0x00, Response Data: , Checksum: 0x20F1

## L-Series RS-485 Serial Command Line Interface

The L-Series RS-485 serial interface is half-duplex, 8N1. A command line is parsed and executed after a carriage return byte is received by the UART. A command line consists of the following: 0 or more space or tab bytes, 2-byte ASCII command, 0 or more ASCII arguments preceded by 1 or more space or tab bytes, 0 or more space or tab bytes, carriage return byte.

The CLI will respond to a command line with the command's returned data, "OK" if the command was executed successfully but has no data to return, or "ERROR: " + *error type* if a command was not executed successfully.

### Command Line Interface Lock

The L-Series will start-up with the CLI locked when the `cliLock` variable is set to 1, which means no received serial commands will be executed. This is to protect the actuator from executing erroneous commands due to noise or user error. To use the command line interface the special command "LK" with the argument "unlock" is required (i.e. "LK unlock\r"). The actuator's CLI will lock after every power cycle unless the configuration variable `cliLock` is set to zero.

## L-Series Serial Commands Quick Reference Table

Command	Description	Group	Arguments
<a href="#">RV</a>	Read configuration variable	Configuration	1 or 0
<a href="#">WV</a>	Write configuration variable	Configuration	2
<a href="#">WF</a>	Write any configuration variable to flash only	Configuration	2
<a href="#">CW</a>	Save configuration settings to non-volatile memory	Configuration	1
<a href="#">CC</a>	Copy configuration block from inactive to active partition	Configuration	1
<a href="#">RR</a>	Read runtime variable	Read	1 or 0
<a href="#">SR</a>	Read status register bits	Read	2 or 0
<a href="#">RI</a>	Read reset information	Read	0
<a href="#">ZD</a>	Read bridge driver FAULT register log	Read	0
<a href="#">FS</a>	Show serial number, part number, firmware version, target PCB	Read	0
<a href="#">HE</a>	Show all available serial commands	Read	0
<a href="#">TA</a>	Trajectory move to absolute position	Motion Control	1
<a href="#">TO</a>	Trajectory move to offset position	Motion Control	1
<a href="#">TR</a>	Trajectory move to fully retracted position ( <i>spMin</i> )	Motion Control	0
<a href="#">TM</a>	Trajectory move to mid-stroke ( $(spMin+spMax)/2$ )	Motion Control	0
<a href="#">TE</a>	Trajectory move to fully extended position ( <i>spMax</i> )	Motion Control	0
<a href="#">TD</a>	Trajectory move to the default position ( <i>defPos</i> )	Motion Control	0
<a href="#">T1</a>	Trajectory move to tPos1	Motion Control	0
<a href="#">T2</a>	Trajectory move to tPos2	Motion Control	0
<a href="#">TK</a>	Interrupt current trajectory move	Motion Control	0
<a href="#">TS</a>	Stop trajectory at maximum acceleration	Motion Control	0
<a href="#">PC</a>	Set target position to current position	Motion Control	0
<a href="#">PA</a>	Set absolute target position	Motion Control	1
<a href="#">PO</a>	Set target position with an offset	Motion Control	1
<a href="#">ZR</a>	Restart actuator controller	System	1
<a href="#">LK</a>	Lock or unlock serial command-line-interface	System	1 or 0
<a href="#">FW</a>	Show firmware installed on active and inactive partitions	System	0
<a href="#">SA</a>	Swap active/inactive firmware partition	System	1
<a href="#">ZC</a>	Run actuator calibration utility	System	1
<a href="#">ZU</a>	Run serial firmware update utility	System	1

Table 22: Serial CLI Commands List

## L-Series Serial CLI Command Details

---

### RV – Read Configuration Variable(s)

This command returns the value of any of the configuration variable. Sending this command with no argument returns the value of all configuration variables.

**Usage:** RV

**Usage:** RV arg1

**arg1 Description:** Variable name (case sensitive)

**arg1 Data Type:** String

### WV –Write Configuration Variable

This command sets any configuration variable in active ram. To save the current configuration to non-volatile memory, use the **CW** command.

**Usage:** WV arg1 arg2

**arg1 Description:** Configuration variable name (case sensitive)

**arg1 Data Type:** String

**arg2 Description:** Value to write

**arg2 Data Type:** Configuration variable's data type

### WF –Write Configuration Variable to Flash Only

This command sets any configuration variable in flash only. This command does not change the configuration variable in active ram, only non-volatile memory. It is primarily used for configuring the operating mode to opMode = 2.

**Usage:** WF arg1 arg2

**arg1 Description:** Configuration variable name (case sensitive)

**arg1 Data Type:** String

**arg2 Description:** Value to write

**arg2 Data Type:** Configuration variable's data type

### CW – Write Configuration Settings to Non-Volatile Memory

This command will take all current configuration settings and write them to non-volatile memory. *This must be done if a setting was changed via a serial command, and you wish to retain this change after a restart or power-cycle.*

**Usage:** CW arg1

**arg1 Description:** Passcode to execute command

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** "321"

### CC – Copy Configuration Block from inactive to active partition

The CC command will copy the actuator configuration from the inactive partition to the active partition. This will overwrite the existing configuration block on the active partition. The command will only execute if the configuration block on the inactive partition is the same size as active partition's configuration block. This command is useful for copying the existing configuration to new firmware after a firmware update. It will only work if the configuration block layout has not changed between old firmware and new. A reset is required to load the copied configuration block after this command is executed.

**Usage:** CC arg1

**arg1 Description:** Passcode to execute command

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** "321"

### RR – Read Run-Time variable

Returns the value of value of any run-time variable. Running this command with no argument shows all of the available run-time variables. The value of some variables can only be read via CAN telemetry. See *Table 14: Telemetry options for the CAN 2.0B protocol used in txNData* for telemetry options accessible via serial. Sending multiple characters will return the requested variables in comma-separated-value format (e.g. RR ABC). Some variables can only be read via CAN telemetry.

**Usage:** RR

**Usage:** RR arg1

**arg1 Description:** One or more run-time variable character designators (see Table 5)

**arg1 Data Type:** String

### SR – Read Status Register Bits

Returns the current value of all status register bytes when the command is sent with no arguments. Returns a particular bit within the status register selected by argument 1 (status byte number) and argument 2 (bit number).

**Usage:** SR

**Usage:** SR arg1 arg2

**arg1 Description:** Status Byte Number

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** 0 to 5

**arg2 Description:** Status Bit Number

**arg2 Data Type:** Unsigned Integer

**arg2 Range:** 0 to 7

### FS – Read Actuator Details

Return the actuator's serial number, part number, firmware version ID, and target PCB.

**Usage:** FS

### HE – Show All Serial Commands

Return a list of all available serial commands and their descriptions.

**Usage:** HE

### RI – Read Reset Information

Read information about the last CPU reset.

**Usage:** RI

### ZD – Read Bridge Driver Fault Register Log

Returns the fault register log of the bridge driver IC.

**Usage:** ZD

### TA – Trajectory Move to Absolute Position

Sending this command result in a trajectory move an absolute encoder position. Trajectory moves obey the [maxSpeed](#) (maximum profile speed) and [accel](#) (profile acceleration) settings.

**Usage:** TA arg1

**arg1 Description:** Absolute encoder position

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** [spMin](#) to [spMax](#)

### TO – Trajectory Move to Offset Position

Trajectory move to the current position + offset. The resultant position must be within the bounds defined by [spMin](#) and [spMax](#)

**Usage:** TO arg1

**arg1 Description:** Relative position offset

**arg1 Data Type:** Signed Integer

**arg1 Range:** [spMin](#) < Target Position < [spMax](#)

### TR – Trajectory Move to Fully Retracted Position ([spMin](#))

Trajectory move to [spMin](#) - "Software Position Minimum" with user defined speed and acceleration.

**Usage:** TR

### TM – Trajectory Move to Mid-Stroke

Trajectory move to midpoint with user defined speed and acceleration. The midpoint is defined by  $\frac{(spMax+spMin)}{2}$

**Usage:** TM

### TE – Trajectory Move to Fully Extended Position ([spMax](#))

Trajectory move to [spMax](#) - "Software Position Maximum" with user defined speed and acceleration.

