LSS - Communication Protocol

Last modified by Coleman Benson (/info/bin/view/XWiki/CBenson) on 2019/07/25 13:31

Page Contents

Serial Protocol

The custom Lynxmotion Smart Servo (LSS) serial protocol was created in order to be as simple and straightforward as possible from a user perspective ("human readable format"), while at the same time compact and robust yet highly versatile. The protocol was based on Lynxmotion's SSC-32 RC servo controller and almost everything one might expect to be able to configure for a smart servo motor is available.

In order to have servos react differently when commands are sent to all servos in a serial bus, the first step a user should take is to assign a different ID number to each servo (explained below). Once this has been done, only the servo(s) which have been assigned to the ID sent as part of the command will follow that command. There is currently no CRC / checksum implemented as part of the protocol.

Session

A "session" is defined as the time between when the servo is powered ON to when it is powered OFF or reset.

Note that for a given session, the action related to a specific commands overrides the stored value in EEPROM.

Action Commands

Action commands tell the servo, within that session, to do something (i.e. "take an action"). The types of action commands which can be sent are described below, and they cannot be combined with other commands such as queries or configurations. Only one action command can be sent at a time. Action commands are session-specific, therefore once a servo is power cycled, it will not have any "memory" of previous actions or virtual positions (described below on this page). Action commands are sent serially to the servo's Rx pin and must be sent in the following format:

1. Start with a number sign # (U+0023)
2. Servo ID number as an integer

3. Action command (one to three letters, no spaces, capital or lower case)
4. Action value in the correct units with no decimal
5. End with a control / carriage return '<cr>'

Ex: #5PD1443<cr>

This sends a serial command to all servo's Rx pins which are connected to the bus and only servo(s) with ID #5 will move to a position in tenths of degrees ("PD") of 144.3 degrees. Any servo on the bus which does not have ID 5 will take no action when receiving this command.

Action Modifiers

Only two commands can be used as action modifiers: Timed Move (T) and Speed (S) described below. Action modifiers can only be used with certain action commands. The format to include a modifier is:

1. Start with a number sign # (U+0023)
2. Servo ID number as an integer
3. Action command (one to three letters, no spaces, capital or lower case)
4. Action value in the correct units with no decimal
5. Modifier command (one letter)
6. Modifier value in the correct units with no decimal
7. End with a control / carriage return '<cr>'

Ex: #5P1456T1263<cr>

This results in the servo with ID #5 rotating from the current angular position to a pulse position ("P") of 1456 in a time ("T") of 1263 milliseconds.

Query Commands

Query commands request information from the servo. They are received via the Rx pin of the servo, and the servo's reply is sent via the servo's Tx pin. Using separate lines for Tx and Rx is called "full duplex". Query commands are also similar to action and configuration commands and must use the following format:

1. Start with a number sign # (U+0023)
2. Servo ID number as an integer
3. Query command (one to three letters, no spaces, capital or lower case)
4. End with a control / carriage return '<cr>'

Ex: #5QD<cr>Query position in degrees for servo #5

The query will return a serial string (almost instantaneously) via the servo's Tx pin with the following format:

1. Start with an asterisk * (U+002A)
2. Servo ID number as an integer
3. Query command (one to three letters, no spaces, capital letters)
4. The reported value in the units described, no decimals.
5. End with a control / carriage return '<cr>'

There is currently no option to control how fast a servo replies after it has received a query command, therefore when sending a query command to the bus, the controller should be prepared to immediately "listen" for and parse the reply. Sending multiple queries on a bus in fast succession may result in replies overlapping and giving incorrect or corrupt data. As such, the controller should receive a reply before sending a new command. A reply to the query sent above might be:

Ex: *5QD1443<cr>

This indicates that servo #5 is currently at 144.3 degrees (1443 tenths of degrees).

Configuration Commands

Configuration commands and corresponding values affect a servo's defaults which are written to and read from the servo's EEPROM. These configurations are retained in memory after the servo is reset or power is cut / lost. Some configuration commands affect the session, while others do not. In the Command table below, the column "Session" denotes if the configuration command affects the session.. Not all action commands have a corresponding configuration command and vice versa. More information about which configuration commands are retained when in RC mode can be found on the LSS - RC PWM page (/info/wiki/lynxmotion/view/lynxmotion-smart-servo/lss-radio-control-pwm/). Configuration commands are not cumulative, in that if two configurations are sent, one after the next, only the last configuration is used and stored. The format to send a configuration command is identical to that of an action command:

1. Start with a number sign # (U+0023)
2. Servo ID number as an integer
3. Configuration command (two to three letters, no spaces, capital or lower case)
4. Configuration value in the correct units with no decimal
5. End with a control / carriage return '<cr>'

Ex: #5CO-50<cr>

This configures an absolute origin offset ("CO") with respect to factory origin to servo with ID #5 and changes the offset for that session to -5.0 degrees (50 tenths of degrees). Once the servo is powered off and then powered on, zeroing the servo will cause it to move to -5.0 degrees with respect to the factory origin and report its position as 0 degrees. Configuration commands can be undone / reset either by sending the servo's default value for that configuration, or by doing a factory reset (clears all configurations) described below.

Session vs Configuration Query

By default, the query command returns the sessions' value. Should no action commands have been sent to change the session value, it will return the value saved in EEPROM which will either be the servo's default, or modified with a configuration command. In order to query the value stored in EEPROM (configuration), add a '1' to the query command:

Ex: #5CSR20<cr> immediately sets the maximum speed for servo #5 to 20rpm (explained below) and changes the value in memory.

