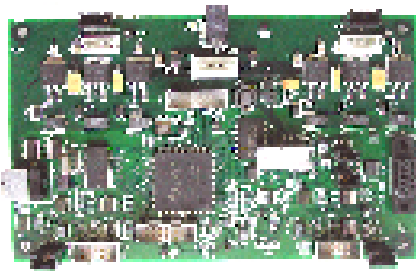


Table of Contents

page 2	kit description
page 3	motor/encoder datasheet
page 4 - 6	setup + connections
page 7	parameter scaling
page 8-9	command-line commands
page 10-12	'how to' command line mode
page 13-14	debug mode commands
page 15-17	'how to' debug mode





second motor optional

Description

The CM-300-VOX developer's kit was specifically designed as a development tool for applications requiring resolution and repeatability of 2 arcseconds or less.

The kit is completely self contained - it interfaces directly to a PC either via an RS-232 or USB-2.0 serial port. The feedback control system and loop filter are pre-programmed into the processor.

The kit typically installs in less than 10 minutes.

The basic package consists of the following elements:

- dual axis controller with dual on-board amplifiers,
- one CM-335-1024 direct drive brushless DC motor with integral sine/cosine encoder
- a 5V, 3A power supply (110 - 220 Vac input)
- USB cable
- CD with software, MatLab & Labview support, examples

Larger motors are available - please contact the factory for options. The CM-300-VOX is also available as a controller - amplifier only. Customer support for different motor/encoder configurations is available.

applications

XY tables, spectrum analyzers, tuneable lasers, programmable attenuators, 3D scanners, monochromators, excentric-based micro-manipulators, seeker heads, mirror positioners.

ordering information

- CM-300-VOX(X)**
- x=0 controller PCB, software and power supply only
 - x=1 one motor complete system
 - x=2 two motor complete system

customization for larger volume applications is welcome.

Specifications :

precision controller:

encoder lines per revolution	up to 16,384 lines (full sine/cosine cycles)
encoder input	sine/cosine, differential
encoder index format	differential/single ended analog or TTL
encoder input pulse rate	500 kHz max
interpolation factor	4,096 per sine/cosine cycle
resolved counts/rev	up to 67,108,864
number of axes	2
loop sampling interval	55 μ s
loop filter	PID
rotation speed range	one rev/10 months to 900 RPM with supplied motor/encoder and power supply
output PWM	33 kHz, 12 bits, 16.5 KHz, 13 bits accessible for driving external high-power amplifiers
communication interfaces	RS-232 and USB-2.0

power inputs:

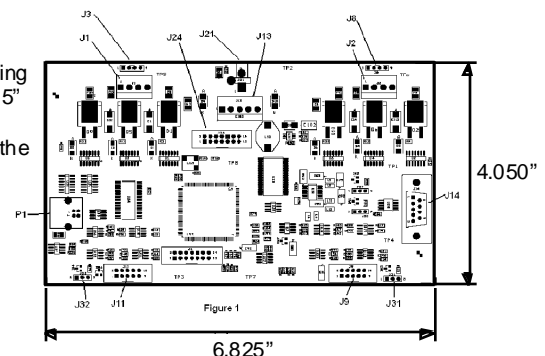
connector J21 (2.5 mm power)	5 V @ 3A for controller and servo
connector J13 (external power)	5V to controller, 5-36V to servo amp for axis0, 5-36V to servo amp for axis 1

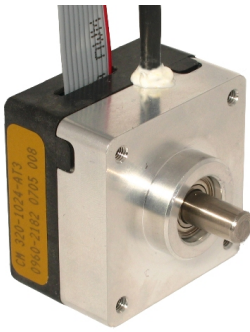
requirements for external user supplied power:

controller power	+5V @ 0.8 A
amplifier current	5 to 36 V, 2A max. per axis
motor drive voltage	5 to 36 V, 2A maximum per phase

controller dimensions:

four mounting holes, 0.125" dia, 0.25" from the corner.





Description

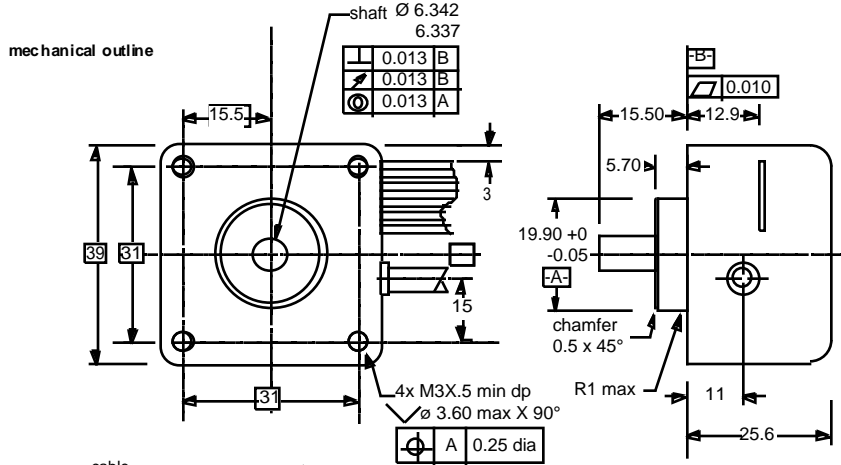
the CM-300-1024 is a high performance brushless DC motor with an integral optical high-resolution encoder.

The motor is characterized by a very high torque to inertia ratio and negligible cogging.

