LSS - Communication Protocol

Last modified by Eric Nantel (/info/bin/view/XWiki/ENantel) on 2021/05/07 14:51

Page Contents

Serial Protocol

The 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 staying compact and robust yet highly versatile. The protocol was based on Lynxmotion's SSC-32 & SSC-32U RC servo controllers and almost everything one might expect to be able to configure for a smart servomotor is available.

In order to be able to control each servo individually with commands, the first step should be to assign a different ID number to each servo (see details on the Configure ID, or "CID" command here). Only the servo(s) which have been configured to a specific ID will act on a command sent to that ID. There is currently no CRC or 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 1: For a given session, the action related to a specific command overrides the stored value in EEPROM.

Note 2: During the power-on / reset process the LSS cannot accept commands for a small amount of time (1.25 s).

Note 3: You can ensure the LSS is ready by using a query command to check for response (ex: #[id]Q\r or #[id]QID\r described below). If the LSS is ready for commands (initialized) it will respond to the query. A timeout between 50-100 ms is recommended to compensate for drivers, OS and buffering delays.

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). 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 **#** (Unicode Character: U+0023)
2. Servo ID number as an integer (assigning an ID described below)
3. Action command (one or more letters, no whitespace, capital or lowercase from the list below)
4. Action value in the correct units with no decimal
5. End with a carriage return **\r** or **<cr>** Unicode Character (U+000D)

Ex: #5D1800<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 (1800 in tenths of degrees) of 180.0 degrees. Any servo on the bus which does not have ID 5 will take no action when receiving this command.

Modifiers

Modifiers can only be used with certain **action commands**. The format to include a modifier is:

1. Start with a number sign **#** (Unicode Character: U+0023)
2. Servo ID number as an integer
3. Action command (one to three letters, no spaces, capital or lowercase from a subset of action commands below)
4. Action value in the correct units with no decimal
5. Modifier command (one or two letters from the list of modifiers below)
6. Modifier value in the correct units with no decimal
7. End with a carriage return **\r** or **<cr>** Unicode Character (U+000D)

Ex: #5D1800T1500<cr>

This results in the servo with ID #5 rotating to a position (1800 in tenths of degrees) of

180.0 degrees in a time ("T") of 1500 milliseconds (1.5 seconds).

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 **#** (Unicode Character: U+0023)
2. Servo ID number as an integer
3. Query command (one to four letters, no spaces, capital or lower case)
4. End with a carriage return **\r** or **<cr>** Unicode Character (U+000D)

Ex: #5QD<cr> Query the position in (tenth of) degrees for servo with ID #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 * (Unicode Character: U+0023)
2. Servo ID number as an integer
3. Query command (one to four letters, no spaces, capital letters)
4. The reported value in the units described, no decimals.
5. End with a carriage return **\r** or **<cr>** Unicode Character (U+000D)

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 to multiple servos 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 query command. A reply to the query sent above might be:

Ex: *5QD1800<cr>

This indicates that servo #5 is currently at 180.0 degrees (1800 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. This means that if two of the same configuration commands 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 **#** (Unicode Character: U+0023)
2. Servo ID number as an integer
3. Configuration command (two to four letters, no spaces, capital or lower case)
4. Configuration value in the correct units with no decimal
5. End with a carriage return **\r** or **<cr>** Unicode Character (U+000D)

Ex: #5CO-50<cr>

This configures an absolute origin offset ("CO") with respect to factory origin of 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 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 that clears all configurations (through the button menu or with DEFAULT command described below).

Session vs Configuration Query

By default, the query command returns the session's 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.

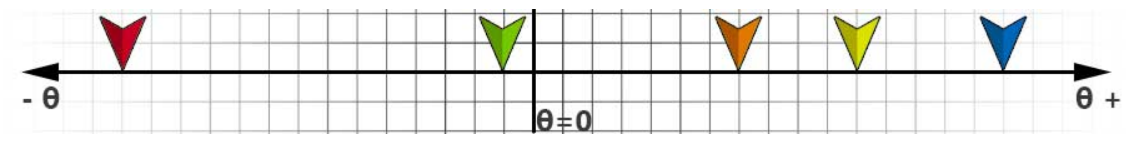
After RESET, a command of #5SR4<cr> sets the session's speed to 4rpm, but does not change the configuration value in memory. Therefore:

#5QSR<cr> or #5QSR0<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. 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 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). The virtual position range at power-up is [-180.0°, 180.0°].