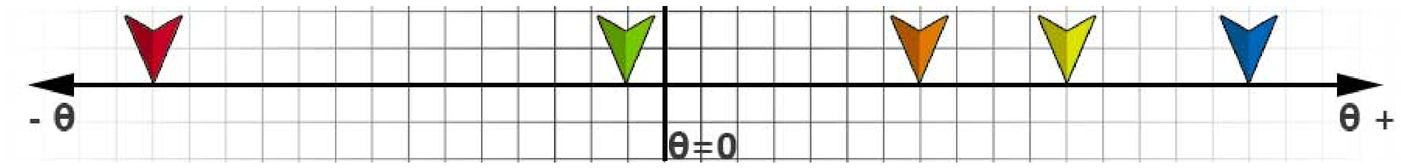
After RESET, a command of #5SR4<cr> sets the session's speed to 4rpm, but does not change the configuration value in memory. Therefore:

#5QSR<cr> would return *5QSR4<cr> which represents the value for that session, whereas

#5QSR1<cr> would return *5QSR20<cr> which represents the value in EEPROM

Virtual Angular Position

The ability to store a "virtual angular position" is a feature which allows for rotation beyond 360 degrees, permitting multiple rotations of the output horn, moving the center position and more. In virtual position mode, the "absolute position" would be the angle of the output shaft with respect to a 360.0 degree circle, and can be obtained by taking the modulus (with respect to 360 degrees) of the value. For example if the virtual position is reported as 15335 (or 1533.5 degrees), taking the modulus would give 93.5 degrees (3600 * 4 + 935 = 15335) as the absolute position (assuming no origin offset).



In this example, the gyre direction (explained below, a.k.a. "rotation direction") is positive (clockwise), and origin offset has not been modified. Each square represents 30 degrees. The following command is sent:

#1D-300<cr> This causes the servo to move to -30.0 degrees (green arrow)

#1D2100<cr> This second position command is sent to the servo, which moves it to 210.0 degrees (orange arrow)

#1D-4200<cr> This next command rotates the servo counterclockwise to a position of -420 degrees (red arrow), which means one full rotation of 360 degrees plus 60.0 degrees (420.0 - 360.0), with a virtual position of -420.0 degrees.

Although the final physical position would be the same as if the servo were commanded to move to -60.0 degrees, the servo is in fact at -420.0 degrees.

#1D4800<cr> This new command is sent which would then cause the servo to rotate from -420.0 degrees to 480.0 degrees (blue arrow), which would be a total of 900 degrees of clockwise rotation, or 2.5 complete rotations.

#1D3300<cr> would cause the servo to rotate from 480.0 degrees to 330.0 degrees (yellow arrow).

If / once the servo loses power or is power cycled, it also loses the virtual position associated with that session. For example, if the virtual position was 480.0 degrees before power is cycled, upon power up the servo's position will be read as +120.0 degrees from zero (assuming center position has not been modified).

Command List

Regular

#	Description	Action	Query	Config	Session	RC	Serial	Units	Notes	Default Value
1	**L**imp	L					✓	none		
2	**H**alt & **H**old	H					✓	none		
3	**T**imed move	T					✓	milliseconds	Modifier only for {P, D, MD}	
4	**S**peed	S	QS				✓	microseconds per second	Modifier only {P}	
5	**M**ove in **D**egrees (relative)	MD					✓	tenths of degrees (ex 325 = 32.5 degrees)		
6	**O**rigin Offset	O	QO	CO	✓	✓	✓	tenths of degrees (ex 91 = 9.1 degrees)		0
7	**A**ngular **R**ange	AR	QAR	CAR	✓	✓	✓	tenths of degrees		1800
8	Position in **P**ulse	P	QP				✓	microseconds	Inherited from SSC-32 serial protocol	
9	Position in **D**egrees	D	QD / QDT				✓	tenths of degrees		

10	**W**heel mode in **D**egrees	WD	QWD				✓	tenths of degrees per second (ex 248 = 24.8 degrees per second)	A.K.A. "Speed mode" or "Continuous rotation"	
11	**W**heel mode in **R**PM	WR	QWR				✓	revolutions per minute (rpm)	A.K.A. "Speed mode" or "Continuous rotation"	
12	Max **S**peed in **D**egrees	SD	QSD	CSD	✓	✓	✓	degrees per second (°/s)	QSD: Add modifier "2" for instantaneous speed. SD overwrites SR / CSD overwrites CSR and vice-versa.	Max per servo
13	Max **S**peed in **R**PM	SR	QSR	CSR	✓	✓	✓	revolutions per minute (rpm)	QSR: Add modifier "2" for instantaneous speed SR overwrites SD / CSR overwrites CSD and vice-versa.	Max per servo
14	**LED** Color	LED	QLED	CLED	✓	✓	✓	none (integer from 0 to 8)	0=Off (black); 1=Red 2=Green; 3=Blue; 4=Yellow; 5=Cyan; 6=Magenta; 7=White;	0 (OFF)

15	**G**yre direction (**G**)	G	QG	CG	✓	✓	✓	none	Gyre / rotation direction: 1= CW (clockwise) -1 = CCW (counter-clockwise)	1
16	**ID** #		QID	CID			✓	none (integer from 0 to 250)	Note: ID 254 is a "broadcast" which all servos respond to.	0
17	**B**aud rate		QB	CB			✓	none (integer)		115200
18	*{coming soon}*									
19	**F**irst Position (**D**eg)		QFD	CFD	X	✓	✓	none		No Value
20	**M**odel **S**tring		QMS					none (string)	Returns the type of servo (ST, HS, HT)	
21	Serial **N**umber		QN					none (integer)	Returns the unique serial number for that servo	
22	**F**irmware version		QF					none (integer)		
23	**Q**uery (gen. status)		Q				✓	none (integer from 1 to 8)	See command description for details	
24	**V**oltage		QV				✓	millivolts (ex 5936 = 5936mV = 5.936V)		
25	**T**emperature		QT				✓	tenths of degrees Celsius	Max temp before error: 85°C (servo goes limp)	
26	**C**urrent		QC				✓	milliamps (ex 200 = 0.2A)		

27	**C**hange to **RC**	CRC	✓	✓	none	Change to RC mode 1 (position) or 2 (wheel).	Serial
28	**RESET**			✓	none	Soft reset. See command for details.	
29	**DEFAULT**			✓	none	Revert to firmware default values. See command for details	
30	**UPDATE**			✓	none	Update firmware. See command for details.	

