ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Velmex VXM-1* at our website: ***Click HERE***

VELMEX, INC.

Document # VXM-UM-E5 12-29-04

Important User Information

This information is for the end user of Velmex VXM Stepping Motor Controllers.

VXM Stepping Motor Controller User's Manual

VXM Stepping Motor Controller User's Manual

VXM Stepping Motor Controller

User's Manual

(Extended Version)

Models VXM-1,2,3,4



This Manual explains the general and advanced operation of the VXM-1, VXM-2, VXM-3, and VXM-4 stepping motor controllers.

Also included on the CDROM:

1. COSMOS utility/controller software (A user friendly Windows program for easy setup, testing, and programming of VXM controllers)
2. Software examples and software drivers

Precautions

⚠ CAUTION:

Controller and AC power supply should be operating in a well ventilated area. Do not use in a wet, dirty, or explosive environment. In industrial environments, repackaging into a NEMA grade enclosure is required.

Do not disconnect motor while running. Keep Motor and Limit cables minimum of 2" apart . Only operate with designated motor. Do not alter cables in any way without first consulting Velmex

⚠ CAUTION

Mismatched Motor Settings Damage Motor & Controller

When Connecting Motor: : Set Controller to the exact motor model/ type before operating.

⚠ CAUTION:

THE VXM MUST BE SET TO THE EXACT MODEL/TYPE MOTOR(S) BEFORE OPERATING.
IMPROPER SETTINGS CAN CAUSE SEVERE DAMAGE TO MOTORS AND CONTROLLER.
Use Velmex COSMOS software to configure VXM before use.



⚠ CAUTION:

MOTOR(S) GET HOT WHEN RUNNING.
Motor(s) must be mounted to a metal surface to dissipate internal heat.

Motors mounted to Velmex actuators/positioners will usually provide sufficient heat dissipation. Motor surface temperature should not exceed 152° F (70° C.) In continuous duty applications when the motor is not mounted to a suitable heat dissipating device, motor surface temperature could exceed 152° F (70° C.)

⚠ WARNING:

TO REDUCE THE RISK OF ELECTRICAL SHOCK, DO NOT ATTEMPT TO REMOVE COVERS ON POWER SUPPLY OR CONTROLLER. THERE ARE NO USER SERVICEABLE PARTS INSIDE. *Any servicing should be done by Velmex qualified service personnel.*

Table of Contents

Precautions.....2
Features.....4
Front VXM.....5
Back.VXM.....5
RS-232 Port.....5
Auxiliary I/O.....6
Motor wiring.....6
Limit Switch wiring.....6
Setup.....7
Jog Mode.....7
Optional Joystick.....7
Communication Methods....8
Linking VXMs for 3 & 4 Motors.....10
Units & Directions.....11
Command Summary (Common Commands).....12
Command Summary (Advanced Commands)....14
Command Reference (Common Commands).....16
Motor Commands.....16
Program Management Commands.....19
Looping/Branching Commands.....20
Pausing & I/O Commands.....22
Operation Commands...23
Status Request Commands.....24
Setup Commands.....25
Examples.....26
Troubleshooting.....30
Specifications.....31
COSMOS Software.....31
Warranty..... 87
Contact Information..... 88

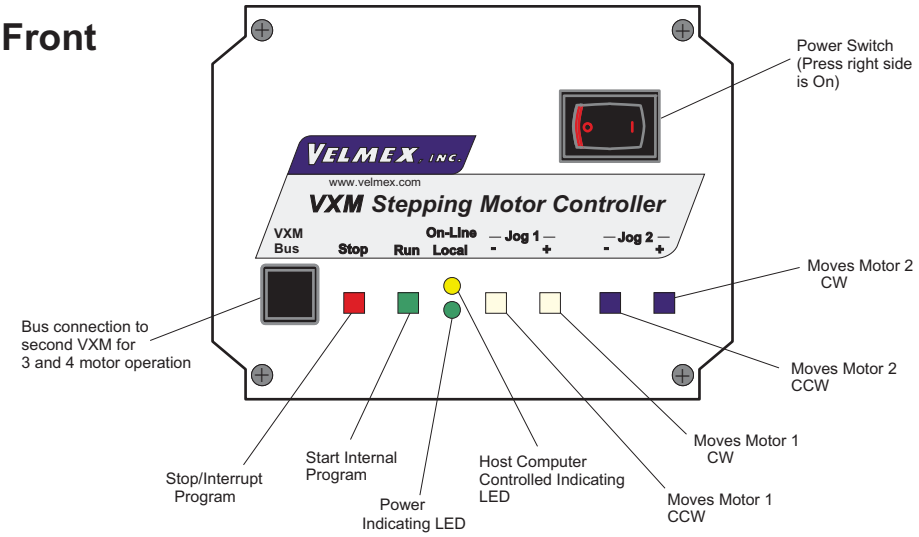
Appendix A
Editing/Debugging Tools.....32
Appendix B
Advanced Input/Output.....34
Appendix C
The Multifunction User Inputs.....38
Appendix D
Producing Trigger Outputs.....39
Appendix E
Getting Motor Position When Moving.....43
Appendix F
More Feedback/ Precision.....45
Appendix G
Complex Profiles & Coordinated Motion...47
Appendix H
Advanced Jog Mode.....49
Appendix I
The Analog Input.....52
Appendix J
The Analog Joystick Option.....54
Appendix K
I/O Electrical Specifications.....58
Appendix L
Motor Torque Curves.....60
Appendix M
Advanced Motor Setup.....64
Appendix N
Limit Switches and Home Switches.....66
Appendix O
Controller Mode.....68
Appendix P
VXM Comparison to NF90/ VP9000.....70
Appendix Q
Outline Dimensions.....75
Appendix R
Model Configurations.....77
Appendix S
Pick-and-Place with JM-x.....79
Appendix T
Stand-alone Methods to Select/Alter Program.....85

Features

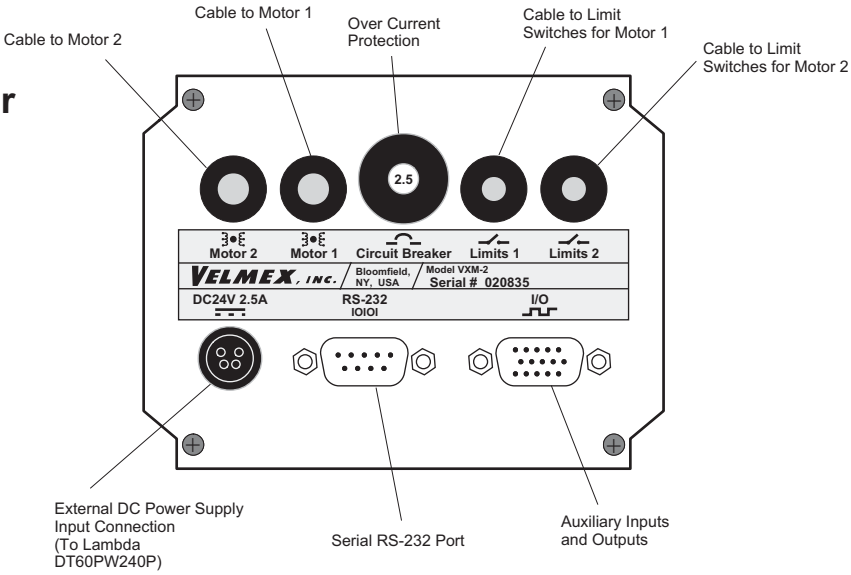
The VXM is a high performance, advanced design stepping motor controller. High reliability, and consistent performance are achieved by these design features:

- Open-loop positioning controller.
- Single chip microcontroller (MCU) digitally controls the motor phase switching and all other interface functions (noise sensitive step and direction translation circuitry are eliminated)
- Pulse width modulated timing is preset by the MCU, eliminating error prone analog feedback circuits
- Regulated power supply with a 100 to 240VAC input range assures consistent motor output torque
- 4X oversize motor drives for long life and overload tolerance
- A single VXM can accept and execute commands for operating 4 motors
- Complete Controller/Indexer/Driver/AC Power Supply with RS-232 interface
- Modulated current control drive has less low speed vibration than typical 400 step/rev controllers
- Nonvolatile memory for user program storage
- Included external desktop type power supply is UL, CE, CSA, and TUV safety agency compliant
- One and two motor versions. Three and four motor capability with two Controls linked by the VXM bus
- Backward compatible with Velmex NF90 and VP9000 Step Motor controllers
- User programmable inputs and outputs
- 10 bit analog input for external sensor, setting speed, or for analog joystick control
- Runs 6 or 8 lead permanent magnet step motors rated from 0.4 to 4.7 amps
- Jog, Run, and Stop input buttons on front panel
- Use interactively with a PC or run standalone
- Optically isolated limit switch inputs
- User resettable circuit breaker protected
- Software settable motor power and motor model selection
- Low voltage 24VDC operation
- Energy saving design, automatically de-energizes motors at a standstill, consuming only 1.4 watts
- FIFO buffer to capture motor positions on external trigger
- Special commands for matrix/array patterns and pick-and-place applications
- Conditional commands to skip or not to skip next command on input state
- Coordinated motion with two VXMs
- Complex motion profiles with “continuous index mode”

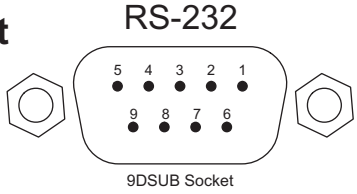
Front



Rear



RS-232 Port



The RS-232 port will connect directly to a COM port of a PC with a straight through 9 pin serial cable (10 foot cable included with the VXM.)

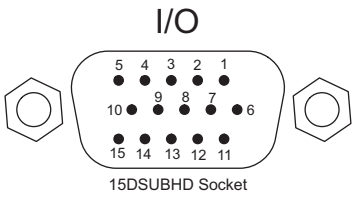
Pin Assignments:

1 N/C
2 T x
3 R x
5 Gnd
4 ┐
6 ┘
7 ┐
8 ┘
9 N/C

N/C = No Connection

Auxiliary I/O Connection

The I/O connections can be used for signaling external equipment or waiting for an external signal. The front panel button inputs are also available on the I/O connector for remote jog, run, and stop.



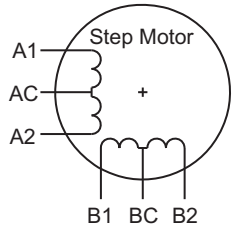
NOTE: All inputs and outputs are TTL levels (0 to +5VDC.) Inputs have resistive pull-ups, and are activated by connecting to 0V. Outputs are normally low, and can sink and source 20 mA max. For more information refer to Appendix K.

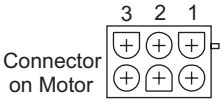
Pin#	Name	Function
1	0V	Logic reference ground for inputs and outputs
2	+5V	+5VDC for Joystick power and other external logic (75mA max. output)
3	Ain	Analog input for Joystick, speed setting, or analog sensor.
4	Run	Run input to start program, same input as Run button (active low)
5	I1	Input 1 (active low)
6	I2	Input 2 (active low)
7	I3	Input 3 (active low)
8	I4	Input 4 and Stop (Same as Stop button on front panel) (active low)
9	0V	Logic reference ground for inputs and outputs
10	J1-	Jog Motor 1 CCW (Same as front panel button) (active low)
11	J1+	Jog Motor 1 CW (Same as front panel button) (active low)
12	J2-	Jog Motor 2 CCW (Same as front panel button) (active low)
13	J2+	Jog Motor 2 CW (Same as front panel button) (active low)
14	O1	Output 1 (normally low)
15	O2	Output 2 (normally low)

Motor Wiring (for Velmex installed step motors)

Pin	Motor	Cable (6 wire)	Slo-Syn	Vexta	Pacific Scientific*
1	BC	W	W	W	W/Y & W/R
2	B2	Gn	Gn	Bu	R
3	AC	Bk	Bk	Y	W/Bk & W/O
4	A2	Or	W/R	Bk	O
5	A1	R	R	Gn	Bk
6	B1	Bu	W/Gn	R	Y

* 8 lead motor with wires combined at AC and BC for 6 lead configuration



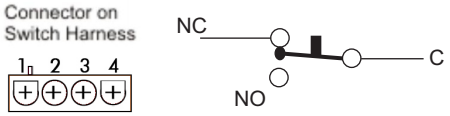


Amp 1-480705-0
(mates with: 1-480704-0 on Cable)

Limit Switch Wiring

	Pin	Switch	Cable
Inner Switch (Motor End)*	1	C	W
	2	NC	R
Outer Switch (End Plate)	3	NC	Gn
	4	C	Bk

* Negative direction on VXM controllers



Amp 1-480703-0 (mates with: 1-480702-0)
Switches are wired on the normally closed (NC) terminals.

CAUTION: The VXM puts 24VDC on the limit switches, do not connect limit inputs to any +5V logic devices

Setup

1. Connect the cables to motors and limit switches (if actuator has limit switches.) Connect the 9 pin serial cable from the VXM's rear panel connector labeled "RS-232" and your computer's serial port (usually labeled "COM" or "|O|O|".) For computers with only USB ports, use a USB to RS-232 adapter.

CAUTION: Motor cables should **never** be bundled together with the Limit Switch, or any I/O cabling. **Never** put any of the VXM's cables with power cables in a common electrical conduit or ducting. **Always** keep Limit Switch and I/O cables at least 2 inches from Motor and Power cables.

CAUTION: Motor cable length or connectors should not be altered without consulting Velmex first. Improper wiring can result in poor performance and damage to the VXM. Altered cables and resultant damage is not covered by the warranty.

IMPORTANT: The VXM can automatically detect limit switch inputs that are wired normally closed to operate (motor stops on open circuit.) Normally closed is the standard used on all Velmex products. However, Velmex rotary tables with the home switch option requires that the limit switch inputs be reconfigured in setup. Refer to the "**setL**" command in Appendix N.

CAUTION: Never connect or disconnect motors with the power on, this can result in severe damage to motor drive electronics.

2. Connect cable from DC power supply to VXM
3. Plug the DC power supply into a AC outlet.
4. Turn on the VXM by pushing the right side of the rocker switch located on the front panel. Both On-Line and Power LEDs will light for 1 second, the On-Line will go out, then the Power LED will flash 6 times.
5. Initially the VXM is set for no motors selected (very low power, motors buzz but will not move.) **Use the COSMOS software included on the CDROM to set VXM for the proper motors.** If your computer is not a Windows based system, refer to the "**setM**" command in the reference section of this manual.

Jog Mode

When the On-Line (yellow) light is not lit, the VXM is in the Local/Jog mode. Using the front panel jog buttons, each motor can be jogged a single step or slewed to 2000 sps (5 revs/sec.) in either direction.
When a Jog button is pressed the motor moves 1 step (1/400 rev.) If the button is held for >0.3 second the motor will accelerate to 2000 sps . Pressing Stop while using the Jog buttons will hold the speed at 39 sps.
Refer to the "**setj**" and "**setJ**" commands in Appendix H for more information about setting jog speeds to different values.

Optional Joysticks

There are two types of external joysticks available for the VXM. One is digital that functions like the front panel jog buttons, and the second is an analog proportional speed type that can operate up to 4 motors with 2 VXMs For more information/configuration refer to Appendix H and J.

Communication Methods

Programming of the VXM is accomplished by sending commands (ASCII characters) to the VXM through the RS-232 interface. The simplest method to send commands is with the Velmex COSMOS program, or with HyperTerminal in Microsoft Windows.

NOTE: All command characters are case sensitive

Another method to send commands is with commercially available languages such as VisualBASIC, C, LabVIEW, etc.

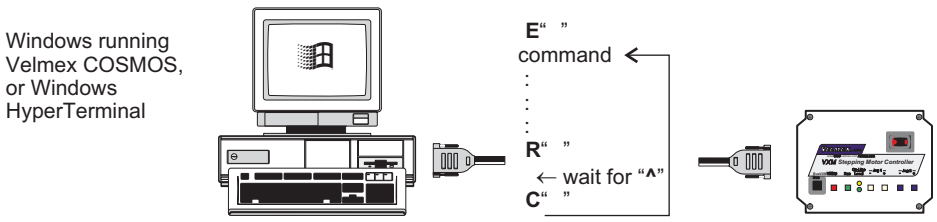
To put the VXM in the On-Line mode/programming mode, the host must send either an "**E**", or "**F**". When the Controller receives an "**E**", or "**F**" the On-line light will light and the Jog inputs will be disabled.

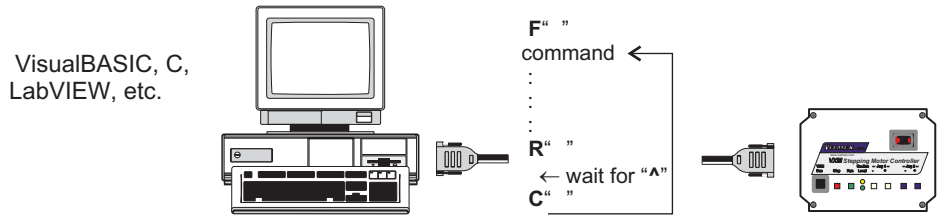
The "**E**" puts the VXM on-line with echo "on" (echoes all characters received back to the host). The "**F**" puts the VXM on-line with echo "off". If you are using HyperTerminal to communicate to the VXM use the "**E**" so typed characters will be displayed. When using a software language to send commands, use the "**F**" so the host's input buffer will not be burdened with echoed characters from the VXM.

RS-232 Interactive Mode

The VXM can be controlled in an interactive mode. Interactive mode is when a host computer sends only the commands necessary to perform a single operation (usually an Index), then the host will wait for the VXM to finish before sending any additional commands. The following procedure would be used for running the VXM in an interactive mode:

1. The host puts the VXM On-Line by sending an "**F**"
2. The host sends a"**N**" to zero position registers if necessary
3. The host sends speed, and acceleration if necessary
4. The host sends an Index ("**I**" command)
5. The host sends a"**R**" to start the Index
6. The host then will wait until it receives a ready prompt ("**^**") from the VXM
   NOTE: The VXM does not send a carriage return or line feed following the "**^**", refer to the "**G**" command in Appendix O for more information
7. The user's routine for outputting, measuring, etc. would be executed by the host
8. A "**C**" would be sent from the host to clear the previous Index command from the VXM's memory
9. The process is repeated from step #3



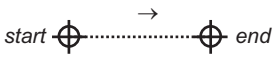


Index Motor 1 Example

```
E
I1M400
R
```

←Enable On-Line mode with echo on
←Incremental Index Motor #1 +400 steps (1rev.)
←Run Index

Graphic Representation:



Comments can be included using a semicolon: **(Never use comments after “R”)**

```
E         ;Enable On-Line mode with echo on
I1M400    ;Incremental Index  Motor #1 +400 steps(1rev.)
R
```

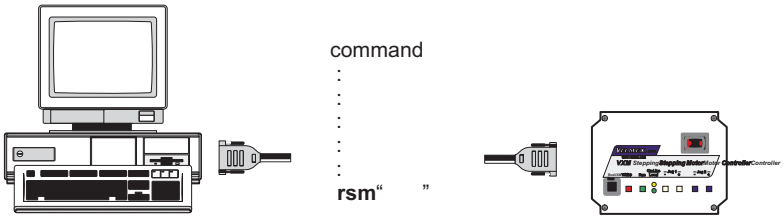
Commands can be on the same line, separated by commas, spaces are optional:

```
E I1M400,R
```

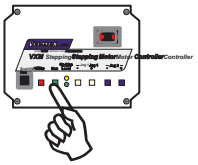
RS-232 Download/ Stand-alone Mode

Entire programs can be transferred to the VXM over the RS-232 interface. When a program(s) has been downloaded to the VXM, it will keep the program(s) in memory until a clear (”**C**”) command is used. To prevent loss of program data when power is turned off, program memory can be permanently saved by using the “**rsm**” command. The ability to retain programs allows the VXM to be used in a stand-alone mode. In a stand-alone mode the operator starts the program by Run button located on the front panel, or by using the Run input on the I/O connector.

Send commands and permanently save them in VXM with “**rsm**” command



VXM holds program(s) that can be activated with the Run input. The default program to run is program 0. Inputs 2 and 3 can configured to binary select and run programs 0 to 3.



Linking VXMs for 3 & 4 Motors (VXM-3,4)

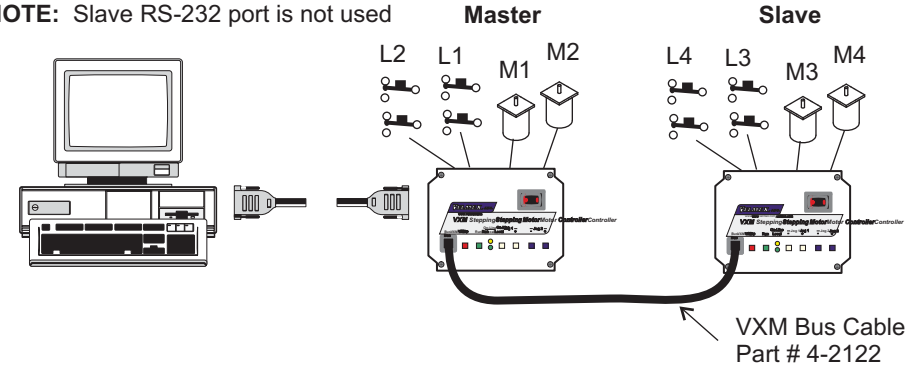
Two controls linked together by the VXM bus make it possible to run 3 or 4 motors with all programs residing in one VXM, and all communication from a host with this one VXM.