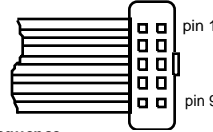
The high resolution incremental sine/cosine encoder was specifically designed to work with high ratio interpolators and is characterized by very low distortion (Lissajous Rmax/Rmin of less than 1.05).

encoder specifications:

Vcc = 5V ± 5%
 Icc = 50 mA max.
 Vpp = 2V differential (4V after receiving differential amplifier)
 linecount = 1024 sine/cosine cycles
 frequency response: flat up to 50 kHz (3000 RPM)
 Zout is the equivalent of a TIL 084 opamp output in series with a 51Ω resistor.



cable, 3M™ p/n 3365 /10, 20 cm ± 1 cm long
 connector, 3M™ p/n 3473-6610



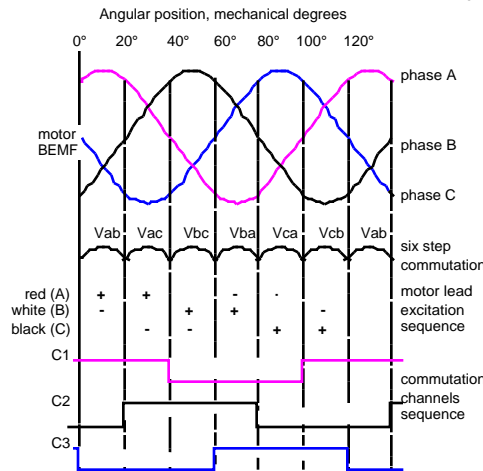
encoder pinouts:

- 1 Vcc
- 2 ground
- 3 C3
- 4 C2
- 5 A inv.
- 6 A
- 7 ind ex
- 8 C1
- 9 B
- 10 B inv.

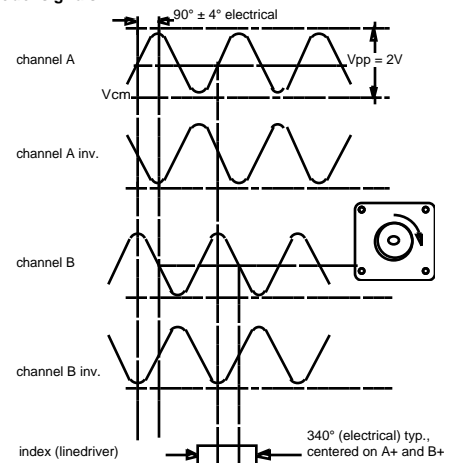
motor pinouts:

- 1 phase A red
- 2 phase B white
- 3 phase C black
- 4 case/shield green

commutation sequence and commutation signals



encoder signals



motor constants @ 20° ambient

PARAMETER	symbol	UNITS	value
maximum rated torque	Tr	in.oz Nm	33 0.232
max. continuous stall torque	Tc	in.oz Nm	4.4 0.031
maximum continuous output power	Pout Smpo	watts RPM	19.9 8,600
motor constant	Km	oz.in/vW Nm/vW	1.7 0.012
electrical time constant	Te	msec.	0.26
mechanical time constant	Tm	msec.	4.46
thermal resistance*	TPR	°C/W	7
maximum cogging torque	Tf	oz.in Nm	0.14 0.99 x 10 ⁻³
viscous damping		in-oz/rpm Nm/rpm	5.1 x 10 ⁻⁵ 3.6 x 10 ⁻⁷
hysteresis drag torque		in-oz Nm	0.15 1.1 x 10 ⁻³
rotor inertia		in-oz-sec ² Kg.m ²	8.9 x 10 ⁻⁵ 6.3 x 10 ⁻⁷
motor weight		oz Kg	7.6 0.2148
number of poles	P		6

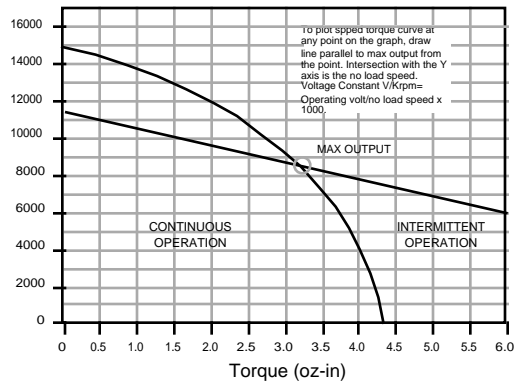
24 V winding constants @ 20°C ambient

parameter	symbol	units	value
peak torque, ± 25%	Vp	in-oz Nm	18 0.13
peak current, ± 15%	Tp	Amps	5.1
torque sensitivity, ± 10%	Kt	in/oz/Amp Nm/Amp	3.65 0.026
no load speed, ± 15%	Snl	rpm rad/sec	8,600 901
voltage constant, ± 10%	Kb	v/Krpm v/rad/sec	2.70 0.026
terminal resistance, ± 12%	Rm	Ohms	4.73
terminal inductance, ± 30%	Lm	mH	1.23
continuous power output @ 75°C temperature rise	power	Watt HP	17.2 0.023
	torque	in-oz Nm	3.5 0.025
	speed	rpm	6,600
	current	Ampers	1.16
	efficiency	percent	61.6

*motor mounted to aluminum heatsink 3.2" x 3.2" x 0.25" thick, no external load

Speed (rpm)

CM-335 speed - torque curve



The Developer's Kit contains a two-axis controller board, a power supply, and a motor/encoder mechanism.

Figure 1 shows the connectors referenced in the discussion which follows. These are:

- J21 +5 power input (low power only)
- J13 +5 V and MOT_0 +V and MOT_1 +V auxiliary inputs
- J1 Axis 0 Motor drive output connector for currents up to 2 Amps
- J3 Axis 0 motor drive output connector for currents up to 0.5 Amp.
- J11 Axis 0 encoder input
- J32 Axis 0 position-capture strobe input.
- J2 Axis 1 Motor drive output connector for currents up to 2 Amps
- J8 Axis 1 motor drive output connector for currents up to 0.5 Amp.
- J9 Axis 1 encoder input
- J31 Axis 1 position-capture strobe inputs
- J14 RS-232 serial communications connector
- P1 USB-2.0 serial communications connector

Connecting Power

The +5 volt power supply provided with the Kit is of the auto-ranging type and can accept input voltages from 100 VAC to 240 VAC, 47 Hz. to 63 Hz., with no need for user adjustment. A CEE-style power cord suitable for use in North America is provided. For European or other markets, you will need to obtain a CEE power cord suitable for your local power environment.

