EZ Stepper®   STEPPER-STICK and EZ10EN

Command Set For Stepper Models: STEPPER-STICK and EZ10EN

Document Revision: A05   03/03/11

The following are Trademarks of AllMotion Inc.:
EZStepper ®
EZServo ®
EasyServo ™
EZBLDC ™
EasyBLDC ™
AllMotion ®

EZ Stepper® STEPPER-STICK and EZ10EN

Overview:
This document describes the operation and command set for the EZ Stepper® EZ10 Model motor drive.

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-(2^31)	**Move Motor to absolute position** (microsteps or quadrature encoder ticks - 32 Bit Positioning**). Eg.** /1A1000R
P	0-(2^31)	**Move Motor relative in positive direction.** (microsteps or quadrature encoder ticks) 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. **Eg.** /1P1000R
D	0-(2^31)	**Move Motor relative in negative direction.** (microsteps or 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 command or by a falling edge on the Switch2 Input. **Eg.** /1D1000R - note ending position must be > 0

EZ Stepper®   STEPPER-STICK and EZ10EN

		HOMING COMMANDS
Z	**0-(2^31)**	**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. **Eg /1Z300000R** See Appendix 2 for further details.
z	**0-(2^31)**	**Change current position without moving.** Sets current position to be position specified without moving motor. New microstep position (preferably) should have the same remainder as old position, when divided by 1024, else the motor may physically move/lose up to 2 steps when this command is issued. **Eg. /1z65536R**
f	**0 or 1**	**Home Flag polarity.** Sets polarity of home sensor, default value is 0. (See Appendix 2) **Eg /1f1R**
F	**0 or 1**	Change direction of rotation considered positive. This should only be done once on power up. **Eg /1F1R**
		SET VELOCITY COMMANDS
V	**160000** (2440)	**In Position Mode Set Max/Slew Speed of motor** Sets microsteps per second. It is recommended that this drive be left in 1/8 micro-step mode. (Since very high microsteps/sec numbers can be issued) **Eg /1V1000R**
		SET ACCELERATION COMMANDS
L	**0-5000** (1)	In EZ10 **Set Acceleration** factor accel = microsteps / sec^2). **Eg /1L50R.** Accel in microsteps / sec^2) = (L Value) x (400,000,000/65536) . **Eg. using t=V/a /1L1R takes 8.192 Seconds to get to a speed of V=50000 microsteps/second**

EZ Stepper® STEPPER-STICK and EZ10EN

		LOOPING AND BRANCHING COMMANDS
g		**Beginning of a repeat loop** marker. See examples below on how to set up a loop. **Eg. /1gP1000M1000G10R**
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**

		ANALOG TO DIGITAL CONVRTER COMMANDS
		All 4 Inputs are ADC inputs and can be read and acted upon by the program. (SEE APPENDIX 9 for details)
?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. This can be used for example to regulate pressure to a given level, by turning a motor on/off at a given voltage.

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.

This command requires V5.93d+ Firmware. |
| **?at** | | Reads back the thresholds for all 4 channels. The Readback order is channels 4:3:2:1 **Eg /1?at<CR>** |

EZ Stepper® STEPPER-STICK and EZ10EN

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. **Eg /1gH02P1000G20R** - waits for switch 2 H command with no number after it waits for Switch 2 Closure (low).
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. **Eg /1gS02A1000A0G20R** - skips on switch 2 Also see "H" command for I/O dependant program execution.

EZ Stepper®   STEPPER-STICK and EZ10EN

		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. (14 full commands max per string) **Eg. /1s1A1000A0R** Note: This command takes approx 1 Second to write to the EEPROM.
e	0-15	**Executes Stored Program** 0-15. **Eg. /1e1R**
		PROGRAM EXECUTION
R		**Run** the command string that is currently in the execution buffer. **Eg. /1R**
X		**Repeat Run** the current command string
		SET MAX MOVE CURRENT / HOLD CURRENT
m	0-100 (25)	**For Steppers " Move" Current on a scale of 0 to 100% of max current.** 100% = 1A for EZ10. **Eg. /1m40R**
h	0-50 (10)	**Sets "Hold" Current** on a scale of 0 to 50% of max current. 100% = 1A for EZ10 **Eg. /1h15R** A value of h=0 will turn the output stage completely off in the Stepper stick. (Sleep mode Implemented in Firmware V5.93B+)
aw	-	Reserved
ax	-	Reserved
ay	-	Reserved

EZ Stepper®   STEPPER-STICK and EZ10EN

		SET MICROSTEP RESOLUTION / ENCODER
j	**1,2, 4, 8,** (8)	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. (Note that this drive can be left in 8 micro-step mode and very large velocity values issued)
N	**1-2** (1)	**Special Modes** 1 = Encoder With No Index (Default). Homes to Opto. 2 = Encoder With Index. Homes to Index.
n	0-4095 (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 **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 Reserved **Bit6 :** /1n64R Reserved **Bit7 :** /1n128R Reserved **Bit8 :** /1n256R Reserved **Bit9 :** /1n512R Reserved **Bit10:** /1n1024R Reserved **Bit11** /1n2048R Reserved.

EZ Stepper®   STEPPER-STICK and EZ10EN

		MISCELLANIOUS
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.
M	**0-30000**	Wait M number of Milliseconds.
		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. Eg **/1J3R** Starting in firmware revision 5.93C the J command can be issued while the motor is running. As an immediate command with no R at the end /1J3<CR> 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. The Baud rate is

EZ Stepper® STEPPER-STICK and EZ10EN

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

		IMMEDIATE QUERY COMMANDS
		These commands do not require an "R" at the end
T		Terminate current command or loop. eg /1T
?	0	Returns the current Commanded motor position eg. /1?0
?	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. Bit 0 = Switch1 Bit 1 = Switch2 Bit 2 = Opto 1 Bit 3 = Opto 2
?	5	Reserved
?	6	Returns the current step size microsteps /step
?	7	Reserved
?	8	Returns Encoder Position. (can be zeroed by "z" command)
?	9	Erases all stored commands in EEPROM.
&		Returns the current Firmware revision and date
Q		Query current status of EZ Stepper/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 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. See Appendix 7
n		The n mode works in both immediate mode and in strings.
		**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.