Advanced

#	Description	Action	Query	Config	Session	RC	Serial	Units	Notes
A1	**A**ngular **S**tiffness	AS	QAS	CAS	✓	✓	✓	none (integer -4 to +4)	Suggested values are between 0 to +4
A2	**A**ngular **H**olding Stiffness	AH	QAH	CAH	✓		✓	none (integer -10 to +10)	Effect is different between serial and RC
A3	**A**ngular **A**cceleration	AA	QAA	CAA	✓		✓	degrees per second squared	Increments of 10 degrees per second squared
A4	**A**ngular **D**eceleration	AD	QAD	CAD	✓		✓	degrees per second squared	Increments of 10 degrees per second squared
A5	**E**nable **M**otion Control	EM	QEM				✓	none	EM0 to disable motion control, EM1 to enable

A6	**C**onfigure **L**ED **B**linking	QLB	CLB	✓	✓	none (integer from 0 to 63)	0=No blinking, 63=Always blink; Blink while: 1=Limp; 2=Holding; 4=Accel; 8=Decel; 16=Free 32=Travel;

Details - Basic

1. Limp (**L**)

Example: #5L<cr>

This action causes the servo to go "limp". The microcontroller will still be powered, but the motor will not. As an emergency safety feature, should the robot not be doing what it is supposed to or risks damage, use the broadcast ID to set all servos limp #254L<cr>.

2. Halt & Hold (**H**)

Example: #5H<cr>

This action overrides whatever the servo might be doing at the time the command is received (accelerating, moving continuously etc.) and causes it to stop immediately and hold that angular position.

3. Timed move (**T**) modifier

Example: #5P1500T2500<cr>

Timed move can be used only as a modifier for a position (P, D, MD) actions. The units are in milliseconds, so a timed move of 2500 milliseconds would cause the servo to rotate from its current position to the desired position in 2.5 seconds. The onboard controller will attempt to ensure that the move is performed entirely at the desired velocity, though differences in torque may cause it to not be exact. This command is in place to ensure backwards compatibility with the SSC-32 / 32U protocol.

Note: If the calculated speed at which a servo must rotate for a timed move is greater than its maximum speed (which depends on voltage and load), then it will move at its maximum speed, and the time of the move may be longer than requested.

4. Speed (**S**) modifier

Example: #5P1500S750<cr>

This command is a modifier only for a position (P) action and determines the speed of the move in microseconds per second. A speed of 750 microseconds would cause the servo to rotate from its current position to the desired position at a speed of 750 microseconds per second. This command is in place to ensure backwards compatibility with the SSC-32 / 32U protocol.

Query Speed (**QS**)

Example: #5QS<cr> might return *5QS300<cr>

This command queries the current speed in microseconds per second.

5. (Relative) Move in Degrees (**MD**)

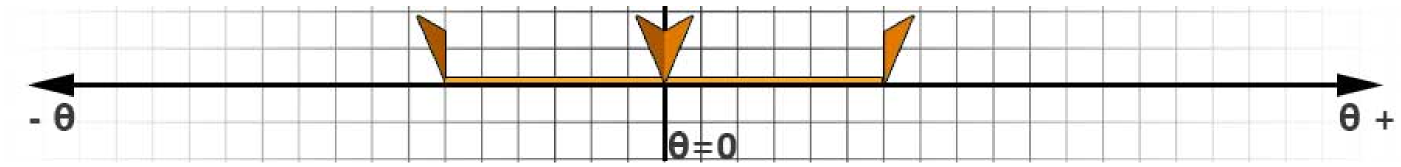
Example: #5MD123<cr>

The relative move command causes the servo to read its current position and move the specified number of tenths of degrees in the corresponding position. For example if the servo is set to rotate CW (default) and an MD command of 123 is sent to the servo, it will cause the servo to rotate clockwise by 12.3 degrees. Negative commands would cause the servo to rotate in the opposite configured direction.

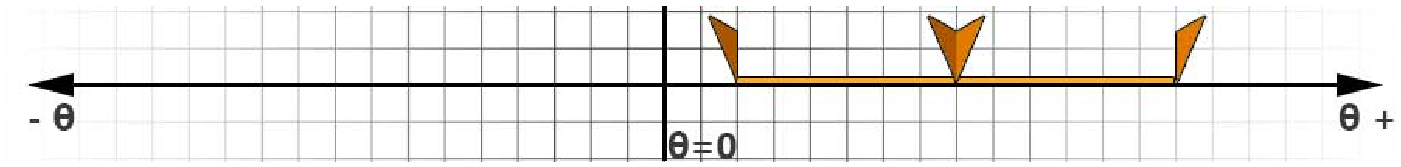
6. Origin Offset Action (**O**)

Example: #5O2400<cr>

This command allows you to temporarily change the origin of the servo in relation to the factory zero position for that session. As with all action commands, the setting will be lost upon servo reset / power cycle. Origin offset commands are not cumulative and always relate to factory zero. In the first image, the origin at factory offset '0' (centered).



In the second image, the origin, and the corresponding angular range (explained below) have been shifted by +240.0 degrees:



Origin Offset Query (**QO**)

Example: #5QO<cr> Returns: *5QO-13

This allows you to query the angle (in tenths of degrees) of the origin in relation to the factory zero position. In this example, the new origin is at -1.3 degrees from the factory zero.

Configure Origin Offset (**CO**)

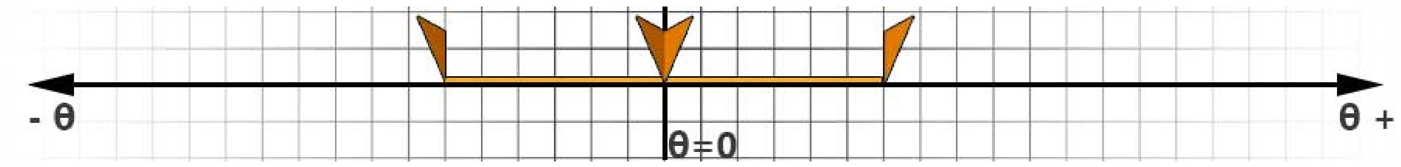
Example: #5CO-24<cr>

This command allows you to change the origin of the servo in relation to the factory zero position in EEPROM. The setting will be saved upon servo reset / power cycle. Origin offset configuration commands are not cumulative and always relate to factory zero. The new origin is also used in RC mode. In the example, the new origin will be at -2.4 degrees from the factory zero.