Connect the power supply to your local mains power environment with a suitable CEE power cord. The power supply will create +5 volts for the controller board.

Connect the power supply output plug to connector J21. You should see a red LED next to J21 light up, indicating power is being supplied to the controller.

The power supply is suitable for powering both the controller board and one or two of the small motor/encoder mechanism provided with the Kit.

For larger motors or where higher voltages for driving the motor are desired, external power must be supplied via connector J13.

Connecting RS-232 Serial Communications

You will need to provide your own serial cable. Connect a suitable RS-232 "DB-9" cable between controller board connector J14, and a communications terminal such as your computer running a terminal emulator. For a typical Intel-processor-based personal computer, the usual RS-232 cable will have a male DB-9 connector on the controller end, and a female DB-9 connector on the computer end.

One example of a cable used successfully with a personal computer is a Belkin model F2N209-06-1 cable. Usually the cable should NOT be a "null modem" type cable. You will have to set your terminal properties correctly. The controller RS-232 port expects a 57.6 kilobaud connection, 8 bits, no parity, 1 stop bit, no flow-control. Check your terminal properties and make any necessary changes. The Windows "Hyper Terminal" terminal emulator program has been used successfully with the above settings.

When correct communications have been established, striking the "RETURN" key should elicit a response from the controller board of:

```
Error performing requested operation.
cmdline>
```

If this does not work, check the connections between the controller board and the terminal, make sure the terminal is set for the correct baud rate etc., that any terminal emulator program is talking to the correct computer connector, and make sure that the TX and Rx connections are not reversed (the classic RS-232 "gender" problem).

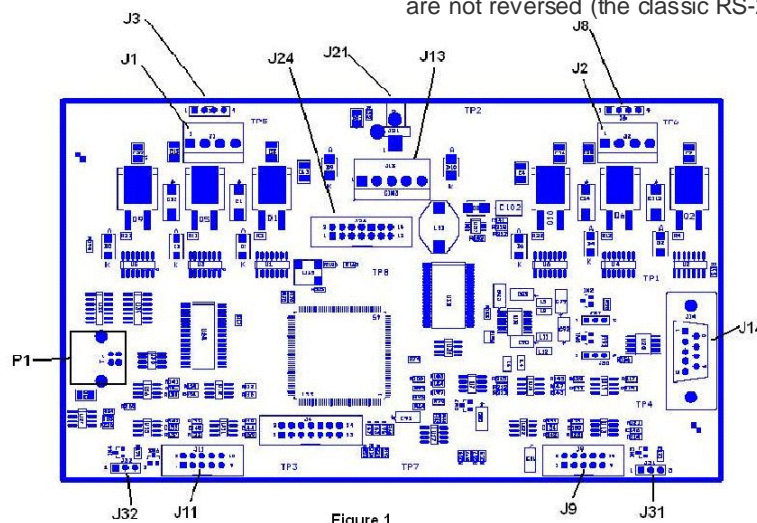


Figure 1

Connecting the Motor/Encoder mechanism

The motor/encoder mechanism provided as part of the VMC motion control kit can be connected to the controller either before or after application of power. For larger motors you may wish to apply motor power after connecting the motor.

The encoder interface for each axis is via a 10-wire ribbon cable and connector, and the motor interface is via a four-wire cable and connector. For Axis # 0, connect the encoder cable to J11 and the motor cable to J3. If Axis #1 is the desired axis, connect the encoder cable to J9 and the motor cable to J8. At this point you are ready to send commands to the controller, and should be able to drive the motor/encoder mechanism as desired, within the limits of the mechanism and the power supply.

(See page xx for examples of how to inform the controller of encoder and motor parameters, set loop-filter gains, read mechanism position information, enable or disable servo control for each axis, etc. via the RS-232 interface).

Connector details

J21 is a coaxial power jack suitable for currents of up to 5 Amps. The center pin is 0.098" (2.49 mm) in diameter, the outer sleeve is 0.25" (6.3 mm) diameter. The center pin is the +5 volt input while the sleeve is the Common return ("ground") connection.

J13 is a five pin connector with pins on 0.156" (3.96 mm) centers, that provides a second means of providing power to the board. The mating connector is an AMP/Tyco p/n 640428-5

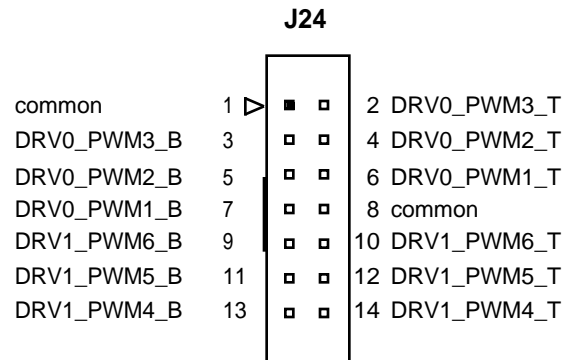
Connections are as follows:

- Pin 1 MOT_0 +V +5 to +36 volts, 2 Amps max supply to Motor on Axis 0
- Pin 2 MOTOR COMMON
- Pin 3 +5 Volts +5 Volts +/- 5%, 2 Amps max
- Pin 4 CONTROLLER COMMON
- Pin 5 MOT_1 +V +5 to +36 volts, 2 Amps max, supply to Motor on Axis 1

Note that the voltages applied to pins 1 and 5 MUST be +5 volts or greater, or they may load down the +5 supply on pin 3.

J24 provides access to the logic level PWM signals from the controller. This allows the user to connect an external "amplifier" if desired, for driving motors requiring more drive capability than that of the PWM half-bridge drivers on the controller board. A suitable mating connector is the 3M p/n 89114-0001.

