EZ Servo®
Command set and Communications protocol

Command Set For Servo Models: EZSV23, EZSV17

Document Revision: A37 5/5/06

INDEX

EZ Servo®
Command set and Communications protocol

Overview:
This document describes the operation and command set for the EZServo® series of motor drives.

Command syntax:
Commands to the EZServo are single alpha characters normally followed by a numeric value. The alpha character represents "what to do" and the numeric value represents "how much to do it".
You can set values for desired velocities, accelerations, and positions. Commands can be issued one at a time or sent in a group. This allows the setting of all move parameters in one command. You can also create loops in the strings and cause the EZ Stepper or EZServo to become a stand-alone device that responds to switch inputs. Finally, storing such strings into the onboard EEPROM allows the EZServo to power up into a mode of your choice, so that it can act with no computer attached.
The Commands are simply typed into a Terminal Program such as "Hyperterminal", no special software is required. The EZServo can even be commanded from a serial enabled PDA.

Command Set: **(Also see examples on page 12)**

Command (**Case sensitive**)	Operand	Description
		POSITIONING COMMANDS
A	0-(2^31)	**Move Motor to absolute (** units are quadrature encoder ticks- 32 Bit Positioning**). Eg /1A1000R**
P	0-(2^31)	**Move Motor relative in positive direction.** **(** units are quadrature encoder ticks**) Eg/1P1000R** A value of zero will cause an endless forwards move at speed V. (i.e. enter into Velocity Mode) The Velocity can then be changed on the fly by using the "V" command. An endless move can be terminated by issuing a T
D	0-(2^31)	**Move Motor relative in negative direction.** **(** units are quadrature encoder ticks**)** (Note: for a finite move, the ending Absolute position must be greater than zero). A value of zero for the operand will cause an endless backwards move at speed V. (i.e. enter into Velocity Mode). The Velocity can then be changed on the fly by using the V command. An endless move can be terminated by issuing a T
B	0-(2^31)	Sets Distance moved in "Bump Jog Mode" see "n" command.

EZ Servo®
Command set and Communications protocol

		HOMING COMMANDS
Z	**0-(2^31)** (400)	**Home/Initialize Motor.** Motor will turn toward 0 until the home opto sensor (opto #1) is interrupted. If already interrupted it will back out of the opto and come back in until re-interrupted. Current motor position is set to zero. See Appendix 2 for further details. **Eg /1Z300000R**
z	**0-(2^31)**	**Change current position without moving.** Sets both Encoder and Commanded position to value given, without moving the motor. **Eg /1z1000R**
r	**0 or 1** (0)	**Recover Servo** Turns the Servo on or off to current encoder position. This is useful in the case where an overload error has occurred and the servo has automatically shut down, but the encoder count is still valid. Operation may be resumed without re-homing. Typically this command would be used in conjunction with the n512 mode in a recovery script in case the servo detects an overload and times out.
f	**0 or 1** (0)	**Home Flag and Limit polarity.** Sets polarity of home sensor, default value is 0. **Eg /1f1R** When f=0 limits are activated when limit inputs are high. When f=1 limits are activated when limit inputs are low
		SET VELOCITY COMMANDS
V	**1- (2^31)** (3,000,000)	**In Position Mode (N1) Set Max/Slew Speed of motor** For Servo version with an encoder, sets (encoder ticks/32.768) per second. (Eg Value of 33 will cause the motor to spin at approx 1 encoder tick/second)
V	**1-1000** (0)	**In Velocity Mode (N0) Set Spin Speed Of Motor** For Servo BLDC Version with only Hall Sensor Feedback (i.e. no encoder). Sets speed in Revs/Second. Use P0 or D0 Command to get started, and set direction.
U	**1- (2^31)** (3,000,000)	**Same as V command for Position Mode except sets velocity in negative direction.** This must be set after setting V, any subsequent setting of V will set both U and V to be the same.
		SET ACCELERATION COMMANDS
L	**0-65000** (4000)	In SV (Servo) version with encoder sets Acceleration in (encoder ticks/32.768) per second squared.

EZ Servo®
Command set and Communications protocol

		LOOPING AND BRANCHING COMMANDS
g		**Beginning of a repeat loop** marker. See examples below on how to set up a loop.
G	0-30000	**End of a repeat loop** marker. Loops can be nested up to 4 levels. A value of 0 causes the loop to be infinite. (Requires T command to Terminate). If no value is specified 0 is assumed. **Eg /1gP1000M1000G10R**
M	0-29000	Wait for this number of milliseconds Eg. **/1gA0M1000A1000M1000G0R** will go back and forth between 0 and 1000 and wait 1 second at each position. (Accurate to +/-10mS)
H	 01 11 02 12 03 13 04 14	**Halt** current command string and wait until condition specified. Wait for low on input 1 (Switch 1) Wait for high on input 1 (Switch 1) Wait for low on input 2 (Switch 2) Wait for high on input 2 (Switch 2) Wait for low on input 3 (Opto 1) Wait for high on input 3 (Opto 1) Wait for low on input 4 (Opto 2) Wait for high on input 4 (Opto 2) If Halted operation can also be resumed by typing /1R Also see "S" command for I/O dependant program execution. If an edge detect is desired, a look for Low and a look for Hi can be placed adjacent to each other eg H01H11 is a rising edge detect. **Eg /1gH12H02P1000G0R** will advance the motor 1000 steps every time the switch #2 is pressed.