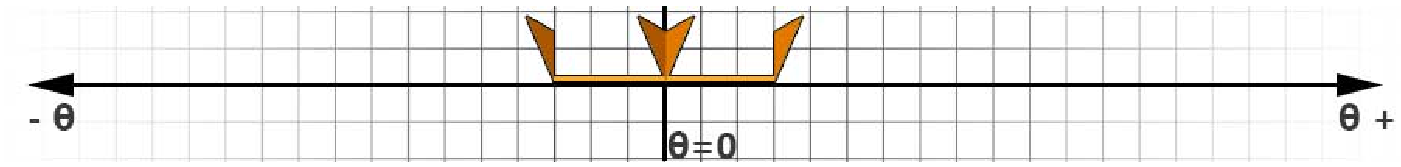
7. Angular Range (**AR**)

Example: #5AR1800<cr>

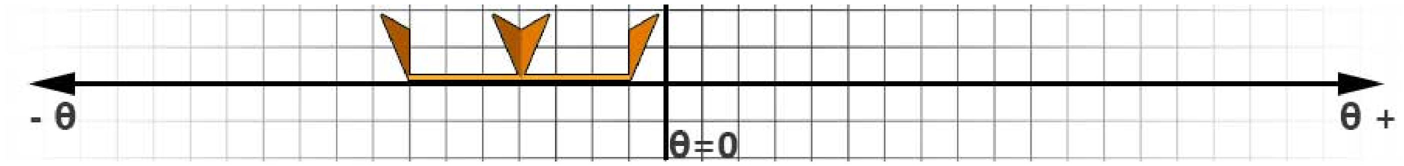
This command allows you to temporarily change the total angular range of the servo in tenths of degrees. This applies to the Position in Pulse (P) command and RC mode. The default for (P) and RC mode is 1800 (180.0 degrees total, or ±90.0 degrees). The image below shows a standard -180.0 to +180.0 range, with no offset:



Below, the angular range is restricted to 180.0 degrees, or -90.0 to +90.0. The center has remained unchanged.



Finally, the angular range action command (ex. #5AR1800<cr>) and origin offset action command (ex. #5O-1200<cr>) are used to move both the center and limit the angular range:



Query Angular Range (**QAR**)

Example: #5QAR<cr> might return *5AR1800, indicating the total angular range is 180.0 degrees.

Configure Angular Range (**CAR**)

This command allows you to change the total angular range of the servo in tenths of degrees in EEPROM. The setting will be saved upon servo reset / power cycle.

8. Position in Pulse (**P**)

Example: #5P2334<cr>

The position in PWM pulses was retained in order to be backward compatible with the SSC-32 / 32U protocol. This relates the desired angle with an RC standard PWM pulse and is further explained in the SSC-32 and SSC-32U manuals found on Lynxmotion.com. Without any modifications to configuration considered, and a ±90.0 degrees standard range where 1500 microseconds is centered, a pulse of 2334 would set the servo to 165.1 degrees. Valid values for P are [500, 2500]. Values outside this range are corrected / restricted to end points.

Query Position in Pulse (**QP**)

Example: #5QP<cr> might return *5QP2334

This command queries the current angular position in PWM "units". The user must take into consideration that the response includes any angular range and origin configurations in order to determine the actual angle.
Valid values for QP are {-500, [500, 2500], -2500}. Values outside the [500, 2500] range are given a negative corresponding end point value to indicate they are out of bounds (note that if the servo is physically located at one of the endpoints, it may return a negative number if it is a fraction of a degree beyond the position).

9. Position in Degrees (**D**)

Example: #5D1456<cr>

This moves the servo to an angle of 145.6 degrees, where the center (0) position is centered. Negative values (ex. -176 representing -17.6 degrees) are used. A full circle would be from -1800 to 1800 degrees. A value of 2700 would be the same angle as -900, except the servo would move in a different direction.

Larger values are permitted and allow for multi-turn functionality using the concept of virtual position.

Query Position in Degrees (**QD**)

Example: #5QD<cr> might return *5QD132<cr>

This means the servo is located at 13.2 degrees.

Query Target Position in Degrees (**QDT**)

Ex: #5QDT<cr> might return *5QDT6783<cr>

The query target position command returns the target angle during and after an action which results in a rotation of the servo horn. In the example above, the servo is rotating to a virtual position of 678.3 degrees. Should the servo not have a target position or be in wheel mode, it will respond without a number (Ex: *5QDT<cr>).

10. Wheel Mode in Degrees (**WD**)

Ex: #5WD900<cr>

This command sets the servo to wheel mode where it will rotate in the desired direction at the selected speed. The example above would have the servo rotate at 90.0 degrees per second clockwise (assuming factory default configurations).

Query Wheel Mode in Degrees (**QWD**)

Ex: #5QWD<cr> might return *5QWD900<cr>

The servo replies with the angular speed in tenths of degrees per second. A negative sign would indicate the opposite direction (for factory default a negative value would be counter clockwise).

11. Wheel Mode in RPM (**WR**)

Ex: #5WR40<cr>

This command sets the servo to wheel mode where it will rotate in the desired direction at the selected rpm. Wheel mode (a.k.a. "continuous rotation") has the servo operate like a geared DC motor. The servo's maximum rpm cannot be set higher than its physical limit at a given voltage. The example above would have the servo rotate at 40 rpm clockwise (assuming factory default configurations).

Query Wheel Mode in RPM (**QWR**)

Ex: #5QWR<cr> might return *5QWR40<cr>

The servo replies with the angular speed in rpm. A negative sign would indicate the opposite direction (for factory default a negative value would be counter clockwise).