Signals are arranged so that in a ribbon cable, each signal and it's complement are adjacent in the cable, for minimum radiation of EMI.



PWM signals from controller

J1 and J2 are four pin connectors with contacts on 0.156" (3.96 mm) centers, that provide a high current means of connecting a motor to the half-bridge PWM drivers for each axis. A suitable mating connector is an AMP/Tyco p/n 640601-4. J1 is for the Axis 0 mechanism while J2 is for the Axis 1 mechanism.

Connections are as follows:

- Pin 1 Motor Phase A
- Pin 2 Motor Phase B
- Pin 3 Motor Phase C
- Pin 4 Motor frame; connects to the controller amplifier motor common via a 10 ohm resistor.

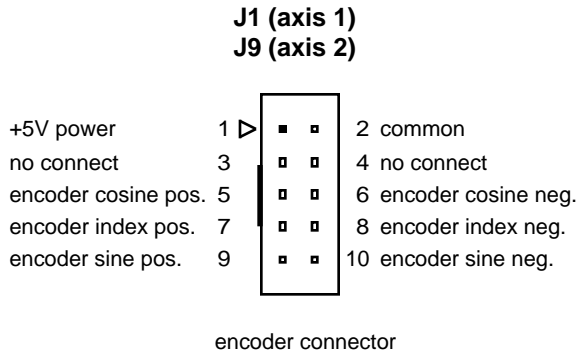
J3 and J8 are four pin connectors with contacts on 0.1" (2.54 mm) centers, that provide an easy means of connecting the motor of the motor/encoder provided with the Motion Control Kit to the half-bridge PWM drivers for each axis. A suitable mating connector is an AMP/Tyco p/n 102241-2 housing with contact pins suitable for the wire being used. Some suitable contacts include the following:

AMP/Tyco p/n	Gauge/AWG
1-87309-4	24-20
87667-2	26-22
102128-1	26-22

J3 is for the Axis 0 mechanism while J8 is for the Axis 1 mechanism.

J9 and J11 are 10-pin connectors with two rows of five contacts each, on 0.10" (2.45 mm) centers. Suitable mating connectors include the 3M 3473-6600 or the 3M 3473-6610. J1 is for Axis 0 while J9 is for Axis 1.

Connections are as follows:

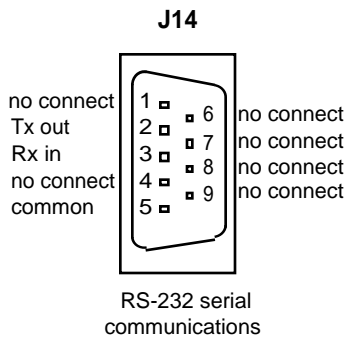


J31 and J32 are three-pin position-capture strobe inputs. Input is a TTL logic level, the position is captured on a falling edge.

Connections are as follows

Pin 1 Common
Pin 2 Strobe input
Pin 3 Common

J14 is a standard female "DB-9" RS-232 serial communications connector. Connections are as follows:



P1 is the USB-2.0 serial communications connector. This is a standard USB type B connector. Connections are as follows

Pin 1 Vcc (sense only, does not use power)
Pin 2 D-
Pin 3 D+
Pin 4 Ground

SCALING OF PARAMETERS FOR THE VOXIS SERVO CONTROLLER BOARD:

The processor used for the controller is a fixed-point processor, so all actual processor computations are done as integers. But for convenience to the user, interface values are usually input as floating point values.

Values may be entered in any valid C "float" format, that is, 25.3, 2.53e1, 0.00253e4, and 2530e-2 are all equivalent values. Integer values may be entered as simple integers, e.g. 25.

PID parameters:

Negative values are not allowed. If a negative value is entered, the previously existing value remains unchanged and an error message is sent to the user.

The smallest non-zero value is 0.004 (actually, 1/256)

The largest allowed value for P, I, or D gains is 32767.0, in increments of 1/256. The largest allowed value for Ilim is 8388607.0, in increments of 1/256.

The dynamic range of the PID values is that of a 23-bit positive number, with the eight LSBs representing a fraction.

Position parameters:

Units are interpolated encoder counts. Note that for position, both positive and negative values are allowed. The ratio of largest-value/smallest-value is 2147483648:1 (32-bit signed value).

Resolution is the product of the number of encoder lines and the interpolation factor (currently a maximum of 4096 counts per encoder line) counts per shaft revolution.

For example, if the mechanism being controlled has a 1024-line encoder, the number of counts per revolution is $1024 * 4096 = 4,196,304$ counts per revolution. For this environment, one LSB of position is an angle of 1.489 micro-radians (0.3 arcseconds).

For a 9000 line encoder, the counts per revolution are $9000 * 4096 = 36,864,000$ and one LSB of position is 0.17 micro-radians, roughly 1/9 of the previous example.

Motion Profile Parameters:

As with the PID parameters, negative values are not allowed for velocity and acceleration, as these are magnitudes only. Negative velocities are achieved by specifying a target position that is less than the current position, such that the difference (target-position - current-position) yields a negative number.

Units for velocity are (position counts/216)/sample-time, and for acceleration are (position counts/216)/sample-time/sample-time.

Note carefully that motion profile target velocity and acceleration values are both scaled by a factor of 65,536 ($=2^{16}$) relative to position values. This means that if the target velocity is set to 1, it will take 65,536 sample intervals for the velocity target value to accumulate enough to cause a 1 bit change in the actual position of the shaft.

The sample time for each axis is $2 * (4128)/150e6 = 55.04$ microseconds.

List of Command-line commands controller over the RS-232 serial port:

Each command is a two character string followed by one or more comma-separated numeric values. Note that some of these are integer, most are decimal (float) values.

The parser will convert decimal values to integer if needed, but will truncate any decimal fraction during the conversion.