Command List

Latest firmware version currently : 368.29.14

Communication Setup

Description	Action	Query	Config	RC	Serial	Default	Unit	Notes
Reset	RESET				✓			Soft reset. See command for details.
Default Configuration	DEFAULT				✓			Revert to firmware default values. See command for details
Firmware **Update** Mode	UPDATE				✓			Update firmware. See command for details.
Confirm	CONFIRM				✓			

Changes							
Change to **RC**			CRC		✓		Change to RC mode 1 (position) or 2 (wheel).
ID #		QID	CID		✓	0	Reset required after change. ID 254 is a "broadcast" which all servos respond to.
Baudrate		QB	CB		✓	115200	Reset required after change.

Motion

Description	Action	Query	Config	RC	Serial	Default	Unit	Notes
Position in **D**egrees	D	QD/QDT			✓		1/10°	
Move in **D**egrees (relative)	MD				✓		1/10°	
Wheel mode in **D**egrees	WD	QWD/QVT			✓		°/s	A.K.A. "Speed mode" or "Continuous rotation"
Wheel mode in **R**PM	WR	QWR			✓		RPM	A.K.A. "Speed mode" or "Continuous rotation"
Position in **P**WM	P	QP			✓		us	Inherited from SSC-32 serial protocol

Move in PWM (relative)	M				✓	us	
Raw **D**uty-cycle **M**ove	RDM	QMD			✓	-1023 to 1023 integer	Positive values : CW / Negative values : CCW
Query Status		Q			✓	1 to 8 integer	See command description for details
Limp	L				✓		
Halt & Hold	H				✓		

Motion Setup

Description	Action	Query	Config	RC	Serial	Default	Unit	Notes
Enable **M**otion Profile	EM	QEM	CEM		✓	1		EM1: trapezoidal motion profile / EM0: no motion profile
Filter **P**osition **C**ount	FPC	QFPC	CFPC	✓	✓	5		Affects motion only when motion profile is disabled (EM0)
Origin Offset	O	QO	CO	✓	✓	0	1/10°	
Angular **R**ange	AR	QAR	CAR	✓	✓	1800	1/10°	
Angular **S**tiffness	AS	QAS	CAS	✓	✓	0	-4 to +4 integer	Suggested values are between 0 to +4
Angular **H**olding	AH	QAH	CAH	✓	✓	4	-10 to +10	

Stiffness							integer	
Angular **A**cceleration	AA	QAA	CAA		✓	100	°/s^2	Increments of 10°/s^2. Only when motion profile is enabled (EM1).
Angular **D**eceleration	AD	QAD	CAD		✓	100	°/s^2	Increments of 10°/s^2. Only when motion profile is enabled (EM1).
Gyre Direction	G	QG	CG	✓	✓	1		Gyre / rotation direction: 1= CW (clockwise) -1 = CCW (counter-clockwise)
First Position (**D**eg)		QFD	CFD	✓	✓	No value	1/10°	Reset required after change.
Maximum **M**otor **D**uty	MMD	QMMD			✓	1023	255 to 1023 integer	
Maximum **S**peed in **D**egrees	SD	QSD	CSD	✓	✓	Max	°/s	SD overwrites SR / CSD overwrites CSR and vice-versa
Maximum **S**peed in **R**PM	SR	QSR	CSR	✓	✓	Max	RPM	SD overwrites SR / CSD overwrites CSR and vice-versa

Modifiers

Description	Modifier	Query	Config	RC	Serial	Default	Unit	Notes
Speed	S	QS			✓		uS/s	For P action command
Speed in **D**egrees	SD				✓		°/s	For D and MD action commands
Timed move	T				✓		ms	Modifier only for P, D and MD. Time can change based on load
Current **H**old	CH				✓		mA	Modifier for D, MD, WD and WR
Current **L**imp	CL				✓		mA	Modifier for D, MD, WD and WR