12. Max Speed in Degrees (**SD**)

Ex: #5SD1800<cr>

This command sets the servo's maximum speed for motion commands in tenths of degrees per second for that session. In the example above, the servo's maximum speed for that session would be set to 180.0 degrees per second. The servo's maximum speed cannot be set higher than its physical limit at a given voltage. The SD action command overrides CSD (described below) for that session. Upon reset or power cycle, the servo reverts to the value associated with CSD as described below. Note that SD and SR (described below) are effectively the same, but allow the user to specify the speed in either unit. The last command (either SR or SD) received is what the servo uses for that session.

Query Speed in Degrees (**QSD**)

Ex: #5QSD<cr> might return *5QSD1800<cr>

By default QSD will return the current session value, which is set to the value of CSD as reset/power cycle and changed whenever an SD/SR command is processed.
If #5QSD1<cr> is sent, the configured maximum speed (CSD value) will be returned instead. You can also query the current speed using "2" and the current target travel speed using "3". See the table below for an example:

Command sent	**Returned value (1/10 °)**
ex: #5QSD<cr>	Session value for maximum speed (set by latest SD/SR command)
ex: #5QSD1<cr>	Configured maximum speed in EEPROM (set by CSD/CSR)
ex: #5QSD2<cr>	Instantaneous speed (same as QWD)
ex: #5QSD3<cr>	Target travel speed

Configure Speed in Degrees (**CSD**)

Ex: #5CSD1800<cr>

Using the CSD command sets the servo's maximum speed which is saved in EEPROM. In the example above, the servo's maximum speed will be set to 180.0 degrees per second. When the servo is powered on (or after a reset), the CSD value is used. Note that CSD and CSR (described below) are effectively the same, but allow the user to specify the speed in either unit. The last command (either CSR or CSD) is what the servo uses for that session.

13. Max Speed in RPM (**SR**)

Ex: #5SD45<cr>

This command sets the servo's maximum speed for motion commands in rpm for that session. In the example above, the servo's maximum speed for that session would be set to 45rpm. The servo's maximum speed cannot be set higher than its physical limit at a given voltage. SD overrides CSD (described below) for that session. Upon reset or power cycle, the servo reverts to the value associated with CSD as described below. Note that SD (described above) and SR are effectively the same, but allow the user to specify the speed in either unit. The last command (either SR or SD) received is what the servo uses for that session.

Query Speed in Degrees (**QSR**)

Ex: #5QSR<cr> might return *5QSR45<cr>

By default QSR will return the current session value, which is set to the value of CSR as reset/power cycle and changed whenever an SD/SR command is processed.
If #5QSR1<cr> is sent, the configured maximum speed (CSR value) will be returned instead. You can also query the current speed using "2" and the current target travel speed using "3". See the table below for an example:

Command sent	**Returned value (1/10 °)**
ex: #5QSR<cr>	Session value for maximum speed (set by latest SD/SR command)
ex: #5QSR1<cr>	Configured maximum speed in EEPROM (set by CSD/CSR)
ex: #5QSR2<cr>	Instantaneous speed (same as QWR)
ex: #5QSR3<cr>	Target travel speed

Configure Speed in RPM (**CSR**)

Ex: #5CSR45<cr>

Using the CSR command sets the servo's maximum speed which is saved in EEPROM. In the example above, the servo's maximum speed will be set to 45rpm. When the servo is powered on (or after a reset), the CSR value is used. Note that CSD and CSR are effectively the same, but allow the user to specify the speed in either unit. The last command (either CSR or CSD) received is what the servo uses for that session.

14. LED Color (**LED**)

Ex: #5LED3<cr>

This action sets the servo's RGB LED color for that session.The LED can be used for aesthetics, or (based on user code) to provide visual status updates. Using timing can create patterns.

0=Off (black); 1=Red 2=Green; 3=Blue; 4=Yellow; 5=Cyan; 6=Magenta; 7=White;

Query LED Color (**QLED**)

Ex: #5QLED<cr> might return *5QLED5<cr>

This simple query returns the indicated servo's LED color.

Configure LED Color (**CLED**)

Configuring the LED color via the CLED command sets the startup color of the servo after a reset or power cycle. Note that it also changes the session's LED color immediately as well.

15. Gyre Rotation Direction (**G**)

"Gyre" is defined as a circular course or motion. The effect of changing the gyre direction is as if you were to use a mirror image of a circle. CW = 1; CCW = -1. The factory default is clockwise (CW).

Ex: #5G-1<cr>

This command will cause servo #5's positions to be inverted, effectively causing the servo to rotate in the opposite direction given the same command. For example in a 2WD robot, servos are often physically installed back to back, therefore setting one of the servos to a negative gyration, the same wheel command (ex WR30) to both servos will cause the robot to move forward or backward rather than rotate.

Query Gyre Direction (**QG**)

Ex: #5QG<cr> might return *5QG-1<cr>

The value returned above means the servo is in a counter-clockwise gyration.

Configure Gyre (**CG**)

Ex: #5CG-1<cr>

This changes the gyre direction as described above and also writes to EEPROM.

16. Identification Number (**ID**)

A servo's identification number cannot be set "on the fly" and must be configured via the CID command described below. The factory default ID number for all servos is 0. Since smart servos are intended to be daisy chained, in order to respond differently from one another, the user must set different identification numbers. Servos with the same ID and baud rate will all receive and react to the same commands (assuming same baud rate).

Query Identification (**QID**)

EX: #254QID<cr> might return *QID5<cr>

When using the query ID command, it is best to only have one servo connected and thus receive only one reply. This is useful when you are not sure of the servo's ID, but don't want to change it. Using the broadcast command (ID 254) with only one servo will have that servo reply with its ID number (assuming the query is sent . Alternatively, pushing the button upon startup and temporarily setting the servo ID to 255 will still result in the servo responding with its "real" ID.

Configure ID (**CID**)

Ex: #4CID5<cr>

Setting a servo's ID in EEPROM is done via the CID command. All servos connected to the same serial bus will be assigned that ID. In most situations each servo must be set a unique ID, which means each servo must be connected individually to the serial bus and receive a unique CID number. It is best to do this before the servos are added to an assembly. Numbered stickers are provided to distinguish each servo after their ID is set, though you are free to use whatever alternative method you like. The servo must be RESET or power cycled in order for the new ID to take effect.

