EZ Stepper®
Command set and Communications protocol

Command Set For Stepper Models: EZ17, EZHR17, EZHR23

Document Revision: A34 4/4/09

INDEX

EZ Stepper®
Command set and Communications protocol

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

Command syntax:
Commands to the EZ Stepper 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 to become a stand-alone device that responds to switch inputs. Finally, storing such strings into the onboard EEPROM allows the EZStepper 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 EZStepper can even be commanded from a serial enabled PDA.

Command Set: **(Also see examples on page 9)**

Command (**Case sensitive**)	Operand	Description
		POSITIONING COMMANDS
A	**0-(2^32)**	**Move Motor to absolute position (** microsteps) 32 Bit Positioning**.**
P	**0-(2^32)**	**Move Motor relative in positive direction.** **(** microsteps **)** 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 command or by a falling edge on the Switch2 Input.
D	**0-(2^32)**	**Move Motor relative in negative direction.** **(** microsteps **)** (Note: for a finite move, the ending Absolute position must be greater than zero, if it is necessary to go further then redefine the current position to be a large positive value by using the lower case z command). 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 command or by a falling edge on the Switch2 Input.

EZ Stepper®
Command set and Communications protocol

		HOMING COMMANDS
Z	**0-(2^32)** (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. Homing is done at speed "v". (lower case "v") See Appendix 2 for further details.
z	**0-(2^32)**	**Change current position without moving.** For Stepper sets current position to be position specified without moving motor.
f	**0 or 1** (0)	**Home Flag polarity.** Sets polarity of home sensor.
		SET VELOCITY COMMANDS
v	 **50-900** (50) **200-2500** (200)	**In Position Mode, Set Start speed of motor.** In EZ17 sets half steps per second, range 50-900 In HR Version sets micro steps** per second, range. 200-2500
V	 **5-5800** (2500) **50-10000** (3700)	**In Position Mode Set Max/Slew Speed of motor** For EZ17 Stepper version sets speed in half steps per second 5-5800 For Stepper HR Version sets micro steps** per second, range 50-10000.
V	 **5-2500** (500) **5-10000** (1000)	**In Velocity Mode Set Spin Speed Of Motor** For EZ17 Upper limit is 2500 halfsteps/sec. For HR Stepper after issuing P0 or D0 Command, allows change of velocity on the fly in microsteps/second.
c	 **50-900** (50) **200-2500** (200)	**In Position Mode, Set Stop speed of motor** In EZ17 sets half steps per second, range 50-900 In HR Version sets micro steps** per second, range 200-2500
		SET ACCELERATION COMMANDS
L	**1-20** (2) **1-20** (2)	In EZ17 **Set Acceleration** factor (accel = "L" * 7500 steps / sec^2) In HR Version sets Acceleration factor (accel = "L" * 7500 microsteps / sec^2)

EZ Stepper®
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.
H		**Halt** current command string and wait until condition specified.
	01	Wait for low on input 1 (Switch 1)
	11	Wait for high on input 1 (Switch 1)
	02	Wait for low on input 2 (Switch 2)
	12	Wait for high on input 2 (Switch 2)
	03	Wait for low on input 3 (Opto 1)
	13	Wait for high on input 3 (Opto 1)
	04	Wait for low on input 4 (Opto 2)
	14	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.
		H command with no number after it waits for Switch 2 Closure (low). This is a legacy command that was the only command available on Non HR units prior to Version 7.20.

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. On non HR units this "S" command is only present in units with V7.20 or later code.
		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)
e	**0-15**	**Executes Stored Program** 0-15.
		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** (30)	**For Steppers sets "Fast Move" Current on a scale of 0 to 100% of max current.** This value is used for move current if V>v. 100% = 1.25A for EZ17. 100% = 3A for EZ23. For conservative operation it is recommended that the duty cycle be reduced linearly, from 100% duty at 50% of rated current, to 25% duty at 100% of rated current.