**Usage:** TE

### T1 – Preset Position 1 Trajectory Move

Initiate a trajectory move to [tPos1](#)

**Usage:** T1

### T2 – Preset Position 2 Trajectory Move

Initiate a trajectory move to [tPos2](#)

**Usage:** T2

### TD – Trajectory Move to the default Position

Trajectory move to [defPos](#) – "Default Position" with user defined speed and acceleration.

**Usage:** TD

### TS – Stop Trajectory Move at Maximum Acceleration

Stop the current trajectory at the maximum acceleration rate.

**Usage:** TS

### TK – Interrupt Current Trajectory Move

Halt the current trajectory motion at current position.

**Usage:** TK

### PA – Set Absolute Target Position

Sending this command with a valid argument will set the current position demand register resulting in an immediate move to the new position. This command writes directly to the position demand register, bypassing the trajectory generator. This leads to a full acceleration, full speed move to the new target position which ignores maximum speed and maximum acceleration settings. Users will typically use this command for maximum dynamic performance in conjunction with an external trajectory generator. Use this command with caution.

**Usage:** PA arg1

**arg1 Description:** Absolute encoder position

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** spMin to spMax

### PO – Set Target Position with an Offset

Set the target position to the current position + the offset. The resultant position must be within the bounds defined by spMin and spMax. This command writes directly to the position demand register, bypassing the trajectory generator. This leads to a full acceleration, full speed move to the new target position and ignores maximum speed and maximum acceleration settings. Use this command with caution.

**Usage:** PO arg1

**arg1 Description:** Relative position offset

**arg1 Data Type:** Signed Integer

**arg1 Range:** spMin < Target Position < spMax

### PC – Set PID Target to Current Position

Set the current position setpoint to the current absolute position value.

**Usage:** PC

### LK – Lock or Unlock Serial Command-Line-Interface

Lock or unlock the serial command line interface.

**Usage:** LK arg1

**arg1 Description:** Command string

**arg1 Data Type:** String

**arg1 Range:** “lock”, “unlock”

### FW – Show Firmware Installed on Active and Inactive Partitions

Show any firmware loaded onto the active and inactive flash memory partitions.

**Usage:** FW

### SA – Swap Active Firmware Partition

Swap the active and inactive flash memory partitions. The CPU must be reset in order to run the new active partition.

**Usage:** SA arg1

**arg1 Description:** Passcode to execute command

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** “321”

### ZC – Run Actuator Calibration Utility

Run the actuator’s calibration utility. The actuator must be uncoupled from any mechanism or load and free to move through its entire mechanical stroke. If successful, the **ZC** command will overwrite certain configuration variables and save all of the configuration settings to non-volatile memory. The actuator must be restarted after this command is issued using the **ZR** command. Consult Ultra Motion engineering before issuing this command.

**Usage:** ZC arg1

**arg1 Description:** Passcode to execute command

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** “321”

### ZR – Reboot Actuator

Restart the actuator controller. Upon restart the actuator will load all configuration settings from non-volatile memory.

**Usage:** ZR arg1

**arg1 Description:** Passcode to execute command

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** “321”

### ZU – Run Serial Firmware Update Utility

The ZU command closes the serial CLI and runs the serial firmware update utility.

**Usage:** ZU arg1

**arg1 Description:** Passcode to execute command

**arg1 Data Type:** Unsigned Integer

**arg1 Range:** “321”

## L-Series Configuration Variables

This section details configuration variables used by this firmware. These variables are stored within the controller's non-volatile memory and are configurable via the RS-485 serial interface. All configuration variables can be set with the **WV** command. For example, "wv [opMode](#) 1". To save all configuration variables to non-volatile memory, use the command "cw 321". Default values shown are for the non-redundant L-Series.

### L-Series Configuration Variables Quick Reference Table

Variable	Description	Group	Default Value	Notes
<a href="#">opMode</a>	Operating Mode Selector	Actuator Operation	0	Default: Serial CLI Mode
<a href="#">rPos</a>	Retracted Hard Stop Position	Actuator Travel Limits	1024	Units: Encoder Counts
<a href="#">ePos</a>	Extended Hard Stop Position	Actuator Travel Limits	<a href="#">rPos</a> + Stroke	Units: Encoder Counts
<a href="#">spMin</a>	Software Position Minimum	Actuator Travel Limits	<a href="#">rPos</a> + 1024	Units: Encoder Counts
<a href="#">spMax</a>	Software Position Maximum	Actuator Travel Limits	<a href="#">ePos</a> - 1024	Units: Encoder Counts
<a href="#">maxCurr</a>	Max Motor Current Demand	Motion Control	10000	
<a href="#">pkd</a>	PID gain kd	Motion Control	1500.0	
<a href="#">pki</a>	PID gain ki	Motion Control	0.2	
<a href="#">pkp</a>	PID gain kp	Motion Control	200.0	
<a href="#">pkdALT</a>	Alternate PID gain kd	Motion Control	1500.0	
<a href="#">pkiALT</a>	Alternate PID gain ki	Motion Control	0.0	
<a href="#">pkpALT</a>	Alternate PID gain kp	Motion Control	200.0	
<a href="#">maxSpeed</a>	Max Speed for Trajectory Moves	Motion Control	400000	
<a href="#">accel</a>	Accel/Decel for Trajectory Moves	Motion Control	2000	
<a href="#">ovbOn</a>	Overvoltage Dynamic Braking	Motion Control	1	Default: Enabled
<a href="#">altGS</a>	Alternate Gain Select	Motion Control	0	Default: Standard Gains
<a href="#">rxData</a>	Position Command Data Format	Motion Control	<>	Default: Position LSB, Position MSB
<a href="#">pMin</a>	Minimum Position Command	Motion Control	0	
<a href="#">pMax</a>	Maximum Position Command	Motion Control	65535	
<a href="#">pInvert</a>	Invert Response to CAN/BSC Command	Motion Control	0	Default: Does Not Invert
<a href="#">inEna</a>	Interpolation Enable Flag	Motion Control	1	Default: Enabled
<a href="#">defPos</a>	Default Position	CAN Mode Operation	$(\text{spMin} + \text{spMax})/2$	Units: Encoder Counts
<a href="#">CANspd</a>	Can Bus Baud Rate Selector	CAN Bus	0	Default: 1 Mbps
<a href="#">CANext</a>	Can Bus Message Type	CAN Bus	1	Default: Extended Message Type (29-bit ID)
<a href="#">rxID</a>	CAN Position Command RX ID	CAN Mode Operation	0x00000003	
<a href="#">rxMask</a>	Mask for CAN RX ID	CAN Mode Operation	0x1FFFFFFF	
<a href="#">canIvl</a>	Interpolation Interval	CAN Mode Operation	50	Units: Milliseconds
<a href="#">canTO</a>	Position Command Timeout	CAN Mode Operation	1250	Units: Milliseconds
<a href="#">cSUact</a>	CAN Startup Action Selector	CAN Mode Operation	0	Default: Hold Position
<a href="#">cTOact</a>	CAN Timeout Action Selector	CAN Mode Operation	0	Default: Hold Last Position, Last Current Command