Command Syntax

Description

CV AX, ACC, VEL, DIR

Constant Velocity mode on specified axis
 AX = AXIS, 0 or 1,
 ACC = encoder counts/65536 per sample-period
 VEL = encoder counts/65536 per sample period
 DIR = 1 for CW, -1 for CCW

EI AX, VAL

Encoder Initialize on specified axis.
 AX = AXIS, 0 or 1,
 VAL = INTEGER encoder lines per rev.
 Max value for encoder lines = 16384 (= 2^{14})

FI AX

Find Index position on specified axis
 AX = AXIS, 0 or 1,
 On completion, the system is actively controlling the mechanism at the index position, which by definition is position 0.0

FP AX, VAL

Find Phases for specified axis.
 AX = AXIS, 0 or 1,
 VAL = INTEGER 0 to 4095
 This is the value to be used by the PWMs during finding of the motor phase positions relative to the encoder. The voltage applied to the motor phase will be approximately $V_{motor} * VAL/4095$

KP AX, VAL

Set K_p (proportional gain) for specified axis
 AX = axis; 0 or 1
 VAL = K_p gain, decimal positive only
 Maximum = 32,767.
 Minimum of 0.000, steps of 0.004.
 See scaling description for more detail.

KI AX, VAL

Set K_i (integral gain) for specified axis
 AX = axis; 0 or 1
 VAL = K_i gain, decimal positive only
 Maximum = 32,767.
 Minimum of 0.000, steps of 0.004.
 See scaling description for more detail.

KD AX, VAL

Set K_d (derivative gain) for specified axis
 AX = axis; 0 or 1
 VAL = K_d gain, decimal positive only
 Maximum = 32,767.
 Minimum of 0.000, steps of 0.004.
 See scaling description for more detail.

KL AX, VAL

Set K_{iLim} (limit on integrated error) for specified axis
 AX = axis; 0 or 1
 VAL = K_i clamping magnitude value, decimal positive only
 Maximum = 8,388,607.0 (= $2^{23} - 1$)
 Minimum of 0.000, steps of 0.004. (actually 1/256)
 See scaling description for more detail.

LE AX, VAL

Loop Enable PID loop.
 AX = axis; 0 or 1
 VAL: 0 = disable, 1 = enable
 Disabling an axis sets the motor drive to 0, e.g. all PWMs at 50%

MI AX, VAL	<p><u>M</u>otor <u>I</u>nitialize on specified axis. AX = axis; 0 or 1 VAL = INTEGER number of motor poles (always an even number)</p>	PE AX, VAL	<p><u>P</u>rofile <u>E</u>nable of motion defined by PS command, for specified axis. AX = axis, 0 or 1 VAL = 0 disables the servo leaving the PWMs at 50% VAL = 1 enables the servo.</p>
SA AX, VAL	<p><u>S</u>et <u>A</u>bsolute position for specified axis. AX = axis; 0 or 1 VAL = new set point in decimal radians. This is an offset from the "home" or index position. This causes the mechanism to traverse as rapidly as it can to the new set point. In effect, this is a "step" input to the target position.</p>		
SR AX, VAL	<p><u>S</u>et <u>R</u>elative position for specified axis. AX = axis; 0 or 1 VAL = distance to move from present set point, in decimal radians</p>		
RP AX	<p><u>R</u>eport current <u>P</u>osition. AX = axis; 0 or 1 Reported position is in radians.</p>		
RC AX	<p><u>R</u>eport <u>C</u>aptured position. AX = axis; 0 or 1 A high-to-low logic level transition on J32 captures the position of Axis 0, a high-to-low transition on J31 captures the position of Axis 1. Capture is asynchronous to the control sample interval. The reported position is in interpolated encoder counts.</p>		
PS AX,AC,VE,EP,MD	<p>motion <u>P</u>rofile <u>S</u>et for specified axis. AX = axis; 0 or 1 AC= acceleration, encoder counts/sample-time/sample-time VE = max velocity, encoder counts/sample-time EP = end pos, encoder counts MD: specify profile mode, MD =1 commands a trapezoid velocity profile, MD =2 commands a repetitive triangle motion MD =3 commands a repetitive saw-tooth motion</p>		

Example for setting commands via RS-232.

Connect +5 Volt power supply to controller board J21 "+5 VOLTS"

Connect an RS-232 cable between controller board J14, "RS-232" and a terminal such as your computer.

The controller RS-232 port expects a 57.6 kilo-baud connection, 8 bits, no parity, 1 stop bit.

When the terminal has established a connection, striking the "RETURN" key should elicit a response from the controller board of

```
servo >
```

If this does not work, check the connections between the controller board and the terminal, make sure the terminal is set for the correct baud rate etc, and make sure that the TX and Rx connections are not reversed (the classic RS-232 "gender" problem).

Controller Command Example:

(See the servo commands list for a detailed list of commands.) The controller can handle two axes independently. The axes are denoted as "0" and "1". Each command is a text sequence made up of the command itself followed by numeric value denoting which axis the command is referring to, followed by further values as defined by the servo commands list. Multiple values in a command are comma-separated. For example, during initialization of a servo axis the controller provides power to the motor windings to determine the commutation positions. The amount of power to be applied to the motor is set by the "FP" (Find Phases) command. A typical command string for "FP" would look like:

```
FP 0, 513 <ret>
```

This would tell the controller that for axis zero, the amount of voltage to be applied to the windings during initialization is to be 513/4096 of the motor supply voltage.

Controlling an axis:

This example will show how to hook up a motor and encoder to axis 0 (zero), inform the controller as to necessary motor and encoder parameters for that axis, set the control loop parameters for that axis, check the position of the mechanism, initialize the motor commutation function, find the index position of the encoder and define that as position "0", set up a motion profile, and have the controller execute the profile.

Proceed as follows:

Connect the mechanism encoder cable to connector J11, "ENCODER #0" Connect the mechanism motor cable to connector J3 "MOTOR #0" (or use J1 with a suitable connector, for larger motors).