Telemetry

Description	Action	Query	Config	RC	Serial	Default	Unit	Notes
Query **V**oltage		QV			✓		mV	
Query **T**emperature		QT			✓		1/10°C	
Query **C**urrent		QC			✓		mA	
Query **M**odel **S**tring		QMS			✓			Returns the model of servo (ex: LSS-ST1, LSS-HS1, LSS-HT1)

Query **F**irmware Version		QF			✓			
Query Serial **N**umber		QN			✓			Returns the unique serial number for the servo

RGB LED

Description	Action	Query	Config	RC	Serial	Default	Unit	Notes
LED Color	LED	QLED	CLED	✓	✓		0 to 7 integer	0=Off; 1=Red; 2=Green; 3=Blue; 4=Yellow; 5=Cyan; 6=Magenta; 7=White
Configure **L**ED **B**linking			CLB	✓	✓		0 to 63 integer	Reset required after change. See command for details.

Details

Communication Setup

Reset

Ex: #5RESET<cr>

This command does a "soft reset" and reverts all commands to those stored in EEPROM (i.e. configuration commands). Note: after a RESET command is received, the LSS will restart and perform initilization again, making it unavailable on the bus for a bit. See Session, note #2 for more details.

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: After the CONFIRM command is sent, the servo will automatically perform a RESET.

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: After the CONFIRM command is sent, the servo will automatically perform a RESET.

Confirm

Ex: #5CONFIRM<cr>

This command is used to confirm changes after a Default or Update command.

Note: After the CONFIRM command is sent, the servo will automatically perform a RESET.

Configure RC Mode (**CRC**)

This command puts the servo into RC mode (position or continuous), where it will only respond to RC PWM signal on the servo's Rx pin. In this mode, 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 rotation (wheel) mode.
ex: #5CRC*<cr>	Where * is any 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.

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.

Query Identification (**QID**)

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

When using the broadcast 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. 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 that have 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.

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 command, all servos in a project should 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 115200.

Query Baud Rate (**QB**)

Ex: #5QB<cr> might return *5QB115200<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.

Motion

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) could also be used. A full circle would be from -1800 to 1800 degrees. A value of 2700 would be the same angle (absolute position) 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 (explained above).

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 virtual position 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 with the last target position used.

(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.

Wheel Mode in Degrees (**WD**)

Ex: #5WD90<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 *5QWD90<cr>

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

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).

Position in PWM (**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 signal and is further explained in the SSC-32 and SSC-32U manuals (https://www.robotshop.com/media/files/pdf2/lynxmotion_ssc-32u_usb_user_guide.pdf#page=24) . Without any modifications to configuration considered, and a ±90.0 degrees standard range where 1500 microseconds is centered, a PWM signal 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).

(Relative) Move in PWM (**M**)

Example: #5M1500<cr>

The relative move in PWM command causes the servo to read its current position and move by the specified number of PWM signal. For example if the servo is set to rotate CW (default) and an M command of 1500 is sent to the servo, it will cause the servo to rotate clockwise by 90 degrees. Negative PWM value would cause the servo to rotate in the opposite configured direction.

Raw Duty-cycle Move (**RDM**)

Example: #5RDM512<cr>

The raw duty-cycle move command (or free move command) will rotate the servo at a specified duty cycle value in wheel mode (a.k.a. "continuous rotation") like a geared DC motor.

The duty values range from 0 to 1023. Negative values will rotate the servo in the opposite direction (for factory default a negative value would be counter clockwise).

Query Move in Duty-cycle (**QMD**)

Example: #5QMD<cr> might return *5QMD512

This command queries the raw duty-cycle move value. 512 value means that the motor is rotating at 50% duty-cycle.

Query Status (**Q**)

The status query describes what the servo is currently doing. The query returns an integer which must be looked up in the table below.

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	Servo is rotating in duty motion / free move using the RDM command
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.

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>.