The VXM bus is a serial bus conforming to the I^2C specification. This bus is used to transfer data back and forth between two VXM controls that are configured as Master and Slave.

Considerations when linking VXM controls together:

- Use only a Velmex approved cable for the bus connection, telephone handset cables will not work. Telephone cables reverse the 1 and 4 connection, a straight through cable is required.
- Bus cables should be short and not be near other cables or electronic devices.
- By default all VXMs are Slaves.
- To link VXMs, one control should be set to be a Master (see Control Mode command "**setDM**" in the Appendix O.) The Velmex COSMOS program will configure the VXMs for Master/Slave operation.
- A designated Master will attempt to establish communications to a Slave on power-up. If the Master can not find the Slave it will try again when it is required to send a motor 3 or 4 command. **NOTE:** It is normal for the Slave not to flash its power light (green LED) at power-up.
- A bus error will occur ("**ËB**" sent to host and VXM resets) if the Master can not find the Slave, either because there is not one connected, or it is not powered.
- The Master is the VXM that runs motors 1 and 2, communicates with a host, and can be started with the Run input and stopped with the Stop input (Input 4.)
- The Slave runs motors 3 and 4 (The Master assigns motors 3 and 4 to motors 1 and 2 on the Slave) and receives all commands over the VXM bus from the Master.
- The Master disables the Run, Stop, and RS-232 inputs on the Slave.

NOTE: Slave RS-232 port is not used



Index Motor 3 Example

```
C I3M400,R
```

←Clear previous entries, Index Motor 3 +400 steps

Graphic Representation:



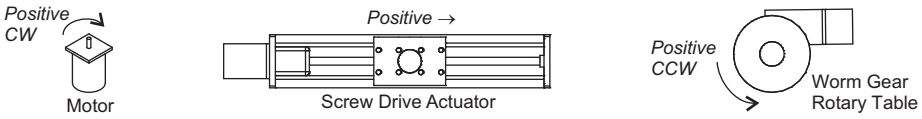
Units & Directions

The VXM uses step units for Index and Speed parameters. One step is 1/400 of a motor revolution. Step units for distance are used with the Index commands ("**I**" command.)

Speed is in units of Steps/ Second (SPS.) Steps/ Second units for speed are used with the Speed commands ("**S**" command.)

Acceleration commands ("**A**" command) are values from 1 to 127 that are relative to steps/sec^2 units. Refer to Application Note #106 for more information about acceleration units and move profiles.

Direction is relative to the device the motor is used on. On screw drive actuators like UniSlides and BiSlides, positive is the direction moving away from the motor.
On worm gear type rotary tables like the Velmex B4800 or B5990, positive is counter clockwise (CCW.) To reorient directions refer to the "**setDM**" command in Appendix O.



Unit Conversion for Velmex Positioners

Lead Screw Models					Speed
UniSlide*		BiSlide**	Advance per turn	Advance per step	@ 1000 SPS (2.5 rev/sec)
			Units	Units	Units
C	P40	E25	0.025 inch	0.0000625 inch	0.0625 inch/sec
B	P20	E50	0.05 inch	0.0001250 inch	0.125 inch/sec
W1	P10	E01	0.1 inch	0.0002500 inch	0.25 inch/sec
W2	P5	E02	0.2 inch	0.0005000 inch	0.5 inch/sec
W4	P2.5	E04	0.4 inch	0.0010000 inch	1 inch/sec
K1	Q1	M01	1 mm	0.0025 mm	2.5 mm/sec
K2	Q2	M02	2 mm	0.0050 mm	5 mm/sec

Rotary Tables				
	Gear Ratio			
B4872	72:1	5 degree	0.0125 degree	12.5 degree/sec
B4836	36:1	10 degree	0.0250 degree	25 degree/sec
B4818	18:1	20 degree	0.0500 degree	50 degree/sec
B5990	90:1	4 degree	0.0100 degree	10 degree/sec

* Typical UniSlide model (where x is from above table): MB4024xJ-S4
** Typical BiSlide model (where x is from above table): MN10-0100-x-21

To convert from "real" units to steps, divide the distance desired to move by the Advance per step. (Distance ÷ Adv per step = Steps)

Example #1: To move 3.000 inches with the BiSlide E04 lead screw (3.000 ÷ 0.001 = 3,000) requires a 3,000 step index.
Example #2: To move 90 degrees with the B5990 rotary table (90 ÷ 0.01 = 9,000) requires a 9,000 step index.
Example #3: To move 4.000 inches with the UniSlide W1 lead screw (4.000 ÷ 0.00025 = 16,000) requires a 16,000 step index.

Other formulas:

1 Motor rev = 400 steps
Linear Speed = Advance per step x steps per second
Rotary Speed = Advance per step x steps per second
Steps per second ÷ 400 = rev/sec

Command Summary (Common Commands)

The following are the most common commands, refer to page 14 for an additional listing of commands for advanced users.

Motor commands:

ImMx Set steps to incremental Index motor CW (positive), m= motor# (1,2,3,4), x=1 to 16,777,215
ImM-x Set steps to incremental Index motor CCW (negative), m= motor# (1,2,3,4), x=1 to 16,777,215
IAmMx Set Absolute Index distance, m=motor# (1,2,3,4), x= ±1 to ±16,777,215 steps
IAmM0 Index motor to Absolute zero position, m=motor# (1,2,3,4)
IAmM-0 Zero motor position for motor# m, m= 1,2,3,4
ImM0 Index motor until positive limit is encountered, m=motor# (1,2,3,4)
ImM-0 Index motor until negative limit is encountered, m=motor# (1,2,3,4)
SmMx Set Speed of motor (70% power), m= motor# (1,2,3,4), x=1 to 6000 steps/sec. (SAmMx is 100% power)
AmMx Acceleration/deceleration, m= motor# (1,2,3,4), x=1 to 127.

Program management commands:

PMx Select Program number x, x= 0 to 4
PM-x Select and clear all commands from Program number x, x= 0 to 4
PM Request the number of the current Program

Looping/branching commands:

L0 Loop continually from the beginning or Loop-to-marker of the current program
LM0 Sets the Loop-to-marker at the current location in the program
LM-0 Resets the Loop-to-marker to the beginning of the current program
Lx Loop from beginning or Loop-to-marker x-1 times (x=2 to 65,535), when the loop reaches its last count the non-loop command directly preceding will be ignored
L-x Loop from beginning or Loop-to-marker x-1 times, alternating direction of motor 1, when the loop reaches its last count the non-loop command directly preceding will be ignored
LAx Loop Always from beginning or Loop-to-marker x-1 times (x=2 to 65,535)
LA-x Loop Always from beginning or Loop-to-marker x-1 times, alternating direction of motor 1
LM-2 Loop once from beginning or Loop-to-marker reversing index direction of motor 2
LM-3 Loop once from beginning or Loop-to-marker reversing index direction of motor 1 and motor 2
Jx Jump to the beginning of program number x, x= 0 to 4
JMx Jump to the beginning of program number x and come back for More after program x ends, x= 0 to 4

Pausing and input/output commands:

Px Pause x tenths of a second, (x=0 to 65,535) tenths of a millisecond when x is negative
PAx Pause x tenths of a second (x=0 to 65,535, 10 µsec pause when x=0) Altering output 1 high for duration of the pause, tenths of a millisecond when x is negative
U0 Wait for a "low" on user input 1
U1 Wait for a low on user input 1, holding user output 1 high while waiting
U4 User output 1 "low" (reset state)
U5 User output 1 high

Operation commands:

Q Quit On-Line mode (return to Local mode)
R Run currently selected program
N Null (zero) motors 1,2,3,4 absolute position registers
K Kill operation/program in progress and reset user outputs
C Clear all commands from currently selected program
D Decelerate to a stop (interrupts current index/ program in progress)
E Enable On-Line mode with echo "on"
F Enable On-Line mode with echo "off"
rsm Run save memory (saves setup/ program values to nonvolatile memory)

***Status request commands*:**

V Verify Controller's status, VXM sends "B" to host if busy, "R" if ready, "J" if in the Jog/slew mode, or "b" if Jog/slewing
X Send current position of motor 1 to host (Motor can be in motion)
Y Send current position of motor 2 to host (Motor can be in motion)
Z Send current position of motor 3 to host (Motor must be stationary)
T Send current position of motor 4 to host (Motor must be stationary)
M Request Memory available for currently selected program
lst List current program to host (ASCII text)
getMmM Read motor type/size selected for axis m

Setup commands:

setMmMx Set axis m for motor type/size x.
setBx Set RS-232 Baud rate (9=9600, 19=19200, 38=38400)

Command Summary (Advanced Commands)

The following commands are for advanced VXM users. For more information on these commands refer to Appendices in this User's Manual.

Motor commands:

SmM-x Read and assign analog input value to motor m speed (70% power), x=speed range (**SAmM-x** is 100% power)

Program management commands:

PMAx Program Associate program x in Master to program x in Slave (Linked VXMs start the same time)

PMA Request the current program associate number

Branching commands:

JM-x Similar to JMx except automatically moves back from absolute indexes after program x ends: For pick-and-place within matrix looping patterns **(available only on VXM firmware versions 1.20 & up)**

Input/output commands:

U2 Enable Jog mode while waiting for an input

U3 Disable Jog mode while waiting for an input

U6 Send "W" to host and wait for a "G" to continue

U7 Start of Continuous Index with pulse on output 2

U77 Start of Continuous Index with no output

U8 Start of Continuous Index sending "@" to the host

U9 End of Continuous Index with autodecel to stop

U91 End of Continuous Index with auto-generate a deceleration Index as next command

U92 End of Continuous Index using next Index for deceleration to stop

U99 End of Continuous Index with instantaneous stop

U11 Skip next command if input 1 is high **(available only on ver. 1.22 & up)**

U12 Skip next command if input 2 is high **(available only on ver. 1.22 & up)**

U21 Skip next command if input 1 is low **(available only on ver. 1.30 & up)**

U22 Skip next command if input 2 is low **(available only on ver. 1.30 & up)**

U13 Wait for a front panel button to jump to a program or continue: "Motor 1 Jog -" button to jump to program #1, "Motor 1 Jog +" button to jump to program #2, "Run" button to proceed in current program.

U14 User output 2 low (reset state)

U15 User output 2 high

U16 Optional User output 3 low (reset state)

U17 Optional User output 3 high

U18 Optional User output 4 low (reset state)

U19 Optional User output 4 high

U23 Wait for a front panel button to jump to a program and come back, or continue: "Motor 1 Jog -" button to jump and return to program #1, "Motor 1 Jog +" button to jump and return to program #2, "Run" button to proceed in current program

U30 Wait for a low to high transition on user input 1

U31 Wait for a low to high transition on user input 1, holding user output 1 high while waiting

U32 Wait for "Motor 1 Jog -" button to be pressed on front panel with debouncing

U33 Wait for "Motor 1 Jog +" button to be pressed on front panel with debouncing

U50 Wait for a low and high on user input 1 with debouncing for a mechanical push-button switch

U51 Wait for a low and high on user input 1 with debouncing for a mechanical push-button switch, holding user output 1 high while waiting

U90 Wait for a low to high on the Run button or connection I/O,4 with debouncing for a mechanical push-button switch

Operation commands:

G Enable On-Line mode with echo off Grouping a <cr> with "^", ":", "W", "O" responses; Also Go after waiting or holding
H Put Controller on Hold (stop after each command and wait for go)
! Record motor positions for later recall with “x”,”y” commands
res Software reset controller
del Delete last command

***Status request commands*:**

x Send last 4 positions of motor 1 to host that were captured by the “!” command or Input 4 trigger
y Send last 4 positions of motor 2 to host that were captured by the “!” command or Input 4 trigger
Request the number of the currently selected motor
✱ Request the position when the last motor started decelerating (shows position when "D" command or Stop/User input 4 used)
? Read state of limit switch inputs for motor 1 and 2 (8 b i t binary value)
~ Read state of User Inputs, Motor 1 and 2 Jog Inputs (8 bit binary value)
$ Read state of User Outputs (8 bit binary value) **(only on ver. 1.22 & up)**
@ Read user analog input value
B Read Backlash compensation setting
O Read Indicate limit switch setting
getDx Read mode/version
getDA Read Joystick Deadband setting
getjmM Read first range Jog Speed for motor m. **getjAmM** for Joystick range setting
getJmM Read second range Jog Speed for motor m. **getJAmM** for Joystick range setting
getLmM Read mode of limits for motor m
getPmM Read “Pulse Every x # Steps” value for axis m Read Pulse width used
getPA by **setPmMx** and **U7** **(only on ver. 1.24 & up)** Read operating mode
getI of user inputs

Commands for two controls connected by VXM bus:

(i3,i1..) Combine Index commands to run simultaneously on two VXM controllers connected by VXM bus
[i1,i2...] Send data to Slave through Master

***Jog mode commands*:**

D Read motor position (Digitize)

Special function and setup commands:

Bx Backlash compensation, 0= off (default), 1= 20 steps, **Ver. 1.25 up: x=0 to 255**
Ox Indicate limit switch Over-travel to host, off when x=0, VXM sends "O" when x=1 and hit limit, x=3 program stops too
setDMx Set VXM/VP9000 or NF90 emulation modes, and other operating parameters
setDAx Set Joystick Deadband value
setjmM Set first range Jog Speed for motor m. **setjAmM** for Joystick range setting
setJmM Set second range Jog Speed for motor m. **setJAmM** for Joystick range setting
setLmMx Set limit switch mode for axis m
setPmMx Set “Pulse Every x # Steps” on output 2 for axis m Set Pulse width used
setPAx by **setPmMx** and **U7**, x=1 to 255 (10 μsec increments)**(available only on versions 1.24 & up)**
setIx Set operating mode of inputs

Command Reference (Common Commands)

This section gives detailed explanations of the most common VXM commands. For the advanced user, refer to the Appendices for more information.
Most commands with variables (except **set** commands) use the VXM's program memory space. The required memory needed per command is specified in this section. The VXM has 256 bytes of program memory for each program. There are 5 (0,1,2,3,4) programs. A program can be cleared by a "**C**" and selected by the "**PMx**" command. The default program when the VXM is powered up is #0. Using different programs is only relevant to users who will be operating the VXM in a stand-alone mode (P8.) Using the VXM in a RS-232 interactive mode (P9.) would only require that the default program be cleared after the **R** command.

The Difference Between Incremental and Absolute Indexes:
An incremental Index is, a move relative to the present position, a distance and direction specified by the Index from the present position.
Incremental: **I1M1200**



An absolute Index is, a move relative to absolute zero position, a distance and direction from the present position calculated by the VXM based on absolute zero position. Absolute zero is established when the VXM is powered-up, by use of the "**N**", or the "**IA***m***M-0**" command.
Absolute: **IA1M2200**



Sending Commands to the VXM:
The standard RS-232 communication settings on the VXM are 9600 baud, 8 data, no parity, and 1 stop bit.
When sending commands that require a value, the commands must end with a carriage return (Enter key or Return on most keyboards), comma, or a period.

Motor Commands

I*m***M***x* Set steps to incremental **I**ndex (move) motor CW (positive, Slider/Carriage will move away from motor end , Rotary Table will rotate CCW), *m*=motor# (1,2,3,4), *x*=1 to 16,777,215.
Memory usage = **4 bytes**.
Examples: **NOTE:** The "<cr>" is a carriage return character (<Enter> key on most keyboards). Command characters are in **LARGE BOLD**.
This example sets motor 1 to index 1200 steps CW: **I1M1200**<cr>
This example sets motor 2 to index 9200 steps CW: **I2M9200**<cr>
This example sets motor 3 to index 10200 steps CW: **I3M10200**<cr>

I*m***M-***x* Set steps to incremental **I**ndex (move) motor CCW (negative, UniSlide Slider will move toward motor end , UniSlide Rotary Table will rotate CW), *m*= motor# (1,2,3,4), *x*=1 to 16,777,215.
Memory usage = **4 bytes**.
Examples:
This example sets motor 1 to index 120 steps CCW: **1M-120**<cr>
This example sets motor 2 to index 20 steps CCW: **I2M-20**<cr>
This example sets motor 4 to index 1 step CCW: **I4M-1**<cr>

IA*m*M*x* Set an Index to an **A**bsolute position, the distance and direction for the move from the present position is calculated by the VXM based on absolute zero position, *m*= motor# (1,2,3,4), *x*= ±1 to ±16,777,215. NOTE: Since the absolute position registers have a range of -8,388,608 to 8,388,607 steps, *x* should not be set to any number less than -8,388,608 or greater than 8,388,607.
Memory usage = **4 bytes**.
Examples:
This example sets motor 1 to index to absolute position 1200 :
IA1M1200<cr>
This example sets motor 4 to index to absolute position -90200 :
IA4M-90200<cr>

IA*m*M0 Moves **I**ndex to **A**bsolute zero position, *m*=motor# (1,2,3,4). When this command is used the VXM calculates the distance and direction to get back to absolute zero position. The "absolute zero" position was established when the "**N**" (Null Absolute Position Registers), "**IA*m*M-0**" command was used, or when the VXM was powered up. (Moves to last known zero, unless control was power off. The position when powered-off becmes zero.)
Memory usage = **4 bytes**.
Examples:
This example sets motor 1 to index to absolute zero position:
IA1M0<cr>
This example sets motor 2 to index to absolute zero position:
IA2M0<cr>
This example sets motor 3 to index to absolute zero position:
IA3M0<cr>

IA*m*M-0 Zero motor position for motor# *m*, *m*=motor# (1,2,3,4). This command clears the position register for the motor selected, making this position absolute zero. The display will show all zeros for the motor selected.
Memory usage = **4 bytes**.
Examples:
This example makes the present position for motor 1 absolute zero:
IA1M-0<cr>
This example makes the present position for motor 2 absolute zero:
IA2M-0<cr>

I*m*M0 Move positive until the positive limit switch is encountered (Home to Positive Limit), *m*=motor# (1,2,3,4). If the limit switch input was disabled in setup, the limit switch input will be re-enabled for the duration of this command. The Index will end if the limit switch is not encountered after 16 million steps.
Memory usage = **4 bytes**.
Example: This example sets motor 1 to seek the positive limit switch:
I1M0<cr>

I*m*M-0 Move negative until the negative limit switch is encountered (Home to Negative Limit), *m*=motor# (1,2,3,4). If the limit switch input was disabled in setup, the limit switch input will be re-enabled for the duration of this command. The Index will end if the limit switch is not encountered after 16 million steps.
Memory usage = **4 bytes**.
Example: This example sets motor 1 to seek the negative limit switch:
I1M-0<cr>

S*m***M***x* Set **S**peed of motor (70% power applied to motor), *m*= motor# (1,2,3,4), *x*=1 to 6000 steps/sec. in 1 step/sec. intervals. If this command is never used, the default speed will be 2000 steps/sec.

NOTE: motor torque decreases as speed increases, and some motors have limited torque above 2000 steps/sec. If the motor torque is below the needed torque to move the load, the motor will stall (lose synchronism and proper position.)
Memory usage = **3 bytes**.
Example:
This example sets the speed of motor 1 to 500 steps/sec at 70% power:
S1M500<cr>

When the "**S**" speed command is used for setting speed, motor running torque will be 70% of the maximum output. For most applications 70% motor torque will be adequate. For moving heavy loads the "**SA**" speed command (100% power) may be needed.

NOTE: Motor power will always be zero when the motor is stationary (motors are normally un-energized at a standstill.

Advantages of the "S" speed command (70% motor power)
1. Saves energy.
2. Motors run smoother and quieter
3. Reduces mid-speed motor resonance.
4. Reduces Motor and Controller heating.

SA*m***M***x* Set **S**peed of motor (100% power applied to motor), *m*= motor# (1,2,3,4), *x*=1 t to 6000 steps/sec. in 1 step/sec. intervals.
Example:
This example sets the speed of motor 2 to 3000 steps/sec at 100% power:
SA2M3000<cr>

How to Determine Maximum Speed: With acceleration set to 2 (default) increase speed until motor stalls, use 75% of this speed as the maximum speed.

CAUTION: Motor and Controller surface temperatures become hot when running motors continuously. Only use 100% ("**SA**" command) motor power if maximum torque is required. For maximum efficiency when lifting heavy loads vertically, use the "**SA**" command to set speed for traversing upwards, and use the "**S**" speed command for the speed down.

A*m***M***x* **A**cceleration/deceleration, *m*= motor# (1,2,3,4), *x*=1 to 127. The default value is 2. The higher the number used, the faster the motor will reach the set speed, and the faster it will slow down to a stop. **NOTE**: motors may stall if this value is set to high.
Memory usage = **2 bytes**
Example:
This example sets the acceleration/deceleration of motor 1 to 3:
A1M3<cr>

How to Determine Maximum Acceleration: With speed set to maximum as determined above, increase acceleration until the motor stalls, use 1/2 of stall acceleration as the maximum See Application Note #106 for more information about acceleration.