EZ Stepper® STEPPER-STICK and EZ10EN

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)

 /1gA1000M500A0M500G10R<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. "**A1000**" makes the motor turn to **A**bsolute position **1000**
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**

EZ Stepper®   STEPPER-STICK and EZ10EN

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)

/1gA100A1000gA100A10G10G100R\<CR>

/1	Talk to motor number 1.
g	Start outer loop
A100	Goto Absolute position 100
A1000	Goto Absolute position 1000.
g	Start inner loop.
A100	Goto Absolute position 100.
A10	Goto Absolute position 10.
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)

/1s0gA0A1000S13e1G0R\<CR>
/1s1gA0A100S03e0G0R\<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 A1000 if input3 is High and between A0 and A100 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
A1000	Goto Absolute position 1000.
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
A1000	Goto Absolute position 1000.
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>
/1s1A1000e0R<CR>
/1s2A2000e0R<CR>
/1s3A3000e0R<CR>
/1s4A4000e0R<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 1000. 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**

EZ Stepper®   STEPPER-STICK and EZ10EN

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

/1A1000<CR>
/2A200<CR>
/AR<CR>

/1A1000<CR>	Set up motor 1 command buffer to go to **A**bsolute position **1000.**
/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.

EZ Stepper® STEPPER-STICK and EZ10EN

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", and "h" commands to set the current regulation at or below the rating for the motor. The default values on power up are h=10% and m=25% and should be safe for most motors.

EZ Stepper® STEPPER-STICK and EZ10EN

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 the current speed "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. The default home mode f = 0 expects the home input to be a TTL high when at home (opto cut) and TTL low when away from home (opto not cut). To set up home: First issue a /1P1000R command the motor should move away from home, if not flip the connections to only one of the windings of the stepper motor to reverse the direction considered positive. Now issue a /1Z1000R , if motor does not go towards home, issue /1f1R to flip sense of home flag.

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.

Details:
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).

EZ Stepper®   STEPPER-STICK and EZ10EN

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). A motor that is good for microstepping will feel smooth when rotated by hand (somewhat like a DC motor) with little tendency to detent.
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. Typically the move current must be set equal to the hold current, when accuracy is required, because if the current is reduced at the end of the move, the motor will fall back into a detent position.

EZ Stepper® STEPPER-STICK and EZ10EN

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 – average over 5 minutes). 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.

Most applications will not require derating of the drive.

EZSteppers are designed with parts rated at 85C or better. This means the PCB copper temperature must remain below 85C. The ambient air temperature allowed depends on the airflow conditions.

MTBF is 20000Hrs at 85C PCB copper temperature and doubles for every 10C lower than 85C.

EZ Stepper®   STEPPER-STICK and EZ10EN

For some applications, which operate without encoder feedback, it may be necessary 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.

EZ Stepper®   STEPPER-STICK and EZ10EN

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.

OEM Protocol requires Version 5.93A or later later firmware. 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.)

EZ Stepper®   STEPPER-STICK and EZ10EN

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.

EZ Stepper®   STEPPER-STICK and EZ10EN

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".

**Note that V5.93 Code had an extra NULL character before the ETX in the response
and this has been removed in V5.93A and beyond.**

EZ Stepper®   STEPPER-STICK and EZ10EN

APPENDIX 8   JOG MODES AND LIMIT SWITCHES

JOG:
The Drive can be placed in a mode that will allow the two switch input to "Jog" the motor backwards and forwards. The command for this is /1n4R. Once issued the motor can be moved by pressing switch 1 or 2. Internally these inputs are "pulled up" with 10K to 3.3V and a switch closure to ground is all that is required.

Note that these are inputs 1 and 2 and the status of the switches can be read via software by /1?4 which returns a binary number between 0-15 which represents the status of the 4 inputs.

LIMIT MODES:
The Drive uses the two opto inputs as limits. These are inputs 3 and 4. It should be noted that these inputs are general purpose inputs and can be driven by switch closures etc or any device that produces a voltage change. The inputs are actually ADC inputs and the One/Zero threshold can be set by the "at" command.

Input 3 is the lower limit and also the home switch. Input 4 is the upper limit. Both limits are simultaneously turned on by the /1n2R command. The default expects optos which are low when away from the limit, this polarity can be changed by issuing /1f1R, such as when normally open switches are used for limits. (Note: normally closed switches are better for limits, since any disconnect of the wires will shut down the motion). When n2 mode is engaged the motor will not move in the direction in which a limit is active, but will back out of the limit.

Note that the limit inputs can be used to "kill moves" and the execution will move to the next command, eg /1gA1000A0GR will stop at 1000 or 0 depending on which limit is active

NOISE CONSIDERATIONS:
The inputs are relatively high impedance at 10K and will pick up noise if bundled with the motor wires etc. For long cable runs each input line should be shielded. The addition of a 0.1uF ceramic capacitor from the input to ground at the Board connector may be an alternative to shielding but may slow the response.

EZ Stepper®   STEPPER-STICK and EZ10EN

APPENDIX 9 ANALOG INPUTS

The 4 Inputs on the EZStepper 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 - 4.95V. 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). This command requires V5.93d+ Firmware.

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

The equivalent input circuit is a 10K Pullup to +5V and 15K pulldown to ground.