Halt & Hold (**H**)

Example: #5H<cr>

This command causes the servo to stop immediately and hold that angular position. It overrides whatever the servo might be doing at the time the command is received

(accelerating, travelling, deccelerating, etc.)

Motion Setup

Enable Motion Profile (**EM**)

EM1 (Enable Motion Profile #1) is the default mode of the LSS and is an easy way to control the servo's position with a single (serial) position command. This mode uses a trapezoidal motion profile which takes care of acceleration, constant speed travel and deceleration. Once the actual position is within a certain value of the target, it switches to a holding algorithm. The LSS commands for Angular Acceleration and Deceleration (AA/CAA/AD/CAD) Angular Stiffness (AS/CAS) and Angular holding stiffness (AH/CAH) affect this motion profile. Modifiers like SD/S and T can be used in EM1.

Ex: #5EM1<cr>

This command enables a trapezoidal motion profile for servo #5

Ex: #5EM0<cr>

This command will disable the built-in trapezoidal motion profile. As such, the servo will move at full speed to the target position using the D/MD action commands. Modifiers like SD/S or T cannot be used in EM0 mode. By default the Filter Position Counter, or "FPC" is active in EM0 mode to smooth out its operation. EM0 is suggested for applications where an external controller will be determining all incremental intermediate positions of the servo's motion, effectively replacing a trajectory manager. To prevent having to send position commands continuously to reach the desired position in EM0/FPC active (FPC >= 2), an internal position engine (IPE) repeats the last position command.

Query Motion Profile (**QEM**)

Ex: #5QEM<cr> might return *5QEM1<cr>

This command will query the motion profile. **0:** motion profile disabled / **1:** trapezoidal motion profile enabled.

Configure Motion Profile (**CEM**)

Ex: #5CEM0<cr>

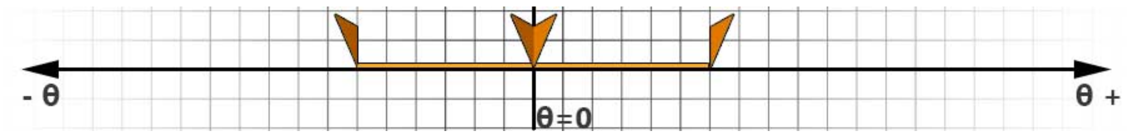
This command configures the motion profile and saves it in the EEPROM. The setting will be saved upon servo reset / power cycle.

Filter Position Count (**FPC**)

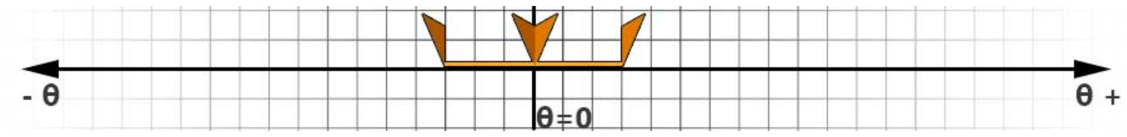
The FPC value relates to the depth of a first order filter (exponential weighted average) over the position change. This has the effect of slowing down both acceleration and deceleration while still allowing the LSS to try to reach the desired position at maximum power at all times. A smaller FPC value will reduce the smoothing effect and a larger value will increase it. To prevent having to send position commands continuously to reach the desired position in EM0/FPC active (FPC >= 2), an internal position engine (IPE) has been put in place, which is also active by default.

Ex: #5FPC10<cr>

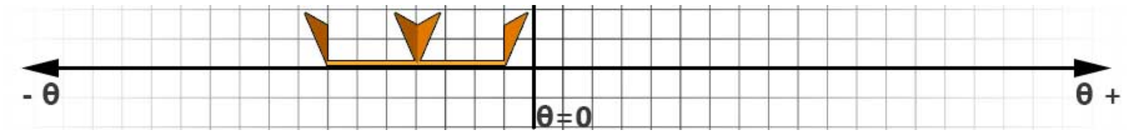
This command allows you to temporarily change the total angular range of the servo in tenths of degrees. This applies to the Position in Pulse (P) command and RC mode. The default for (P) and RC mode is 1800 (180.0 degrees total, or ±90.0 degrees). The image below shows a standard -180.0 to +180.0 range, 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.