Programming Shortcut:
The motor designation in Acceleration, Speed, and Index commands is optional if the desired motor has already been set as the current motor. The current motor is motor 1 when the Controller is first turned on. The last motor jog/slewed will be the current motor number. The current motor will be the number used in the last Acceleration, Speed, or Index command. Users that have only a one motor VXM (Model VXM-1) do not have to use the motor designation in a command. For example, these commands would always be motor 1 commands of a one motor VXM:

A2,S4000,I400,

For running a particular motor of a multi-motor VXM, only the first Command needs the motor number. For example, all of these commands would be for motor 2:

I2M200,I-200,S2000,IA0,

Program Management Commands

PM*x* Select **P**rogram number *x* as the current program, *x*= 0 to 4. Each program can hold 256 bytes of commands. The default program number is 0.
Program 3 can be interactive with user input 3, and Program 4 can be interactive with user input 4. See the'**setI** " command in the Reference Manual on the CDROM for more information.
Memory usage = **0 bytes**. This command is immediate (not stored)

Example:
This example selects program #1 for the current program:

PM1<cr>

PM-*x* Select and clear all commands from Program number x, x= 0 to 4. This command will select program *x* as the current program and delete all commands from this program.
Memory usage = **0 bytes**. This command is immediate (not stored)

Example:
This example selects program #0 and erases all commands within it:

PM-0<cr>

PM Request the number of the current program. the VXM will send a value between 0 and 4 indicating the program number selected.

Example:

PM<cr>

If the current program is 3, the VXM will send the following to the host:

3<cr>

Looping/Branching Commands

L0 Loop continually from the beginning or Loop-to-marker of the current program. The loop will occur to the last Loop-to-marker of the current program if it was set previously. This command can be used once in a program as the last command, it functions the same as a "continuous run input".
Memory usage = **1 byte**.

LM0 Sets the **L**oop-to-marker at this point in the current program. All looping commands in the current program that follow will branch to here. Any loop commands in the program prior to this marker will branch to the beginning of the program or a previous marker.
NOTE: Multiple markers can be used in a program, the number is only limited by the program memory available (256 bytes per program).
Memory usage = **1 byte**

LM-0 Resets the **L**oop-to-marker to the beginning of the current program.
NOTE: Multiple resets can be used in a program, the number is only limited by the program memory available (256 bytes per program).
Memory usage = **1 byte**

L*x* **L**oop from beginning or Loop-to-marker of the current program x-1 times (x=2 to 65,535). A maximum of 10 nested loop commands can be used per run.
NOTE: When the Loop reaches its last count, the non-loop command directly preceding the Loop will be ignored.
Memory usage = **3 bytes**.
Example:
This example sets a loop to repeat, any previous commands 4000-1 times, while repeating the directly preceding non-loop command 4000-2 times:

L4000<cr>

L-*x* **L**oop from beginning or Loop-to-marker of the current program *x*-1 times alternating direction of motor 1 indexes (*x*=2 to 65,535). A maximum of 10 nested loop commands can be used per run.
NOTE: When the Loop reaches its last count, the non-loop command directly preceding the Loop will be ignored.
Memory usage = **3 bytes**.
Example:
This example sets a loop to repeat, any previous commands 100-1 times alternating motor 1 direction every repeat, while repeating the directly preceding non-loop command 100-2 times: **L-100**<cr>

LA*x* **L**oop Always from beginning or Loop-to-marker of the current program *x*-1 times (*x*=2 to 65,535). Maximum 10 nested loop commands per run allowed.
Memory usage = **3 bytes**.
Examples:
This example sets a loop to repeat all previous commands 4000-1 times:

LA4000<cr>

Consecutively nested loops are equal to the product of their loop values. For example, the following loops together are equal to 10,000,000-1 (50,000 x 200):
`

LA50000,LA200<cr>

LA-*x* **L**oop **A** lways from beginning or Loop-to-marker of the current program *x*-1 times alternating direction of motor 1 indexes (*x*=2 to 65,535). A maximum of 10 nested loop commands can be used per run.
Memory usage = **3 bytes**.
Examples:
This example sets a loop to repeat 100-1 times all previous commands alternating motor 1 direction every repeat: **LA-100**<cr>

Consecutively nested loops are equal to the product of their loop values. For example, the following loops together are equal to 2,500,000,000-1 (50,000 x 50,000): **LA-50000,LA50000**<cr>

LM-2 **L**oop once from the beginning or Loop-to-marker of the current program, reversing index direction of motor 2. See "Example Programs" section for use of this command.
Memory usage = **1 byte**

LM-3 **L**oop once from beginning or Loop-to-marker of the current program, reversing index direction of motor 1 and motor 2. See "Example Programs" section for use of this command.
Memory usage = **1 byte**.

J*x* **J**ump to the beginning of program number ,*x* *x* = 0 to 4. Program number *x* will temporarily be the current program, all commands will be executed starting from the first one that was previously entered into program *x*. If there is not any commands in program *x*, or after executing the last command, the program will end, and the VXM will send the ready prompt to the host ("^"). The current program number will still be the program that was originally selected with a "**PMx**" or "**PM-x**" command. Linking multiple programs (maximum of 5) together is possible by using a jump command, as the last command, to make a jump to a different program. All looping commands in program *x* will be local to this program only.
Memory usage = **2 bytes**
Example:
This example will jump to program #1 : **J1**<cr>

JM*x* Jump to the beginning of program number x and come back for **M**ore after program x ends, *x*= 0 to 4. Program number *x* will temporarily be the current program, all commands will be executed starting from the first one that was previously entered into program *x*. If there is not any commands in program *x*, or after executing the last command, control will be transferred back to the program that initiated the **J**ump, then the next command in the initiating program will be executed. The maximum **JM*x*** commands active at a time is 4. This command can be used to make programming more modular, having a main program that jumps to other programs (modules) and returns, can make long programs easier to maintain and edit. All looping commands in program *x* will be local to this program.
CAUTION: Motor reverse-direction-flags are set by "**L-x**", "**LM-2**", and "**LM-3**" looping commands. If a **JM*x*** command is used "inside" one of these loops, motor 1 and motor 2 direction may be reversed in program *x*.
Memory usage **= 2 bytes.**
Example:
This example will jump to program #3 and return: **JM3**<cr>

Pausing and Input/Output Commands

P*x* **P**ause tenths of a second (xx=0 to 65,535).
Memory usage = 3 bytes.

Examples:
This example pauses for 1 second: **P10**<cr>

This example pauses for 15 seconds: **P150**<cr>

This example pauses for 1 hour: **P36000**<cr>

P-*x* Pause *x* tenths of a millisecond (x=1 to 65,535).
Memory usage = **3 bytes.**

Example:
This example pauses for 50 milliseconds (0.050 seconds): **P-500**<cr>

PA*x* **P**ause tenths of a second **A**ltering the state of output 1 (xx=0 to 65,535, 10 µsec pause when x=0). The user output 1 (I/O,14) will go to +5V for the duration of the pause.
Memory usage = **3 bytes.**

Example:
This example pauses for 15 seconds holding output 1 high: **PA150**<cr>

PA-*x* **P**ause tenths of a millisecond ltering the state of output 1 (=1 to 65,535). The user output 1 (I/O,14) will go to +5V for the duration of the pause. Memory usage = **3 bytes**.

Example:
This example pauses for 15 milliseconds (0.015 seconds) holding output 1 high:
PA-150<cr>

U0 Wait for a "**low**" on the user input 1. A "**low**" is a voltage less than 0.8 VDC (not to be less than 0V) applied to I/O,5. A simple push-button or toggle switch can be used between Gnd (I/O,1) and input 1 (I/O,5) to satisfy this input. The input level must be high for at least 1 ms to be a valid input. This command is best used when interfacing to other solid-state logic devices, refer to the "**U50**" command (Appendix B) for push-button switch input.
Memory usage = **2 bytes.**

U1 Wait for a "low" on the user input1holding user output 1 "high" (+5V) while waiting. A "low" is a voltage less than 0.8 VDC (not to be less than 0V) applied to I/O,5. User output 1 (I/O,14) will go to +5V for the duration of the wait. A simple push-button or toggle switch can be used between Gnd (I/O,1) and input 1 (I/O,5) to satisfy this input. The input level must be high for at least 1 ms to be a valid input. This command is best used when interfacing to other solid-state logic devices, refer to the "**U50**" command (Appendix B) for push-button switch input.
Memory usage = **2 bytes**.

U4 User output 1 "low". The user output 1 (I/O,14) will go to 0V. This is the state of the user output 1 on power-up. This command is used in conjunction with the "**U5**" command.
Memory usage = **2 bytes**.

U5 User output 1 high. The user output 1 (I/O,14) will go to +5V. This command is used in conjunction with the "**U4**" command. Memory usage = **2 bytes**.

Operation Commands

These commands are immediate (not stored), they do not use the VXM's program memory, and do not need an ending carriage return or comma.

Q **Q**uit On-Line mode (return to Local Jog/slew mode.) The "**Q**" command is used to get back to the power-up state, where the VXM is in the Local Jog/slew mode (On-Line light is off.)

R **R**un currently select program. The "**R**" command will start execution of commands stored (of current program) in the VXM's memory. At the end of the "run" the single character "^" (no carriage return or line feed follows the "^" unless the VXM was put On-Line with the "**G**" command) will be transmitted to the host. Additional "**R**" commands received by the VXM will repeat the same program. Refer to the "**C**" and "**PMx**" command to clear a program from memory. The Run input (I/O,4) and the front panel Run button function the same as this "**R**" command.

N **N**ull (zero) motors 1,2,3,4 Absolute Position Registers. This command can be used in the Local Jog/slew or the On-Line mode. The "**N**" command zeros the position registers that have been counting steps from indexing and/or jog/slewing the motor(s).

K **K**ill operation in progress. This command will immediately interrupt any running program. The user outputs will be reset, all looping and hold flags will be reset, and if a motor is moving it will be stopped immediately. If the motor speed is above 1000 steps/sec. when the interrupt occurs, the motor may loose position due to mechanical overshoot (see the "**D**" command for a less abrupt method to interrupt indexes). The VXM will transmit the "^" to the host after receiving the "K" command.

C **C**lear all commands from the currently selected program. All setup values, motor position values, and the state of user outputs will not be altered.

D **D**ecelerate to a stop (interrupts current index in progress, default function of Stop button too). When the VXM receives the single character "D" while it is indexing a motor, that motor will be decelerated to a stop at the set deceleration. The motor position prior to decelerating is saved, refer to the "*" command to request this position. The VXM will then proceed to the next command in the program. The "**D**" command has a different function when in the Local Jog/slew mode, refer to the section on "Digitizing With a Host" in Appendix H for more information.

E **E**nable On-Line mode with echo on. The single character "**E**" is used to put the VXM in the On-Line mode after power-up. All characters the VXM receives will be echoed back to the host. Refer to the section "Communication Methods " (p.8) for more information.

F Enable On-Line mode with echo o**FF**. The single character "**F**" is used to put the VXM in the On-Line mode after power-up. No characters will be echoed back to the host. The VXM will still respond to motor position and status requests. Refer to the section "Communication Methods " (p.8) for more information.

rsm Run save memory, saves setup/ program values to nonvolatile memory. The VXM will send the single character "^" after completion of the save. Use this command to:

1. To permanently save setup/special function (**setx** commands) values that have been modified.
2. To save programs/commands for stand-alone use.

CAUTION: When using the "**rsm**" command power should not be interrupted otherwise data loss may occur. The host should always wait for the "^" before sending another command.
The nonvolatile memory has a limited write life (100,000 erase/write cycles), therefore, do not use "**rsm**" more than necessary. It would typically be used to save motor selection that has been updated, or to keep a program in the VXM for use without a host computer (stand-alone use.)

Status Request Commands

V **V**erify Controller's status, when On-Line the VXM sends a "B" to the host if it is busy, or an "R" if it is ready. The "**V**" command is used to poll the VXM to see if it is busy running a program, or ready to receive more commands.

NOTE: Use of this command is optional, since the VXM automatically transmits a "^" character to the host when a program has finished.
If the VXM is running a program when it receives a "**V**" the VXM will respond by transmitting the single character "B". If the VXM is idle waiting for a command the VXM will respond by transmitting the single character "R".
When in the Local Jog/slew mode the VXM will respond by sending a "J" if a motor is not moving and a "b" if a motor is moving.

X Send position of motor 1 to the host. When the VXM receives the single character "**X**" it will transmit the value from it's motor 1 Absolute Position Register. Below is what the host would receive if motor 1 is at negative 1200. This command can be used when the motor is indexing. See the "**N**" command for information on zeroing the Absolute Position Registers.

-0001200<cr>

Y Send position of motor 1 to the host. When the VXM receives the single character "**Y**" it will transmit the value from it's motor 2 Absolute Position Register. Below is what the host would receive if motor 2 is at positive 9201. This command can be used when the motor is indexing. See the "**N**" command for information on zeroing the Absolute Position Registers.

+0009201<cr>

Z Send position of motor 3 to the host. When the VXM (Master) receives the single character "**Z**" it will transmit (from motor 1 in Slave) the value of the motor 3 Absolute Position Register. Below is what the host would receive if motor 3 is at negative 20. This command cannot be used when the motor is indexing. See the "**N**" command for information on zeroing the Absolute Position Registers.

-0000020<cr>

T Send position of motor 4 to the host. When the VXM (Master) receives the single character "**T**" it will transmit (from motor 2 in Slave) the value of the motor 4 Absolute Position Register. Below is what the host would receive if motor 4 is at negative 200000. This command cannot be used when the motor is indexing. See the "**N**" command for information on zeroing the Absolute Position Registers. **-0200000**<cr>

M Request **M**emory available for the currently selected program. The VXM will send the number of bytes that are unused of the current program. The value will be 0 to 256 followed by <cr> (carriage return.)

lst List commands in current program to host (ASCII format.) Returns program number and memory remaining prior to listing commands.
Example listing: **PM0 M252**
I1M400

getM*m***M** Get motor type/size selected for axis ,*m* *m* =axis# (1,2,3,4.)
Value returned will be a number between 0 and 6. Refer to the table below for value to motor model cross reference.

Setup Commands

These commands do not use program memory, they have their own reserved space.

setM*m***M***x* **S**et motor type/size selected for axis *m*, *m*=axis# (1,2,3,4.)
Value for *x* should be a number between 0 and 6. Refer to the table below for the proper value to use.

CAUTION: THE VXM MUST BE SET TO THE EXACT MODEL/TYPE MOTOR(S) BEFORE OPERATING. IMPROPER SETTINGS CAN CAUSE SEVERE DAMAGE TO MOTORS AND CONTROLLER.

x	Motor Model (Amps)
0	Default (0.4A to 0.7A)
1	Vexta PK245 (1.2A)
2	Slo-Syn M061 (3.8A)
3	Slo-Syn M062 (4.7A) Vexta PK264 (3A)
4	Slo-Syn M063 (4.6A) Vexta PK266 (3A)
5	Slo-Syn M091 (4.7A) Vexta PK268 (3A)
6	Slo-Syn M092 (4.6A) Vexta PK296 (4.5A)

setB*x* Set RS-232 **B**aud rate to value *x*. (default=9600) *x*=9 for 9600 baud, *x*=19 for 19,200 baud, *x*=38 for 38,400 baud. Data bits are always 8, with no parity, and 1 stop bit.

Examples

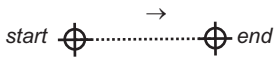
The following examples can be keyed into the VXM with a terminal program like HyperTerminal in Microsoft Windows, or the Velmex COSMOS software. Another method to send commands is with commercially available languages such as VisualBASIC, C, LabVIEW, etc.

The "**<cr>**" is a carriage return character (<Enter> key on most keyboards). Command characters are in a rectangle like this:

```
C I3M400<cr>
```

A diagram of the resultant motion of a screw driven linear actuator is included showing start/end points, direction and commands. A letter over a point on the diagram represents the function occurring at that point. A "P" is a pause command, "U" user I/O command, and a "Z" is a motor 3 Index. Numbers shown in the diagrams represent Loop count values.

Example diagram:



Example #1	Motors run	RAM used	Function
On-Line	-	-	Enable On-Line mode with echo on

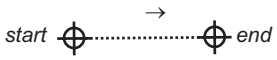
```
E
```

Example #2	Motors run	RAM used	**Function** Incremental
Index	1	4	Index Motor #1 400 steps (1 rev.) CW

```
I1M400,R
```

or

```
I1M400<cr>
R
```



Example #3	Motors run	RAM used	Function
Clear	-	-	Clear all commands from current program

```
C
```

Example #4	Motors run	RAM used	**Function** Incremental
Index	1	4	Index Motor #2 600 steps CCW

```
I2M-600,R
```

or

```
I2M-600<cr> R
```

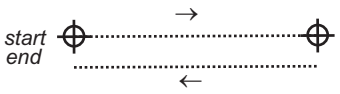


Example #5	Motors run	RAM used	Function
Auto-Reverse	1	8	Index Motor #1 both directions

```
I1M800,I-800,R
```

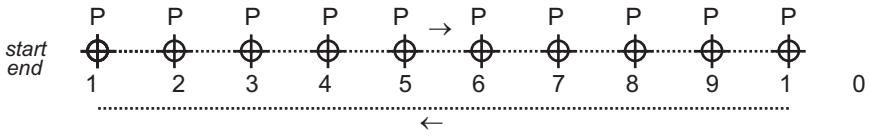
or

```
I1M800<cr>
I1M-800<cr>
R
```



Example #6	Motors run	RAM used	Function
Repeating Index	1	14	Repeating Index pausing 1 second between Indexes, return to start

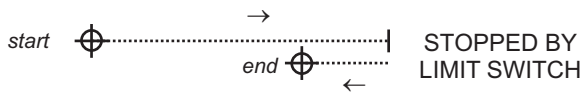
```
P10,I1M400,L10,I1M-3600,R
```



Example #7	Motors run	RAM used	Function
Home to Limit	1	15	Home Motor 1 to Positive Limit Switch and move 200 steps from Limit Switch and zero position

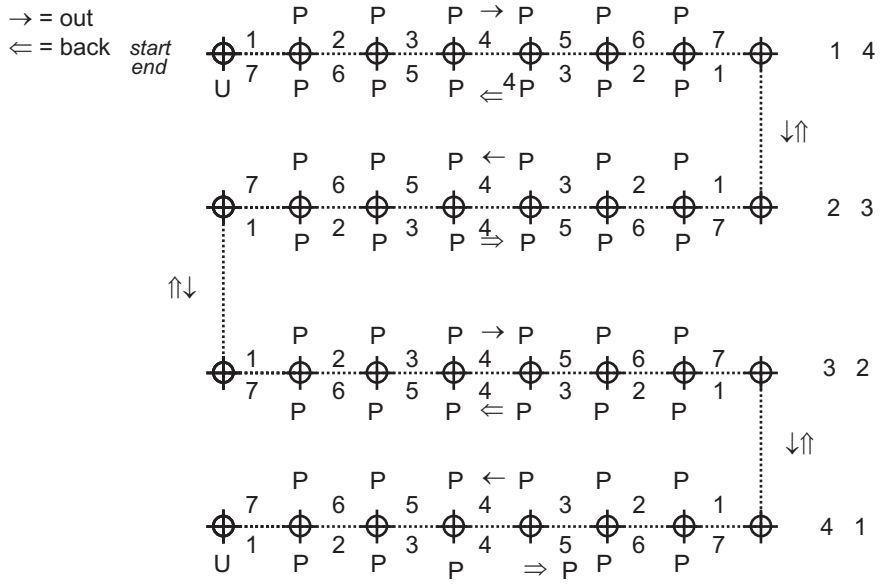
CAUTION: Positioning may be unreliable and limit switches may be damaged if speeds above 1000 steps/second are used for homing.

```
S1M600,I1M0,I1M-200,IA1M-0,R
```



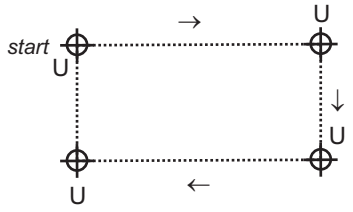
Example #8	Motors run	RAM used	Function
Raster Scan	2	23	Raster scan with 1 sec. pauses and waiting for “G” at the end; then run backwards through raster scan

```
I1M200,P10,L7,I2M400,L-4,U6,LM-2,R
```



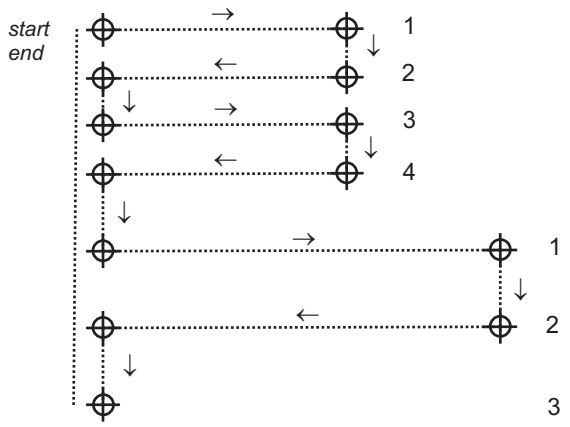
Example #9	Motors run	RAM used	Function
Rectangle	2	14	Rectangle, with Output 1 and Wait for Input 1 at each corner

```
I1M2000,U1,I2M1000,U1,LM-3,L0,R
```