S		**Skip next instruction depending on status of switch.**
	01	Skip next instruction if low on input 1 (Switch 1)
	11	Skip next instruction if high on input 1 (Switch 1)
	02	Skip next instruction if low on input 2 (Switch 2)
	12	Skip next instruction if high on input 2 (Switch 2)
	03	Skip next instruction if low on input 3 (Opto 1)
	13	Skip next instruction if high on input 3 (Opto 1)
	04	Skip next instruction if low on input 4 (Opto 2)
	14	Skip next instruction if high on input 4 (Opto 2)
		Program branching to a complex subroutine can be implemented by making the next instruction a stored string execution. (See examples). Loops can be escaped by branching to a stored string with no commands.
		Also see "H" command for I/O dependant program execution.
i	0	**Skip on status of 2 switches.**
	1	The "i" command interprets 2 switches as a combined
	2	binary number, with values 0,1,2,3
	3	The next instruction is skipped unless the number is what is read back.
		Eg /1d1000gi0A1000i1A2000i2A2000i3A3000G0R
		It is possible to implement a debounce on this command in case the switches do not simultaneously change by use of the "d" or debounce command. Allowed values of "d" are 1-30000.
d	1-30000	**Debounce** -Number of times the binary pattern on the 2 switches must be constant for prior to acting upon the "i." command.
		PROGRAM STORAGE AND RECALL
s	0-15	**Stores a program** 0-3 or 0-15 depending on model, Program 0 is executed on power up. (25 full commands max per string). Note: This command takes approx 1 Second to write to the EEPROM.
e	0-15	**Executes Stored Program** 0-15.

EZ Servo®
Command set and Communications protocol

		PROGRAM EXECUTION
R		**Run** the command string that is currently in the execution buffer.
X		**Repeat Run** the current command string
		SET MAX MOVE CURRENT / PID CONSTANTS / TORQUE MODE / HOLD CURRENT
m	**0-100** (50)	**Sets Max current allowed during a move.** 100% = 5A for EZSV23 . (Default is 50%) 100% = 2A for EZSV17 . (Default is 50%)
h	**0-50** (0)	**Sets Current in Torque mode.** 100% = 5A for EZSV23 . (Default is 0) 100% = 2A for EZSV17 . (Default is 0) When a value is written in here the servo automatically enters Torque mode. Issue P0 or D0 to change direction.
u	**1-25000** (25000)	**Overload Timeout (Milliseconds)** When the Servo detects an overload condition it will shut down after this number of Milliseconds. Default 25000. See appendix 8 for recovery from an overload .
au	**1-65000** (100)	**Overload Retry Count** When the n modes 512 and 1024 are activated to retry a stalled move, then that move must complete before this number of retries. After this number of retries is exhausted then stored program 12 is executed. See appendix 8 for recovery from an overload .
ar	**5073**	**Processor Reset** This command will initiate a Processor Reset the same as that which happens on power up. 5073 is chosen to avoid inadvertent resets. Eg: /1ar5073R<CR>
w	**0-65530** (1000)	**Set Servo Proportional gain.** (Default value is 1000, and is stable with most motors, when equipped with 400-1000 line encoders – 1600-4000 quadrature ticks). Default is 1000. See Appendix 6
x	**0-65530** (0)	**Set Servo Set Integral gain.** (Default value is 0, and is stable with most motors, when equipped with 400-1000 line encoders – 1600-4000 quadrature ticks). Default is 0. See Appendix 6
y	**0-65530** (2500)	**Set Servo Set Differential gain.** (Default value is 2500, and is stable with most motors, when equipped with 400-100 line encoders – 1600-4000 quadrature ticks). Default is 2500. See Appendix 6

		SPECIAL MODES COMMANDS
j	**2, 4, 8, 16, 32, 64, 128, 256** (256)	(Future Release) Pulses Received on the Step and direction inputs are multiplied by this number and divided by 256. The Resulting number is then added/ subtracted from the current position measured in encoder ticks.
N	**0-3** (1)	**Sets Modes – Interpret as combination of Binary Bits** 0 = No Encoder – Uses Hall Sensors for Velocity Control. 1 = Encoder With No Index (Default). Homes to Opto. 2 = Encoder With Index. Homes to Index. Note: Hall sensors can be wired as an encoder for crude position control. (Must Be 5Volt). See Appendix 5 3 = Uses Potentiometer 1 as an encoder. See Appendix 9.
n	**0-65535** (0)	**Sets Modes – Interpret as combination of Binary Bits** **Bit0 (LSB)** : /1n1R Enable Pulse Jog Mode. Jog distance is given by "B" command. Velocity is given by "V" command. The Switch Inputs become the Jog Inputs. **Bit1 :** /1n2R Enable Limits. (The two optos become limits switches). The polarity of the limits is set by the "f" command. The Home opto is the lower limit. **Bit2 :** /1n4R Enable Continuous Jog Mode. Continuous run of motor while switch is depressed. Velocity is given by "V" command. Note that the jog mode allows moves below zero, which will be interpreted by any subsequent "A" command as a large positive number. If this is undesirable, please use the "z" command to define zero position to be some positive number so that underflow will not occur. **Bit3 :** /1n8R Reserved **Bit4 :** /1n16R Reserved **Bit5 :** /1n32R Enable Step And Direction Mode if (1) or enable Dual Encoder Mode if (0) **Eg /1n96<CR>** (96=32+64) Enables step and dir mode and slaves the motor to it. **Bit6 :** / 1n64R Enable Motor slave to encoder/step-dir. **Bit7 :** /1n128R Reserved Encoder or Step and direction Input. (SV23 Only) **Bit8 :** /1n256R When Set this bit will disable the response from the servo. (SV23 Only) **Bit9 and Bit10:** When set these bit will execute one of the stored programs 13, 14 or 15 whenever the servo shuts down due to an overload or an error. /1n512R will execute program 13