Variable	Description	Group	Default Value	Notes
<a href="#">txEna</a>	Telemetry Enable Flags	CAN Telemetry	0	Default: Disabled
<a href="#">tx1Data</a>	Telemetry Data Format 1	CAN Telemetry	GKPCD	
<a href="#">tx1ID</a>	Telemetry Message Destination ID 1	CAN Telemetry	0x0000007F	
<a href="#">tx1Ivl</a>	Telemetry Interval 1	CAN Telemetry	1000	Units: Milliseconds
<a href="#">tx2Data</a>	Telemetry Data Format 2	CAN Telemetry	klmnpb	
<a href="#">tx2ID</a>	Telemetry Message Destination ID 2	CAN Telemetry	0x000027F	
<a href="#">tx2Ivl</a>	Telemetry Interval 2	CAN Telemetry	2500	Units: Milliseconds
<a href="#">tx3Data</a>	Telemetry Data Format 3	CAN Telemetry	wxy	
<a href="#">tx3ID</a>	Telemetry Message Destination ID 3	CAN Telemetry	0x000037F	
<a href="#">tx3Ivl</a>	Telemetry Interval 3	CAN Telemetry	5000	Units: Milliseconds
<a href="#">evntID</a>	Event Message Destination ID	CAN Event Messages	0x0000001F	
<a href="#">evntIvl</a>	Event Message Min Interval	CAN Event Messages	4000	Units: Milliseconds
<a href="#">evntWrn</a>	Warning Event Mask	CAN Event Messages	0x0000	
<a href="#">evntWrnC</a>	Warning Event Mask - Clear on Broadcast	CAN Event Messages	0x3FFF	
<a href="#">evntMsc</a>	Miscellaneous Event Mask	CAN Event Messages	0x0000	
<a href="#">IDbyte</a>	Identification Byte	CAN Event Messages	255	
<a href="#">sBaud</a>	Serial Baud Rate	RS485 Serial CLI	115200	Units: Baud
<a href="#">cliLock</a>	CLI Lock on Startup	RS485 Serial CLI	1	Default: Unlocked
<a href="#">cliBanr</a>	Startup Banner	RS485 Serial CLI	1	Default: Enabled
<a href="#">tPos1</a>	Trajectory Position 1	RS485 Serial CLI	3072	Units: Encoder Counts
<a href="#">tPos2</a>	Trajectory Position 2	RS485 Serial CLI	4096	Units: Encoder Counts
<a href="#">posLe</a>	Position Less than Threshold	Status Bits	0	Units: Encoder Counts
<a href="#">posGr</a>	Position Greater than Threshold	Status Bits	50175	Units: Encoder Counts
<a href="#">stplvl</a>	Stopped Speed Threshold	Status Bits	10	Units: Milliseconds @ 0 velocity count
<a href="#">ovCurr</a>	Motor Current Threshold	Status Bits	8200	
<a href="#">ovTemp</a>	Over Temperature Threshold	Status Bits	60.0	Units: Celsius
<a href="#">unTemp</a>	Under Temperature Threshold	Status Bits	-30.0	Units: Celsius
<a href="#">ovHumi</a>	Over Humidity Threshold	Status Bits	85.0	Units: Percent
<a href="#">bscAddr</a>	BSC Actuator Address	RS-485 BSC Protocol	128	
<a href="#">bSUact</a>	BSC Startup Action Selector	RS-485 BSC Protocol	0	Default: Hold Position
<a href="#">bTOact</a>	BSC Timeout Action Selector	RS-485 BSC Protocol	0	Default: Hold Last Position, Last Current Command
<a href="#">bscTO</a>	BSC Interpolation Interval	RS-485 BSC Protocol	1250	Units: Milliseconds
<a href="#">bscIvl</a>	BSC Position Command Timeout	RS-485 BSC Protocol	50	Units: Milliseconds

Table 23: Configuration variable reference list

Note: Default values for [rPos](#), [ePos](#), [spMin](#), [spMax](#), and [defPos](#) are set during actuator calibration.

### opMode – Operating Mode

This setting determines whether the actuator's motion is controlled by the RS-485 command line interface (**opMode** = 0), CAN bus command frames (**opMode** = 1), or the binary serial control protocol (**opMode** = 2)

**Data Type:** Integer

**Valid Range:** 0 to 2

**Default:** 0

### rPos, ePos – Retracted Hardstop Position, Extended Hardstop Position

These two configuration variables define the position of the actuator's fully-retracted and fully-extended hardstops. By default, the **rPos** variable is set along with the zero position of the Phase Index absolute position sensor such that position 0 is 1 screw revolution beyond the actuator's retracted hardstop (**rPos** = 1024). The full mechanical stroke length of the actuator should be **ePos** – **rPos**. These variables usually should not be modified by the user except through the use of the built-in actuator calibration routine ("ZC 321" CLI command). The values are expressed in Phase Index encoder counts. There are 1024 Phase Index encoder counts per screw revolution in the L-Series actuator.

**Data Type:** Integer

**Valid Range:** 0 – 50175

**Rules:**  $rPos \leq spMin < spMax \leq ePos$

**Default:** **rPos** = 1024, **ePos** = **rPos** + Stroke Length

### spMin, spMax - Software Position Minimum, Software Position Maximum

These two configuration variables set the minimum and maximum allowable position command in the L-Series actuator. The values are expressed in Phase Index encoder counts.

**Data Type:** Integer

**Valid Range:** **rPos** to **ePos**, **spMax**  $\geq$  **spMin** + 1

**Rules:**  $rPos \leq spMin < spMax \leq ePos$

**Default:** **spMin** = **rPos** + 1024, **spMax** = **ePos** – 1024

### sBaud – Serial Baud Rate

This setting sets the Baud rate for serial communication. Serial is accessible in all control modes and is used for diagnostics, configuration, initial setup, and control with **opMode** = 0 and **opMode** = 2. Default Baud rate is 115200. Lower baud rates will be more tolerant to noise and crosstalk at the expense of data bandwidth.

**Data Type:** Integer

**Valid Range:** 4800 to 460800

**Default:** 115200

### cliLock – Command Line Interface Lock

This setting determines if the CLI will startup locked or unlocked. A setting of 1 is unlocked on startup.

**Data Type:** Integer

**Valid Range:** 0 to 1

**Default:** 1

### cliBanr – CLI Power-On Banner

This setting determines if the serial banner that includes the actuator's firmware and hardware revisions, serial number, etc. is transmitted on reset or power-cycle. A setting of 0 disables the banner on startup.

**Data Type:** Integer

**Valid Range:** 0 to 1

**Default:** 1

### pkp, pki, and pkd – PID Gains $k_p$ , $k_i$ , and $k_d$ (Respectively)

These three values represent the gains for the proportional, integral, and derivative terms of the position PID control loop. Internal scaling of these gains is unique to this PID algorithm. Default factory values used represent typical stable gains. Adjusting these values is not recommended unless first discussed with Ultra Motion engineering.

**Data Type:** Float

**Valid Range:** -

**Default:**  $pkp = 200.0$ ,  $pki = 0.200$ ,  $pkd = 1500.0$

### pkpALT, pkiALT, and pkdALT– Alternate PID Gains $k_p$ , $k_i$ , and $k_d$ (Respectively)

These three values represent an alternate set of gains for the proportional, integral, and derivative terms of the position PID control loop. The gains used at startup by the actuator can be configured with the [altGS](#) configuration variable. The active gain set can be controlled with BSC or CAN packets if the Control Word is included in [rxData](#). See Table 9: Command options for L-Series Actuator Control for more information.

Internal scaling of these gains is unique to this PID algorithm. Default factory values used represent typical stable gains with zero i-term for load sharing configurations of the dual redundant L-Series. Adjusting these values is not recommended unless first discussed with Ultra Motion engineering.

**Data Type:** Float

**Valid Range:** -

**Default:**  $pkpALT = 200.0$ ,  $pkiALT = 0.0$ ,  $pkdALT = 1500.0$

### altGS – Default PID gain selection

This setting determines whether the standard or alternate position PID gains are used on startup by default. A setting of 0 sets the position PID gains to the standard values, a setting of 1 sets the PID gains to the alternate set.

**Data Type:** Integer

**Valid Range:** 0 to 1

**Default:** 1

### maxSpeed - Maximum Speed

This variable sets the top speed for trapezoidal profile trajectory moves. Note that it is possible to set [maxSpeed](#) higher than the maximum physical speed capability of the actuator. This can lead to control instability. The maximum speed that the actuator can achieve depends on the operating voltage, load, gear reduction, and other factors. Exceeding the maximum physical speed with the trajectory generator can result in integral windup, target position overshoot, and deviation from the defined trajectory profile.

**Data Type:** Integer

**Valid Range:** 1 to 10000000

**Default:** 400000

### maxCurr - Max Motor Current Demand

This setting limits the motor current demand signal that commands the current loop, thereby limiting the force produced by the L-Series. The value represents a percentage of full current output where 32767 equals 100%. The relationship is linear with a slight offset do to unloaded running friction of the system. Contact Ultra Motion engineering for more detailed information. Care must be taken if exceeding the continuous load rating of the actuator or if there's a potential to drive into a hardstop.