Note that the default motor supply voltage is +5 volts. For larger motors or large loads, higher motor supply voltages up to +36 volts can be provided by the user, via connector J13.

For the provided VM-17 mechanism with no attached load and running at the default motor supply voltage of +5 volts, here is a set of commands that will get a VM-17 on axis 0 initialized, set up suitable loop filter parameters, search for the mechanism index position, servo at the index position, and then execute a controlled move to a position one half of a revolution away from the index position. At the end of the motion, the last command disables servo control.

command	comment
EI 0, 1024 <ret>	encoder attached to axis 0 has 1024 lines
MI 0, 6 <ret>	motor attached to axis 0 has six poles
FP 0, 527 <ret>	on axis 0, use 527 as magnitude value for finding phases
KP 0, 4 <ret>	For axis 0, set loop filter parameter Kp to 4
KI 0, 1 <ret>	For axis 0, set loop filter parameter Ki to 1
KD 0, 10 <ret>	For axis 0, set loop filter parameter Kd to 10
KL 0, 32000 <ret>	For axis 0, set loop filter parameter KiLim to 10.0
LE 0, 1 <ret>	For axis 0, enable servo loop control
FI 0 <ret>	For axis 0, find the "index" position.
SA 0, 333772 <ret>	Set axis 0 loop target position to 333772
PS 0, 337224, 3338,	1688887, 1 Parameters are: Axis = 0 *Target velocity = 337,224 * Target Acceleration = 3338 Target end position = 1,688,887 position counts Desired motion = 1 (= triangle)
PE 0, 1	* Keep in mind that acceleration and velocity are scaled by a factor of 216 For axis 0, start the most recently specified move.
LE 0, 0	Disable servo control on axis 0. E.g., stop servo control, motor coasts.

What follows is a more detailed description of what is being done with each command.

It is necessary to let the controller have some information regarding the mechanism it is to control.

Encoder:

Be aware that most incremental encoder vendors like to talk about the number of positions that can be resolved, rather than the number of lines on the encoder. For the common square-wave (TTL) quadrature incremental encoder, the number of positions can be four times the number of lines. What is needed here is the number of lines, not the "quadrature" value which is four times larger. Be careful to use the correct value.

For the VM-17 provided with the kit, the encoder has 1024 lines. So the encoder initialization instruction will be

```
EI 0, 1024 <ret>
```

Motor:

The controller synthesizes sinusoidal voltages for each phase of the motor. It does not make use of traditional Hall commutation, rather it extracts commutation information from the position encoder of the mechanism. This results in significantly reduced motor torque ripple compared to commutation from Hall sensors.

But to do this it is necessary for the controller to know the number of motor magnetic poles (which due to the physics of magnetism will always be an even number). For the VM-17 mechanism provided with the Voxis "kit", the brushless DC (BLDC) motor has six poles. So the motor initialization instruction would be:

```
MI 0, 6 <ret>
```

Knowing the number of poles in the motor is only part of the needed information. It is also necessary to know the zero torque position of each winding in the coordinates of the encoder. This is determined by applying power for a brief period to each phase and watching to see what the motor response is.

An optimal value for finding the phase location can be pre-calculated but it requires knowledge of the motor torque constant, the winding resistance, the motor damping behavior, the load inertia, and bearing friction. A value that is larger than optimum will cause the motor to settle with a damped oscillatory motion, a value too small may not allow the motor to settle to the proper position and may not cause the motor to move at all. The easiest way to determine a good value for finding the phase location is to make a few experiments and set a value by observation.

For the CM-300-VOX motor/encoder mechanism provided with the kit, a typical command to set the motor initialization power value would be:

```
FP 0, 527 <ret>
```

This will cause the controller to apply a voltage of 527/4096 of the motor supply voltage (5 volts in this case) to each winding of the motor on axis #0 during the "Find Phases" sequence.

Loop Filter:

Now that the encoder, motor, and commutation values are determined, it is time to set the loop filter parameters. The loop filter is a simple PID filter, with the usual "anti-windup" limit on the integrated position error as an additional settable parameter.

Suitable values for the unloaded motor/encoder provided with the "kit" are:

```
Kp = 4  
Ki = 1  
Kd = 10  
KiLim = 32000
```

Other values will be necessary for other mechanisms or for the provided mechanism if it is driving a load. These can be determined either by calculation (see the discussion of gain setting, elsewhere) or by experimentation.

The classical experimental means of setting a PID loop is to set all loop values to zero, except the KiLim value which is usually set to it's maximum value. First the Kp term is increased until the step response of the mechanism is ringing and on the verge of oscillation. Next the Kd term is increased until a good settling behavior is obtained.

Finally, the Ki term is increased until the settling behavior begins to degrade, as shown by the return of ringing to the step response.

At this time, the KiLim term can be reduced until the controller is no longer able to maintain a zero position error, at which time the KiLim term can be increased (perhaps doubled) to provide some guard band.

Issuing the following commands will set the controller appropriately:

```
KP 0, 4 <ret>  
KI 0, 1 <ret>  
KD 0, 10 <ret>  
KL 0, 32000 <ret>
```

Up to this point, there has been no servo control activity. The next step is to enable the servo control function:

```
LE 0, 1 <ret>
```

Now the servo system can be used to locate the index position. The encoder has an index pulse output that the servo control can search for. On execution of the FI command the index position will be located and established as the "home" position.

```
FI 0 <ret>
```

Note that it is not strictly necessary to find the index position, but if it is not located, the controller will take it's "home" position as whatever is present at power-up in it's incremental position registers, that is to say, a random position. If it is desired to be able to cycle power and then return to a previously determined position, the best way is usually to do so in terms of some recorded offset from the index position. Next, one can send the mechanism to a targeted position using the SA command.

This will set the target position to the specified location, and the servo system will respond by moving to the target position as fast as possible.

In the following example, the mechanism is commanded to servo on the index position:

```
SA 0, 0 <ret>
```

There are three motion profiles available: trapezoidal move, a repetitive "triangle" motion, and a repetitive "saw-tooth" motion.