l	**0-100** (30)	**For Steppers sets "Slow Move" Current on a scale of 0 to 100% of max current.** This value is used for move current if V<v. This command also sets the Move current in Velocity Mode. 100% = 1.25A for EZ17. 100% = 3A for EZ23 For conservative operation it is recommended that the duty cycle be reduced linearly, from 100% duty at 50% of rated current, to 25% duty at 100% of rated current.
h	**0-50** (12)	**For Steppers sets Hold Current** on a scale of 0 to 50% of max current. 100% = 1.25A for EZ17. 100% = 3A for EZ23.
		SET MICROSTEP RESOLUTION
j	**2, 4, 8, 16, 32, 64, 128, 256** (2)	For HR version only, adjusts the resolution in micro-steps per step. Resolution depends on model. For best micro step results, a motor must be selected that is capable of micro step operation. (Default is 2) For changing resolution on standard drives see the "N"command
N	**0-1** (0) **8** (0)	**For Stepper EZ17 Model Only**: Set micro step mode on or off. This command switches between half step mode and eighth step mode. See the "j" command for changing resolution on HR Drives. **For Stepper EZHR23 Model Only**: Sets Solenoid Outputs to drive Step and Direction so that an external Driver can be slaved to this drive. (requires Pullups on Solenoid pins)
n	**0-2047** (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 (lower case "v"). The Switch Inputs become the Jog Inputs. (Not Available in EZHR17) **Bit1 :** /1n2R Enable Limits. (The two optos become limits switches). The polarity of the limits is set by the "f" command. . (Not Available in EZHR17) **Bit2 :** /1n4R Enable Continuous Jog Mode. Continuous run of motor while switch is depressed. Velocity is given by "v" command (lower case "v"). 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. . (Not Available in EZHR17) **Bit3 :** /1n8R Reserved **Bit4 :** /1n16R Reserved
o	**0-250** (30)	For Stepper HR version only, allows the user to correct any unevenness in microstep size. It is best to adjust this with a current probe, but adjusting for lowest audible noise is a good approximation. This command can be executed while the motor is running.
		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.
		DEVICE RESPONSE PACKET
		See Appendix 7 for detailed description of device response to commands.

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

EZ Stepper®
Command set and Communications protocol

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

		IMMEDIATE QUERY COMMANDS
T		Terminate current command or loop. (example: /1T)
?	0	Returns the current Commanded motor position
?	1	Returns the current Start Velocity
?	2	Returns the current Slew/Max Speed for Position mode /1?2
?	3	Returns the current Stop Speed
?	4	Returns the status of all four inputs, 0-15 representing a 4 bit binary pattern. Bit 0 = Switch1 Bit 1 = Switch2 Bit 2 = Opto 1 Bit 3 = Opto 2
?	5	Returns the current Velocity mode Speed
?	6	Returns the current step size microsteps /step (HR Version Only). /1?6
?	7	Returns the current 'o' value. (HR Version Only)
?	8	Returns Encoder Position. (can be zeroed by "z" command)
?	9	Erases all stored commands in EEPROM. EZ17 and EZHR17 Only. In EZ23 Use /1s0R to erase Prog 0 etc.
&		Returns the current Firmware revision and date
Q		Query current status of EZ Stepper 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 EZStepper/EZServo is ready to accept commands. It is cleared when the EZStepper/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.
$		/1$ Reads Current Program String EZ17, EZHR17 Only
		**Micro steps for HR version is in whatever resolution currently selected using "j" command.
		Some commands are new and present only in later models. AllMotion reserves the right to enhance the specifications at

		any time.

Examples of a command strings in DT protocol are:

Example #1 (A Move to Absolute Position)

/1A12345R<CR>

This breaks down to:
1. "/" is the start character. It lets the EZ Steppers 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 Stepper to **R**un the command that it has received.

<CR> is a carriage return that tells the EZ Stepper 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 Steppers 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 Stepper 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 Stepper to **R**un the command that it has received.
10. **<CR>** is a carriage return that tells the EZ Stepper 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 , Wait For Switch2 closure, Home to Opto)

/1s0m75h10gJ3M500J0M500G10HZ10000A1000A0R<CR>

/1s0	Store following program in motor number 1 stored string 0 (string 0 is executed on power up).
m75	Set move current to 75% of max
h10	Set hold current to 10% of max
g	Start a loop
J3	Turn on both on off drivers.
M500	Wait 500 mS
J0	Turn off both on off drivers.
M500	Wait 500 mS
G10	Repeat loop above 10 times.
H	Wait for a switch2 closure. (Switch 2 automatically selected if no number)
Z10000	Home the stepper to opto #1. Max Steps allowed to find opto = 10000.
A1000	Move to position 1000
A0	Move to position 0
R	Run

Note: This program string will abort after the Z command if it does not find a flag.

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 stepper 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 Steppers 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 Steppers 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 stepper #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
STEPPER MOTOR ELECTRICAL SPECIFICATION

The EZ Stepper ™ will work with most stepper motors, however the performance achieved will be a function of the motor used.

A Stepper Motor moves by generating a rotating magnetic field, which is followed by a rotor. This magnetic field is produced by placing a Sine Wave and a Cosine Wave on two coils that are spaced 90 degrees apart. The torque is proportional to the magnetic field and thus to the current in the windings.

As the Motor spins faster, the current in the windings need to be changed faster in a sinusoidal fashion. However the inductance of the motor will begin to limit the ability to change the current. This is the main limitation in how fast a given motor can spin.