Example #10	Motors run	RAM used	Function
Raster Scans	2	27	Two Different Raster Scans using Loop-to-marker

```
I1M2000,I2M300,L-4,LM0,I2M600,I1M3000,L-3,IA2M0<cr>
```

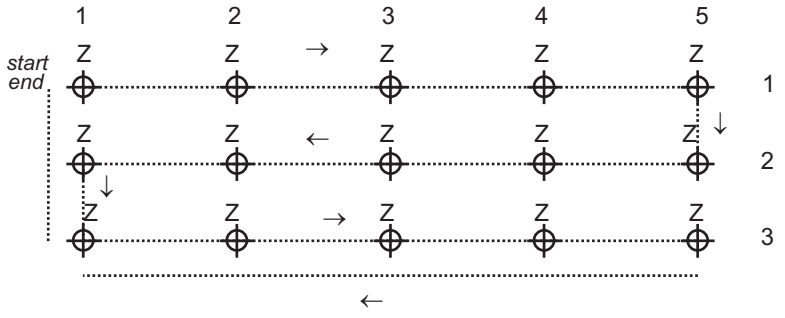


This would do the entire pattern of the above example 5 times:

```
I1M2000,I2M300,L-4,LM0,I2M600,I1M3000,L-3,IA2M0,LM-0,LA5<cr>
```

Example #11	Motors run	RAM used	Function
X,Y Matrix	3	30	X,Y Matrix Moving Z Axis Up then Down at each Position

```
I3M2000,I3M-2000,I1M1600,L5,I2M400,L-3,IA1M0,IA2M0<cr>
```



Troubleshooting

Symptom	Possible Cause
Power (Green LED) light does not come on	Power supply not connected or AC cord not attached. Power switch not on. Circuit breaker tripped (white center protruding from breaker.)
Motor makes noise but is not moving (Stalled)	VXM not configured for Motor, speed too high, broken wiring, or jammed mechanism/motor.
Motor is not making any noise and is not moving (no power to motor)	Limit switches not connected or set for wrong type switches
Circuit Breaker trips when power applied to VXM	Voltage > 28 volts or polarity reversed
Circuit Breaker trips while running motor for a short time	Wrong motor selected. Shorted wiring.
Circuit Breaker trips while running motor for a long time	Wrong motor setting. Shorted wiring. Controller overheating from lack of ventilation or ambient temperature too high
Controller is too hot to touch (It is normal for Controller and Motor do get very warm when running continuously)	Wrong motor setting. Lack of ventilation or ambient temperature too high. **NOTE:** Motor should always be mounted for heat conduction.
Motor runs erratically (at lowest speeds goes either direction, and has low torque at mid speeds)	Broken wire to motor or broken connector pin
Motor always goes opposite direction	Directions were inverted with the "**setDM**" command. Refer to the "**getDM** and "**setDM"** command.
VXM resets itself and sends "**ËB**" to the host.	A motor 3 or 4 command was sent to the VXM when there is not a second VXM connected to the VXM bus.
VXM continuously resets itself (flashes power light slowly) and sends "**ËB**"s to the host.	A master VXM attempts communication with a Slave that is off.
Power (Green LED) light flashes rapidly and continuously on power-up	Run, Stop, or a Jog input is pulled low. The VXM does not allow button Jog inputs to be activated at power up. (release button/input to recover.)
On-Line light flashing continuously (yellow LED)	RS-232 overrun error, host sent commands while VXM was busy sending requested data (power off/on to recover.)
Power and On-Line light flash rapidly and VXM sends "**ËM**" to the host.	Program memory is full (send "**K**" to recover.)
Power and On-Line light flash rapidly and VXM sends "**ËL**" to the host.	More than 10 nested Loop commands encountered per run (send "**K**" to recover.)
Power and On-Line light flash rapidly and VXM sends "**ËJ**" to the host.	More than 4 nested "JMx"s encountered per run (send "**K**" to recover.)
Power and On-Line light flash rapidly and VXM sends "**ËC**" to the host.	When "**U9**x" missing or motor not the same when continuous indexing (send "**K**" to recover.)

Specifications

Physical:
Weight.(VXM-1)...2.6 lbs (1.2 kg)
Weight.(VXM-2)...2.9 lbs (1.3 kg)
Height .....3.27" (83 mm)
Width .....4.37" (111 mm)
Length .....6.89" (175 mm)

AC Power Supply
Weight.....1.0 lbs (0.45kg)
Height .....1.57" (40 mm)
Width .....2.72" (69 mm)
Length .....5.14" (131 mm)

Electrical Requirements:
AC Power Supply..... 100-240VAC 2A 50-60Hz
VXM Controller 24VDC 2.5A

Environmental:
Operating Temperature .... 35° -95° F (2° - 35° C)
Relative Humidity..... 10%-90% (noncondensing)

Basic Models:
VXM-1 (one motor version)
VXM-2 (two motor version, one motor operates at a time)

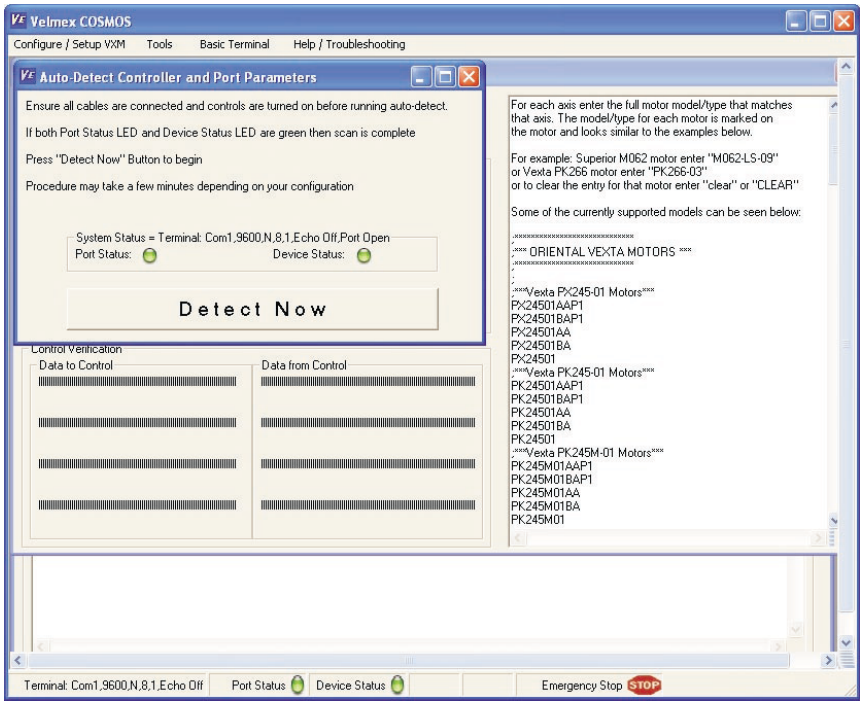
RS-232 Port Configuration: 8 Data, No Parity, 1 Stop, 9600 baud rate default

Velmex COSMOS Software

The COSMOS software for Windows is the easiest way to configure, program, and become familiar with the features of the VXM controller. COSMOS has the following capabilities:

1. Test serial port for communications
2. Retrieve and update setup information
3. Display status and error messages
4. Enter commands directly into the VXM

COSMOS is included on the VXM CDROM.



Appendix A

Editing/Debugging Tools

H Put the Controller on **H**old (single step through program). When the VXM receives the single character "**H**" the Hold Flag will be set. With the Hold Flag set, a "running" program stops before each command and sends a "**:**" followed by the listing* of the command to the host. When stopped, the "**X**", "**Y**", "**Z**", and "**T**" commands can be used to read motor position. A "**G**" will start the listed command and advance to the next. An "**H**" toggles the flag off and the program continues as normal. The "**K**" terminates the program and clears the Hold Flag. This Command allows single stepping through a program for debugging or as a program interrupt from the host.

***Revised: Command listing only on VXM firmware versions 1.21 & up**

del Delete the last command in current program. This command should only be used for manual editing of programs with a terminal/ terminal program. Use the "**C**" command for complete deletion of commands in a program.

res Reset VXM to power-up state. This is the exact same state when power is turned off then back on.

Request the **number** of the currently selected motor. The VXM will send the number of last motor run or last command that set a motor value. The value will be 1 to 4 followed by <cr> (carriage return).

~ Get State of User Inputs, Motor 1 and Motor 2 Jog Inputs. Each input represents one bit of a binary value. The value the VXM sends is represented in the table below.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Decimal Value	128	64	32	16	8	4	2	1	Decimal Value
1=high	Input 4	Input 3	Input 2	Input 1	Jog 2+	Jog 2-	Jog 1+	Jog 1-	
No Inputs Activated	1	1	1	1	1	1	1	1	255
Input 3 Activated	1	0	1	1	1	1	1	1	191

NOTE: In response to this command the VXM sends a single character equal to the above 8 bit value

? Read State of Limit Inputs. Each input represents one bit of a binary value. The value the VXM sends is represented in the table below.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Decimal Value	128	64	32	16	8	4	2	1	Decimal Value
1=high	Limit 4+	Limit 4-	Limit 3+	Limit 3-	Limit 2+	Limit 2-	Limit 1+	Limit 1-	
No Limits Activated or Connected	1	1	1	1	1	1	1	1	255
Limit 1+ Low	1	1	1	1	1	1	0	1	253

NOTE: In response to this command the VXM sends a single character equal to the above 8 bit value

$ Get State of User Outputs. Each Output represents one bit of a binary value. The value the VXM sends is represented in the table below.*

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Decimal Value	128	64	32	16	8	4	2	1	Decimal Value
1=high					Output 4	Output 3	Output 2	Output 1	
No Outputs Activated (default)	0	0	0	0	0	0	0	0	0
Output 2 High	0	0	0	0	0	0	1	0	2

NOTE: In response to this command the VXM sends a single character equal to the above 8 bit value

***NEW COMMAND: available only on VXM firmware versions 1.22 & up**

See Also
Ist, X, Y, Z, T, V

Appendix B

Advanced Input/Output

Input commands:

U6

single character "W" to the host when this command is executed. The VXM will wait until a "**G**" is received from the host before proceeding in the program.
Memory usage = **2 bytes**.

U13 Wait for a Jog button to be pressed. This command allows user interaction, by initiating a jump to a specific program, or allowing the current program to proceed.
The Jog 1- button will cause a jump to program #1.
The Jog 1+ button will cause a jump to program #2.
The “Run” button will cause the current program to continue to the next command
Memory usage = **2 bytes**.

U23 Wait for a Jog button to be pressed. This command allows user interaction, by initiating a jump-and-come-back-for-more to a specific program, or allowing the current program to proceed.
The Jog 1- button will cause a jump to program #1 and return.
The Jog 1+ button will cause a jump to program #2 and return.
The “Run” button will cause the current program to continue to the next command
Memory usage = **2 bytes**.

U30 Wait for a low to high transition on the user input 1. A "high" is a voltage between +1.5VDC and +5VDC applied to I/O,5. A simple pushbutton or toggle switch can be used between 0V (I/O,1) and input 1 (I/O,5) to satisfy this input. The input level must be low (less than 0.8V) for at least 1 ms, and go high for at least 1 ms to be a valid input. This command is best used when interfacing to other solid-state logic devices, refer to the "**U50**" command for push-button switch input.
Memory usage = **2 bytes**.

U31 Wait for a low to high transition on the user input 1 holding user output 1 "high" (+5V) while waiting. A "high" is a voltage between +1.5VDC and +5VDC applied to I/O,5. User output 1 (I/O,14) will go to +5V for the duration of the wait. A simple pushbutton or toggle switch can be used between 0V (I/O,1) and input 1 (I/O,5) to satisfy this input. The input level must be low (less than 0.8V) for at least 1 ms, and go high for at least 1 ms to be a valid input. This command is best used when interfacing to other solid-state logic devices, refer to the "**U51**” command for push-button switch input.
Memory usage = **2 bytes**.

U32 Wait for the Jog 1- button to be pressed and debounced.
Memory usage = **2 bytes**.

U33 Wait for the Jog 1+ button to be pressed and debounced.
Memory usage = **2 bytes**.

U50 Wait for a low to high transition on the user input 1with debouncing for a mechanical push-button switch. A "high" is a voltage between +1.5VDC and+5VDC applied to I/O,5. A simple pushbutton or toggle switch can be used between 0V (I/O,1) and input 1 (I/O,5) to satisfy this input.

When a push-button switch is pressed, the switch's electrical contacts will bounce off each other a few times before settling into their final position. This bouncing will produce a series of highs and lows, which could result in several consecutive wait commands to see these electrical bounces as valid inputs from just one push-button press. When using the "**U50**" command, the VXM will filter out the electrical bounces associated with mechanical switches.
Memory usage = **2 bytes**.

U51 Wait for a low to high transition on the user input 1 with debouncing for a mechanical push-button switch, holding user output 1 "high" (+5V) while waiting. A "high" is a voltage between +1.5VDC and +5VDC applied to I/O,5. User output 1 (I/O,14) will go to +5V for the duration of the wait. A simple pushbutton or toggle switch can be used between 0V (I/O,1) and input 1 (I/O,5) to satisfy this input.

When a push-button switch is pressed, the switch's electrical contacts will bounce off each other a few times before settling into their final position. This bouncing will produce a series of highs and lows, which could result in several consecutive wait commands to see these electrical bounces as valid inputs from just one push-button press. When using the "**U51**" command, the VXM will filter out the electrical bounces associated with mechanical switches.
Memory usage = **2 bytes**.

U90 Wait for a low to high transition on the Run input/button with debouncing for a mechanical push-button switch. Pressing the front panel Run button or a connection between I/O,4 and I/O,1 (0V) will activate this input.
Memory usage = **2 bytes**.

Programming Tip: Use the **"U90"** as the first command if you want a run to only start with a press and the release of the Run button.

CAUTION: The Run input also starts the current program when the VXM is in an idle state either On-Line or in Local Jog/slew mode.

See Also
U0, U1
U11,U12,U21,U22 (Appendix T)

Output commands:

U14 User output 2 "low". The user output 2 (I/O,15) will go to 0V. This is the state of the user output 2 on power-up. This command is used in conjunction with the "**U15**" command.
Memory usage = **2 bytes**.

U15 User output 2 high. The user output 2 (I/O,15) will go to +5V. This command is used in conjunction with the "**U14**" command.
Memory usage = **2 bytes**.

U16 User output 3 "low". The user output 3 (Optional Connection) will go to 0V. This is the state of the user output 3 on power-up. This command is used in conjunction with the "**U17**" command.
Memory usage = **2 bytes**.

U17 User output 3 high. The user output 3 (Optional Connection) will go to +5V. This command is used in conjunction with the "**U16**" command.
Memory usage = **2 bytes**.

U18 User output 4 "low". The user output 4 (Optional Connection) will go to 0V. This is the state of the user output 4 on power-up. This command is used in conjunction with the "**U19**" command.
Memory usage = **2 bytes**.

U19 User output 4 high. The user output 4 (Optional Connection) will go to +5V. This command is used in conjunction with the "**U18**" command.
Memory usage = **2 bytes**.

See Also
U4, U5

Using the I/O on Slave (Bussed VXMs)

Most of the inputs and outputs on a Slave VXM (in a Master/ Slave bussed configuration) are available through the Master. To access the user I/O on the second VXM (Slave) from the Master use the standard "**Ux**" commands +100. The Master will subtract 100 from the "**Ux**" command and send the result to the Slave.

The following are valid commands for I/O addressing on a Slave:

U100 Wait for a "low" on user input 1
U101 Wait for a low on user input 1, holding user output 1 high while waiting
U104 User output 1 "low" (reset state)
U105 User output 1 high
U102 Enable Jog mode while waiting for an input
U103 Disable Jog mode while waiting for an input
U113 Wait for a front panel button to jump to a program or continue: "Motor 1 Jog -" button to jump to program #1, "Motor 1 Jog +" button to jump to program #2, "Run" button to proceed in current program.
U114 User output 2 low (reset state)
U115 User output 2 high
U116 Optional User output 3 low (reset state)
U117 Optional User output 3 high
U118 Optional User output 4 low (reset state)
U119 Optional User output 4 high
U123 Wait for a front panel button to jump to a program and come back, or continue: "Motor 1 Jog -" button to jump and return to program #1, "Motor 1 Jog +" button to jump and return to program #2, "Run" button to proceed in current program
U130 Wait for a low to high transition on user input 1
U131 Wait for a low to high transition on user input 1, holding user output 1 high while waiting
U132 Wait for "Motor 1 Jog -" button to be pressed on front panel with debouncing
U133 Wait for "Motor 1 Jog +" button to be pressed on front panel with debouncing
U150 Wait for a low and high on user input 1 with debouncing for a mechanical push-button switch
U151 Wait for a low and high on user input 1 with debouncing for a mechanical push-button switch, holding user output 1 high while waiting
U190 Wait for a low to high on the Run button or connection I/O,4 with debouncing for a mechanical push-button switch

Appendix C

The Multifunction User Inputs

setI*x* Set operating mode of User Inputs. The value for *x* is a number between 0 and 255 that can be derived from the table below.

Example:
This example enables binary selection of programs 0 to 3 with user inputs 2 and 3

```
setI67,
```

getI Get operating mode of User Inputs. The value returned will be a number between 0 and 255 (see table below.)

The program select feature of inputs 2 and 3 can be used to select programs 0 to 3 for stand-alone applications. A rotary type binary switch would be attached to inputs 2 and 3 for program selection. Following the program selection, the user would press/ activate the Run input/button. See truth table below for function of each input.

Program #	Input 3	Input 2
0	1	1
1	1	0
2	0	1
3	0	0

1=high (no connection)
0= low (connected to 0V)

Bit	7	6	5	4	3	2	1	0	
Decimal Value	128	64	32	16	8	4	2	1	Decimal Value
	Capture Motor Position on Input 4 trigger 0=disable (bit 1 will be 0)	Program # Select with Inputs 2 and 3 0=disable (bit 2 will be 0)	Jump to Program 4 after Stop input (Input 4) 0=disable	Stop (Input 4) Decel/Hard stop 1=Hard stop 0=Decel to stop	Low valid time for Run and Input 1 1=100usec 0=1ms	Input 3 Interrupt User Waits and Jump to Program 3 enable/disable 0=disable	Stop (Input 4) enable/disable 0=disable "**ËS**" is sent ** to host when enabled, and Stop input	Run enable /disable 0=disable	**x**
Default	0	0	0	0	0	1	1	1	7
Pgm Select on inputs 2,3	0	1	0	0	0	0 *	1	1	67
Record motor position on Input 4	1	0	0	0	0	1	0 *	1	133
Joystick for motors 3/4	0	0	0	0	0	0	1	1	3
Jump to program 4 after Stop input	0	0	1	0	0	1	1	1	39

* VXM clears these bits automatically

**** New feature: Only on VXM firmware versions 1.21 & up**
Also on versions 1.21 an up: When Input 4 (Stop) is held low (0V) program will not run

When Setting Bit 7 See Also x, y

Appendix D

Producing Trigger Outputs

There are three different methods to produce a pulse at an exact position:

1. Index to a position, then while stopped use "**PA**", "**U1**", or "**U5**"/"**U4**" command.

2. Use the Continuous indexing method with the "**U7**" command to produce a pulse out instead of stopping at the end of indexes.

3. Set VXM to Pulse every "n" ("n" is a number from 0 to 32,767) number of steps on output 2

Index, Stop, Output

Example:
This example Indexes 400 steps and makes a pulse on output 1 for 1 second

```
I1M400,PA10,R
```

Continuous Indexing

U7 Start of Continuous Index with pulse output. This command is used when it is desirable to make several Indexes on one axis without stopping or slowing between each Index. Instead of stopping a positive going pulse will appear on user output 2 (I/O,15) at each Index distance. Pulse width is settable with the "**setPAx"** command described on page 43 (default width is 10 μsec.) This pulse would be used to trigger measurement/sampling equipment. The "**U9**" or "**U91**" command must be used as the last command to decelerate to a stop from the last Index.
Memory usage =**2 bytes**.

Continuous Indexes have the following limitations:

a) Each Index must be the same motor, and direction should not be changed unless speed is below 800 steps/sec.

b) Only motors 1 and 2 can be run in this mode

c) The acceleration value set before the "**U7**" command will be used in the continuous index.

d) Speed settings, Jumps, Loops, and Output commands can be used between indexes. Pauses and Wait commands are not allowed.

NOTE: The Power and On-Line light will flash rapidly and VXM sends "**ËC**" to the host when a"**U9**x" command is missing or motor is not the same.

U7 Command Examples:

This example makes an index on motor 1, producing a pulse at positions 1000,1100,1150,1250, and then runs motor 1 back to the start position:
S1M1500,U7,I1M1000,I1M100,I1M50,I1M100,U9,IA1M0<cr>

This example will Index motor 2 and pulse 100 times:
U7,I2M400,LA100,U9<cr>

This example makes an index on motor 2, producing a pulse with speed changes between each index:
S2M1500,U7,I2M2000,S2M3000,I2M4000,S2M500,I2M800,U9,S2M3000,IA2M0<cr>

U8 Start of Continuous Index sending "**@**" to the host. This command is the same as the "**U7**" except the single character "**@**" is transmitted at each Index distance, instead of a pulse on the user output 2.
Memory usage = **2 bytes**.