Each is specified in terms of the same parameters; target acceleration, target velocity, and target end position. These control the motion profile generator, which the servo system will attempt to follow as closely as possible.

To set up a trapezoid motion, pick the desired end position, the desired velocity for slewing to that position, and the desired acceleration for ramping up to the target velocity and back to zero velocity at the end target position. Then enable the motion. Note that acceleration and velocity are scaled by a factor of 2^{16} relative to the position values.

```
PS 0, 337224, 3338, 1688887, 1
    Set up parameters for a
    triangular move.
```

Parameters are:

Axis = 0

Target velocity = 337,224

Target Acceleration = 3338

Target end position = 1,688,887 position counts

Desired motion = 1 (= triangle)

```
PE 0, 1          Start the specified move.
```

Commands for “Debug” mode:

dg

This is actually a command in the command-line mode, not debug mode. It is used to get into the “debug” style of commands documented below, from the (default) command-line style that is present at power-up.

q - Quit from debug mode.

This command is used to exit from “debug” style commands and return to command-line style commands.

a - Display various angle parameters, mainly having to do with angular position of the mechanism. A single line of output will be displayed. The line is continuously overwritten with fresh data. A typical line of output will look like:

```
QEP: ab9, Drv: 0, Cur: 0.337221, Set:
0.00, Err: -0.337221, T: db
```

A - Essentially the same as “a”, except that the displayed lines scroll up the screen as each new line of data is written.

```
QEP: ab9 Drv: 0 Ang: 0.336832 Err: -
0.336832 Sin: -833 Cos: 1099 Atan:
0.92222 db db
```

e - Initialize controller axis with number of encoder lines and number of motor poles. The command will prompt the user for the values it needs. First prompt is

```
Enter enc lines:
```

User should respond with the number of encoder lines, followed by a <ret>.

Second prompt is

```
Enter num poles:
```

User should respond with the number of motor poles, followed by a <ret>. From the physics of magnetism, this number must be an EVEN value.

The e command has the property of initializing all controller parameters not prompted for to zero.

This command MUST be executed for any of the other commands can do anything sensible.

i - find index

Once controller has been given the initialization commands (e, G, p, t) the controller will drive the mechanism to locate the index position. When the Index position has been found, the “home” position will be set to the index position and all future motion will be referenced to this position.

G - set gains

prompts for values for p, i, d, and iLim. On completion, reports entered values. These values MUST be entered before the controller can function. These values are set to zero at power-up and upon execution of the e command. When a value is prompted for, enter the desired value followed by a <ret>. If no value is entered, the existing value is retained without change. NO OTHER COMMAND EXHIBITS THIS BEHAVIOR.

g - report gains

Displays the gain values that are current at the time of the command. Does not modify them. To modify gain values, use the G command.

p - find phases

Prompts for a value to use in driving the motor phases during the motor phase position search. Value can either integer or hexadecimal. Minimum is 0, maximum is 4095 (0xFFF)

t - enable/disable pid control

Toggles between active control and open loop.

s - specify set point

Units are encoder counts, and are offset from the “home” or “index” position. If pid control is active, the controller will attempt to drive the mechanism to the specified position as rapidly as possible.

w - specify motion profile

prompts for each of the following:

New end point:	units are position counts
New acc:	units are position counts/ sample-period/sample-period
Max vel:	units are position counts/ sample-period
Mode:	1 = trapezoidal motion profile 2 = triangle motion profile (repetitive motion) 3 = saw-tooth motion profile (repetitive motion)

c - specify constant velocity rotation mode

prompts for each of the following:

acceleration:	units are position counts
velocity:	units are position counts
direction:	1 = clockwise rotation (= increasing position value) -1 = counterclockwise rotation

x - toggle axis

Changes between axis 0 and axis 1 for entering parameters

In the following example, commands and responses entered by the user are in **CourierNew BOLD** type, responses from the controller are in normal CourierNew type. Comments about a command or response on the same line as an example line are in *italics*.

Connect the RS-232 communications, motor, and power supply to the controller board, as instructed in the chapter 'Setup'.

Press the "Enter" or "Return" key, denoted here as **<ret>**

```
<ret>           Controller will respond with:  
cmdline>       The controller prompt when in  
                "command line" mode  
Error performing requested operation.  
  
cmdline>
```

This indicates that the controller is in "Command line" style command syntax. If you want to use this command style, refer to the command-line instructions chapter for details. A somewhat more user-friendly syntax is the "debug" syntax, which is entered by typing:

```
cmdline> dg <ret>
```

The controller will respond with:

```
servo 0>
```

which is the controller "debug" mode prompt for servo axis #0.

The basic behavior of this mode is for the user to type a single letter command. The controller will begin executing the requested command; no **<ret>** is needed after a command letter.

If values are needed to execute the command, the controller will prompt for the needed value. The prompted-for value must be terminated with a **<ret>** to tell the controller the value is complete. To change to servo axis #1, type

```
servo 0> x
```

The controller will respond with

```
Selected AXIS 1
```

```
servo 1>
```

To get back to axis #0 enter another **x** at the servo 1> prompt.

Simply hitting **<ret>** at the prompt will elicit a response of:

```
- unknown command
```

```
servo 0>
```

or

```
- unknown command
```

```
servo 1>
```

depending on which axis you have previously selected with the "x" command.

If an unrecognized command is entered, the controller will echo the character it received and respond with

```
servo 0> k
```

```
k - unknown command
```

```
servo 0>
```

From here the user can begin to drive the mechanism via the controller. It is assumed for this example that servo axis 0 is the one being communicated with and controlled.

The first step is to tell the controller the number of encoder lines of the encoder and the number of magnet poles of the motor, for the mechanism attached to the axis being communicated with.

For this example, the mechanism has a 1024 line encoder and a six pole motor.