17. Baud Rate

A servo's baud rate cannot be set "on the fly" and must be configured via the CB command described below. The factory default baud rate for all servos is 115200. Since smart servos are intended to be daisy chained, in order to respond to the same serial bus, all servos in a project should ideally be set to the same baud rate. Setting different baud rates will have the servos respond differently and may create issues. Available baud rates are: 9600 bps, 19200 bps, 38400 bps, 57600 bps, 115.2 kbps, 230.4 kbps, 250.0 kbps, 460.8 kbps, 500.0 kbps. Servos are shipped with a baud rate set to 9600. The baud rates are currently restricted to those above.

Query Baud Rate (**QB**)

Ex: #5QB<cr> might return *5QB9600<cr>

Since the command to query the baud rate must be done at the servo's existing baud rate, it can simply be used to confirm the CB configuration command was correctly received before the servo is power cycled and the new baud rate takes effect.

Configure Baud Rate (**CB**)

Important Note: the servo's current session retains the given baud rate and the new baud rate will only take effect when the servo is power cycled / RESET.

Ex: #5CB9600<cr>

Sending this command will change the baud rate associated with servo ID 5 to 9600 bits per second.

18. {*Coming soon*}

Command coming soon....

19. First Position (Degrees)

In certain cases, a user might want to have the servo move to a specific angle upon power up; we refer to this as "first position" (a.k.a. "initial position"). The factory default has no first position value stored in EEPROM and therefore upon power up, the servo remains limp until a position (or hold command) is assigned.

Query First Position in Degrees (**QFD**)

Ex: #5QFD<cr> might return *5QFD64<cr>

The reply above indicates that servo with ID 5 has a first position pulse of 1550 microseconds. If there is no first position value stored, the reply will be DIS

Configure First Position in Degrees (**CFD**)

Ex: #5CD64<cr>

This configuration command means the servo, when set to smart mode, will immediately move to 6.4 degrees upon power up. Sending a CFD command without a number (Ex. #5CFD<cr>) results in the servo remaining limp upon power up. In order to remove the first position, send no value, ex: #5CFD<cr>

20. Query Model String (**QMS**)

Ex: #5QMS<cr> might return *5QMSLSS-HS1cr>

This reply means the servo model is LSS-HS1, meaning a high speed servo, first revision.

21. Query Serial Number (**QN**)

Ex: #5QN<cr> might return *5QN12345678<cr>

The number in the response (12345678) would be the servo's serial number which is set and should not be changed by the user.

22. Query Firmware (**QF**)

Ex: #5QF<cr> might return *5QF411<cr>

The number in the reply represents the firmware version, in this example being 411.

23. Query Status (**Q**)

The status query described what the servo is currently doing. The query returns an integer which must be looked up in the table below. Use the CLB advanced command to have the LED blink for certain statuses.

Ex: #5Q<cr> might return *5Q6<cr>, which indicates the motor is holding a position.

*Value returned (Q)	Status	Detailed description
ex: *5Q0<cr>	0: Unknown	LSS is unsure / unknown state
ex: *5Q1<cr>	1: Limp	Motor driving circuit is not powered and horn can be moved freely

ex: *5Q2<cr>	2: Free moving	Motor driving circuit is not powered and horn can be moved freely
ex: *5Q3<cr>	3: Accelerating	Increasing speed from rest (or previous speed) towards travel speed
ex: *5Q4<cr>	4: Traveling	Moving at a stable speed
ex: *5Q5<cr>	5: Decelerating	Decreasing from travel speed towards final position.
ex: *5Q6<cr>	6: Holding	Keeping current position
ex: *5Q7<cr>	7: Outside limits	{More details coming soon}
ex: *5Q8<cr>	8: Stuck	Motor cannot perform request movement at current speed setting
ex: *5Q9<cr>	9: Blocked	Similar to stuck, but the motor is at maximum duty and still cannot move (i.e.: stalled)
ex: *5Q10<cr>	10: Safe Mode	A safety limit has been exceeded (temperature, peak current or extended high current draw). Send a Q1 command to know which limit has been reached (described below).

If a safety limit has been reached and exceeded, the LED will flash red and the servo will stop providing torque (no longer react to commands which cause the motor to rotate). In order to determine which limit has been reached, send a Q1 command. The servo must be RESET in order to return to normal operation, though if a limit is still detected (for example the servo is still too hot), it will revert back to Safe Mode.

***Value returned (Q1)**	**Status**	**Detailed description**
ex: *5Q0<cr>	No limits have been passed	Nothing is wrong
ex: *5Q1<cr>	Current limit has been passed	Something cause the current to either spike, or remain too high for too long
ex: *5Q2<cr>	Input voltage detected is below or above acceptable range	Check the voltage of your batteries or power source
ex: *5Q3<cr>	Temperature limit has been reached	The servo is too hot to continue operating safely.

24. Query Voltage (**QV**)

Ex: #5QV<cr> might return *5QV11200<cr>

The number returned has one decimal, so in the case above, servo with ID 5 has an input voltage of 11.2V (perhaps a three cell LiPo battery).

25. Query Temperature (**QT**)

Ex: #5QT<cr> might return *5QT564<cr>

The units are in tenths of degrees Celcius, so in the example above, the servo's internal temperature is 56.4 degrees C. To convert from degrees Celcius to degrees Farenheit, multiply by 1.8 and add 32. Therefore 56.4C = 133.52F.

26. Query Current (**QC**)

Ex: #5QC<cr> might return *5QC140<cr>

The units are in milliamps, so in the example above, the servo is consuming 140mA, or 0.14A.

27. Configure RC Mode (**CRC**)

This command puts the servo into RC mode (position or continuous), where it will only respond to RC pulses. Note that because this is the case, the servo will no longer accept serial commands. The servo can be placed back into smart mode by using the button menu.