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 higher 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 lower 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-2<cr>

Writes the desired angular stiffness value to EEPROM.

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.

Ex: #5AH3<cr>

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

Query Angular Holding Stiffness (**QAH**)

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

This returns the servo's angular holding stiffness value.

Configure Angular Holding Stiffness (**CAH**)

Ex: #5CAH2<cr>

This writes the angular holding stiffness of servo #5 to 2 to EEPROM.

Angular Acceleration (**AA**)

The default value for angular acceleration is 100. Accepts values of between 1 and 100. Increments of 10 degrees per second squared.

Ex: #5AA30<cr>

This sets the angular acceleration for servo #5 to 30 degrees per second squared ($°/s^2$).

Query Angular Acceleration (**QAA**)

Ex: #5QAA<cr> might return *5QAA30<cr>

This returns the servo's angular acceleration in degrees per second squared ($°/s^2$).

Configure Angular Acceleration (**CAA**)

Ex: #5CAA30<cr>

This writes the angular acceleration of servo #5 to 30 degrees per second squared (°/s^2) to EEPROM.

Angular Deceleration (**AD**)

The default value for angular deceleration is 100. Accepts values of between 1 and 100. Increments of 10 degrees per second squared.

Ex: #5AD30<cr>

This sets the angular deceleration for servo #5 to 30 degrees per second squared (°/s^2).

Query Angular Deceleration (**QAD**)

Ex: #5QAD<cr> might return *5QAD30<cr>

This returns the servo's angular deceleration in degrees per second squared (°/s^2).

Configure Angular Deceleration (**CAD**)

Ex: #5CAD30<cr>

This writes the angular deceleration of servo #5 to 30 degrees per second squared (°/s^2) to EEPROM.

Gyre 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. By default: CW = 1; CCW = -1.

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. Sending a #5WR30 command will rotate the servo in a counter-clockwise gyration at 30 RPM.

Configure Gyre (**CG**)

Ex: #5CG-1<cr>

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

First Position

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. Note that the number should be restricted to -1790

(-179.0 degrees) to +1790 (179.0 degrees) and values beyond this will be changed to 1800.

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

Ex: #5QFD<cr> might return *5QFD900<cr>

The reply above indicates that servo with ID 5 has a first position of 90.0 degrees. If there is no first position value stored, the reply will be DIS.

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

Ex: #5CFD900<cr>

This configuration command means the servo, when set to smart mode, will immediately move to 90.0 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>

Maximum Motor Duty (**MMD**)

This command allows the user to limit the duty cycle value sent from the servo's MCU to the DC Motor driver. The duty cycle limit value can be within the range of 255 to 1023. The default value is 1023. A typical use-case for this command is active compliance.

Ex: #5MMD512<cr>

This will set the duty-cycle to 512 for servo with ID 5 for that session.

Query Maximum Motor Duty (**QMMD**)

Ex: #5QMMDD<cr> might return *5QMMD512<cr>

This command returns the configured limit of the duty cycle value sent from the servo's MCU to the Motor Controller. The default value is 1023.

Maximum 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.

Maximum Speed in RPM (**SR**)

Ex: #5SR45<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. SR overrides CSR (described below) for that session. Upon reset or power cycle, the servo reverts to the value associated with CSR 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 RPM (**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 QWD)
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.

Modifiers

Speed (**S**, **SD**) modifier

Example: #5P1500S750<cr>

Modifier (S) is 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.

Example: #5D0SD180<cr>

Modifier (SD) is only for a position (D) or relative position (MD) action and determines the speed of the move in degrees per second. A speed modifier (SD) of 180 would cause the servo to rotate from its current position to the desired absolute or relative position at a speed of 180 degrees per second.

Query Speed (**QS**)

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

This command queries the current speed in microseconds per second.

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.

Current Halt & Hold (**CH**) modifier

Example: #5D1423CH400<cr>

This has servo with ID 5 move to 142.3 degrees but, should it detect a current of 400mA or higher before it reaches the desired position, will immediately halt and hold position.