Each winding of the motor can be modeled as an Inductor in series with a resistor. If a step in Voltage is applied then the current will rise with time constant L/R. If L is in Henrys and R is in ohms then L/R is the time it takes in Seconds for the current to reach 63% of its final value.

The current I for a step function of voltage V into a coil is given by

$$I = (V/R)\ (1-\exp(tR/L))$$

This equation is a standard response of a first order system to a step input. The final value of current is seen to be V/R. (This system is similar to a spring (L) in parallel with a Damper (R) being acted upon by a Step in Force (V) giving a resulting velocity (I).)

There are two methods by which the current can be made to change faster.
(1) Reduce the Inductance of the motor
(2) Increase the forcing function voltage V.

For (1) it is seen that for high performance, a motor with low inductance is desired.

For (2) the trick is to use a motor which is rated at about ¼ of the supply voltage (V). This means that it takes less time to ramp the current to a given value. (Once the current reaches the desired value the "Chopper" type drive used in the EZ Steppers ™ will "Chop" the input voltage in order to maintain the current. – So the current never actually gets to the final value of V/R, but the advantage of "heading towards" a higher current with the same time constant means that the current gets to any given value faster.)

So for example, for a 24V supply use a motor rated at around 6V, and then use the "m", "l" and "h" commands to set the current regulation at or below the rating for the motor. The default values on power up are l=h=12% and m=30% and should be safe for most motors.

EZ Stepper®
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" (lower case).

The maximum number of micro 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. For the EZHR17 and EZHR23 the units for this number is the current micro-step resolution for the drive. The EZ17 always homes in 1/8th step mode.

To setup home: **First** issue a /1P1000R command, if the motor does not go away from home, flip the connections to only one of the windings of the stepper motor.
Second, issue a /1Z10000R command, if the motor does not go towards home, then issue a /1f1R command to invert the polarity of the home flag. /1f1Z100000R etc.

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. Note: Homing resets speeds to default values.

Homing Algo Detail: There are four full steps in a single electrical cycle that moves the stepper motor. (A+, B+, A-, B-). For repeatability in homing, the home position is set to first step in that cycle that occurs after the flag edge has been seen. (This means that the home position is defined some ways beyond the middle of the flag).

However there is a small but finite chance that an ambiguity in home position may occur in the rare case that the exact point of switching into A+ occurs at the same point at which the flag gets cut. In which case a 4 step ambiguity in home position may exist, because sometimes the flag may cut just before and sometimes just after. The procedure below describes a method by which the ambiguity can be removed. However, this procedure need not be followed if a 4 step inaccuracy in Home position is acceptable.

To eliminate the home position ambiguity. First issue the Z command, allow the motor to home. Then move 2 full steps (in any direction), now mechanically move the flag edge (or sensor) such that it trips in the middle of the sensor by adjusting it while watching the status LED on the board which shows the status of the home sensor. This will ensure that the flag trips at A- and thus the motor will home to a unique position of A+.

Another way to do this, if it is not hazardous, is to put the motor in an endless homing loop "/1gZ10000GR", then move the flag/opto around while the motor is homing. It will be noticed that the motor will home to two distinct positions that are 4 steps apart. Make sure the Hi to Low transition point of the Opto is NOT near these positions (exact position does not matter as long as it is not near the place where it homes to).

APPENDIX 3
MICROSTEPPING PRIMER

First lets consider a Full Stepping Driver:
A Stepper motor moves by having two windings that are orthogonal to each other and sequencing the current in these windings.
When full stepping a typical sequence is:

A+ (Only winding A current applied in +ve Direction)
B+ (Only winding B current applied in +ve Direction)
A- (Only winding A current applied in -ve Direction)
B- (Only winding B current applied in -ve Direction)

A full electrical cycle consists of 4 steps.

It can be seen that if the windings are not physically placed orthogonally then the 4 steps may not be of equal size, and the delta in motion will only be a constant if the number of steps is divisible by four, even when in full step mode.

Now consider Microstepping:
Microstepping is achieved by placing two sinusoidally varying currents that are 90 degrees apart, in the windings of the stepper. This causes a torque vector of equal length to rotate causing smooth inter step motion of the rotor.

However in order to get even motion in every step it is necessary

1) That the windings be mechanically orthogonal
2) That the windings produce equal torques for equal currents.
3) That there is no other "detent torque" that is acting upon the rotor in the absence of current. This detent torque is easily felt by rotating the stepper (with windings disconnected and not shorted).
4) The current not be so small that the driver cannot regulate it to the microstepping accuracy desired.

In general most inexpensive stepper motors cannot microstep with any accuracy. Typically, a special motor designed for microstepping must be run at a significant current in order to get even microsteps.

APPENDIX 4
HEAT DISSIPATION

Most stepper applications require intermittent moving of the motor. In the EZ Stepper, the current is increased to the "move" current, the move is performed, and the current is then reduced to the "hold" current (automatically). The dissipation in the drive is proportional to the current flowing in the drive, and therefore the dissipation occurs primarily during the "move".