Command sent	**Note**
ex: #5CRC1<cr>	Change to RC position mode.
ex: #5CRC2<cr>	Change to RC continuous (wheel) mode.
ex: #5CRC*<cr>	Where * is any number or value other than 1 or 2 (or no value): stay in smart mode.

EX: #5CRC2<cr>

This command would place the servo in RC wheel mode after a RESET or power cycle. Note that after a RESET or power cycle, the servo will be in RC mode and will not reply to serial commands. Using the command #5CRC<cr> or #5CRC3<cr> which requests that the servo remain in serial mode still requires a RESET command.

Important note: To revert from RC mode back to serial mode, the LSS - Button Menu (/info/wiki/lynxmotion/view/lynxmotion-smart-servo/lss-button-menu/) is required. Should the button be inaccessible (or broken) when the servo is in RC mode and the user needs to change to serial mode, a 5V constant HIGH needs to be sent to the servo's Rx pin (RC PWM pin), ensuring a common GND and wait for 30 seconds. Normal RC PWM pulses should not exceed 2500 milliseconds. After 30 seconds, the servo will interpret this as a desired mode change and change to serial mode. This has been implemented as a fail safe.

28. **RESET**

Ex: #5RESET<cr> or #5RS<cr>

This command does a "soft reset" (no power cycle required) and reverts all commands to those stored in EEPROM (i.e. configuration commands).

29. **DEFAULT** & CONFIRM

Ex: #5DEFAULT<cr>

This command sets in motion the reset of all values to the default values included with the version of the firmware installed on that servo. The servo then waits for the CONFIRM command. Any other command received will cause the servo to exit the DEFAULT function.

EX: #5DEFAULT<cr> followed by #5CONFIRM<cr>

Since it it not common to have to restore all configurations, a confirmation command is needed after a firmware command is sent. Should any command other than CONFIRM be received by the servo after the firmware command has been received, it will exit the command.

Note that after the CONFIRM command is sent, the servo will automatically perform a RESET.

30. **UPDATE** & CONFIRM

Ex: #5UPDATE<cr>

This command sets in motion the equivalent of a long button press when the servo is not powered in order to enter firmware update mode. This is useful should the button be broken or inaccessible. The servo then waits for the CONFIRM command. Any other command received will cause the servo to exit the UPDATE function.

EX: #5UPDATE<cr> followed by #5CONFIRM<cr>

Since it it not common to have to update firmware, a confirmation command is needed after an UPDATE command is sent. Should any command other than CONFIRM be received by the servo after the firmware command has been received, it will leave the firmware action.

Note that after the CONFIRM command is sent, the servo will automatically perform a RESET.

Details - Advanced

The motion controller used in serial mode is not the same as the motion controller use in RC mode. RC mode is intended to add functionality to what would be considered "normal" RC behavior based on PWM input.

A1. Angular Stiffness (**AS**)

The servo's rigidity / angular stiffness can be thought of as (though not identical to) a damped spring in which the value affects the stiffness and embodies how much, and how quickly the servo tried keep the requested position against changes. There are no units.

A positive value of "angular stiffness":

- The more torque will be applied to try to keep the desired position against external input / changes
- The faster the motor will reach its intended travel speed and the motor will decelerate faster and nearer to its target position

A negative value on the other hand:

- Causes a slower acceleration to the travel speed, and a slower deceleration
- Allows the target position to deviate more from its position before additional torque is applied to bring it back

The default value for stiffness depending on the firmware may be 0 or 1. Greater values produce increasingly erratic behavior and the effect becomes extreme below -4 and above +4. Maximum values are -10 to +10.

Ex: #5AS-2<cr>

This reduces the angular stiffness to -2 for that session, allowing the servo to deviate more around the desired position. This can be beneficial in many situations such as impacts (legged robots) where more of a "spring" effect is desired. Upon reset, the servo will use the value stored in memory, based on the last configuration command.

Ex: #5QAS<cr>

Queries the value being used.

Ex: #5CAS<cr>

Writes the desired angular stiffness value to memory.

A2. Angular Holding Stiffness (**AH**)

The angular holding stiffness determines the servo's ability to hold a desired position under load. The default value for stiffness depending on the firmware may be 0 or 1. Greater values produce increasingly erratic behavior and the effect becomes extreme below -4 and above +4. Maximum values are -10 to +10. Note that when considering altering a stiffness value, the end effect depends on the mode being tested.

Ex: #5AH3<cr>

This sets the holding stiffness for servo #5 to 3 for that session.

Query Angular Hold Stiffness (**QAH**)

Ex: #5QAH<cr> might return *5QAH3<cr>

This returns the servo's angular holding stiffness value.

Configure Angular Hold Stiffness (**CAH**)

Ex: #5CAH2<cr>

This writes the angular holding stiffness of servo #5 to 2 to EEPROM. Note that when considering altering a stiffness value, the end effect depends on the mode being tested.

A3: Angular Acceleration (**AA**)

The default value for angular acceleration is 100, which is the same as the maximum deceleration. Accepts values of between 1 and 100. Increments of 10 degrees per second squared.

Ex: #5AA30<cr>

Query Angular Acceleration (**QAD**)

Ex: #5QA<cr> might return *5QA30<cr>

Configure Angular Acceleration (**CAD**)

Ex: #5CA30<cr>

A4: Angular Deceleration (**AD**)

The default value for angular deceleration is 100, which is the same as the maximum acceleration. Values between 1 and 15 have the greatest impact.

Ex: #5AD8<cr>

Query Angular Deceleration (**QAD**)

Ex: #5QD<cr> might return *5QD8<cr>

Configure Angular Deceleration (**CAD**)

Ex: #5CD8<cr>

A5: Motion Control (**EM**)

The command EM0 disables use of the motion controller (acceleration, velocity / travel, deceleration). As such, the servo will move at full speed for all motion commands. The command EM1 enables use of the motion controller.

Note that if the modifiers S or T are used, it is assumed that motion control is desired, and for that command, EM1 will be used.