U9 End of Continuous Index. This command is used, as the ending command of a Continuous Index, in conjunction with the "**U7**" or "**U8**" commands. This command will start the motor into a deceleration to a stop an equal time and distance it took to get to the present speed.
Memory usage = **2 bytes**.

U91 End of Continuous Index. This command is similar to the "**U9**" except it creates an index move in the program for decelerating to a stop. When the VXM sees this command it will change it into a "**U92**" followed by an Index that has a value equal to the distance required to decelerate to a stop.
Memory usage = **6 bytes**

U92 End of Continuous Index. This command is similar to the "**U9**" except it requires an index move directly after it in the program for decelerating to a stop. When the VXM sees this command it will look ahead for the index command and use it as the deceleration distance.
Memory usage = **2 bytes**

NOTE: The "**U91**" command described previously will automatically create this command and the proper index value.



See Also
U77, U99, Appendix G

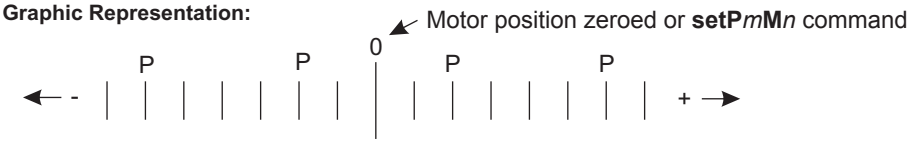
Pulse Every “n” Steps

The following command makes it possible to produce a repeating pulse for any number of motor steps from 1 to 32,767.

setP*m*M*n* Set “Pulse Every *n* # Steps” on output 2* for axis *m, m*= motor# (1,2,3,4), *n*= 0 to 32,767 (0= disable/default.) Pulse width is settable with the “**setPAx”** command (default width is 10 μsec.)
If *n* >1 when this command is set, or motor position is zeroed, (”**N**”) the location from the present position where the pulse will occur is 1/2 *n* going in either direction. However, if *n* is an odd value the extra step to produce a pulse will be in the positive direction (pulse at the integer of 1/2 n going negative, and pulse at the integer of 1/2 n+1 going positive.)
* Master’s output 2 for motor 1 or 2, and Slave’s output 2 for motor 3 or 4.
NOTE: Pulse will be generated when indexing and when jogging.

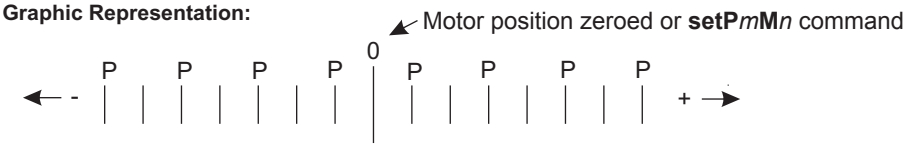
Example:
This example is will pulse on output 2 for every 4 steps of Motor 1
setP1M4

Graphic Representation:



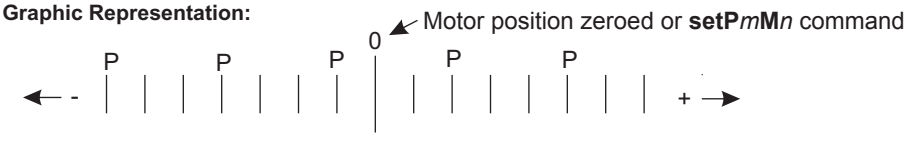
Example:
This example is will pulse on output 2 for every 2 steps of Motor 1
setP1M2

Graphic Representation:



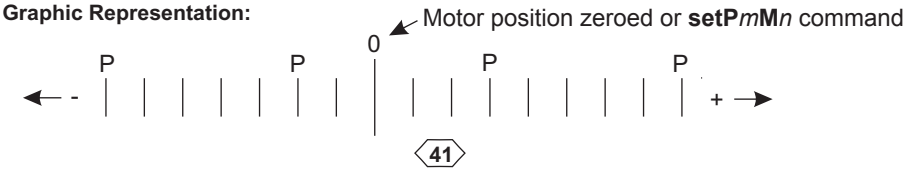
Example:
This example is will pulse on output 2 for every 3 steps of Motor 2
setP2M3

Graphic Representation:



Example:
This example is will pulse on output 2 for every 5 steps of Motor 1
setP1M5

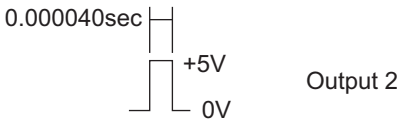
Graphic Representation:



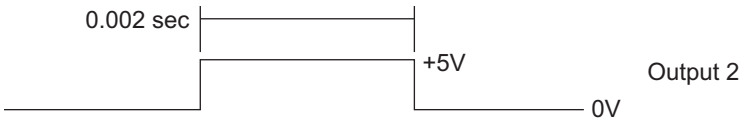
getP*m*M Get value for "Pulse Every *n*# Steps" for axis m .
m= motor# (1,2,3,4)
Value returned will between 0 and 32,767
(0= disabled/default)

setPA*x* Set Pulse width used by **setP*m*M*n*** and **U7** commands. *x*= 1 to 255
(1=default.) Units are 10µsec increments (10 x 10^{-6} seconds)
NEW COMMAND: available only on VXM firmware versions 1.24 & up

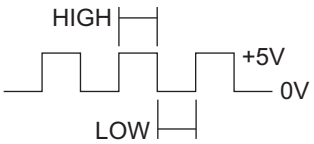
Example:
This example will set the pulse width to 40 microseconds
setPA4



Example:
This example will set the pulse width to 2 milliseconds
setPA200



$$\textbf{Maximum Speed}^{***} = n \times \frac{1}{(\textbf{HIGH}^{*} + \textbf{LOW}^{**})}$$



n is value from **setP*m*M*n***, or **I*m*M*n*** following a **U7**
* Pulse width value *x* from **setPA*x*** command
** Low time needed between pulse
*** Exceeding this speed will truncate pulse timing

getPA Get Pulse width value used for **setP M*m* *n*** and **U7** commands
Value returned will between 1 and 255
1= 10µsec (default)
NEW COMMAND: available only on VXM firmware versions 1.24 & up

Appendix E

Getting Motor Position When Moving

It is possible to read motor position directly while motor is in motion using the "**X**" and "**Y**" commands. At high motor speeds it might not be timely or accurate enough to transfer motor position over the serial port while the motor is in motion. Another approach is to use an external trigger to tell the VXM to capture motor position(s) for later retrieval (after motion has ended.)

The Automatic Deceleration Capture

Every time the motor starts to decelerate to a stop the motor position is saved for later retrieval with the "*****" command. One use of this feature would be to stop the motor with the decelerate-to-a-stop command ("**D**") when an event has occurred. Then after waiting for the move to end, (wait for "^") and reading the position where deceleration started.

***** (Asterisk) Request motor position when the last deceleration occurred. This position can be from a normal index decelerating to a stop, or an interrupted index from a "**D**" (Decelerate to a stop) command or Stop input/button (User input 4.) Below is what the host would receive if the last motor indexing started it's deceleration at position negative 14901.

-0014901<cr>

The Triggered Position Capture

The VXM can capture motor 1 and motor 2 position(s) either by the host sending a "**!**" or from a pulse on input 4 (see Appendix C.) Up to 4 positions will be recorded (one for each trigger input.) Then after waiting for the move to end, (wait for "^") the positions can be requested with the "**X**" and "**Y**" commands.

! Capture motor 1 and motor 2 positions into FIFO buffer (4 positions maximum.) Use the "X" and "Y"commands to retrieve buffer data after move(s) are complete.
NOTE: buffer data is automatically zeroed at the start of every run.

See Also
X, Y, setIx

x Request captured motor 1 positions from FIFO buffer (all 4 positions.)
NOTE: buffer data is automatically zeroed at the start of every run.

Example:
This example shows the values the VXM returned when the "**!**" was sent at positions 521,919, and 1149 while motor 1 was indexing.

+0000521
+0000919
+0001149
+0000000

y Request captured motor 2 positions from FIFO buffer (all 4 positions.)
NOTE: buffer data is automatically zeroed at the start of every run.

Example:
This example shows the values the VXM returned when the "**!**" was sent at position negative 2004 while motor 2 was indexing.

-0002004
+0000000
+0000000
+0000000

See Also
X, Y,
setlx

Appendix F

More Feedback/ Precision

There are three ways to improve precision and get more information from the VXM.

1. Use the limit switch to establish a home position.

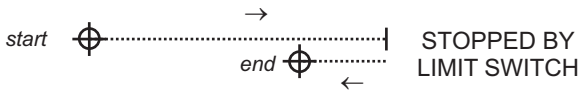
2. Turn on backlash compensation feature if high accuracy is needed when moving both directions.

3. Set the limit switch output feature to let the host computer know that a positioner has inadvertently hit a limit switch.

Use Limit Switch for Home

The limit switch on a typical Velmex slide assembly can provide a repeatability of better than 0.0004" (0.010 mm.) Using a limit switch on initial power-up to set a home is a good way to insure consistent performance from a precision positioning system. See Appendix O for setting program 4 to do homing routines at power-up.

```
S1M500,I1M0,I1M-200,IA1M-0,R
```



Backlash Compensation Improves Accuracy

Mechanical devices have some clearances between mating parts. Whenever such a device is commanded to reverse direction there can be some lost motion. The VXM has the following command to overcome this situation.

B*x* **B**acklash Compensation, compensation is 20 steps when =1, off when =0 (default). Desired number of "comp" steps can be input directly (*x*= 2 to 255)* The VXM can compensate for mechanical backlash by ending every index in the positive direction. When backlash compensation is on, and a motor makes a negative Index, "comp" number of steps will be added to the Index. The Motor will then immediately reverse, indexing positive "comp" number of steps. The VXM will not do the ending positive "comp" step move if the index is the **I*m*M-0** (Index until negative limit encountered.)
Memory usage = **0 bytes** (this command is immediate, and uses a reserved memory location)

B<cr> Read **B**acklash Compensation setting, off when value returned=0(default)
The number of "comp" steps will be returned when set**

****Only on VXM firmware versions 1.25 & up, older versions return "1" to indicate on for 20 steps of compensation**

Indicate Limit Switch Encounter to Detect Faults

A properly connected/ configured limit switch will always stop a motor immediately. Normally the host computer would not know if a limit switch stopped a move. With the "**O**" command described below, the VXM can be set to notify the host computer that it has hit a limit switch and stop the entire program too.

O*x* Indicate Limit Switch **O**ver-travel to the host.

x Value	Function
0	Disabled (default)
1	Send O to host when hit Limit Stop
2	program if hit Limit Send O to host
3	and Stop program when hit Limit

The VXM sends "**O**" when *x*=1 and a limit switch is encountered. This command is useful when the host needs to know if a positioner's travel has been exceeded due to a motor stall or an index(es) that are too long. When Indicate-Limit-Switch-Over-travel = 1, the VXM transmits the single character "**O**" to the host when an indexing motor activates it's limit switch input.
Memory usage = **0 bytes** (this command is immediate, and uses a reserved memory location)

NOTE: Home to limit switch commands never stop the program, but the "O" will be sent if enabled.

O<cr> Read current setting for Indicate Limit Switch **O**ver-travel to the host.

See Also
V, ?, ImM0, ImM-0, Example #7

Appendix G

Complex Profiles & Coordinated Motion

Complex Profiles

The VXM automatically does a simple profile move every time it is required to perform an index. This profile consists of a acceleration segment, slew segment, and a deceleration segment. Using the VXM's Continuous indexing feature more complex motion profiles are possible. In addition to the Continuous Index commands described in Appendix D, below are two more specifically for making complex profiles.

U77 Start of Continuous Index with no output. This command is same as the "**U7**" except it does not produce the pulse on user output 2.
Memory usage = **2 bytes**.

U99 End of Continuous Index with no deceleration. This command is similar to the "**U9**" command without the deceleration move after the last index.
Memory usage = **2 bytes**.

CAUTION: The motor speed should be below 800 steps/second, when the VXM executes this command, to prevent an excessively hard stop that may cause a mechanical overshoot of intended position.

As mentioned in Appendix D, speed commands can be used between indexes in the continuous index mode. The number of speed changes is limited only by available program memory space. Changes in speed will occur at the acceleration rate set before the continuous index mode.

Coordinated Motion

The most common method to move is one axis at a time. Using one VXM controller this is the only option available. However, with two VXMs connected together with the VXM bus two axes at a time can be run. Assuming there are two VXMs connected by a bus cable, the following commands can be utilized to produce complex coordinated motion profiles.

This is a typical example of running two motors sequentially when each axis is on a different VXM:

I1M400,I3M800,R

To run these two motors (motor 1 and motor 3) the same time requires adding "**( )**" around the indexes. This example will combine the index commands to run simultaneously:

(i3,i1,..) Combine Index commands to run simultaneously on two VXM controllers connected by VXM bus. **i3,i1,..** = index commands, slave motor (3,4) first.
Memory usage = **2 bytes**.
Example:

(I3M800,I1M400,)R

NOTE: A motor 3 or motor 4 index command must be before a motor 1 or motor 2 index for simultaneous operation to occur.

Example:

This will not run the motors same time!



Master/Slave Associated Programs*

Two VXM controllers that are bussed together can be set to execute entire programs simultaneously. The procedure to coordinate VXM program to VXM program is as follows:

1. Transfer a program to the Slave using "**[ ]**" Program
2. associate the Master to the Slave with the "**PMAx**" command*

[i1,i2...] Send data to Slave through the Master. **i1,i2...** = commands for Slave.
NOTE: Status requests, "**R**", and "**Q**" commands are not allowed.

PMA*x* Program associate program number *x* in Master to the same program number in the Slave. Program *x* in the Slave will run the same time when program *x* in the Master is run. *x*= 0,1,2,3,4*

PMA-*x* Program associate all programs in the Master to all the programs in the Slave except for program number . Programs numbers except*x* in the Slave will run the same time when a program in the Master is run. *x*= 0,1,2,3,4*

PMA255 Disable Master/ Slave Program association (default)

* **CAUTION:** Motor 3 or 4 commands should not be used if Program Associate is set to program 0. A Master uses the Slave's program 0 memory space to run a motor 3 or 4 command (I3Mx, I4Mx, S3Mx, etc.) Any existing commands in the Slave's program 0 will be erased when running a motor 3 or 4 command!

There are special Excel spreadsheet files on the CDROM to create the VXM commands for producing sinusoidal motion, triangles, rectangles with radius/ chamfer corners, and circles.

See Also

U7, U8, U9, U91, U92, Appendix D

Appendix H

Advanced Jog Mode

When the On-Line (yellow) light is not lit, the VXM is in the Local/Jog mode. Using the front panel jog buttons, each motor can be jogged a single step or slewed to a default speed of 2000 sps (5 revs/sec.) in either direction.

When a Jog button is pressed the motor moves 1 step (1/400 rev.) If the button is held for >0.3 second the motor will accelerate to a default speed of 2000 sps.

Pressing/holding the Stop button while using the Jog buttons will hold the speed at 39 sps.

The default speed of 2000 mentioned above is settable by "**setj**" command.
Additionally there is a second jog speed "**setJ**" that can be selected by activating input 2

setj*m***M***x* Set primary maximum jog speed. *x*= 1 to 6000 sps (default=2000)
m= motor# (1,2,3,4)

setj*m***M0** Disable jog input for motor *m*. This command will deactivate the jog buttons for motor *m* and the corresponding auxiliary I/O jog inputs.
m= motor# (1,2,3,4)

setJ*m***M***x* Set secondary maximum jog speed. When input 2 is low (I/O,6) this speed will be the applied maximum jog speed. *x*= 40 to 6000 sps (default=2000)
m= motor# (1,2,3,4)

getj*m***M** Get the primary maximum jog speed. The value returned will be a number between 0 and 6000 (default=2000)
m= motor# (1,2,3,4)

getJ*m***M** Get the secondary maximum jog speed. The value returned will be a number between 40 and 6000 (default=2000)
m= motor# (1,2,3,4)

See Also
Auxiliary I/O Connection

Optional Digital Joystick

The optional digital joystick allows remote jog control of a one or two axis VXM controller. The joystick provides four momentary outputs that are connected to the Jog Motor inputs on the Auxiliary I/O (internally connected to same inputs as the front panel jog buttons.) See previous page for information on jog function and speed settings.

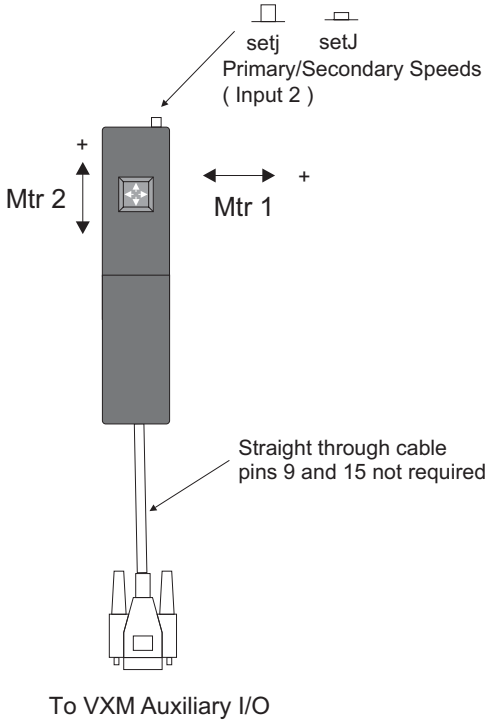
The joystick without the enclosure is available for OEM applications.

Enclosure Dimensions: 1.45" x 5.75" x 0.85" (Width x Depth x Height)
Overall height: 1.6" (bottom of enclosure to top of joystick lever)

The digital joystick has a button switch connected to Input 2 for toggling between the primary and secondary settable jog speeds.

NOTE: The default primary and secondary speeds are by default both set to 2000.

NOTE: It is possible to disable/remove the button switch if Input 2 is needed for a function other than jog speed selection. To disable/remove the button, with the button in the out position, use pliers to pull the button cap off the switch actuator. The switch actuator should now be below the surface enough to prevent unintended input. An alternate method to disable the button switch is by clipping off pin 6 on the cables connector.



Enabling Jog When a Program is Running

U2 Enable Jog while waiting for an input. This command will allow motor jogging, with the jog button/inputs, during the following wait commands: **U0,U1,U30,U31,U50,U51,** or **U90**.
Memory usage = **2 bytes**.

U3 Disable Jog while waiting for an input (default setting on power-up.) This command will disable motor jogging during a wait command.
Memory usage **2 bytes**.

Digitizing With a Host

The VXM stores its absolute position (relative to when registers were zeroed) in memory. The absolute registers reflect the accumulated distance from operating the motors in the Jog mode and/or under program control. These registers can hold from -8,388,608 to +8,388,607 steps.

With a host terminal or computer connected via the RS-232 interface, the VXM can be used as a digitizer.In the Jog mode the VXM will send motor position when it receives a"**D**" from the host.

Here is an example of what the host would receive when Motor 1 is at absolute 201, Motor 2 is at absolute -1294010, Motor 3 is at absolute 0, and Motor 4 is at 80000:

```
X+0000201<cr>
Y-1294010<cr>
Z+0000000<cr>
T+0080000<cr>
```

NOTE: VXM will only send the X and Y motor positions in a single VXM system. The above example is from a Master/Slave VXM system.

Appendix I

The Analog Input

The VXM has a 10-bit analog to digital converter for general use, motor speed setting, or for use with the optional Analog Joystick (see Appendix J.)

The analog reference voltage is the internal +5VDC which is also used for the VXM's internal logic. This +5VDC is brought out on I/O pin 2 for use with additional analog circuitry.

CAUTION: The analog input (Ain) voltage must not exceed the +5VDC

Internally Ain has a 100K ohm resistor to the +5VDC, and a 100K ohm resistor to 0V. There is also a 100 ohm resistor between the converter and Ain.

By default the voltage at the analog input Ain (I/O,3) is +2.5VDC. The digital value of the Ain can be read directly with the "**@**" command. Since this is a 10-bit converter, the 2.5VDC would be equal to 512 (½ of 1024.) Connecting Ain to the +5VDC would return a value of 1024. If Ain is connected to 0V (I/O,1) the returned value will be 0.

NOTE: There is a ± 2 digit margin for conversion/ circuitry error

External potentiometers should be between 2K and 10K ohms.

@ Read user analog input value Ain (I/O,3.) The value returned will be a number between 0 and 1024.

S*m***M-***x* Set Speed to motor *m* from analog input value and range (70% power),*x*
m= motor# (1,2,3,4) *x*=speed range:
1,2,3,4,5,6,11,12,13,14,15,21,22,23,24,31,32,33,41,42,51.
See table at right to determine actual range of speed.
Memory usage = **3 bytes**.

Example:
This example proportions the full range of the analog input to a speed range from 1000 to 3000 steps/sec for motor 1.