Note that setting **maxCurr** low does not lead to a freely backdriveable actuator with the L-Series and will instead cause a dynamic braking behavior. The actuator's enable input must be de-energized, control word commanded to COAST, or the CAN startup/timeout action set to COAST the motor in order to have the actuator backdrive freely.

**Data Type:** Integer

**Valid Range:** 0 to 32767

**Default value:** 10000 (approximate continuous limit at 25°C with natural convection)

### accel - Acceleration and Deceleration Rate

This setting defines the acceleration and deceleration the L-Series will use in profile trajectory moves. Note: the acceleration and deceleration will be equal. Setting the acceleration value greater than what's physically achievable by the actuator for a given **maxCurr** setting and load can lead to control loop instability.

**Data Type:** Integer

**Valid Range:** 0 to 131071

**Default:** 2000

### ovbOn – Overvoltage Dynamic Braking

When set to 1 value, the actuator will dynamically brake when the sensed bridge voltage is greater than 55.0 VDC and will disable dynamic braking when the sensed voltage drops below 52.0 VDC. This helps to prevent the actuator's regenerated energy from spiking the bridge voltage to levels that can damage the actuator.

Setting this value to zero disables this dynamic braking functionality and should only be done if the power supply can handle the regenerated energy from a backdriving or deceleration event, or if power shunt electronics are used to prevent the bus voltage from spiking to an unsafe level. The overvoltage dynamic braking will not engage if the enable input is inactive.

**Data Type:** Integer

**Valid Range:** 0 to 1

**Default:** 1

### CANspd – CAN Bus Baud Rate

This variable defines the baud rate of the CAN interface. There are 8 options designated by the integers 0 to 7.

0 = 1 Mbps, 1 = 500 Kbps, 2 = 250 Kbps, 3 = 125 Kbps, 4 = 100 Kbps, 5 = 50 Kbps, 6 = 20 Kbps, 7 = 10 Kbps

**Data Type:** Integer

**Valid Range:** 0 to 7

**Default:** 0

### CANext – CAN Bus Message Type

This variable defines whether the L-Series is configured for CAN 2.0B standard message type (11-bit identifiers), or extended message type (29-bit identifiers). **CANext** = 0 configures standard message type (11-bit ID). A value of 1 in **CANext** sets the actuator to extended message type (29-bit ID).

**Data Type:** Integer

**Valid Range:** 0 to 1

**Default:** 1

### rxID – CAN RX Identifier

**rxID** is the actuator's network address and is used along with **rxMask** to filter incoming CAN command messages. For acceptance, the message's identifier must match **rxID** where each corresponding bit in **rxMask** is set to '1'. If **CANext** = 0, only the lower 11 bits of **rxID** and **rxMask** are examined (standard message type). If **CANext** is 1, the lower 29 bits of **rxID** and **rxMask** are examined (extended message type).

**Data Type:** Integer

**Valid Range:** 0x00000000 to 0x1FFFFFFF

**Default:** 0x00000003

### rxMask – Mask for CAN RX Identifier

**rxMask** is used along with **rxID** to filter incoming CAN command messages. For acceptance, the message's identifier must match **rxID** where each corresponding bit in **rxMask** is set to '1'. If a bit in **rxMask** is set to '0', the corresponding bit of an incoming can message is matched for acceptance regardless of its value. If **rxMask** is set to 0x00000000, CAN messages with any identifier will be accepted. If **rxMask** is set to 0x1FFFFFFF, only CAN messages with an identifier that exactly matches the relevant bits of **rxID** will be accepted. If **CANext** = 0, only the lower 11 bits of **rxID** and **rxMask** are examined (standard message type). If **CANext** is 1, the lower 29 bits of **rxID** and **rxMask** are examined (extended message type).

**Data Type:** Integer

**Valid Range:** 0 to 0x1FFFFFFF

**Default:** 0x1FFFFFFF

### rxData – CAN Command Frame Data Format

This variable configures the data format of received position command messages. **rxData** is a string with a length from 1 to 8 characters. Each character in **rxData** designates a function for the corresponding data byte in the command message frame. The number of data bytes in the command message frame must match the number of characters in **rxData**. For example, if the master is sending out command messages with 8 data bytes, then **rxData** should be padded with 'X' or 'x' like this: "xxxx<>xx", where '<' is the position low byte, '>' is the position high byte, and 'x' means ignore this byte. The characters '(' and ')' can also be used to set the max motor current demand. '(' is max current demand low byte, and ')' is max current demand high byte. The acceptable range for the position command value is **pMin** to **pMax**. The acceptable range of the max current demand value is 0 to 32,767. The max current demand value is not updated if '(' and ')' characters are not included in **rxData**.

**Data Type:** String

**Valid Range:** <>()\*Xx

**Default:** <>

### cSUact – CAN Startup Action

This variable defines the behavior of the actuator after start-up and before receiving a valid CAN position command message.

cSUact = 0 holds the current position

cSUact = 1 executes a trajectory move to the default position [defPos](#)

cSUact = 2 COAST the motor

cSUact = 3 Dynamic brake the motor

**Data Type:** Integer

**Valid Range:** 0 to 3

**Default:** 0

### cTOact – CAN Timeout Action

This variable defines the behavior of the actuator after not receiving a valid CAN position command message within the timeout period ([canTO](#) ms). The actuator must receive at least 1 valid CAN position command message before a timeout can occur.

cTOact = 0 Holds the last valid position with the last valid max motor current demand value

cTOact = 1 Starts a trajectory move to the default position [defPos](#) with the last valid max current demand value

cTOact = 2 COAST the motor

cTOact = 3 Dynamic brake the motor

cTOact = 4 Holds the last valid position with the configured [maxCurr](#) setting

cTOact = 5 Executes a trajectory move to the default position [defPos](#) with the configured [maxCurr](#) setting

**Data Type:** Integer

**Valid Range:** 0 to 5

**Default:** 0

### canTO – CAN Position Command Timeout Period

This variable defines the CAN position command message timeout period in increments of 1 ms.

**Data Type:** Integer

**Valid Range:** 0 to 65535

**Default:** 1250

### pMin, pMax – CAN Position Command Range

This variable defines the valid command range that will be sent to the actuator via CAN messages. The value of [pMin](#)/[pMax](#) will be mapped to the actuator's travel range [spMin](#)/[spMax](#). If [pInvert](#) is set to 1, [pMax](#) will map to [spMin](#), and [pMin](#) will map to [spMax](#).

**Data Type:** Integer

**Valid Range:** 0 to 65535

**Default:** [pMin](#) = 0, [pMax](#) = 65535

### pInvert – Inversion of Response to CAN command

Inverts the response of the actuator with respect to the CAN position command. A value of 0 does not invert. A value of 1 inverts.

**Data Type:** Integer

**Valid Range:** 0 to 1

**Default:** 0

### defPos – Default Position

defPos is the absolute encoder value of the default position. The actuator can move to defPos upon startup if no valid position command messages have been received, or in the event of a position command message not being received in the canTO timeout period. The behavior of the actuator in these situations is defined by the cSUact and cTOact variables.

**Data Type:** Integer

**Valid Range:** spMin to spMax

**Default:** (spMin + spMax)/2

### canIvl – CAN Interpolation Interval

This variable defines the interpolation period in units of 1 ms for the received CAN commands. The default value is 50 (50 milliseconds), which is equivalent to a 20 Hz position update rate.

**Data Type:** Integer

**Valid Range:** 1 to 65535

**Default:** 50

### inEna – Interpolation Enable Flag

Setting inEna to 0 disables position interpolation, setting inEna to 1 enables position interpolation.

**Data Type:** Integer

**Valid Range:** 0 to 1

**Default:** 1

### txEna – Telemetry Enable Flags

Setting txEna to 0 disables all telemetry messages from the actuator. Setting to a value from 1 to 7 (0b001 to 0b111) will activate the corresponding telemetry message. For example, txEna=3 (0b011) would enable the broadcasting of tx1Data and tx2Data

**Data Type:** Integer

**Valid Range:** 0 to 7

**Default:** 0

### txNID – CAN Telemetry Message ID

These variables are the 29-bit or 11-bit CAN 2.0B identifiers for the telemetry messages. They represent the destination address of each telemetry message. If CANext=0, only the lower 11 bits are used.