		/1n1024R will execute program 14 /1n1536R will execute program 15 See appendix 8 for an example. **Bit11** /1n2048R Reserved. **Bit12 :** /1n4096R Reserved. **Bit13:** /1n8192R Uses Potentiometer 2 to command the motion of the motor. See Appendix 9. **Bit14:** /1n16384R When Set this bit will kill any move if switch 1 is pushed. **Bit15:** /1n32768R When Set this bit will kill any move if switch 2 is pushed. **Bit16:** /1n65536R When Set Potentiometer on Opto 2 input will set velocity. (Joystick Mode)
b	**9600** **19200** **38400** (9600)	**Adjustable baud rate** **Eg /1b19200R** This command will usually be stored as program zero and execute on power up. Default baud rate is 9600.
p	**0-65000**	**Ping Command (lower case "p")** This command will send a numeric message back to the host, when that point in the string is reached. **Eg /1gA1000p3333A0G0R** Will send the number 3333 every time through the loop. Care must be taken when using this command because it can tie up he 485 bus.
		ON/OFF POWER DRIVER
J	**0-3** (0)	On/Off Driver – Interpret as 2 bit Binary Value, 3=11= Both Drivers On, 2=10=Driver2 on Driver 1 Off etc. Note: Only the EZSV23 Has the On/Off Drivers
K	**0-65536** (0)	(Future Release) Pulse Width Modulation of On/Off Drivers. The number is interpreted as 2 separate Bytes. Each Byte programming one of the On/Off Drivers, to a value from 0-255. (Drivers Must be derated to 0.5A in this mode, due to switching losses)
		DEVICE RESPONSE PACKET
		See Appendix 7 for detailed description of device response to commands.

		ANALOG TO DIGITAL CONVRTR COMMANDS
		All 4 Inputs are ADC inputs and can be read and acted upon by the program. Please see Appendix 9
at	**100000-416368**	Thresholds where the Drive calls a one or a zero for each input is varied by this command.
?aa		Reads back all 4 Input ADC Values **E.g. /1?aa<CR>** The Readback order is channels 4:3:2:1
at	**100000 to 116368 200000 to 216368 300000 to 316368 400000 to 416368** (6144)	The "at" command sets the threshold, upon which a "one" or "zero" is called for each of the 4 channels. The Number represents the channel number followed by a 5 digit number from 00000-16368 which represents the threshold on a scale from a 0-3.3V. The default values are 6144 for all 4 channels which represents 1.24V. Changing the threshold allows the H and S commands to work on a variable analog input value which essentially allows the program to act upon an analog level. **Eg /1at106144R** sets the threshold of channel 1 to 6144, Note that leading zeros are required for the threshold value which is always 5 digits plus the channel number.
?at		Reads back the thresholds for all 4 channels. The Readback order is channels 4:3:2:1 **Eg /1?at<CR>**
		POTENTIOMETER POSITION COMMAND
		The motion of the Servo can be slaved to value read from Potentiometer2. Please see appendix 9.
ao	**0-20000** (0)	After multiplication by the am value this offset is added to obtain the position command. **Eg /1ao1000R<CR>**
am	**0-20000** (256)	The Pot value is multiplied by this value and divided by 256 to get the position command. **Eg /1am512R<CR>**
ad	**0-20000** (50)	Sets a deadband around the pot value used for the last move, which must be exceeded before a new move command is issued. **Eg /1ad100R<CR>**

		POTENTIOMETER VELOCITY COMMAND
		(Joystick Mode) - Future Command not yet available - The velocity of the Servo can be slaved to value read from Potentiometer2. Please see appendix 9. Typically type /1n65536zP0R to enter this mode
am	**0-120000** (256)	The Pot value is multiplied by this value and divided by 256 to get the position command. **Eg /1am512R<CR>**
ad	**0-20000** (50)	Sets a deadband around the mid pot value, which must be exceeded before movement will start **Eg /1ad100R<CR>**
z	**0-16384** (current pot value)	Sets pot value for zero velocity. i.e. midpoint of joystick. **Eg /1z8000R<CR>** If zero value is specified, the current value of the pot at the time of issuing this command will be used as the midpoint of the pot. **Eg /1z0R**

Hardware protocol:
The EZ Servo communicates over the RS485 bus at 9600 baud, 1 Stop bit, No Parity, No flow control.

EZ Servo®
Command set and Communications protocol

Commands Below are "Immediate" Commands, and cannot be cascaded in strings or stored. These commands execute while others commands are running.