S1M-12<cr>

SA*m***M-***x* Set Speed to motor *m* from analog input value and range (100% power),*x*
m= motor# (1,2,3,4) *x*=speed range:
1,2,3,4,5,6,11,12,13,14,15,21,22,23,24,31,32,33,41,42,51.
See table at right to determine actual range of speed.
Memory usage = **3 bytes**.

NOTE: When a "**SmM-*x*"** is stored in a program, the original *x* value will be kept with the command. However, when the "**lst**" command is used to list the program, the *x* value displayed will be the steps/sec speed converted from reading Ain.

This feature allows viewing the actual derived speed the VXM is going to use.

Connection to Potentiometer

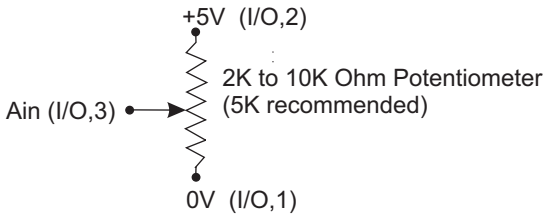


Table for analog assigned motor speed

x	**Speed Range (steps/sec.)**
1	1 - 1000
2	1 - 2000
3	1 - 3000
4	1 - 4000
5	1 - 5000
6	1 - 6000
11	1000 - 2000
12	1000 - 3000
13	1000 - 4000
14	1000 - 5000
15	1000 - 6000
21	2000 - 3000
22	2000 - 4000
23	2000 - 5000
24	2000 - 6000
31	3000 - 4000
32	3000 - 5000
33	3000 - 6000
41	4000 - 5000
42	4000 - 6000
51	5000 - 6000

Appendix J

The Analog Joystick Option

The VXM Proportional Speed Joystick provides a precise efficient one, two, three, or four axis variable speed positioning system when used with one or two VXM Stepping Motor Controllers.
The Joystick is a compact long life 1 million cycle design packaged in a hand held size enclosure.
The joystick has two button switches, one for speed range selection, and the other to select motors 1,3 or 2,4.
To achieve simultaneous motion two VXM Controllers are required for this system. The VXMs can be either one or two axis versions.
The joystick without the enclosure is available for OEM applications.

Enclosure Dimensions: 1.45" x 5.75" x 0.85" (Width x Depth x Height)
Overall height: 1.6" (bottom of enclosure to top of joystick lever)

setjA*m*M*x* Set primary joystick speed range. *x*= 1 to 24.
See table at right to determine actual range of speed.
m= motor# (1,2,3,4)
NOTE: Factory set to 1 when VXM and joystick ordered the same time

setjA*m*M0 Disable joystick for motor *m.* This command will deactivate the joystick for motor *m* (default.)
m= motor# (1,2,3,4)

setJA*m*M*x* Set secondary joystick speed range. *x*= 1 to 24. When input 2 is low (I/O,6) this speed range is used by the joystick.
See table at right to determine actual range of speed.
m= motor# (1,2,3,4)
NOTE: Factory set to 10 when VXM and joystick ordered the same time

getjA*m*M Get the primary joystick speed range. The value returned will be a number between 0 and 24 (default=0)
m= motor# (1,2,3,4)

getJA*m*M Get the secondary joystick speed range. The value returned will be a number between 0 and 24 (default=0)
m= motor# (1,2,3,4)

CAUTION: The joystick must be at it's self centered position (middle) at power-up. The VXM reads the joystick value at power-up and assigns this value as the no motion setting. If the joystick is off-center on power-up, the motor will start moving when the joystick returns to center.

setDA*x* Set Joystick Deadband value. *x*= 20 to 100 (default=40)

NOTE: Setting *x* to a low value makes it difficult to move just one axis without inducing motion on the opposite axis. Setting *x* to a high value produces a noticeable delay when changing direction.

getDA Get Joystick Deadband value. Value returned is a number between 20 to 100 (default=40)

By default the joystick operates motor 1 of the Master VXM and motor 1 of the Slave VXM. If the VXM controls are two motor versions, motor 2 of each VXM can be operated with the joystick by applying a low on input 3 (I/O,7)

This capability allows four motors to be operated from one joystick.

Table for joystick assigned motor speed

x	**Speed Range (steps/sec.)**
0	Disable Joystick
1	1 - 250
2	2 - 500
3	3 - 750
4	4 - 1000
5	5 - 1250
6	6 - 1500
7	7 - 1750
8	8 - 2000
9	9 - 2250
10	10 - 2500
11	11 - 2750
12	12 - 3000
13	13 - 3250
14	14 - 3500
15	15 - 3750
16	16 - 4000
17	17 - 4250
18	18 - 4500
19	19 - 4750
20	20 - 5000
21	21 - 5250
22	22 - 5500
23	23 - 5750
24	24 - 6000

Analog Joystick with One VXM

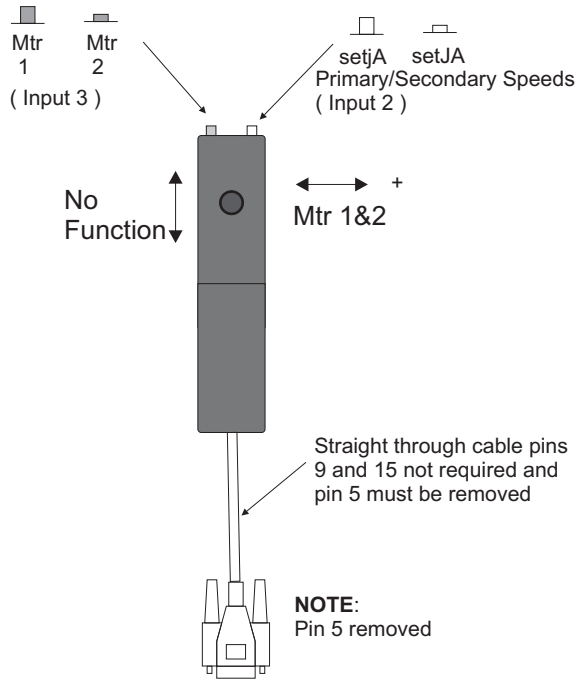
The optional analog joystick can be used with a single VXM control.
The joystick provides analog outputs for two VXM controls. With a single VXM one output is not connected resulting in “no function” in the one direction of the joystick.

The analog joystick has a button switch connected to Input 2 for toggling between the primary and secondary settable jog speeds.

There is also a second button switch connected to Input 3 for alternating between motors 1 and 2.

NOTE: The default primary and secondary speeds are by default both set to 0 (disabled joystick.)

NOTE: It is possible to disable/remove the button switch(es) if Input 2 and Input 3 are needed for another function. To disable/remove the button(s), with the button in the out position, use pliers to pull the button cap off the switch actuator. The switch actuator should now be below the surface enough to prevent unintended input. An alternate method to disable the button switches is by clipping off pin 6 and pin 7 on the cables connector.



To VXM Auxiliary I/O

Analog Joystick with Two VXMs

The joystick provides analog outputs for two VXM controls. A connection to each VXM is accomplished with a special Velmex "Y" cable.

The analog joystick has a button switch connected to Input 2 for toggling between the primary and secondary settable jog speeds.

There is also a second button switch connected to Input 3 for alternating between motors 1,3 and 2,4.

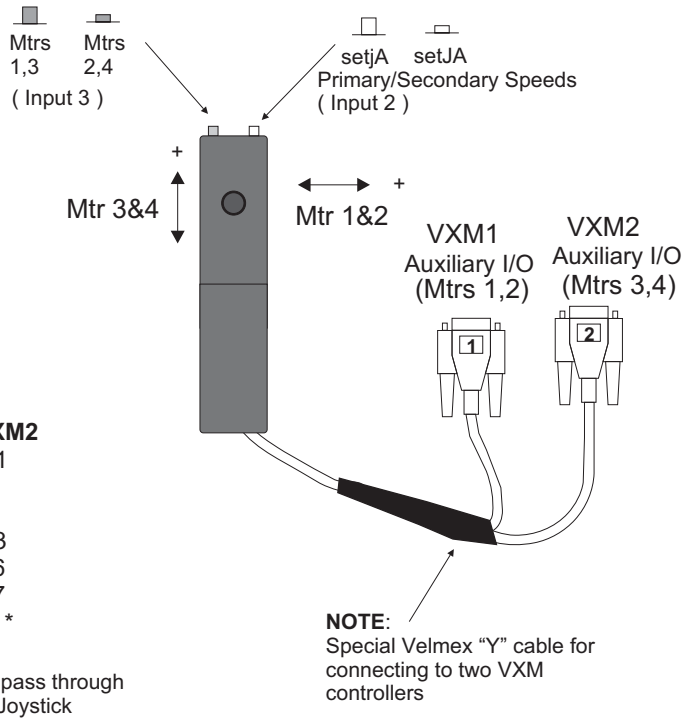
NOTE: The default primary and secondary speeds are by default both set to 0 (disabled joystick.)

NOTE: It is possible to disable/remove the button switch(es) if Input 2 and Input 3 are needed for another function. To disable/remove the button(s), with the button in the out position, use pliers to pull the button cap off the switch actuator. The switch actuator should now be below the surface enough to prevent unintended input. An alternate method to disable the button switches is by clipping off pin 6 and pin 7on the cables connectors.

NOTE: The joystick will not operate VXM2 if the "Y" cable is not connected to VXM1 or if VXM1 is off. This is because the common +5V reference voltage for the joystick comes from VXM1 (pin 2.)

⚠ CAUTION:
If VXM1 is turned off with VXM2 on, motor(s) on VXM2 will run!

Use a common power strip to turn on/off VXMs



"Y" Cable Pinouts

Joystick	VXM1	VXM2
1	1	1
2	2	-
3	3	-
5	-	3
6	6	6
7	7	7
*	*	* *

* Other connection may pass through but are not used by the Joystick

** No Other Connections

Appendix K

I/O Electrical Specifications

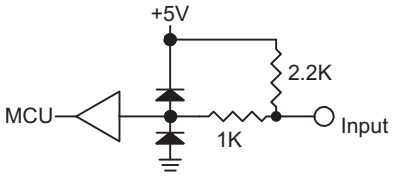
All User I/O inputs and outputs (except limit switch inputs) are TTL levels (0 to +5VDC.)

Inputs have a 2200 ohm resistor to +5VDC, and are activated by connecting to 0V.

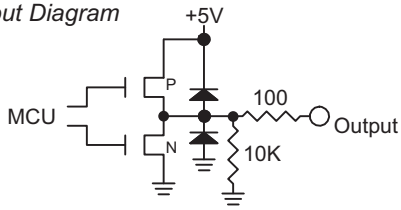
NOTE: When Input 4 (Stop) is held low (0V) program will not run.*

Outputs are normally low, and can sink and source 20 mA max.

Input Diagram



Output Diagram



Limit switch inputs are optically isolated. Limit inputs operate on 24VDC through a 10K ohm resistor to power the LED in the optical isolator (see Appendix N for more information.)

The +5VDC on I/O,2 is intended for use with additional analog input circuitry. Current draw should not exceed 75mA.

⚠ CAUTION:

Optically isolated relays may be required on all user I/Os to insure long term reliable operation.

Never directly connect a VXM I/O to an inductive load, any device that is not within 10 feet of the VXM, or anything not powered at the same AC source.

Damage due to improperly interfacing VXM controllers to other devices is not covered under the warranty.

As a minimum precaution against electrostatic discharge (ESD) damage follow these guidelines:

1. Provide the shortest conductive path possible to earth ground from user designed panels or enclosures that have switches or buttons the operator will come in contact with.

2. Use metal panels and enclosures to house buttons or switches electrically bonded to a protective earth ground.

3. Use shielded cables on all VXM I/O.

4. If no other protective earth ground is available, use the earth ground on the VXMs Auxiliary I/O connector shell or connector shell on shielded cable.

*** New feature: Only on VXM firmware versions 1.21 & up**

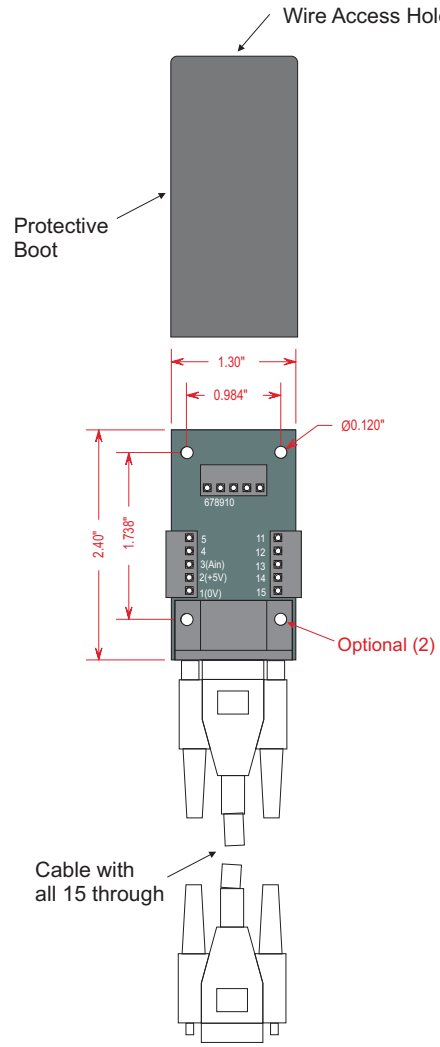
Optional Auxiliary I/O Breakout Module

The optional auxiliary I/O breakout module is a convenient method to interface to the VXMs auxiliary I/O. Wire connections can be made to all 15 I/O connections using the screw type terminal blocks.

Specifications

Wire size: 26 to18 AWG
Boot material: PVC
Boot dielectric strength: 700 V/mil

Pin#	Name
1	0V
2	+5V
3	Ain
4	Run
5	I1
6	I2
7	I3
8	I4
9	0V
10	J1-
11	J1+
12	J2-
13	J2+
14	O1
15	O2

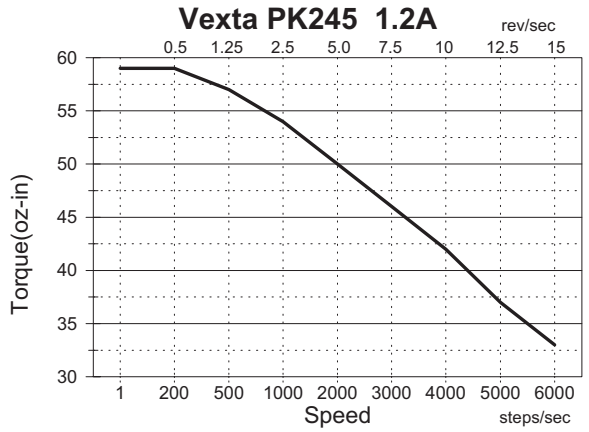


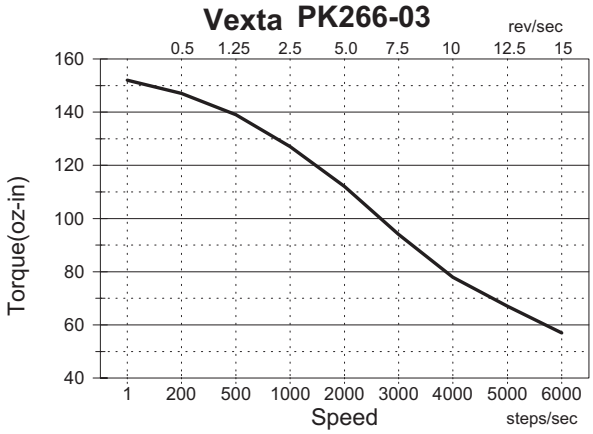
See Also
Auxiliary I/O Connection (page 6)

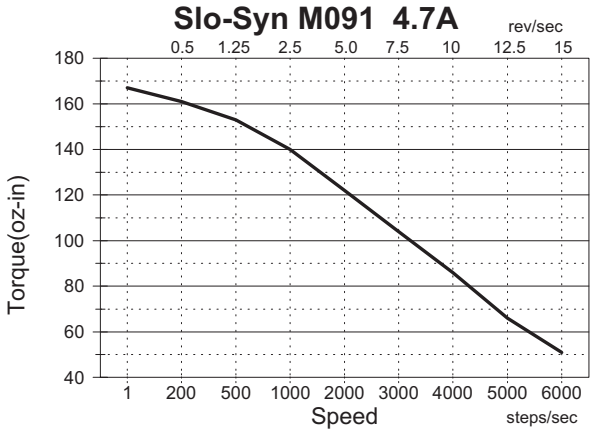
Appendix L

Motor Torque Curves

(Torque measured at 100% power settings)

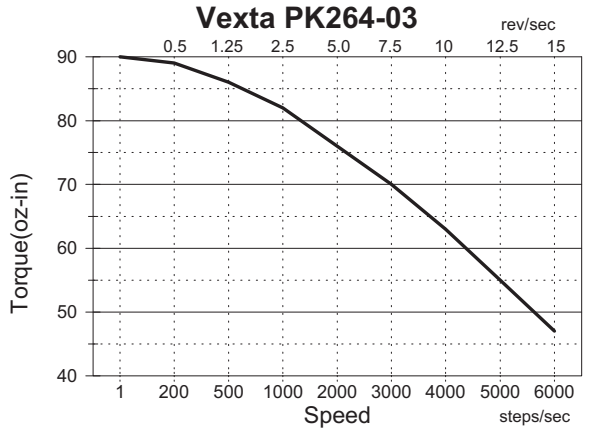


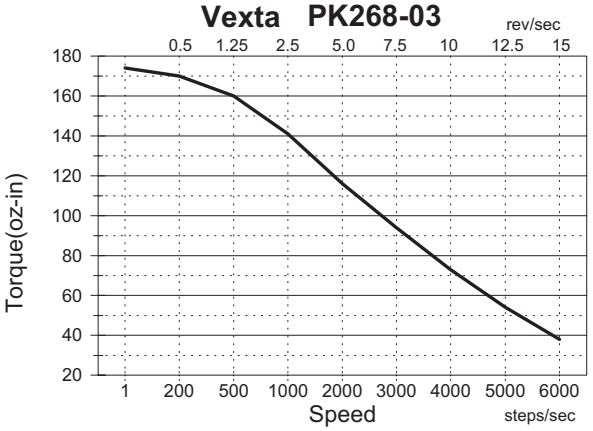


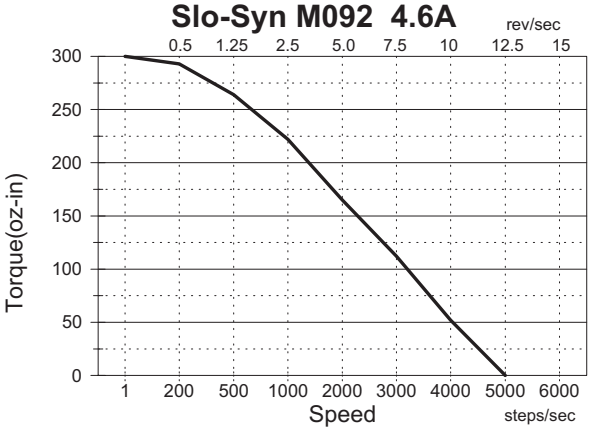


Motor Torque Curves

(Torque measured at 100% power settings)

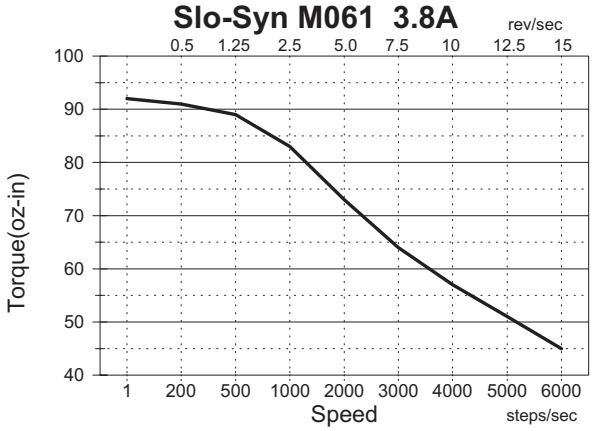


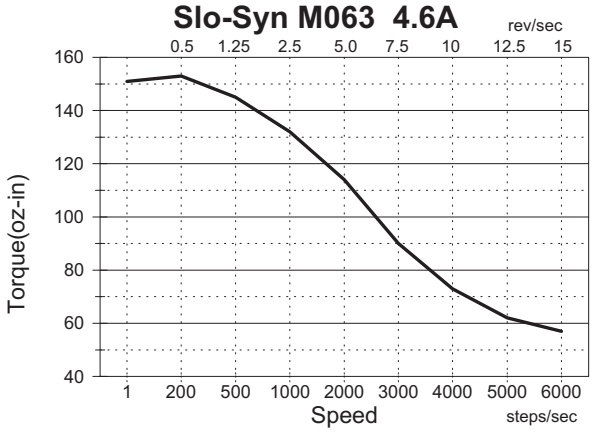




Motor Torque Curves (continued)

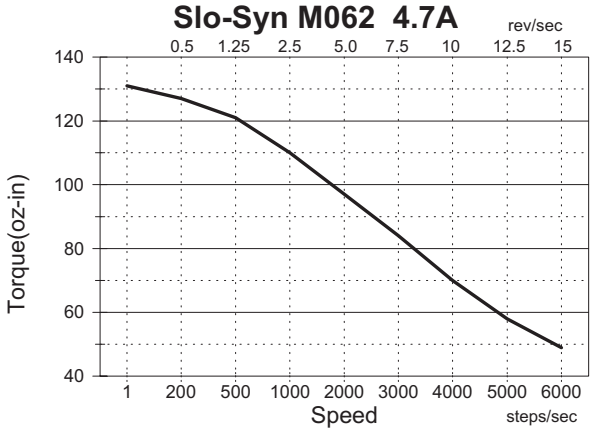
(Torque measured at 100% power settings)





Motor Torque Curves (continued)

(Torque measured at 100% power settings)



Appendix M

Advanced Motor Setup

The VXM applies energy to the motors based on the size of the motor. The physical size is the important aspect that determines the amount energy the VXM will apply. The size of the motor is proportional to the motor's current times it's inductance.