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.

For EZ Steppers: The electronics for the EZ Steppers are fully capable of running at the rated voltage and current. However due to the small size of the boards, which limits the steady state heat transfer to the ambient, care must be taken when the drive is used in high duty cycle and/or high current applications. For conservative operation it is recommended that the duty cycle be reduced linearly, from 100% duty at 50% of rated current, to 25% duty at 100% of rated current. (Duty cycle means the percentage of the time that the drive is moving the load). Conservatively, the maximum continuous run at 100% current is about 1minute. An on board thermal cutout typically trips after about 2 minutes at 100% current. (This cutout is self-resetting when the drive cools). Of course, at 50% of current, the drive will run continuously with no time limit.
Applications that require 100% duty at 100% current will need forced are cooling or the addition of a larger heatsink.

APPENDIX 5
STEP LOSS DETECTION USING OPTO

For some applications it may be useful to detect loss of steps due to the mechanism stalling for any reason.

Step loss is easily detected by following the algorithm below:

1) Home the Stepper to the Opto using the Z Command.
2) Move out of the flag a little by issuing say a A100 Command
3) Figure out the exact step on which the Flag gets cut by issuing D1 commands followed by /1?4 commands to read back the Opto. Let's call this value Y. (This only needs to be done once during initial set up)***.
4) Execute the move sequence for which step loss detection is needed.
5) Issue a command to go back to absolute position Y+1
6) Check the opto, it should not be cut (read Opto back with /1?4 command).
7) Now issue a command to go to position Y-1.
8) Check the opto, it should be cut (read Opto back with /1?4 command).
9) If the Opto was not at the state expected , then steps may have been lost.
10) Step loss detection can also be done by looking for changes on the other inputs.

*** See Homing algorithm detail in Appendix2.

APPENDIX 6
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 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 parse the communications on the bus continuously for responses starting with /0. (Do NOT for example look for the next character coming back after issuing a command because glitches on the bus when the bus reverses direction can sometimes be interpreted as characters)

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)

Example Initialization Error Response:
Note that the Upper Nibble only typically takes on values of 4 or 6 (Hex)
An initialization error has response has 1 in the lower Nibble. So the response is 41 Hex or 61 Hex which corresponds to ASCII character upper case "A" or lower case "a", depending on if the device is busy or not.

Example Invalid Command Response:
Note that the Upper Nibble only typically takes on values of 4 or 6 (Hex)
An invalid command has response has 2 in the lower Nibble. So the response is 42 Hex or 62 Hex which corresponds to ASCII character upper case "B" or lower case "b", depending on if the device is busy or not.

Example Operand Out of range Response:
Note that the Upper Nibble only typically takes on values of 4 or 6 (Hex)
An operand out of range has response has 3 in the lower Nibble. So the response is 43 Hex or 63 Hex which corresponds to ASCII character upper case "C" or lower case "c", depending on if the device is busy or not.

Example Overload Error Response:
Note that the Upper Nibble only typically takes on values of 4 or 6 (Hex)
An overload error has response has 7 in the lower Nibble. So the response is 47 Hex or 67 Hex which corresponds to ASCII character upper case "I" or lower case "i", depending on if the device is busy or not.

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".

EZ Stepper®
Command set and Communications protocol

APPENDIX 8
Special Commands not present in all drives

These special commands have been added to various models as a result of customer special requests. If any of these commands are required for your application, please request them explicitly.

1) Jog and Limit mode - Available in EZ17 and EZHR23
Engaged by lower case "n" command (Example setup /1V5000v1000B20n6R)
The number after the "n" is a binary bit wise number in which each bit has meaning.
Bit0 (LSB) Enable Pulse Jog Mode. Jog distance is given by "B" command. Velocity is given by "v" command (lower case "v").
Bit1 : Enable Limits. (The two optos become limits switches). The polarity of the limits is set by the "f" command
Bit2 : Enable Continuous Jog Mode. Continuous run of motor while switch is depressed. Velocity is given by "v" command (lower case "v").

The limits are the optos and the Jog inputs are the switches. 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. For jog mode to operate correctly V>v . (If v>V an endless move will be started by pushing the jog buttons, which can only be stopped by a /1T command.)

2) Step and Direction Output. Available in EZHR23.
N command, this command can allow output of step and direction pulses on the two solenoid driver outputs. These can be used to drive a higher power driver. The EZ stepper essentially being used as a controller.

3) Command Readback.
/1$<CR> will read back the current command string that is being run. The values stored command, this command can allow output of step and direction pulses on the two solenoid driver outputs. These can be used to drive a higher power driver. The EZ stepper essentially being used as a controller.