**Data Type:** Integer

**Valid Range:** 0 to 0x1FFFFFFF

**Default:** tx1ID = 0x0000007F, tx2ID = 0x000027F, tx3ID = 0x000037F

### txNData – Telemetry Data Selection

The specific telemetry data bytes to be broadcast are selected with these configuration variables. The string may have a length from 1 to 8 characters. Each character in txNData designates one variable in the telemetry message. Variables may be one or more bytes in length each. The total number of data bytes cannot exceed 8 for a CAN 2.0B telemetry message.

**Data Type:** String

**Valid Range:** A-Z, a-z, 3, 4, 6, 7, 8, 9

**Default:** tx1Data = GKPCD, tx2Data = klmnbp, tx3Data = wxy

### txNlvl – Telemetry Interval

The telemetry intervals define the broadcast period of each telemetry message from the actuator in milliseconds. Care must be taken to ensure the CAN bus is not overloaded by too rapid a transmission rate.

**Data Type:** Integer

**Valid Range:** 2 to 10000

**Default:** tx1lvl = 1000, tx2lvl = 2500, tx3lvl = 5000

### bscAddr – BSC Address

bscAddr is the actuator's RS-485 serial address and is used to filter incoming BSC command messages. For acceptance, the message's Actuator ID byte must match bscAddr

**Data Type:** Integer

**Valid Range:** 1 to 128

**Default:** 128

### bSUact – BSC Startup Action

This variable defines the behavior of the actuator after start-up and before receiving a valid BSC position command message.

bSUact = 0 holds the current position

bSUact = 1 executes a trajectory move to the default position defPos

bSUact = 2 COAST the motor

bSUact = 3 Dynamic brake the motor

**Data Type:** Integer

**Valid Range:** 0 to 3

**Default:** 0

### bTOact – BSC Timeout Action

This variable defines the behavior of the actuator after not receiving a valid BSC position command message within the timeout period (bscTO ms). The actuator must receive at least 1 valid BSC position command message before a timeout can occur.

bTOact = 0 Holds the last valid position with the last valid max motor current demand value

bTOact = 1 Starts a trajectory move to the default position defPos with the last valid max current demand value

bTOact = 2 COAST the motor

bTOact = 3 Dynamic brake the motor

bTOact = 4 Holds the last valid position with the configured maxCurr setting

bTOact = 5 Executes a trajectory move to the default position defPos with the configured maxCurr setting

**Data Type:** Integer

**Valid Range:** 0 to 5

**Default:** 0

### bscTO – BSC Position Command Timeout Period

This variable defines the BSC position command message timeout period in increments of 1 ms.

**Data Type:** Integer  
**Valid Range:** 0 to 65535  
**Default:** 1250

### bscIvl – BSC Interpolation Interval

This variable defines the interpolation period in units of 1 ms for the received BSC commands. The default value is 50 (50 milliseconds), which is equivalent to a 20 Hz position update rate.

**Data Type:** Integer  
**Valid Range:** 1 to 65535  
**Default:** 50

### ovCurr – Motor Current Threshold

Defines the motor current threshold that causes the “Over Current” status bit to go high (when Motor Current > ovCurr). This variable is only used to define the behavior of the related status bit.

**Data Type:** Integer  
**Valid Range:** 0 to 32766  
**Default:** 8200

### stpIvl – Stopped Speed Threshold

Defines the number of milliseconds with a 0 velocity count before the actuator is considered to be “stopped” and the “stopped” status bit is set high. This variable is only used to define the behavior of the related status bit.

**Data Type:** Integer  
**Valid Range:** 1 to 65535  
**Default:** 10

### ovTemp – Over Temperature Threshold

Defines the temperature threshold that causes the “Over Temperature” status bit to go active (when Temp > ovTemp). This variable is only used to define the behavior of the related status bit. Units are in °C

**Data Type:** Float  
**Valid Range:** -50.0 to 149.0  
**Default:** 60.0

### unTemp – Under Temperature Threshold

Defines the temperature threshold that causes the “Under Temperature” status bit to go active (when Temp < unTemp). This variable is only used to define the behavior of the related status bit. Units are in °C

**Data Type:** Float  
**Valid Range:** -49.0 to 150.0  
**Default:** -30.0

### ovHumi – Over Humidity Threshold

Defines the relative humidity threshold that causes the “Over Humidity” status bit to go active (when RH > ovHumi). This variable is only used to define the behavior of the related status bit. Units are in %

**Data Type:** Float  
**Valid Range:** 0 to 100.0  
**Default:** 85.0

### tPos1 – Trajectory Position 1

Default position that can be used to command trajectory moves “T1”

**Data Type:** Integer  
**Valid Range:** spMin to spMax  
**Default:** 3072

### tPos2 – Trajectory Position 2

Default position that can be used to command trajectory moves “T2”

**Data Type:** Integer  
**Valid Range:** spMin to spMax  
**Default:** 4096

### posLe – Position less than threshold

Used as an additional position threshold that controls the posLe status bit. The status bit will go high if absolute position is less than the posLe threshold. This variable is only used to define the behavior of the related status bit.

**Data Type:** Integer  
**Valid Range:** 0 to 50175  
**Default:** 0

### posGr – Position greater than threshold

Used as an additional position threshold that controls the posGr status bit. The status bit will go high if absolute position is greater than the posGr threshold. This variable is only used to define the behavior of the related status bit.

**Data Type:** Integer  
**Valid Range:** 0 to 50175  
**Default:** 50175

### evntID – Event ID

This is the CAN ID that will be used for asynchronous event messages. If CANext=0, only the lower 11 bits are used.

**Data Type:** Integer  
**Valid Range:** 0 to 0x1FFFFFFF  
**Default:** 0x0000001F

### evntMsc– Miscellaneous Event Mask

Bit mask for miscellaneous event enable

**Data Type:** Integer  
**Valid Range:** 0 to 65535  
**Default:** 0x0000

### evntWrn– Warning Event Mask

Bit mask for warning event enable

**Data Type:** Integer

**Valid Range:** 0 to 16383

**Default:** 0x0000

### evntWrnC– Warning Event Mask – Clear on Broadcast

Bit mask for warning bit is cleared on event message broadcast.

**Data Type:** Integer

**Valid Range:** 0 to 16383

**Default:** 0x3FFF

### evntIvl – Minimum Event Message Interval

The minimum delay between successive warning event messages for the same warning bit in milliseconds.

**Data Type:** Integer

**Valid Range:** 100 to 65535

**Default:** 4000

### IDbyte – Identification Byte

An 8-bit value that can be used to identify actuators on the CAN bus. The byte is used in asynchronous event messages and can be added to telemetry messages. This can be useful for identification if multiple actuators are sending messages to a single address.

**Data Type:** Integer

**Valid Range:** 0 to 255

**Default:** 255

## Tertiary Encoder Overview

The following sections detail the configuration and operation of the dual redundant L-Series' tertiary Phase Index absolute position sensor.

### Tertiary Sensor - System Errors Register

The System Errors Register contains error status bits that result in automatic loss of encoder feedback

**Tertiary Sensor - System Error Register**

Error Bit	Bit Description
0	Configuration not loaded
1	Invalid Configuration in NVM
2	Phase Index encoder initialization failure
3 to 7	Reserved

Table 24: Tertiary Sensor - System Errors Register bit descriptions

### Tertiary Sensor - System Warning Register

The System Warning Register contains error status bits that do NOT result in automatic loss of encoder feedback, but may indicate issues with the sensor's accuracy, reliability, or configuration.

**Tertiary Sensor - System Warning Register**

Warning Bit	Bit Description
0	Bad configuration block in memory, majority of redundant copies are valid
1	Invalid configuration
2	Erroneous reset
3	UART error
4	Phase Index sensor communication error
5	Phase error above threshold
6	Phase Index QEI error
7	Phase Index encoder diagnostic error
8	Voltage high (> 55 VDC)
9	CAN initialization error
10	CAN transmit error
11 to 15	Reserved

Table 25: Tertiary Sensor - System Warnings Register bit descriptions

## Tertiary Sensor - System Status Register

The status register bytes contain bits that represent different aspects of the L-Series health and state. The configurable telemetry options include latched and non-latched versions of the individual status bytes.