Motors of the same current are not the same to the VXM. When using motors not listed for use with the VXM, set the VXM to motor setting that has the same or higher current times inductance value than the motor you wish to use.

setMA*m*M*x* Set motor type/size selected for axis *m* and applies 100% power to motor in jog mode (normally 70% power), *m*=axis# (1,2,3,4.)
Value for *x* should be a number between 0 and 6. Refer to the table below for the proper value to use.

NOTE: The 100% motor power setting only applies to operation in jog mode. Under program control the "**S**" and "**SA**" speed settings set power to motor.

CAUTION: THE VXM MUST BE SET TO THE EXACT MODEL/TYPE MOTOR(S) BEFORE OPERATING. IMPROPER SETTINGS CAN CAUSE SEVERE DAMAGE TO MOTORS AND CONTROLLER.

x	Motor Model (Amps)
0	Default (0.4A to 0.7A)
1	Vexta PK245 (1.2A)
2	Slo-Syn M061 (3.8A)
3	Slo-Syn M062 (4.7A) Vexta PK264 (3A)
4	Slo-Syn M063 (4.6A) Vexta PK266 (3A)
5	Slo-Syn M091 (4.7A) Vexta PK268 (3A)
6	Slo-Syn M092 (4.6A) Vexta PK296 (4.5A)

See Also
setMmMx, getMmM

Setting for other Motors

The VXM can be used with motors not listed in the table on the previous page. Motors that are compatible with the VXM **must** meet the following criteria:

1. **6 or 8 wire permanent magnet step motor**
2. **Motor rated at a unipolar per phase current of 0.4 to 4.7 Amps**

If the motor you want to use does not meet the above requirements, STOP!The motor is not compatible with the VXM.

Two important specifications are needed about the motor to be used with the VXM. The motor's Unipolar per phase current rating in amps, and the motor's inductance rated in Millihenries (MH.) Refer to the motor manufacturer's data sheet to obtain this information.

Multiply the current (I) times the inductance (L) to arrive at a product value (IxL) to compare in the table below.

When determining the VXM value x from the table below, the motor must be within both the current range and the IxL product.

x	Unipolar Current Rating	Current Inductance Product (I x L)
0	0.4 to 0.7 Amps	1 to 14
1	0.8 to 2 Amps	3.3 to 10
2	2.5 to 4 Amps	2.4 to 3.6
	1.5 to 1.7 Amps	13 to 14
3	3.5 to 4.7 Amps	3.7 to 5.1
	1.8 to 2.0 Amps	11 to 12
4	3.5 to 4.7 Amps	5.2 to 6.9
5	3.5 to 4.7 Amps	7.0 to 12.5
6	3.5 to 4.7 Amps	12.6 to 13

CAUTION: THE VXM MUST BE SET CORRECTLY BEFORE OPERATING. IMPROPER SETTINGS CAN CAUSE SEVERE DAMAGE TO MOTORS AND CONTROLLER.

Appendix N

Limit Switches and Home Switches

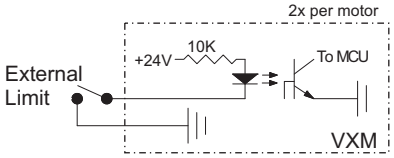
The VXM by default recognizes normally closed (N/C to run) limit switches. The default mode is auto-detect normally closed switches. If the VXM is used with normally open or with a home switch, the limit switch mode will need to be changed. The following command sets the operating mode of the limit switch inputs.

setL*m*M*x* Set Limit Switch mode for axis *m*, *m*= motor# (1,2,3,4)

x
0= Auto-detect N/C to run (default)*
1= Enabled N/C to run
2= Disabled N/C for Home Switch use
-1= Enabled N/O to run
-2= Disabled N/O for Home Switch use

getL*m*M Get Limit Switch mode setting for axis *m*, *m*= motor# (1,2,3,4) value returned is either -2,-1,0,1,2 (default=0)

Limit switch inputs are optically isolated, from the main control logic of the VXM, as shown in the diagram at the right.

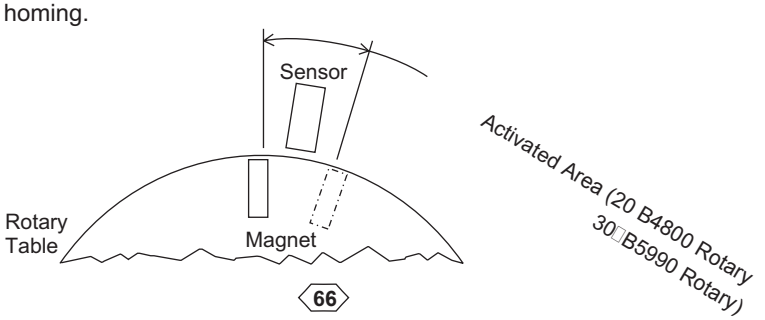


Using Home Switch on Rotary Tables

It is important to understand how a home switch works, and what programming procedure to use to get a high degree of precision and accuracy obtainable by these switches. Repeatability of 1 motor step is achievable if the proper procedures are followed when referencing to a home switch.

The most common home switch used on rotary tables is a magnetic reed type sensor. The magnetic sensor is usually connected directly to the positive and negative limit switch inputs on the VXM.

Home switches have an active area of several degrees. Because of this large area where the switch is activated, it is important to always approach the switch from the same direction when homing.



Procedure to configure the VXM for use with a home switch:
(For rotary tables only, see page 27 & 46 for linear actuator homing example)
1. Determine which axis (axes) will be connected to a home switch and connect motor, limit cables, and set VXM for motor type/model attached.

2. Set limit switch function to a value of -2. Example to set motor 1 for use with a normally open home switch:

setL1M-2<cr>

3. Run the "**rsm**" command to permanently save limit setting(s)

Programming sequence for homing to the switch:
(For rotary tables only, see page 27 & 46 for linear actuator homing example)
1. Set a speed for homing (maximum of 1000.) Always use this selected speed for homing to maintain repeatability.

2. Set move to limit command (**I***m***M0**, or **I***m***M-0**) Pick a direction to move to the home switch. Always use this direction to maintain accuracy. NOTE: if the table is already in the active area of the switch the table will not move. Steps 3 and 4 below will compensate for this situation.

3. Set an Index to move back from home switch area to insure table will be totally out of activated area of the switch before doing final homing. Indexing 4000 steps should be adequate to move beyond the active switch area.

4. Set move to limit command (I*m***M0**, or **I*m*M-0**) and zero motor position at this position desired.

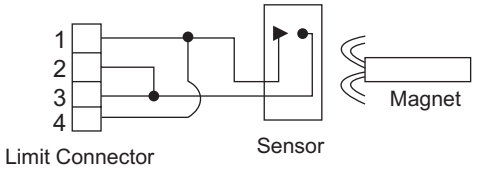
This example homes motor 1 moving negative direction into home switch and zeroes position:

```
C S1M600,I1M-0,I1M4000,I1M-0,IA1M-0,R
```

This example homes motor 2 moving positive direction into home switch and zeroes position 1000 steps away from switch:

```
C S2M800,I2M0,I2M-4000,I2M0,I 2M-1000,IA2M-0,R
```

Magnetic Reed Switch Wiring
(Activated in both directions)



Limit Connector
Amp 1-480703-0

See Also
Limit Switch Wiring (page 6)

Appendix O

Controller Mode

The VXM has a main mode register that can be set with the "**setDMx**" command.

setDM*x* Set operating mode of VXM. The value for *x* is a number between 0 and 255 that can be derived from the table below.

Example:
This example would invert the direction of motor 1 from the standard.

```
setDM3,
```

getDM Get operating mode of VXM. The value returned is a number between 0 and 255 (see table below.) default=1

Use this feature for homing VXM every power-up in stand-alone applications.
Put home routine in program #4.
First run after power up will run program # 4 first.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Decimal Value	128	64	32	16	8	4	2	1	Decimal Value
	Insert/Combine Program # 4 on 1st Run	Set This VXM as Master	Invert Motor 4 Direction	Invert Motor 3 Direction	Emulate NF90	Invert Motor 2 Direction	Invert Motor 1 Direction	1	***x***
Default	0	0	0	0	0	0	0	1	1
Emulate older NF90	0	0	0	0	1	0	0	1	9

Always 1

The VXM can be set back to factory defaults with the "**setD0**" command. This command will erase all programs, modes, and motor settings.

setD0 Set VXM back to factory defaults
CAUTION: All programs, settings, motor selections will be erased

NOTE: The VXM will send "**ËN**" to the host indicating memory has been erased

There is three special commands that get additional status from the VXM

getD0 Gets the VXM's firmware version in the format X.XX

getD1 Gets the VXM's firmware date code in the format XX-XX-XX (month,day,year)

getD2 Returns 2 if system is a single VXM, returns 4 if VXM is a Master with a detected Slave.

G Enable On-Line mode with echo off, **G**rouping a <cr> (carriage return) with the "**^**", "**:**", "W", "O" character responses.

This command has the functionality of the "F" command except a <cr> will be appended to: The "^" (Ready prompt) after finishing a program; the "R" response to a "V" command; an "O" response to Limit Switch Over-travel; the ":" response to a "H" command; and the "W" when a "U6" command is executed.

This command would be beneficial when the VXM is used with a Serial to IEEE488 converter. The terminator on the Converter should be set to "CR".

NOTE: A <cr> will not be added to the "**B**", "**b**" or "**J**" response to a "**V**" command.

See Also
E, F

Appendix P

VXM Comparison to NF90/ VP9000

Motor commands:

ImMx Set steps to incremental Index motor CW (positive), m= motor# (1,2,3,4), x=1 to 16,777,215

ImM-x Set steps to incremental Index motor CCW (negative), m= motor# (1,2,3,4), x=1 to 16,777,215

IAmMx Set Absolute Index distance, m=motor# (1,2,3,4), x= ±1 to ±16,777,215 steps

IAmM0 Index motor to Absolute zero position, m=motor# (1,2,3,4)

IAmM-0 Zero motor position for motor# m, m= 1,2,3,4

ImM0 Index motor until positive limit is encountered, m=motor# (1,2,3,4)

ImM-0 Index motor until negative limit is encountered, m=motor# (1,2,3,4)

SmMx Set Speed of motor (70% power), m= motor# (1,2,3,4), x=1 to 6000 steps/sec. (SAmMx is 100% power)

SmM-x Read and assign analog input value to motor m speed (70% power), x=speed range (**SAmM-x** is 100% power)

AmMx Acceleration/deceleration, m= motor# (1,2,3,4), x=1 to 127.

Program management commands:

PMx Select Program number x, x= 0 to 4

PM-x Select and clear all commands from Program number x, x= 0 to 4

PM Request the number of the current Program

PMAx Program Associate program x in Master to program x in Slave (Linked VXMs start the same time) -x or x =255 is disable

PMA Request the current program associate number

Special looping/branching commands:

L0 Loop continually from the beginning or Loop-to-marker of the current program

LM0 Sets the Loop-to-marker at the current location in the program

LM-0 Resets the Loop-to-marker to the beginning of the current program

Lx Loop from beginning or Loop-to-marker x-1 times (x=2 to 65,535), when the loop reaches its last count the non-loop command directly preceding will be ignored

L-x Loop from beginning or Loop-to-marker x-1 times, alternating direction of motor 1, when the loop reaches its last count the non-loop command directly preceding will be ignored

LAx Loop Always from beginning or Loop-to-marker x-1 times (x=2 to 65,535)

LA-x Loop Always from beginning or Loop-to-marker x-1 times, alternating direction of motor 1

LM-2 Loop once from beginning or Loop-to-marker reversing index direction of motor 2

LM-3 Loop once from beginning or Loop-to-marker reversing index direction of motor 1 and motor 2

Jx Jump to the beginning of program number x, x= 0 to 4

JMx Jump to the beginning of program number x and come back for More after program x ends, x= 0 to 4

JM-x Similar to JMx except automatically moves back from absolute indexes after program x ends: For pick-and-place within matrix looping patterns

Legend:

New Commands for VXM not available on VP9000 or NF90

Different input/output/range/additional values from VP9000

Different command/ function for NF90 mode

Pausing and input output commands:

Px Pause x tenths of a second, (x=0 to 65,5350) tenths of a millisecond when x is negative

PAx Pause x tenths of a second (x=0 to 65,535, 10 µsec pause when x=0) Altering output 1 high for duration of the pause, tenths of a millisecond when x is negative

U0 Wait for a "low" on user input 1

U1 Wait for a low on user input 1, holding user output 1 high while waiting

U2 Enable Jog mode while waiting for an input

U3 Disable Jog mode while waiting for an input

U4 User output 1 "low" (reset state)

U5 User output 1 high

U6 Send "W" to host and wait for a "G" to continue

U7 Start of Continuous Index with 10 µsec pulse on output 2

U77 Start of Continuous Index with no output

U8 Start of Continuous Index sending "@" to the host

U9 End of Continuous Index with autodecel to stop

U91 End of Continuous Index with auto-generate a deceleration Index as next command

U92 End of Continuous Index using next Index for deceleration to stop

U99 End of Continuous Index with instantaneous stop

U11 Skip next command if input 1 is high

U12 Skip next command if input 2 is high

U13 Wait for a front panel button to jump to a program or continue: "Motor 1 Jog -" button to jump to program #1, "Motor 1 Jog +" button to jump to program #2, "Run" button to proceed in current program.

U14 User output 2 low (reset state)

U15 User output 2 high

U16 Optional User output 3 low (reset state)

U17 Optional User output 3 high

U18 Optional User output 4 low (reset state)

U19 Optional User output 4 high

U21 Skip next command if input 1 is low

U22 Skip next command if input 2 is low

U23 Wait for a front panel button to jump to a program and come back, or continue: "Motor 1 Jog -" button to jump and return to program #1, "Motor 1 Jog +" button to jump and return to program #2, "Run" button to proceed in current program

U30 Wait for a low to high transition on user input 1

U31 Wait for a low to high transition on user input 1, holding user output 1 high while waiting

U32 Wait for "Motor 1 Jog -" button to be pressed on front panel with debouncing

U33 Wait for "Motor 1 Jog +" button to be pressed on front panel with debouncing

U50 Wait for a low and high on user input 1 with debouncing for a mechanical push-button switch

U51 Wait for a low and high on user input 1 with debouncing for a mechanical push-button switch, holding user output 1 high while waiting

U90 Wait for a low to high on the Run button or connection I/O,4 with debouncing for a mechanical push-button switch

Legend:

New Commands for VXM not available on VP9000 or NF90

Different input/output/range/additional values from VP9000

Different command/ function for NF90 mode

Operation commands:

Q Quit On-Line mode (return to Local mode)
R Run currently selected program
N Null (zero) motors 1,2,3,4 absolute position registers
K Kill operation/program in progress and reset user outputs
C Clear all commands from currently selected program
D Decelerate to a stop (interrupts current index/ program in progress)
E Enable On-Line mode with echo "on"
F Enable On-Line mode with echo "off"
G Enable On-Line mode with echo off Grouping a <cr> with "^", ":", "W", "O" responses; Also Go after waiting or holding
H Put Controller on Hold (stop after each command and wait for go)
! Record motor positions for later recall with “x”,”y” commands
res Software reset controller
del Delete last command

***Status request commands*:**

V Verify Controller's status, VXM sends "B" to host if busy, "R" if ready, "J" if in the Jog/slew mode, or “b” if Jog/slewing
X Send current position of motor 1 to host (Motor can be in motion)
Y Send current position of motor 2 to host (Motor can be in motion)
Z Send current position of motor 3 to host (Motor must be stationary)
T Send current position of motor 4 to host (Motor must be stationary)
x Send last 4 positions of motor 1 to host that were captured by the “!” command or Input 4 trigger
y Send last 4 positions of motor 2 to host that were captured by the “!” command or Input 4 trigger
M Request Memory available for currently selected program
Request the number of the currently selected motor
***** Request the position when the last motor started decelerating (shows position when "D" command or Stop/User input 4 used)
? Read state of limit switch inputs for motor 1 and 2 (8 b i t binary value)
~ Read state of User Inputs, Motor 1 and 2 Jog Inputs (8 bit binary value) Read
$ state of User Outputs (8 bit binary value)
@ Read user analog input value
B Read Backlash compensation setting
O Read Indicate limit switch setting
getDx Read mode/version
getDA Read Joystick Deadband setting
getjmM Read first range Jog Speed for motor m. **getjAmM** for Joystick range setting
getJmM Read second range Jog Speed for motor m. **getJAmM** for Joystick range setting
getLmM Read mode of limits for motor m
getMmM Read motor type/size selected for axis m
getPmM Read “Pulse Every x # Steps” value for axis m
getPA Read Pulse width used by setPmMx and U7
getI Read operating mode of user inputs
lst List current program to host (ASCII text)

Legend:

New Commands for VXM not available on VP9000 or NF90
Different input/output/range/additional values from VP9000
Different command/ function for NF90 mode

Commands for two controls connected by VXM bus:

(i3,i1... Combine Index commands to run simultaneously on two VXM controllers connected by VXM bus

) Send data to Slave through Master

***Jog mode commands*:**

D Read motor position (Digitize)

Special function and setup commands:

Bx Backlash compensation, on when x=1, off when x=0

Ox Indicate limit switch Over-travel to host, off when x=0, VXM sends "O" when x=1 and hit limit, x=3 program stops too

setDMx Set VXM/VP9000 or NF90 emulation modes, and other operating parameters

setDAx Set Joystick Deadband value

setjmM Set first range Jog Speed for motor m. **setjAmM** for Joystick range setting

setJmM Set second range Jog Speed for motor m. **setJAmM** for Joystick range setting

setLmMx Set limit switch mode for axis m

setMmMx Set axis m for motor type/size x. Also sets default (jog/joystick) motor power to 70%. **setMAmMx** is 100% power

setPmMx Set "Pulse Every x # Steps" on output 2 for axis m

setPAx Set Pulse width used by setPmMx and U7, x=1 to 255

setIx Set operating mode of inputs

setBx Set RS-232 Baud rate (9=9600, 19=19200, 38=38400)

Memory save commands

rsm Run save memory (saves setup/ program values to nonvolatile memory)

Legend:

New Commands for VXM not available on VP9000 or NF90

Different input/output/range/additional values from VP9000

Different command/ function for NF90 mode

NF90 emulation mode:

ImM0 Index motor m to absolute zero position

L-0 Sets the Loop-to-marker at the current location in the program

U2 Disable user output when pausing

U3 Enable output when pausing (reset state)

Acceleration/deceleration values will be internally doubled to match NF90's 2x ramp rate

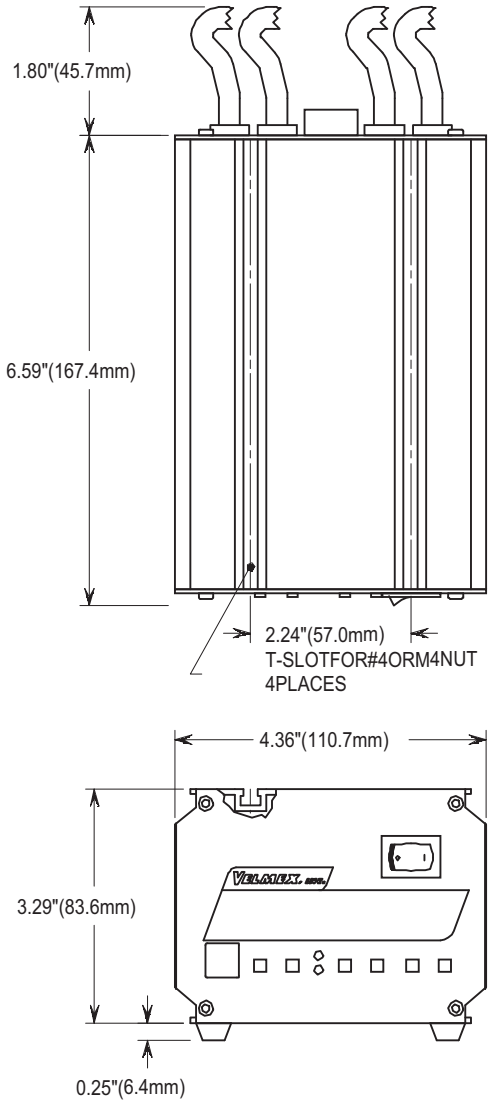
VP9000 Commands not supported by VXM: **U10, U40, U41, U60, U61, U70, U71, U72, U73, { }, %, &**