		IMMEDIATE QUERY COMMANDS
T		Terminate current command or loop. **(Eg: /1T <CR>)**
?	0	Returns the current Commanded motor position **/1?0<CR>**
?	1	Reserved
?	2	Returns the current Slew/Max Speed for Position mode
?	3	Reserved
?	4	Returns the status of all four inputs, 0-15 representing a 4 bit binary pattern. **Eg /1?4<CR>** Bit 0 = Switch1 , Bit 1 = Switch2 Bit 2 = Opto 1, Bit 3 = Opto 2
?	5 , 6, 7	Reserved
?	8	Returns Encoder Position. (can be zeroed by "z" command)
?	9	Erases all stored commands in EEPROM.
?	10	(Future Release) Returns Second Encoder or Step and Direction Counter Readback.
?	w	Returns Proportional Gain Value **Eg /1?w<CR>**
?	x	Returns Integral Gain Value **Eg /1?x<CR>**
?	y	Returns Differential Gain Value **Eg /1?y<CR>**
&		Returns the current Firmware revision and date **/1&<CR>**
Q		Query current status of EZServo Returns the Ready/Busy status as well as any error conditions in the "Status" byte of the return string. **The Return string consists of the start character (/), the master address (0) and the status byte.** Bit 5 of the status byte is set when the EZServo is ready to accept commands. It is cleared when the EZServo is busy. The least significant four bits of the Status byte contain the completion code. The list of the codes is: 0 = No Error 1 = Initialization error 2 = Bad Command 3 = Operand out of range Errors in OpCode will be returned immediately, while Errors in Operand range will be returned only when the next command is issued.
		Some commands are new and present only in later models. .

Examples of a command strings in DT protocol are:

Please first see Appendix 5 and 6 to ensure correct wiring and stability of motor.

Example #1 (A Move to Absolute Position)

/1A12345R<CR>

This breaks down to:
1. "/" is the start character. It lets the EZ Servos know that a command is coming in.
2. "**1**" is the device address, (set on address switch on device).
3. "**A12345**" makes the motor turn to **A**bsolute position **12345**
4. "**R**" Tells the EZ Servo to **R**un the command that it has received.

<CR> is a carriage return that tells the EZ Servo that the command string is complete and should be parsed.

Note: Hyperterminal issues each character as you type it in. Therefore it is not possible to cut and paste in Hyperterminal. Backspace is allowed only upto the address character. If backspace is used, all characters "backspaced" must be retyped in. If a typing error is made, typically hit enter and type it all in again – what was typed in will be overwritten as long as the R command at the end was not present.

Example #2 (Move loop with waits)

/1gA10000M500A0M500G10R<CR>

This breaks down to:
1. "/" is the start character. It lets the EZ Servos know that a command is coming in.
2. "**1**" is the device address, (set on address switch on device).
3. "**g**" is the start of a repeat loop
4. "**A10000**" makes the motor turn to **A**bsolute position **10000**
5. "**M500**" causes the EZ Servo to wait for **500 M**illiseconds.
6. "**A0**" makes the motor turn to **A**bsolute position **0**.
7. "**M500**" is another wait command for **500 M**illiseconds.
8. "**G10**" will make the string repeat 10 times starting from the location of the small "g"
9. "**R**" Tells the EZ Servo to **R**un the command that it has received.
10. **<CR>** is a carriage return that tells the EZ Servo that the command string is complete and should be parsed.

To Terminate the above loop while in progress type **/1T**

Example #3 (Program Storage and Recall)

An example of a storing a command string for later execution:

/1s2gA10000M500A0M500G10R<CR>

The program outlined in the prior example is stored as Program 2

/1e2R<CR>

Will execute the previously stored program #2. (Note: program 0 is always executed on power up, if we use 0 instead of 2 in the above example then this program would execute automatically on power up).

Example #4 (Set Current , Move upon Switch2 closure)

/1s0m50Z20000gH02P1000G10R<CR>

/1s0	Store following program in motor number 1 stored string 0 (string 0 is executed on power up).
m50	Set move current to 50% of max
Z20000	Home to opto with max move of 20000 encoder ticks to find opto.
g	Start a loop
H02	Wait for Zero on Switch2
P1000	Advance 1000 Encoder Ticks
G10	Repeat loop above 10 times.
R	Run

Note the above loop will terminate after the Z command if a Home Flag is not found.
Type /1e0R to execute.
To Terminate the above loop while in progress type **/1T**

Example #5 (Nested loop example)

/1gA10A1000gA10A100G10G100R<CR>

/1	Talk to motor number 1.
g	Start outer loop
A10	Goto Absolute position 10
A1000	Goto Absolute position 1000.
g	Start inner loop.
A10	Goto Absolute position 10.
A1000	Goto Absolute position 100.
G10	Do inner loop 10 times. (End of Inner Loop)
G100	Do outer loop 100 times. (End of outer loop)
R	Run.

To Terminate the above loop while in progress type **/1T**

Example #6 (Skip / Branch instruction)

/1s0gA0A100S13e1G0R<CR>
/1s1gA0A10S03e0G0R<CR>

Two "Programs" are stored in string0 and string1 and the code switches from one Program to the other depending on the state of input3. In the example given the code will cycle the motor between position A0 and A100 if input3 is High and between A0 and A10 if input 3 is Low.

Stored string 0:

/1	Talk to motor 1
s0	Store following in string0 (executed on power up).
g	Start loop
A0	Goto Absolute position 0
A100	Goto Absolute position 100.
S13	Skip next instruction if 1 (hi) on input 3
e1	Jump to string1
G0	End of loop (infinite loop).
R	Run.

Stored string 1:

/1	Talk to motor 1
s0	Store following in string0 (executed on power up).
g	Start loop
A0	Goto Absolute position 0
A10	Goto Absolute position 100.
S03	Skip next instruction if 0 (low) on input 3
e0	Jump to string0
G0	End of loop (infinite loop).
R	Run.

Example #7 (Monitor 4 Switches and execute 4 different Programs depending on which switch input is pushed)

/1s0gS11e1S12e2S13e3S14e4G0R<CR>
/1s1A100e0R<CR>
/1s2A200e0R<CR>
/1s3A300e0R<CR>
/1s4A400e0R<CR>

Five program strings are stored. Upon power up String 0 automatically executes and loops around sampling the switches one by one, and skipping around the subsequent instruction if it is not depressed. Then for example when Switch1 is depressed stored String 1 is executed, which moves the Servo to position 100. Execution then returns to Stored String 0, due to the e0 command at the end of the other stored strings. If the switch is still depressed it will jump back to String 1 again, but since it is already at that position there will be no visible motion.

To Terminate the above endless loop type **/1T**
Note that using an "e" command to go to another program is a more of a "GOTO" rather than a "GOSUB" since execution does not automatically return to the original departure point.

Example #8 (Move 100 Steps forwards on every rising edge of Switch2)

/1gH02H12P100G0R

The endless loop first waits for a 0 level on switch1 then waits for a "1" level on Input2. Then A relative move of 100 Steps is issued, and the program returns to the beginning to look for another rising edge.

To Terminate the above endless loop type **/1T**

Coordinated motion between multiple axes

For the simple case of motors 1-9, the EZ Servos are addressed as /1, /2, etc. as shown above.
Up to 16 motors can be addressed individually or in banks of 2, 4 ,or "All", increasing versatility and ease of programming. Synchronized motion is possible by issuing commands addressed to individual EZ Servos without the "R" (Run) command, which sets up the command without executing it. At the proper time, the "R" command is sent to a bank of motors to start several actions in concert.

Addressing motors 10-16
Use the ASCII characters that are the ones above 1-9, which are
10 = ":" (colon)
11 = ";" (semi colon)
12 = "<" (less than)
13 = "=" (equals)
14 = ">" (greater than)
15 = "?" (question mark)
16 = "@" (at sign) – use setting zero on the address switch for this.

For example /=A1000R moves Servo #13 to position 1000.

Addressing banks of motors:
Global addressing of more than one motor is also possible.
The same command can be issued to a bank, or different commands issued to the motors individually, (minus a "Run" at the end) and then a global "Run" command issued to a bank or to All.

The Banks of two are:
Motors 1 and 2 : "A"
Motors 3 and 4 : "C"
Motors 5 and 6 : "E"
Motors 7 and 8 : "G"
Motors 9 and 10 : "I"
Motors 11 and 12 : "K"
Motors 13 and 14 : "M"
Motors 15 and 16: "O"
The Banks of four are:
Motors 1,2,3, and 4 : "Q"
Motors 5,6,7, and 8 : "U"
Motors 9,10,11, and 12: "Y"
Motors 13,14,15 and 16 : "]"- (close bracket)

For All motors:
Use the Global address "_" (underscore).

Example #9 Coordinated Motion with axes doing same motion.

/_A10000R<CR>

/_	(Slash then Underscore) Talk to all 15 Motors.
A1000	Goto Absolute position 1000.
R	Run. All motors will go to Absolute position 1000

Example #10 Coordinated Motion with axes doing different motions

/1A10000<CR>
/2A200<CR>
/AR<CR>

/1A10000<CR> Set up motor 1 command buffer to go to **A**bsolute position **10000.**
/2A200<CR> Set up motor 2 command buffer to go to **A**bsolute position **200.**
/AR Execute current commands in buffer for **Bank Address "A"** which is motors 1 and 2. (The "A" here is an Address of a Bank of motors 1&2 because it comes after the slash and should not be confused with the "A" that means absolute position.) Both moves will start at the same time, and complete at a time determined by the Velocity set for each axis.

APPENDIX 1
SERVO MOTOR ELECTRICAL SPECIFICATION

The EZ Servo® will work with most Servo motors, however the performance achieved will be a function of the motor used.

Select the following:

Kv: Back EMF constant. Chose a motor such that the back EMF at the max speed required is about half of the supply voltage. This will allow for good controllability at the top speed. Eg use a 12V Motor with a 24V supply. (The EZServo regulates the current so the motor will not be damaged.)

L: Motor Inductance, a general guideline is:
For EZSV17 the inductance must be >500uH
For the EZ23 the Inductance must be > 100uH.

Operation of lower inductance motors is possible but depends on exact values of supply voltage, motor voltage, inductance and resistance. Please contact the factory for assistance. Typically motors with inductance less than stated above can be run with the addition of an extra series inductance of about 500uH or so. Inductor must be capable of handling the full motor current.

R: Motor Resistance. In most cases when Kv and L is selected as stated above, this value will be acceptable.

EZ Servo®
Command set and Communications protocol

APPENDIX 2
HOMING ALGORITHM IN DETAIL

The "Z" command is used to initialize the motor to a known position. When issued the Motor will turn toward 0 until the home opto sensor is interrupted. If already interrupted it will back out of the opto and come back in until re-interrupted. Current motor position is set to zero. The Homing is done at a speed set by "V".

The maximum number of steps allowed to go towards home is defined by the Z command operand + 400. The maximum number of steps away from home (while sensor is cut) is 10000. If motor goes away from home, rephasing of the motor power and hall sensors wires is required to change the direction of rotation of rotation that the motor considers positive.

Opto and flag should be set up to be unambiguous, i.e. when motor is all the way at one end of travel, flag should cut the opto, when at other end of travel flag should not cut opto. There should only be one black to white transition possible in the whole range of travel.

To set up the home flags:

1) First ensure that a positive move e.g. /1P100R moves away from home and the home flag.
2) The Default condition expects the output of the Home flag to be low when away from home (as is the case in an opto). If Home flag is high when away from home (as in the case of the "Normally Open" switch) then issue the command /1f1R to reverse the polarity that is expected of the home flag.
3) Homing should be done at a slow speed, especially if homing to a narrow Index pulse on an encoder, which may be missed at high speeds.
4) Issue /1N1R if home is Opto 1. Issue /1N2R if home is the index on the encoder. Subsequent Z commands will then home to the chosen signal.

EZ Servo®
Command set and Communications protocol

APPENDIX 3
HEAT DISSIPATION

Most applications require intermittent moving of the motor. In the EZ Servo, the current is increased while a move is performed, and the current is then reduced at the end of the move. The dissipation in the drive is proportional to the current flowing in the drive, and therefore the dissipation occurs primarily during the "move". The one exception to this is the rare instance where the drive has to fight a constant load such as gravity.

When the Drive generates heat, the heat first warms the circuit board and heatfin (if any). Only then does the heat transfer to the surroundings. For intermittent moves that are less than one minute in duration, the Drive primarily cools using this thermal inertia of the board and heatfin, and not by steady state dissipation to the surrounding ambient.

The EZServos are designed to work beyond the voltage and current that they are rated for, however the small size of the boards limit their ability to dissipate heat in steady state. It is recommended that the drive be derated linearly from 100% Duty Operation at 50% current to 25% Operation at 100% of rated current. (Ie if an application requires 100% current (m=100) during the entire move, moves should only performed about 25% of the time – average over about 5 minutes). Typically servos only require significant current when accelerating or decelerating, so even if "m" is set to 100, it does not mean that 100% of current is flowing into the motor during the entire move.

Most applications will not require derating of the drive.

EZ Servo®
Command set and Communications protocol

APPENDIX 4
OEM PROTOCOL WITH CHECKSUM

The Protocol described in the majority of this manual is DT (Data Terminal) protocol. There is however a more robust protocol known as OEM protocol that includes checksums. AllMotion Drives work transparently in both protocols. And switch between the protocols depending on the start transmission character seen.

The OEM protocol uses 02 hex (Ctrl B) as the start character and 03 Hex (Ctrl C) as the stop character. The 02 Hex Start Character is equivalent to the / character in DT protocol.

OEM PROTOL EXAMPLE1:
/1A12345R(Enter) in DT Protocol is equivalent to
(CtrlB)11A12345R(Ctrl C)# in OEM protocol

Explanation	Typed	Hex
Start Character	Ctrl B	02
Address	1	31
Sequence	1	31
Command	A	41
Operand	1	31
Operand	2	32
Operand	3	33
Operand	4	34
Operand	5	35
Run	R	52
End Character	Ctrl C	03
Check Sum	#	23

The Check Sum is the binary 8 bit XOR of every character typed from the start character to the end character, including the start and end character. (The Sequence character should be kept at 1 when experimenting for the first time.) Note that there is no need to issue a Carriage return in OEM protocol.

Note that some earlier models require the first command issued after power up to be a DT protocol command. Subsequent commands can be in DT protocol or in OEM protocol. Very early models do not have OEM Protocol implemented at all.

OEM PROTOL EXAMPLE 2:
/1gA1000M500A0M500G10R(Return) in DT Protocol is equivalent to
(CtrlB)11gA1000M500A0M500G10R(CtrlC)C in OEM protocol

The C at the end is Hex 43 which is the checksum (Binary XOR of all preceding Bytes).

Sequence Character:
The Sequence Character comes into effect if a response to a command is not received from the Drive. In this instance the same command can be resent with bit 3 (repeat bit) of the sequence byte set, and bits 0-2 representing the sequence number.

When the repeat bit is set consecutive commands received by the drive must have a different sequence number in order to get executed. (Only the sequence number is looked at – not the command itself)

This covers both possibilities that (a) the Drive didn't receive the command and (b) The Drive received the command but the response was not received.

The sequence number can take the following values.
31-37 without the repeat bit set or 39-3F with the repeat bit set.
(The upper nibble of the sequence byte is always 3.)

APPENDIX 5 BLDC MOTOR WIRING

The procedure below describes how to figure out the phasing of the encoder and the hall sensors. However, please note that AllMotion can perform this process for no extra charge. Just ship us a motor and we will ship it back fully wired with a board.

BLDC Motor Wiring:

First it is necessary to set the phasing of the Hall Sensors:

The phasing of the hall sensors is unrelated to the encoder connection, and no encoder connection is necessary at this time.

1) The procedure is to wire the BLDC motor nominally, as shown in the wiring diagram, and then try all 6 combinations of the 3 hall sensor wires until one combination works.
2) Set the current limit low using /1m10R. (Use of current limited supply set to about quarter of the motor rated current is also recommended).
3) With the encoder unplugged issue the command /1A10000R and the motor should spin smoothly in one direction for about 5 seconds before giving up.
4) While the motor spins hold the shaft of the motor (Only if no danger of injury) and ensure that there are no "Dead Spots" in the torque. Four of the six combinations of the 3 hall sensor wires will have obvious dead spots. Of the remaining two, one will have subtle dead spots and the other will be perfect. The Combinations are: 123,132, 213,231, 312,321 .
   Only the Hall sensor wires need to be switched – don't switch power wires.
   Turn Off Power when Switching Wires or Disconnecting Motor.

Second it is necessary to make sure we have negative feedback from the encoder.

Plug in the encoder and issue a command /1A1000R . If the motor spins endlessly and then gives up, then try again with the encoder A, B channel wires switched.

Motor Tuning:

Typically (in 99% of cases) the motor will be stable with the default constants that are loaded on power up.

If the behavior is oscillatory, then increase the value if the Differential constant.

Eg . /1y3000R

If increasing the Differential term does not work and the motor is noisy and the encoder ticks can be "heard", then simultaneous reduction of both the Proportional and Differential constant is recommended. /1w250y500R works well especially with high (4000) line count encoders.

Motor Overload:

If motor gives up half way through a move, and gives an overload error (Upper or lower case I when queried with /1Q after error) this is due to the fact that the motor could not keep up with the commanded trajectory. Typically increase the value of the move current "m" to allow the motor to move faster.

Continued……

APPENDIX 5 Continued .....

Motors with no encoders:

In N=0 mode Motors with no encoders will work in velocity control mode in by using the hall sensor crossings as a gage of velocity. In this mode only the Integrator is active and The "x" value controls the acceleration between speeds. (L command is inactive in this mode). The velocity is set in Revs per second using the V command.

In N= 1 mode, It is also however possible to run motors with no encoders by using the Hall sensors as a position encoder, by wiring over two of the hall lines over to the encoder A and B inputs in addition to the regular Hall connection. Any 2 of the hall lines can be wired over, however as with any encoder the A and B lines must be connected to count up when the motor moves in the positive direction.
This mode typically has better performance than the N0 mode in velocity control applications, and can also act as a crude position control mode.
Typically in this mode very low Accelerations and Velocity values must be used since the halls act as a very low line count encoder. Try /1L1V10000R.
It is also necessary to run the hall sensors on 5V since the encoder connections are 5V. In the EZSV17 which puts out 15V hall power, it will be necessary to use the encoder power at 5V to power the hall sensors.

APPENDIX 6 BLDC MOTOR TUNING

Typically (in 99% of cases) the motor will be stable with the default constants that are loaded on power up.
If the motor Vibrates or oscillates on power up, try issuing /1w250y500R
The motor should be stable but the response will be somewhat slow.
Increasing both P and D values in proportion will "stiffen" the servo.

If zero following error is needed then the Integrator will need to be turned on:
(eg /1x100R) , however the system will become more unstable and harder to compensate when the integrator is turned on. Try to achieve the best possible result with the P and D values only and then turn on just a little I.
Eg /1w700x100y30000R

The default values work well with 512 –2048 line count encoders. For higher line count encoders use smaller values of PID.

The use of a current limited lab supply is recommended while tuning the motor. Large currents may be drawn if oscillations occur.

Motor Overload:
If motor gives up half way through a move, and gives an overload error (Upper or lower case I when queried with /1Q after error) this is due to the fact that the motor could not keep up with the commanded trajectory. Typically increase the value of the move current "m" to allow the motor to move faster.

EZ Servo®
Command set and Communications protocol

APPENDIX 7
DEVICE RESPONSE PACKET

EZ Servos and EZ Servos respond to commands by sending messages addressed to the "Master Device". The Master Device (which for example is a PC) is assumed always has Address zero. The master device should look for responses starting with /0.

After the /0 the next is the "Status Character" which is actually a collection of 8 bits. These Bits are:
Bit7 … Reserved
Bit6 … Always Set
Bit5 … Ready Bit - Set When EZ Servo is ready to accept a command.
Bit4 … Reserved

Bits 3 thru 0 form an error code from 0-15
0 = No Error
1 = InitError
2 = Bad Command (illegal command was sent)
3 = Bad Operand (Out of range operand value)
4 = N/A
5 = Communications Error (Internal communications error)
6 = N/A
7 = Not Initialized (Controller was not initialized before attempting a move)
8 = N/A
9 = Overload Error (Physical system could not keep up with commanded position)
10 = N/A
11 = Move Not Allowed
12 = N/A
13 = N/A
14 = N/A
15 = Command Overflow (unit was already executing a command when another command was received)

Continued…

Example Response to command /1?4

FFh: RS485 line turn around character. It's transmitted at the beginning of a message.
2Fh: ASCII "/"Start character. The DT protocol uses the '/' for this.
30h: ASCII "0" This is the address of the recipient for the message.
In this case ASCII zero (30h) represents the master controller.
60h: This is the status character (as explained above
31h:
31h: These two bytes are the actual answer in ASCII.
This is an eleven which represents the status of the four inputs.
The inputs form a four bit value. The weighting of the bits is:
Bit 0 = Switch 1
Bit 1 = Switch 2
Bit 2 = Opto 1
Bit 3 = Opto 2
03h: This is the ETX or end of text character. It is at the end of the answer string.
0Dh: This is the carriage return…
0Ah: ...and line feed.

A program that receives these responses must continuously parse for /0 and take the response from the bytes that follow /0. The first Character that comes back may be corrupted due to line turn around transients, and should not be used as a "timing mark".

APPENDIX 8
AUTO RECOVERY OF A STALLED SERVO

The EZServo determines a stalled or overload condition, by checking to see if is following a commanded trajectory. If the following error is too great for a period given by the "u" command then the servo will shut down and report an error condition.

Typically the following error is great due to the fact that either:

1)"m" value (current) set too low.
2)"L" value (acceleration) set too high, for Torque available from motor.
3)"V" value (Velocity) set too high, for Torque available from motor.
4) Physical obstruction, or excessive friction.

When an overload condition is detected it will be reported back as an upper or lower case I when the status is quizzed. This status can be used by an external computer to execute a recovery script.