Status Byte	Bit	Status Bit#	Description
0	0	0	System error register is non-zero
0	1	1	Nominal 3.3V bus
0	2	2	Nominal 5.0 V bus
0	3	3	Nominal 7.5 V bus
0	4	4	Input voltage less than minimum (9 VDC)
0	5	5	Input voltage greater than maximum (55 VDC)
0	6	6	Temperature at PCB is greater than ovTemp value
0	7	7	Temperature at PCB is less than unTemp value
1	0	8	Absolute position less than or equal to rPos
1	1	9	Absolute position greater than or equal to ePos
1	2	10	Absolute position less than posLe
1	3	11	Absolute position greater than posGr
1	4	12	Direction of travel is extend
1	5	13	Actuator stopped
1	6	14	System Warning register is non-zero
1	7	15	Reserved
2	0	16	Phase Index A error register is non-zero*
2	1	17	Phase Index A "Offset Loops Finished"
2	2	18	Phase Index A CORDIC overflow
2	3	19	Phase Index B error register is non-zero
2	4	20	Phase Index B "Offset Loops Finished"
2	5	21	Phase Index B CORDIC overflow
2	6	22	Encoder phase error warning limit exceeded
2	7	23	Phase Index to QEI difference exceeded
3	0	24	CAN RX error counter > 0 and < 127
3	1	25	CAN TX error counter > 0 and < 127
3	2	26	CAN RX Warning
3	3	27	CAN TX Warning
3	4	28	CAN RX error passive state
3	5	29	CAN TX error passive state
3	6	30	CAN TX Bus Off
3	7	31	CAN TX error flag

Table 26: Tertiary Sensor - Status Register bit descriptions

\*Status Byte 2 indicates issues with the sensor's accuracy, reliability, or operation.

Char	Description	Type	Num Bytes	Available via RS485
A	ID Byte	UINT8	1	Yes
B	Critical Errors Byte	UINT8	1	Yes
C	Warning Byte 0	UINT8	1	Yes
D	Warning Byte 1	UINT8	1	Yes
E	Warning Register – Clear on Broadcast	UINT16	2	Yes
K	Encoder Position Feedback	UINT16	2	Yes
L	Encoder Phase Error	UINT16	2	Yes
M	Encoder Velocity Count	INT16	2	Yes
N	Encoder Velocity Interval	UINT32	4	Yes
O	Encoder QEI Error	UINT16	2	Yes
P	Primary Sensor Error Register	UINT16	2	Yes
Q	Secondary Sensor Error Register	UINT16	2	Yes
R	Primary Sensor Diagnostics Register	UINT16	2	Yes
S	Primary Sensor Diagnostics Register	UINT16	2	Yes
a	Supply Voltage – Instantaneous	UINT16	2	Yes
b	Supply Voltage – Average over interval	UINT16	2	No
c	Supply Voltage – Minimum over interval	UINT16	2	No
d	Supply Voltage – Maximum over interval	UINT16	2	No
e	Status Register Byte 0	UINT8	1	Yes
f	Status Register Byte 1	UINT8	1	Yes
g	Status Register Byte 2	UINT8	1	Yes
h	Status Register Byte 3	UINT8	1	Yes
k	Status Register Byte 0 – Latched High	UINT8	1	No
l	Status Register Byte 1 – Latched High	UINT8	1	No
m	Status Register Byte 2 – Latched High	UINT8	1	No
n	Status Register Byte 3 – Latched High	UINT8	1	No
q	Status Register Byte 0 – Latched Low	UINT8	1	No
r	Status Register Byte 1 – Latched Low	UINT8	1	No
s	Status Register Byte 2 – Latched Low	UINT8	1	No
t	Status Register Byte 3 – Latched Low	UINT8	1	No
w	PCB Temperature (2 Hz update)	UINT8	1	Yes
z	PCB Temperature – 32 bits (2 Hz update)	FLOAT32	4	Yes
7	Serial Number	UINT32	4	Yes
8	UART Status Register	UINT16	2	Yes
9	CAN Errors (MSB = TX errors, LSB = RX errors)	UINT16	2	Yes
@	BSC CRC Errors	UINT16	2	Yes
\$	BSC Timeouts	UINT16	2	Yes
%	Serial TX Dropped	UINT16	2	Yes
!	opMode	UINT8	1	Yes

Table 27: Tertiary Sensor - Telemetry options for the CAN 2.0B protocol used in txNData

## Tertiary Sensor Serial Commands Quick Reference Table

Command	Description	Group	Arguments
<a href="#">RV</a>	Read configuration variable	Configuration	1 or 0
<a href="#">WV</a>	Write configuration variable	Configuration	2
<a href="#">WF</a>	Write any configuration variable to flash only	Configuration	2
<a href="#">CW</a>	Save configuration settings to non-volatile memory	Configuration	1
<a href="#">CC</a>	Copy configuration block from inactive to active partition	Configuration	1
<a href="#">RR</a>	Read runtime variable	Read	1 or 0
<a href="#">SR</a>	Read status register bits	Read	2 or 0
<a href="#">RI</a>	Read reset information	Read	0
<a href="#">FS</a>	Show serial number, part number, firmware version, target PCB	Read	0
<a href="#">HE</a>	Show all available serial commands	Read	0
<a href="#">ZR</a>	Restart actuator controller	System	1
<a href="#">LK</a>	Lock or unlock serial command-line-interface	System	1 or 0
<a href="#">FW</a>	Show firmware installed on active and inactive partitions	System	0
<a href="#">SA</a>	Swap active/inactive firmware partition	System	1
<a href="#">ZC</a>	Run actuator calibration utility	System	1
<a href="#">ZU</a>	Run serial firmware update utility	System	1

Table 28: Tertiary Sensor - Serial CLI Commands List

## Tertiary Sensor - Configuration Variables Quick Reference Table

Variable	Description	Group	Default Value	Notes
<a href="#">opMode</a>	Operating Mode Selector	Actuator Operation	0	Default: Serial CLI Mode
<a href="#">rPos</a>	Retracted Hard Stop Position	Actuator Travel Limits	1024	Units: Encoder Counts
<a href="#">ePos</a>	Extended Hard Stop Position	Actuator Travel Limits	<a href="#">rPos</a> + Stroke	Units: Encoder Counts
<a href="#">CANspd</a>	Can Bus Baud Rate Selector	CAN Bus	0	Default: 1 Mbps
<a href="#">CANext</a>	Can Bus Message Type	CAN Bus	1	Default: Extended Message Type (29-bit ID)
<a href="#">rxID</a>	CAN Position Command RX ID	CAN Mode Operation	0x00000005	
<a href="#">rxMask</a>	Mask for CAN RX ID	CAN Mode Operation	0x1FFFFFFF	
<a href="#">txEna</a>	Telemetry Enable Flags	CAN Telemetry	0	Default: Disabled
<a href="#">tx1Data</a>	Telemetry Data Format 1	CAN Telemetry	BDE	
<a href="#">tx1ID</a>	Telemetry Message Destination ID 1	CAN Telemetry	0x00000081	
<a href="#">tx1Ivl</a>	Telemetry Interval 1	CAN Telemetry	1000	Units: Milliseconds
<a href="#">tx2Data</a>	Telemetry Data Format 2	CAN Telemetry	K	
<a href="#">tx2ID</a>	Telemetry Message Destination ID 2	CAN Telemetry	0x0000281	
<a href="#">tx2Ivl</a>	Telemetry Interval 2	CAN Telemetry	2000	Units: Milliseconds
<a href="#">tx3Data</a>	Telemetry Data Format 3	CAN Telemetry	bw	
<a href="#">tx3ID</a>	Telemetry Message Destination ID 3	CAN Telemetry	0x0000381	
<a href="#">tx3Ivl</a>	Telemetry Interval 3	CAN Telemetry	3000	Units: Milliseconds
<a href="#">evntID</a>	Event Message Destination ID	CAN Event Messages	0x00000021	
<a href="#">evntIvl</a>	Event Message Min Interval	CAN Event Messages	4000	Units: Milliseconds
<a href="#">evntWrn</a>	Warning Event Mask	CAN Event Messages	0x0000	
<a href="#">evntWrnC</a>	Warning Event Mask - Clear on Broadcast	CAN Event Messages	0x0000	
<a href="#">evntMsc</a>	Miscellaneous Event Mask	CAN Event Messages	0x0000	
<a href="#">IDbyte</a>	Identification Byte	CAN Event Messages	255	
<a href="#">sBaud</a>	Serial Baud Rate	RS485 Serial CLI	115200	Units: Baud
<a href="#">cliLock</a>	CLI Lock on Startup	RS485 Serial CLI	1	Default: Unlocked
<a href="#">cliBanr</a>	Startup Banner	RS485 Serial CLI	1	Default: Enabled
<a href="#">posLe</a>	Position Less than Threshold	Status Bits	0	Units: Encoder Counts
<a href="#">posGr</a>	Position Greater than Threshold	Status Bits	50175	Units: Encoder Counts
<a href="#">stplvl</a>	Stopped Speed Threshold	Status Bits	10	Units: Milliseconds @ 0 velocity count
<a href="#">ovTemp</a>	Over Temperature Threshold	Status Bits	60.0	Units: Celsius
<a href="#">unTemp</a>	Under Temperature Threshold	Status Bits	-30.0	Units: Celsius
<a href="#">bscAddr</a>	BSC Actuator Address	RS-485 BSC Protocol	128	