Tell the controller these values using the "e" command:

```
servo 0> e  
Enter enc lines: 1024 <ret>  
1024 encoder lines  
Enter num poles: 6 <ret>  
6 poles
```

Note that invoking the "e" command will cause the controller to restore all previously set parameters (for things like PID gains, commutation phase information, and motion profile values among others) to zeros. The "e" command functions as a master RESET for the axis being controlled.

Now the controller needs to find the location of the motor phases relative to the encoder, in order to be able to properly drive the motor.

Command the controller to find the phase value using the "p" command. The controller will prompt for a value to use in finding phases. A "reasonable" value for driving the motor of this example is 300 (or 0x121C in hexadecimal).

```
servo 0> p
Phase mag value: 300 <ret>
```

```
servo 0>
New prompt indicates
completion of
instruction
```

The motor/encoder mechanism will move back and forth two or three times as the controller establishes the mechanical relation between the encoder and motor magnets.

Next enter suitable p, i, d, and iLim values using the "G" command. Some "reasonable" values for the mechanism supplied with the kit are p = 0.1, i = .0001, d = 0.4, iLim = 10.

```
servo 0> G
Note that "G" is
uppercase, not
lower case "g".
```

```
New p gain: 4 <ret>
New i gain: 1 <ret>
New d gain: 10 <ret>
New iLim gain: 32e3. <ret>
Gains (p, i, d, iLim): 4.000, 0.098,
10.000, 32000.000
```

```
servo 0>
New prompt indicates
completion of
instruction
```

If you want to see what gains are already entered without making any changes, use the "g" command:

```
servo 0> g
Cur gains (p, i, d, iLim): 4.000, 0.098,
10.000, 32000.000
servo 0>
```

Now the controller can be commanded to enable the servo control process using the "t" command (toggles active servo control between Enabled (= 1) and Disabled (= 0)).

```
servo 0> t
Enable is 1
servo 0>
```

If all is well, the mechanism is now stable and being controlled to maintain position at wherever it was when control was enabled. The set-point and actual position, as well as the difference between the two can be observed using the "a" command.

```
servo 0> a
QEP: f20, Drv: -5, Cur: 6.282958, Set:
0.00, Err: 0.000227, T: ffff
servo 0>
```

These values will be continuously updated, with each new line overwriting the previous line, until any key is pressed.

The "A" command behaves similarly except that each line ends in a line-feed character so the lines scroll up the screen. As with the "a" command this behavior will continue until any key is pressed.

The next step is to have the controller locate the index position of the mechanism:

```
servo 0> i
Found index at 4554224!
servo 0>
```

If the controller is unable to locate the index position after about 60 seconds, the command will time out and report an error.

The mechanism can now be commanded to go to a desired set-point using the "s" command. This will cause the servo controller to send the mechanism to the new set-point as rapidly as the mechanism and available power will allow.

```
servo 0> s
New set pt: 1000000 <ret>
New set pt 1000000
servo 0>
```

A more gradual motion can be commanded using the motion profile command "w". This command prompts for motion parameters and motion type.

```
servo 0> w
New end point: 2000000 <ret>
2,000,000 position
counts ccw from "home"
index position.
New acc: 33377 <ret>
33377 * 1/65565
position counts/sample-
time/sample-time
Max vel: 3337721 <ret>
333,7721 * 1/65565
position counts/sample-
time
Mode: 2 <ret>
requested motion is
"triangle"
```

The mechanism will traverse between the index position (0 position counts) and a position of 2000000 position counts, repeating this endlessly until a key is pressed on the keyboard. During motion, lines of position information will scroll up the screen similar to the "A" command. Capture of these lines allows analysis of the motion being generated if desired.

NOTE: For any motion to take place, the start and end positions must be different. If they are the same value, the controller firmware will cause the mechanism to "traverse" between two identical points, resulting in no motion.

For constant velocity mode, press the 'c' key, and then enter the desired acceleration, velocity, and direction of rotation when prompted. Acceleration and velocity must both be positive numbers:

```
servo 0>c
New acc: 333777
Max vel: 33377219
Direction (CW: 1, CCW: -1): 1
1822 1 -205 135724797 33377219 135698443
```

As always, entering any character exits the command.

To return to "computer" style syntax, type "q" after the prompt:

```
servo 0> q
Completed command successfully.
```

```
cmdline>
```

The controller is now back in the "computer" style syntax it was in by default at powerup. Below are the commands used in the above example, with the explanations removed so just the contents of the screen are shown (minus any scrolling data that may be generated):

```
<ret>
cmdline>
Error performing requested operation.
```

```
cmdline> dg <ret>
```

```
servo 0> e
Enter enc lines: 1024 <ret>
1024 encoder lines
Enter num poles: 6 <ret>
6 poles
```

```
servo 0> p
Phase mag value: 300 <ret>
```

```
servo 0> G
New p gain: 4 <ret>
New i gain: 1 <ret>
New d gain: 10 <ret>
New iLim gain: 32e3 <ret>
Gains (p, i, d, iLim): 4.000, 0.098,
10.000, 32000.000
```

```
servo 0> g
Cur gains (p, i, d, iLim): 4.000,
0.098, 10.000, 32000.000
```

```
servo 0> t
Enable is 1
```

```
servo 0> a
QEP: f20, Drv: -5, Cur: 6.282958, Set:
0.00,Err: 0.000227, T: ffff
```

```
servo 0> i
Found index at 4554224!
```

```
servo 0> s
New set pt: 1 <ret>
New set pt 1
```

```
servo 0> w
New end point: 2000000 <ret>
New acc: 333777 <ret>
Max vel: 33377219 <ret>
Mode: 2 <ret>
```

```
servo 0>c
New acc: 333777
Max vel: 33377219
Direction (CW: 1, CCW: -1): 1
1822 1 -205 135724797 33377219
135698443
```

```
servo 0> q
Completed command successfully.
```

```
cmdline>
```