However it may be desired that the servo recover by itself in the case of a stand-alone application. For this purpose we have the "n" mode bits n512, and n1024 . Depending on which of these bits is set the servo will execute stored program 13, 14 or 15 when an overload is detected. Program12 is also executed as a last resort if programs 13,14,15 cannot auto recover by a after retrying a number of times given by the "au" command.

Example:
Set m=5 so that we can stall a motor easily.
Set u=1000 so the motor overloads after 1 second.
Set au = 100 so that 100 retries max are allowed
Set n =512 so that Stored Program 13 will be issued on error condition.

```
/1s0m5n512u1000e1R
/1s1gA10000A0G0R
/1s13r1e1R
/1s12A0R
```

Program zero enables Auto-recovery mode and calls program 1 on power up.
Program 1 cycles between 10000 and 0
When a stall occurs while running program 1, program13 is automatically called. This program recovers the servo to the current encoder position and then calls program 1 again. The servo will attempt auto-recovery 100 times as given by the "au" command, then, if no move completes after 100 retries, it will execute stored program 12.

EZ Servo®
Command set and Communications protocol

APPENDIX 9
ANALOG INPUTS AND ANALOG FEEDBACK

Available on Software version 3.90 and later

ANALOG INPUTS

The 4 Inputs on the EZServo are all ADC Inputs.

1) These ADC values can be read via RS485. E.g**. /1?aa<CR>.** These values are on a scale of 0-16368 as the input varies from 0-3.3V. The inputs as shipped are good to about 7 bits, but can be made to be better than 10 bits with the removal of the input over-voltage protection circuitry, (call factory for details).
2) The threshold upon which a Digital "One" or "Zero" is called can be varied with the "at" command and hence affect the Halt H command or Skip S command. **E.g. /1at309999R<CR>.** – This sets the threshold on input 3 to be 09999. Note that it is necessary to insert leading zeros after the Input Number (3) since the threshold value must always be entered as 5 digits (00000-16368).
3) The thresholds for all 4 inputs can be read back with the **/1?at<CR>** command. The units have a default threshold value of 6144 = (1.24V).
4) It is possible for example to regulate pressure, by turning a pump on or off depending on an analog value read back , by setting the threshold of the One/Zero call to be the regulation Point. E.g. /1at308000gS03P1000G0R
5) A potentiometer can be placed as shown in the wiring diagram and its position read back via the /1?aa<CR> command. Note that the supply provided (which normally drives an LED) has 200 ohm in series to 5V, so the use of a 500 ohm potentiometer will give almost a 0-3.3V range on the inputs.

POTENTIOMETER POSITION COMMAND

Potentiometer 2 can be used to command the position of a motor. The value read back on potentiometer 2 from 0-16368 is multiplied by the multiplier "am" and then divided by 256, then an offset given by "ao" is added. The motor will then use this number just as if it had been commanded by a /1A12345R type command. There is further a "deadband" command "ad" which sets a dead band on the 0-16368 pot value read back such that a value outside this deadband must be seen before a command is issued to cause a move.

/1n8192R<CR> enables potentiometer command mode.

Stepper position = ((analog value from pot / 256) x ("am" multiplier)) + ("ao" offset value)

Use 500 Ohm Potentiometer.
Supply pin of pot already has 200 Ohm in series on the board to 5V
Value from pot = 0 to 16368 for 0-3.3V on wiper.

Continued…….

Eg /1ao1228R<CR> sets ao offset to 1228 . Default is 0
Eg /1am256R<CR> sets am multiplier to 256. Default is 256.
Eg /1ad100<CR> sets dead band on pot to 100 . Default is 50
Then issue /1n8192R to enter the mode.

POTENTIOMETER POSITION FEEDBACK

Potentiometer 1 can be used as an encoder. The value read back is from 0-16368.
/1N3R enables Potentiometer 1 as the encoder, (instead of the quadrature encoder).

To make this mode work:

1) Use a 500 Ohm Pot wired in to the Potentiometer 1 position (see wiring diagram).
2) Connect motor shaft to shaft of pot.
3) Issue /1w200x0y500R<CR> to reduce PID values.
4) Issue /1N3r1R<CR> to enable the pot as encoder and turn the servo on.
5) The shaft of the motor should now be holding position.
6) If motor slams to the end of the pot, then reverse the pot ends. (positive feedback).
7) Issue a /1?aa<CR> which will return the pot value. Issue a /1?<CR> which will return the motor position. The pot value and the motor position should now be the same.
8) Issue /1A2000R<CR> then issue /1A4000R<CR> The motor should move between 1/8th and ¼ of the pot full range (16368).
9) It may be necessary to have stops such that the pot ends do not take the full force of the motor, if the motor is commanded to the ends.

POTENTIOMETER VELOCITY COMMAND (Joystick Mode) (-Future Release-)

Potentiometer 2 can be used to command velocity of the drive.
Velocity in encoder ticks/sec =(Pot 2 Value – (deadband/2)) x (gain-multiplier / 256)

To enter this mode typically hook up a 500 Ohm Pot to Opto2 as shown in the wiring diagram . Then set it to mid position and issue /1ad100am1000n65536z0P0R

“ad” = dead-band and “am” = gain-multiplier must be set prior to entering the mode. If needed issue /1n0R then /1T to disable the mode and then change these values as necessary.

A zero z value in the above string will set will set the zero velocity position to be the current pot position of the pot. Any non zero z value will cause that value from the pot to be considered the zero velocity point.

To exit this mode typically type /1n0R to remove the mode and then type /1T to terminate any move in progress.