Table 29: Tertiary Sensor - Configuration variable reference list

## Redundant Operation

The dual redundant configuration of the L-Series contains two isolated controllers that are configured and controlled identically to the non-redundant L-Series. An isolated tertiary Phase Index sensor allows for voting by the host system to validate the position feedback.

The optically isolated enable inputs control the supply of power to the bridge drivers and motor phases directly in hardware. The enable input bypasses the controller's microcontroller, providing the host with full authority to directly disable the motor in hardware in the event of a controller failure. This authority is critical to ensure the host can independently disable either of the dual redundant actuator's motors, allowing the functional motor to freely backdrive the failed motor and maintain overall actuator functionality. The below section "Operating Limits and Thermal Management" details the potential for total loss of actuator performance due to a thermal failure.

### Methods of Operation

---

#### *Cold-Standby*

In cold-standby, the primary motor/controller is commanded to drive the load which includes backdriving the secondary motor. The secondary controller must be configured to allow free backdriving by commanding COAST through the Control Word or removing power to the motor via the optically isolated enable input.

Both primary and secondary controllers should have their telemetry monitored to ensure operational health, and all three position feedback values should be monitored for agreement.

A user may decide to switch control from the primary motor to the secondary motor due to a loss of primary communication, primary temperature feedback exceeding a threshold, position feedback discrepancy, etc.

#### *Load-Sharing*

Both motors drive the load simultaneously in a load-sharing configuration, reducing the phase current required per motor, lowering heat generation, and increasing bandwidth capability. Load sharing offers multiple performance benefits at the cost of additional control complexity. Gains sets may need to be tuned for particular applications to achieve the required results and prevent force-fighting.

#### **Load-Sharing Recommendations:**

- The packet update rate and interpolation intervals should be at least 200 Hz.
- Ensure both controllers receive identical position commands in rapid succession to avoid fighting.
- Set startup and timeout actions to COAST, preventing one motor from locking in position if communication fails.
- Configure the timeout value to be less than twice the update rate to prevent a scenario where one controller loses a packet and holds the previous position.
- Use a leader/follower control scheme:
  - The leader operates with standard PID gains.
  - The follower uses alternate PID gains with the I-gain set to zero to prevent oscillation, force-fighting, and instability.
- Ensure command packets include the Control Word so each controller can be independently commanded to:
  - COAST – cut all power to the motor
  - Switch between standard and alternate PID gains dynamically.
  - Promote the follower to leader by switching to the primary PID gain set.

## Operating Limits and Thermal Management

---

To ensure reliable performance, always operate within the actuator's continuous load capacity or carefully monitor motor phase current and temperature telemetry when exceeding it.

### Key Considerations:

- The L-Series temperature feedback does not directly measure motor winding temperature, causing a delay in detecting heat buildup after/during peak load events.
- Driving into a hard stop or excessive load can overheat the motor windings, leading to thermal failure and degraded performance when switching to the secondary motor. Special care must be taken to avoid overdriving the actuator, especially with elevated maxCurr settings.
- A thermal failure results in a viscous damping effect, substantially increasing the torque required to backdrive the failed motor. This will significantly reduce the functional motor's effectiveness and possibly lead to a total loss of performance.

To prevent performance loss, actuator failure, and extend actuator lifespan, it is essential to configure the actuator properly for the expected loads, monitor motor phase current to prevent excessive stress, and track actuator temperature to avoid overheating.

## Tertiary Sensor Voting

---

The dual-redundant L-Series features two isolated control and power electronics for controlling its two motors, along with an isolated tertiary absolute position sensor used by the host or flight computer to confirm the validity of the position feedback. The host should monitor all three absolute position feedback values. A significant deviation in one sensor's reported position compared to the others indicates a potential failure in the corresponding electronics, requiring immediate action to disable control and quarantine the suspect system.

The status registers provide additional health information on the absolute position sensors and can help detect anomalies. Due to differences in telemetry sampling rate, speed of actuation, and electromagnetic variations, minor deviations between the three absolute position sensors are expected during operation. At a drive speed of 0.25 in/s, the typical peak-to-peak relative error is 15 encoder counts.

## Dynamic Performance

---

All performance graphs shown for the L-Series represent the dual redundant actuator's capability in a cold-standby load arrangement.

When both motors operate simultaneously, load sharing reduces the required motor phase current for a given load, effectively increasing the actuator's continuous load capacity, provided proper communication/control is maintained.

For a conservative assessment, only single-motor cold-standby performance data is provided.

If a motor experiences thermal failure due to overload, the redundant actuator's performance may be severely degraded or lost entirely due to viscous drag from the failed motor.

## Dual Redundant Controllers - Default Configurations

The table below outlines the default configuration variables for the primary and secondary controllers that differ from the non-redundant default configuration. The values of spMin and spMax for primary and secondary controllers are set equal to one another by default.

Variable	Description	Group	Primary Default	Secondary Default	Notes
<a href="#">altGS</a>	Alternate Gain Select	Motion Control	0	1	Default: Standard Gains
<a href="#">rxData</a>	Position Command Data Format	Motion Control	<>*	<>*	Default: Position LSB, Position MSB
<a href="#">rxID</a>	CAN Position Command RX ID	CAN Mode Operation	0x00000003	0x00000004	
<a href="#">canIvl</a>	Interpolation Interval	CAN Mode Operation	4	4	Units: Milliseconds
<a href="#">canTO</a>	Position Command Timeout	CAN Mode Operation	8	8	Units: Milliseconds
<a href="#">cSUact</a>	CAN Startup Action Selector	CAN Mode Operation	2	2	Default: COAST
<a href="#">cTOact</a>	CAN Timeout Action Selector	CAN Mode Operation	2	2	Default: COAST
<a href="#">tx1ID</a>	Telemetry Message Destination ID 1	CAN Telemetry	0x0000007F	0x00000080	
<a href="#">tx2ID</a>	Telemetry Message Destination ID 2	CAN Telemetry	0x000027F	0x00000280	
<a href="#">tx3ID</a>	Telemetry Message Destination ID 3	CAN Telemetry	0x000037F	0x00000380	
<a href="#">evntID</a>	Event Message Destination ID	CAN Event Messages	0x0000001F	0x00000020	
<a href="#">IDbyte</a>	Identification Byte	CAN Event Messages	253	254	
<a href="#">bscAddr</a>	BSC Actuator Address	RS-485 BSC Protocol	126	127	
<a href="#">bsUact</a>	BSC Startup Action Selector	RS-485 BSC Protocol	2	2	Default: COAST
<a href="#">bsTOact</a>	BSC Timeout Action Selector	RS-485 BSC Protocol	2	2	Default: COAST
<a href="#">bscTO</a>	BSC Interpolation Interval	RS-485 BSC Protocol	4	4	Units: Milliseconds
<a href="#">bscIvl</a>	BSC Position Command Timeout	RS-485 BSC Protocol	8	8	Units: Milliseconds

Table 30: Dual Redundant – Primary and Secondary default configuration variables

## General Actuator Design Guidelines

Below are some common pitfalls and guidelines to consider when using our linear servo actuators. Contact Ultra Motion Engineering for advice regarding the successful integration of the L-Series into an application.