A6. Configure LED Blinking (**CLB**)

This command allows you to control when the RGB LED will blink the user set color (see 16. RGB LED for details). This is very useful when visually seeing what the servo is doing. You can turn on or off blinking for various LSS status. The command requires that the servo be RESET. Here is the list and their associated value:

Blink While:	**#**
No blinking	0
Limp	1
Holding	2
Accelerating	4
Decelerating	8

Free	16
Travelling	32
Always blink	63

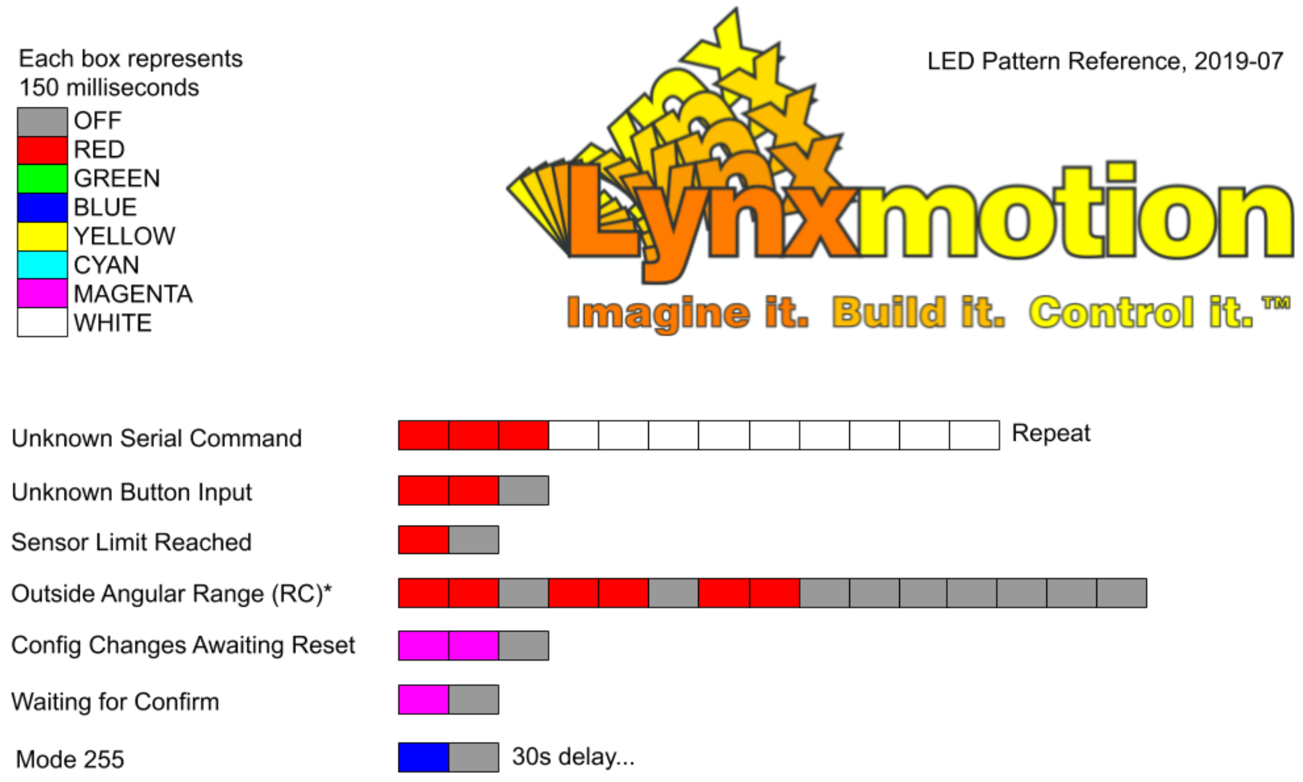
To set blinking, use CLB with the value of your choosing. To activate blinking in multiple status, simply add together the values of the corresponding status. See examples below:

Ex: #5CLB0<cr> to turn off all blinking (LED always solid)
Ex: #5CLB1<cr> only blink when limp (1)
Ex: #5CLB2<cr> only blink when holding (2)
Ex: #5CLB12<cr> only blink when accel or decel (accel 4 + decel 8 = 12)
Ex: #5CLB48<cr> only blink when free or travel (free 16 + travel 32 = 48)
Ex: #5CLB63<cr> blink in all status (1 + 2 + 4 + 8 + 16 + 32)

RESETTING the servo is needed.

RGB LED Patterns

The LED patterns below do not include those which are part of the button menu, which can be found here: LSS Button Menu (/info/wiki/lynxmotion/view/lynxmotion-smart-servo/lss-button-menu/)



Each box represents 150 milliseconds

▨	OFF
🟥	RED
🟩	GREEN
🟦	BLUE
🟨	YELLOW
▢	CYAN
🟪	MAGENTA
⬜	WHITE

LED Pattern Reference, 2019-07

Unknown Serial Command	🟥🟥🟥⬜⬜⬜⬜⬜⬜⬜⬜⬜	Repeat
Unknown Button Input	🟥🟥▨	
Sensor Limit Reached	🟥▨	
Outside Angular Range (RC)*	🟥🟥▨🟥🟥▨🟥🟥▨▨▨▨▨▨▨	
Config Changes Awaiting Reset	🟪🟪▨	
Waiting for Confirm	🟪▨	
Mode 255	🟦▨	30s delay...

Tags: LSS (/info/wiki/lynxmotion/view/Main/Tags?do=viewTag&tag=LSS) communication (/info/wiki/lynxmotion/view/Main/Tags?do=viewTag&tag=communication) protocol (/info/wiki/lynxmotion/view/Main/Tags?do=viewTag&tag=protocol) programming (/info/wiki/lynxmotion/view/Main/Tags?do=viewTag&tag=programming) firmware (/info/wiki/lynxmotion/view/Main/Tags?do=viewTag&tag=firmware) control (/info/wiki/lynxmotion/view/Main/Tags?do=viewTag&tag=control) LSS-Ref (/info/wiki/lynxmotion/view/Main/Tags?do=viewTag&tag=LSS-Ref)

Created by RB1 (/info/bin/view/XWiki/RB1) on 2019/03/20 15:44