This modifier can be added to the following actions: D; MD; WD; WR.

Current Limp (**CL**) modifier

Example: #5D1423CL400<cr>

This has servo with ID 5 move to 142.3 degrees but, should it detect a current of 400mA or higher before it reaches the desired position, will immediately go limp.

This modifier can be added to the following actions: D; MD; WD; WR.

Telemetry

Query Voltage (**QV**)

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

The number returned is in milliVolts, so in the case above, servo with ID 5 has an input voltage of 11.2V.

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.

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.

Query Model String (**QMS**)

Ex: #5QMS<cr> might return *5QMSLSS-HS1<cr>

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

Query Firmware (**QF**)

Ex: #5QF<cr> might return *5QF368<cr>

The number in the reply represents the firmware version, in this example being 368.

The command #5QF3<cr> can also be sent and the servo will reply with a 3 numbers firmware version, for example, 368.29.14

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.

RGB LED

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**)

Ex: #5CLED3<cr>

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. The command above will configure the servo's LED to a Blue color.

Configure LED Blinking (**CLB**)

This command allows you to control when the RGB LED will blink the user set color (see RGB LED command 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 to turn off all blinking (LED always solid)

Ex: #5CLB1 only blink when limp (1)

Ex: #5CLB2 only blink when holding (2)

Ex: #5CLB12 only blink when accel or decel (accel 4 + decel 8 = 12)

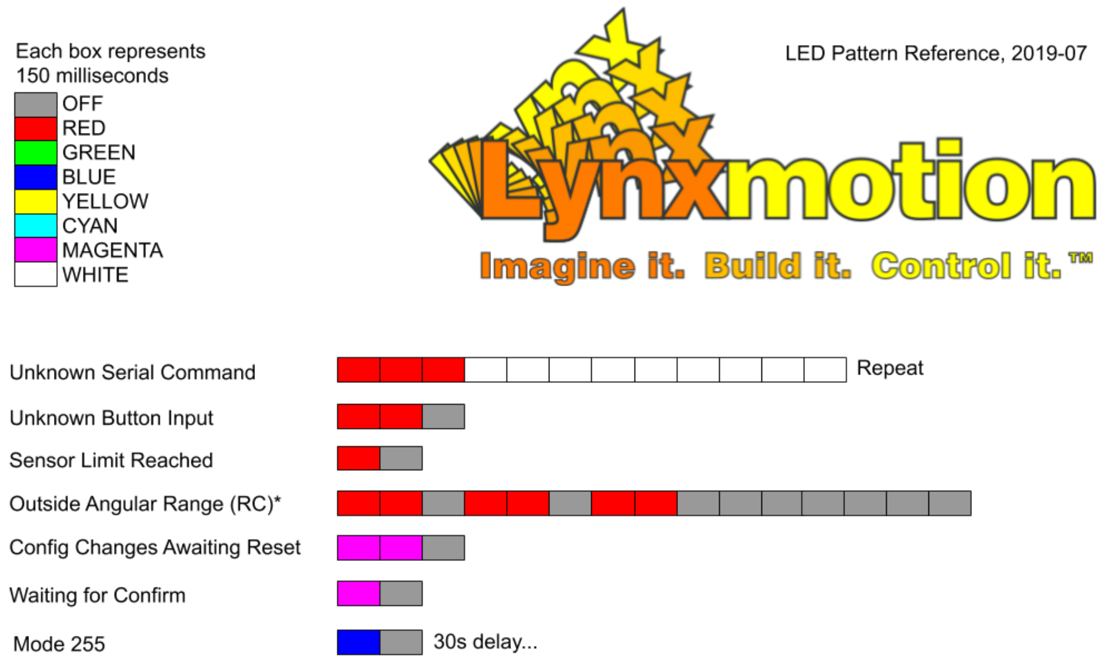
Ex: #5CLB48 only blink when free or travel (free 16 + travel 32 = 48)

Ex: #5CLB63 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
Lynxmotion
Imagine it. Build it. Control it.™
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

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

(/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)