### *Mechanical vs. Software Position Limits*

---

The fully retracted position shown in the interface control drawings and 3D models represents the actuator's nominal position at the internal retracted hard stop. By default, the L-Series ships with software-defined retracted and extended position limits set one revolution inwards from the mechanical hard stops. Users can adjust these limits closer to the hard stops but should maintain at least 0.5 screw revolutions of headroom to accommodate mechanical tolerances and control system overshoot.

### *Non-Axial Loading*

---

Rod-style linear actuators such as the L-Series are not rated for bending moments, loads perpendicular to the output shaft axis, or any loads other than axial tension or compression. Significant "side-loads" often occur inadvertently with over-constrained actuator mounting. Side-loads result in accelerated wear, reduced performance, or failure of the actuator. Mounting methods must be designed with adequate rotational degrees of freedom, such as pivoting joints (like a clevis or trunnion), or spherical bearings on both ends of the actuator, to ensure the actuator is only loaded in the axial direction when accounting for assembly motion and tolerances in your structure.

### *Torque at the Output Shaft*

---

The L-Series has an internal anti-rotation feature that holds the shaft from rotating during operation. All torque applied to the actuator's shaft during accessory installation *must* be reacted through the hex on the actuator's shaft.

### *Overloading and Mechanical Stops*

---

When using an actuator with a BLDC motor, take care that you do not run the actuator into a physical "hard-stop", such as a machine element or any physical body which the actuator impacts on its travel and which disallows movement. If this happens, the control system will attempt to obey position commands by powering through the physical block, setting the motor current command to the maximum. This can result in the motor over-heating and burning out, especially with higher maximum motor current settings or extended overload durations.

The exact continuous limit is difficult to accurately predict due to the unique heat transfer environments from application to application. A **maxCurr** setting of 10,000 is a rough approximation of the continuous capability of the actuator assuming a 25°C ambient environment and natural convection.

### *Regenerative Energy*

---

The actuator can push regenerated energy back into the power supply during deceleration or backdriving events. The power supply being used in the application must be sized adequately to prevent unwanted bus spikes during use. Regulated power supplies are particularly susceptible to bus spikes due to regenerated energy. Consider adding shunt electronics to the power system if significant amounts of regenerative energy are expected, a regulated power supply is being used, or if the bus is shared with other sensitive electronics. Bus voltage spikes above the actuator's rated voltage can lead to damage.

### *Solid Particle and Liquid Exposure*

---

The L-Series makes use of a high durometer shaft scraper, spring energized PTFE shaft seal, and O-ring seals on all static mechanical joints to provide robust IP67 sealing. Care should be taken to avoid the ingress of solid particles into an unmated connector by always using a connector cap when unmated.

### *Hot Plugging*

---

The actuator should not be connected/disconnected to the power supply while the power supply is active.

## L-Series Firmware Update

The L-Series can be updated to new firmware in the field by using Ultra Motion's Firmware update utility. `fw_loader_v1_0.exe` is a lightweight Windows console application for loading firmware onto Ultra Motion L-Series actuators over a serial (COM) port. It reads an Intel HEX formatted file and writes to the flash memory of the actuator's control unit using a proprietary serial protocol. The actuator must be running its firmware update utility to interact with this program. Windows 7 or later is required and L2 firmware v1.25 or later is required.

### Firmware Updating Procedure

The firmware updating utility and the most up to date firmware file can be received from Ultra Motion Engineering. The L-Series actuator must receive the "ZU321" command to begin running its firmware update utility. After navigating to the `fw_loader_v1_0.exe` directory, the following command line instruction can be sent:

```
fw_loader_v1_0.exe -f <file.hex> -p <COMx> [-b <baud>] [-q] [-n]
```

#### Firmware Update Command Line Options

Option Code	Description	Optionality
-f, --file	Path to the Intel HEX file	Required
-p, --port	COM port (e.g. COM3)	Required
-b, --baud	Baud rate (4800 to 460800)	Optional, default: 115200
-q, --quiet	Verbose output OFF	Optional
-n, --noexec	Skip automatic jump to application	Optional

Table 31: Firmware update command line options

After successfully loading the firmware, the actuator should be sent the "CC321" command to copy the previous calibration and configuration variables to the updated firmware.

Example:

```
C:\>fw_loader_v1_0.exe -p COM3 -b 115200 -f firmware.hex
```

### Firmware Update Tool Verification

To verify the integrity of the `fw_loader_v1_0.exe` file, compare its SHA-256 hash with the value received from Ultra Motion Engineering or available on our website at the following location:

[https://www.ultramotion.com/downloads/SHA256\\_list.html](https://www.ultramotion.com/downloads/SHA256_list.html)

The following commands can be used to compute the local hash:

- Using Command Prompt: `certutil -hashfile fw_loader_v1_0.exe SHA256`
- Using PowerShell: `Get-FileHash .\fw_loader_v1_0.exe -Algorithm SHA256`

Verify the output of the local hash and the hash received from Ultra Motion engineering or Ultramotion.com are equal e.g: `0xcc756ef5ec8dbd215508a37c9f729d621cca76e7b4dd9ad824ce05378c79b58c`

# Unit Conversions

## Definitions

Variable	Definition	Units	Note
$N$	Sensor Counts	–	$N$ denotes sensor counts for the specific data type in question
$P$	Position	$in$	Linear Position with respect to retracted endstop (rPos) of the actuator. The retracted endstop is always at $N = 1024$ .
$L$	Screw Lead	$in$	The lead of the drive screw
$a$	Acceleration	$in/s^2$	Linear acceleration, which is always positive (scalar) for setting 'accel'
$v$	Velocity	$in/s$	Linear velocity of the actuator shaft
$V_{bus}$	Bus Voltage	$V$	Supply voltage to the actuator (before bridge enable switch)
$V_{bridge}$	Voltage at Bridge	$V$	Voltage at the bridge driver (after bridge enable switch)
$T$	Temperature	$^{\circ}C$	Internal temperature, as measured at the controller PCB
$H_R$	Relative Humidity	%	Relative humidity, actual values will be between 0 and 100%
$I_{motor}$	Motor Current	$A$	Current through the BLDC motor
$I_{bridge}$	Bridge Current	$A$	Current through the bridge enable switch

Table 32: Unit conversion definitions

## Equations

Variable	Conversion to physical units
<b>Position</b>	$P = \frac{(N - 1024)L}{1024}$
<b>Speed</b>	$v = maxSpeed * 1000 * \frac{L}{65535 * 1024}$
<b>Accel</b>	$a = accel * 1000^2 * \frac{L}{65535 * 1024}$
<b>Bus Voltage</b>	$V_{bus} = VinSenADC * \frac{55.0}{16380}$
<b>Bridge Voltage</b>	$V_{bridge} = VinSenADC * \frac{55.0}{16380}$
<b>Motor Current</b>	$I_{motor} = motorCurrent * \frac{20.0}{32752}$
<b>Bridge Current</b>	$I_{bridge} = ibrSenADC * \frac{30}{16380}$
<b>8-bit Temperature</b>	$Temperature (^{\circ}C) = Feedback - 50$

Table 33: Unit conversion equations

## Contact Information

If you have any questions about the L-Series or any of our other products, contact us by one of the following methods:



### INQUIRY

Leave a web inquiry (to be replied to within one business day):

[ultramotion.com/contact](http://ultramotion.com/contact)



### LIVE CHAT

Live Chat directly with one of our engineers:

[ultramotion.com](http://ultramotion.com)



### EMAIL

Email (to be replied to within one business day):

[info@ultramotion.com](mailto:info@ultramotion.com)



### CALL

PH: 888-321-9178, 631-298-9179

Fax: 631-298-6593



### ADDRESS

Ultra Motion  
22355 County Road 48, #21  
Cutchogue, NY 11935



### HOURS

Our Business Hours:

Monday-Friday

9AM – 5PM EST