Feature	VXM
Addressable Axes/ Control	1,2,3*,4* * By 2nd VXM linked with VXM bus
Motor Compatibility	Size 17 to size 34 (0.4 to 4.7 amp)
Motor output torque as percent of NF90 (M091 Motor)	185% @ 0.2 rev/sec 200% @ 5 rev/sec 340% @ 10 rev/sec
Steps/ revolution	400
Speed Range (steps/sec)	1 to 6000
Program Storage (memory type)	5 (RAM/ FLASH)
Range for user “Pulse Every x # Steps”	0 to 32,767
User Inputs	Run (Active Low) In 1** (Active Low) In 2** (Multifunction) In 3** (Multifunction) In 4** (Stop Interrupt) In A** (Analog)
User Outputs	Out 1** Out 2** Optional Out 3** Optional Out 4** +5VDC** ** x2 for Linked VXMs
Other Inputs	AC/DC Power Limit Switch (Optically Isolated)
Other Outputs	Motor (6 wire Unipolar)
User Interfaces	RS-232 (Tx, Rx, Gnd) Run,Stop, and Jog Keys Opt. Speed Pot. Analog Joystick Remote Jog Opt. Program Sel. Switch
RS-232 Configuration	8 Data, No Parity, 1 Stop
Default Baud Rate	9600
Maximum Baud Rate	38,400

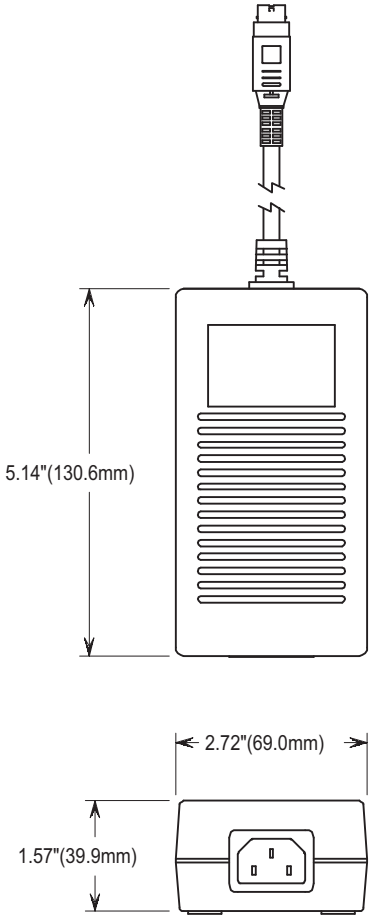
Appendix Q

Outline Dimensions

VXM

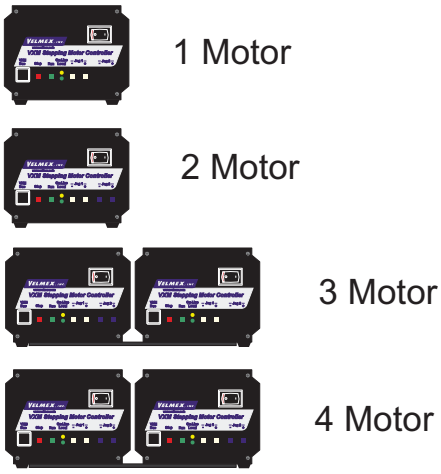


Power Supply



Appendix R

Model Configurations



For OEM applications contact Velmex Sales/ Engineering department to find out more about how we can accommodate your special requirements.

Possible Options:

1. Half “U” enclosure
2. Din rail mountable version
3. Din rail power supply
4. Lower voltage or battery operation
5. Integrated input device. (See website for options)
6. Custom programming
7. Special cables or connectors
8. Higher power or lower power
9. Synchronized motor operation



Rack Panel 1 Motor



Rack Panel 2 Motor



Rack Panel 3 Motor



Rack Panel 4 Motor

Appendix S

Pick and Place Using JM-x*

There are applications that require moving objects from a common pick location to an array of placement locations (or vice of versa.) One way to accomplish this would be to write individual moves to every array position followed by a move to the absolute pick position. The drawback to this approach is the unwieldy quantity of indexes to write if the array/matrix pattern is very large. The solution is to use the **JM-***x* command which is specifically designed for pick and place applications.

The **JM-***x* command is similar to the **JM***x* except the **JM-***x* automatically moves back from absolute indexes after program *x* ends.

The **JM-***x* is not suitable for any use other than pick-and-place within matrix looping patterns.

JM-*x* **J**ump to the beginning of program number x, come back for More after program *x* ends, and automatically move back from absolute indexes that were in program *x* (= 0 to 4).

Program number *x* will temporarily be the current program, all commands will be executed starting from the first one that was previously entered into program *x*.

The VXM will record motor 1 and 2 absolute indexes while in program *x*. When program *x* ends, the VXM will look ahead to the next incremental index, combining this index with the return distance of the recorded absolute index of the same motor. The other motor index recorded will also be moved back it's recorded distance.

1. The motor **reverse-direction-flags** set by "**L-x**", "**LM-2**", and "**LM-3**" looping commands will be **disabled while in program *x*.**

2. **Do not use absolute indexes for motor 3 or 4 in program *x*,** use only incremental indexes for motor 3 or 4.

3. Any **absolute indexes** encountered **on return from program *x*** will **clear** all **recorded return distances** that were saved in program *x*

Memory usage = **2 bytes**

The following examples are based on loops to produce multiple moves. By modifying the loop values, the number of moves per row and the number of rows can be changed.

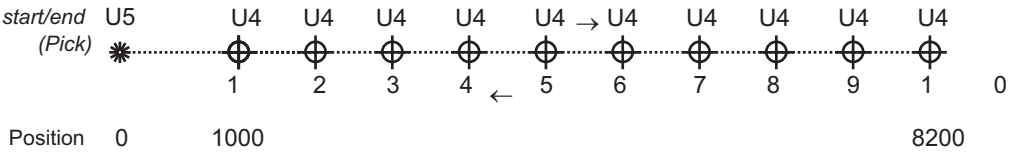
***NEW COMMAND: available only on VXM firmware versions 1.20 & up**

Example #12	Motors run	Function
Pick and Place	1	Pick from common point and place in row

```
;This Pick and  Place example moves one axis
;Placing parts 10x in one row
;The common pick/start  location is set to zero absolute
;Output 1 is used to signal  an external gripper
;Gripperis used to move parts from Pick to Place location

E
                ;Program 1 is subroutine to go to pick location
PM-1            ;Select and clear program 1
IA1M0           ;Go to start
U5U5            ;Output 1 on to activate gripper
P2P2            ;Pause.2 sec.to let gripper grab part

                ;Program 0 is main program
PM-0            ;Select and clear program 0
IA1M-0          ;Make this location zero (Start)
U5              ;Output 1 on to  activate ripper
P2              ;Pause .2 sec.  to let gripper grab part
IA1M1000        ;Move from pick location to place 1
LM0             ;Set loop marker  here, loops will branch here
U4              ;Output 1 off to ungrip part
JM-1            ;Jump/return to program 1 to get another part
I1M800          ;Advance to  next  place location
L10             ;Do from loop  marker 10x
U4              ;End with part not gripped
```



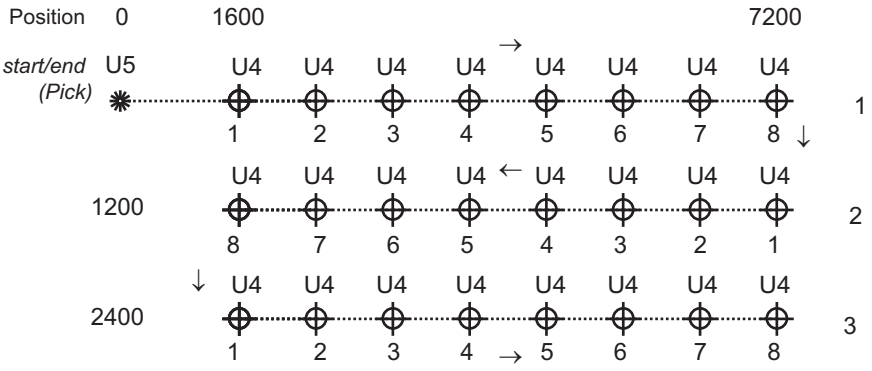
Pick and Place Using JM-x (continued)

Example #13	Motors run	Function
Pick and Place	2	Pick from common point and place in 3 rows

```
;This Pick and Place example moves X,Y
;Placing parts 8x in 3 rows
;The common pick/start location is set to zero absolute
;Output 1 is used to signal anexternal gripper

E
PM-1              ;Select and clear program 1(subroutine)
IA1M0,IA2M0       ;Go  to start
U5                ;Output 1 on (Hold part)

PM-0              ;Select and clear program 0 (main program)
IA1M-0,IA2M-0     ;Make this location zero (Start)
U5                ;Output 1 on (Hold part)
IA1M1600          ;Move from pick location to place 1
LM0               ;Set loop marker here, loops will branch here
U4                ;Output 1 off (Drop part)
JM-1              ;Jump/return to program 1 to get another part
I1M800            ;Advance  to next place location
L8                ;Do from loop  marker 8x
I2M1200           ;Advance Y to  next  row
L-3               ;Do 3 rows
U4                ;End with  Output 1 off (Drop part)
```

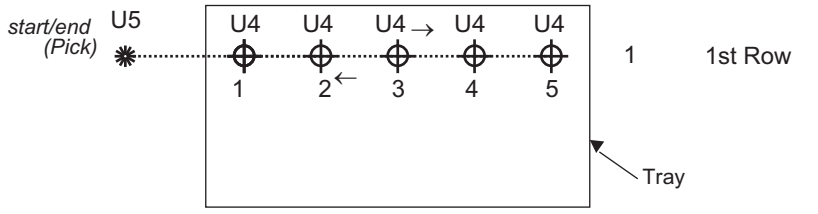


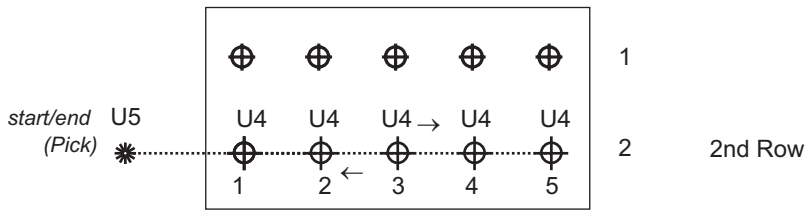
Example #14	Motors run	Function
Pick and Place	2	Pick from common point and place in rows, Y axis moves tray of parts to next row

```
;This Pick and Place example moves X,Y
;Y axis moves tray of parts to advance next row
;Placing parts 5x in 2 rows
;The common pick/start location is set to zero absolute
;Output 1 is used to signal an external gripper
E
PM-2            ;Program 2 for getting part from Pick location
IA1M0           ;Move to home
U5,P1           ;Output 1 on to activate gripper

PM-1            ;Program 1 for placement of parts in row
U4              ;Output 1 off (release gripper)
JM-2            ;Go home and pick up another part
I1M800          ;Advance to next place location
L5              ;Do 5 places per row

PM-0            ;Program 0 is start/setup program
IA1M-0,IA2M-0   ;Make this location zero (Start)
SA1M3000,SA2M3000,A1M6,A2M6   ;Set speeds & accels
U5,P1           ;Output 1 on to activate gripper
LM0             ;Set loop marker, all loops will branch here
IA1M1600        ;Move out to 1st place location
JM1             ;jump to program  1 to do row
IA1M0           ;Clear out absolute saved moves from JM-X
I2M1200         ;move motor 2 over next row
L2              ;Advance another row (2 rows)and repeat
IA2M0,IA1M0     ;Motors home
U4              ;Let go with gripper
```





Pick and Place Using JM-x (continued)

The following example is the similar to example 14 except for a third axis (Z) moves part up from pick location and down at place location, there are 3 rows instead of 2, The program will start again with the Run button.

Example #15	Motors run	Function
Pick and Place	3	Pick from common point and place in rows, Y axis moves tray of parts to next row

```
;This Pick and Place example moves X,Y,Z
;Y axis moves tray of parts to advance next row
;Placing parts 5x in 3 rows
;The common pick/start location is set to zero absolute
;Output 1 is used to signal an external gripper

E
PM-2            ;Program 2 for getting part from Pick location
IA1M0           ;Move to home
I3M400          ;Index Z down to part
U5,P1           ;Output 1 on to grab part, pause too
I3M-400         ;Have part, now Index Z up

PM-1            ;Program 1 for placement of parts in row
I3M400          ;Index Z down to part
U4              ;Let part go
I3M-400         ;Z back up
JM-2            ;Go home and pick up another part
I1M800          ;Advance to next place location
L5              ;Do 5 places per row


PM-3            ;Program 3 is main program after start
IA1M1600        ;Move out to 1st place location
JM1             ;jump to program 1 to do row
IA1M0           ;Clear out absolute saved moves from JM-X
I2M1200         ;move motor 2 over next row
L3              ;Advance another row (3 rows)and repeat
IA2M0           ;Y home
U90             ;Wait for operator to change tray& press Run
L0              ;Start; again (loop forever)


PM-0           ;Program; 0 is start/setup program
IA1M-0,IA2M-0 ;Make this location zero (Start)
S1M3000,S2M3000,A1M6,A2M6  ;Set; speeds & accels
S3M2000,A3M4   ;Set Zspeed &accel
JM2            ;Grab 1st part
J3             ;Jump to Program 3 (it will take over now)
```

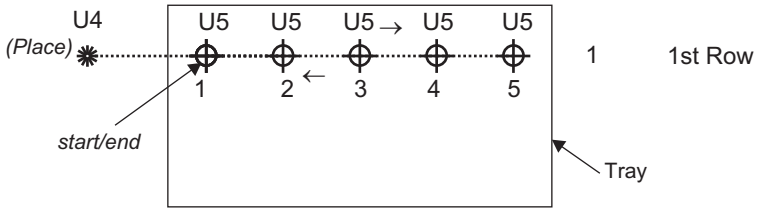
Example #16	Motors run	Function
Pick and Place	3	Pick from rows and place at common location, Y axis moves tray of parts to next row

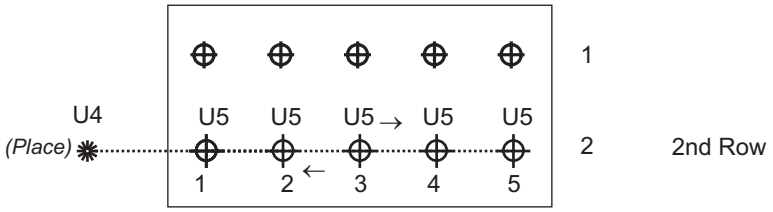
```
;This Pick and Place example moves X,Y,Z
;Parts are Picked 5x in 2 rows & Placed at one point

E
PM-2            ;Program 2 places parts at one location
I3M400          ;Index Z down to part
U5,P1           ;Output 1 on to grab part, pause too
I3M-400         ;Have part, now Index Z up
IA1M-1000       ;Move to place location
I3M400          ;Index Z down to part
U4              ;Let part go
I3M-400         ;Now Index Z up

PM-1            ;Program 1 for getting parts in row
JM-2            ;Grab and drop off another part
I1M800          ;Advance to next place location
L5              ;Do 5 places per row

PM-0            ;Program 0 is start/setup program
IA1M-0,IA2M-0 ;Make this location zero (Start)
S1M3000,S2M3000,A1M6,A2M6   ;Set speeds & accels
S3M2000,A3M4  ;Set Z speed & accel
LM0            ;Set loop marker, all loops will branch here
JM1
IA1M0           ;jump to program 1 to do row
I2M1200         ;Move back to Home (start of row)
L2              ;move motor 2 over next row
IA2M0           ;Do 2 rows
                ;Y home
```





Appendix T

Stand-alone Methods to Select/Alter Program

Sometimes it is necessary for the user to interact directly with the VXM to select a program, stop program, change speed, or change to different routine based on an external input. Below are the four categories and the methods for user interaction with the VXM in standalone applications.

External Selection of Programs
The default program the VXM uses is program #0. By pressing the Run button or a button connected to the Run input on the I/O (see Appendix K) program #0 will be started. If a binary switch is connected, to Inputs 2 and 3, programs #0 to #3 can be selected. See Appendix C and Application Note #AN103 for more information.

External Setting of Speed
The **SmM-x** and **SAmM-x** commands that assign analog input value to motor speed can be used to vary speeds based on an external potentiometer setting. Refer to Appendix I and Application Note #AN102 for more information.

Program Interruption
The Stop button or a button connected to Input 4 normally will stop a program in a controlled manner.
When Stop is pressed, the VXM will decelerate a running motor to a stop, and end the program. The VXM can be optionally set to run program #4 after stopping. See Appendix C and Appendix K for more information and other stop settings.

⚠ CAUTION:

THE STOP BUTTON/INPUT 4 IS NOT AN EMERGENCY STOP. FOR EMERGENCY STOP, POWER TO VXM SHOULD BE DISCONNECTED.

Conditional Branching
Branching/jumping to a specific part of a program, when an external event occurs, is a way to produce an alternate function from a user input. There are three different ways the VXM can perform conditional branching:

1. **U13** and **U23** commands wait for a front panel button or I/O input to jump to a program or continue: "Motor 1 Jog -" button (I/O,10) will jump to program #1; "Motor 1 Jog +" button (I/O,11) will jump to program #2; the "Run" button (I/O,4) will continue in the current program. See Appendix B for more information.

2. Input 3 (I/O,7) is a special interrupt of a wait for input command. For example, the **U0** command will wait for a low on input 1 (I/O,5). When the VXM is waiting for Input 1 and Input 3 (I/O7) is pulled low the program will immediately branch to program #3. See Appendix C for more information about enabling/disabling this feature.

3. The **U11/U21** and **U12/U22** commands to skip-next-command-on-input-high/low are the most versatile conditional branch commands. These commands allow jumps or speed changes to occur based on the state of input 1 or 2. See the following page for more information about these commands.

Skip Next Command If Input High*/Low**

U11 Skip next command if Input 1 (I/O,5) is high.
Memory usage = **2 bytes**.

U21 Skip next command if Input 1 (I/O,5) is low.
Memory usage = **2 bytes**.

U12 Skip next command if Input 2 (I/O,6) is high.
Memory usage = **2 bytes**.

U22 Skip next command if Input 2 (I/O,6) is low.
Memory usage = **2 bytes**.

Example #17	Motors run	Function
Change Speed	1	Speed will be set to 2000 if input 1 is low, or 4000 if input 1 is high

```
E
PM-0      ;Select and clear Program 0
U11       ;Skip next command if Input 1 high
S1M2000   ;Set Speed to 2000
U21       ;Skip next command if Input 1 low
S1M4000   ;Set Speed to 4000
I1M8000   ;Index out (Forward)
I1M-8000  ;Index back (Reverse)
```

Example #18	Motors run	Function Changes
Change Program	1	to program 1 if input 2 is low. Program 1 runs motor to zero

```
E
PM-1          ;Select and clear Program 1
S1M4000       ;Speed to 4000
IA1M0         ;Go to Home position

PM-0          ;Select and clear Program 0
U12           ;Skip nextcommand if Input 2 high
J1            ;Jump to Program 1
S1M2000       ;Speed to 2000
I1M8000       ;Index
```

***NEW COMMANDS: available only on VXM firmware versions 1.22 & up,**
**** available on VXM firmware versions 1.30 & up**

Warranty

Stepping Motor Controllers manufactured by Velmex are warranted to be free from defects for a period of two (2) years on all parts. Velmex's obligation under this warranty does not apply to defects due, directly or indirectly, to misuse, abuse, negligence, accidents, or unauthorized repairs, alterations, or cables/connectors that require replacement due to wear. Claims must be authorized, and a return authorization number issued before a product can be returned.
The warranty does not cover items which are not manufactured or constructed by Velmex, Inc. These components are warranted by their respective manufacturer. Under the above warranty, Velmex will, at its option, either repair or replace a nonconforming or defective product.

The above warranty is the only warranty authorized by Velmex. Velmex shall in no event be responsible for any loss of business or profits, downtime or delay, labor, repair, or material costs, injury to person or property or any similar or dissimilar incidental or consequential loss or damage incurred by purchaser, even if Velmex has been advised of the possibility of such losses or damages.

Inasmuch as Velmex does not undertake to evaluate the suitability of any Velmex product for any particular application, the purchaser is expected to understand the operational characteristics of the product, as suggested in documentation supplied by Velmex, and to assess the suitability of Velmex products for this application.

This limited warranty give you specific legal rights which vary from State to State.

Record Controller and Motor information here for future reference:

Model#: VXM-__

Serial #: _____

Motor #1 _____ **Motor #2** _____

Motor #3 _____ **Motor #4** _____

Notes:



Contact Information

By Phone: 585-657-6151 and 800-642-6446
By Fax: 585-657-6153
Email: info@velmex.com
On the Internet: www.velmex.com and www.bislide.com
By mail: Velmex, Inc.
7550 State Route 5 & 2 0
Bloomfield, NY 14469 USA



Copyright 2002 Velmex Inc. All rights reserved. Velmex, the Velmex logo, UniSlide, and BiSlide are trademarks of Velmex Inc. All other trademarks are the property of their respective owners.

Document # VXM-UM-E5 12-29-04

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP