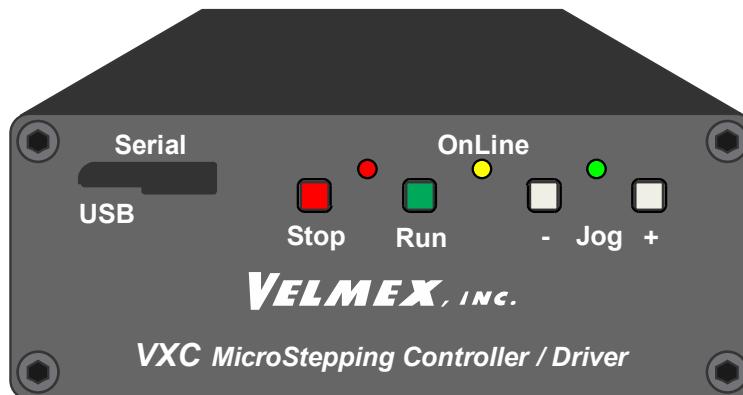


VXC MICROSTEPPING MOTOR CONTROLLER / DRIVER

User's Manual



*“Positioning Systems for Science and Industry”
Since 1967*

Table of Contents

.....	1
Precautions	12
High Integration Stepping Motor Controller/Drive	14
Model Numbers	14
Function	15
External Features	16
Setup	17
Configuring VXC With Device ID #.....	18
Device ID for Limit, Home/Stall Settings.....	19
Jog Mode.....	19
Optional Joysticks	19
Serial Communication.....	20
Units & Direction.....	21
Units for Velmex Positioners	22
Unit Conversion.....	22
Other Formulas	22
Motion Commands	23
The Difference Between Incremental and Absolute Indexes	23
Incremental: I1M1000,.....	23
Absolute: IA1M2000,.....	23
Sending Commands to the VXC	23
Incremental Index (Move an Increment).....	24
Absolute Index (Move to Position)	25
Index to Absolute Zero (Move to Zero).....	26
Zero Motor Position.....	26
Move to Positive to Home/Limit.....	27
Move to Negative to Home/Limit	27
Speed	28
How to Determine Maximum Speed	29
Smart Speeds	29
Acceleration	30
How to Determine Maximum Acceleration.....	30
Smart Acceleration.....	30
Program Coding Shortcut.....	31

Interactive Mode.....	32
Interactive Examples.....	33
Standalone Mode.....	34
Standalone Examples.....	35
Command Summary (Common & Advanced).....	36
Operation Commands.....	36
Editing/Debugging Tools.....	36
Setup Commands.....	36
Status Request Commands.....	37
Motion Commands.....	37
Program Management Commands.....	38
Looping & Branching Commands.....	38
Pausing, Input/Output Commands.....	38
Program Management Commands.....	40
Editing/Debugging Tools.....	42
Program Examples.....	43
Example 1 Enable On-Line Mode Echo on.....	43
Example 2 Single Axis Incremental Index Move 1 st Motor 400 Steps.....	43
Example 3 Single Axis Incremental Index Move 1 st Motor 400 Steps.....	43
Example 4 Clear all Commands from Current Program & Zero Motor Position.....	43
Example 5 Clear Previous & Index Move 2nd Motor 600 Steps.....	44
Example 6 Index Motor 1 Both Directions (Auto Reverse).....	44
Example 7 One-Axis Index Pausing 0.5 Sec, Loop, Return to Start.....	44
Example 8 One-Axis Index Using Index Absolute Zero to Return to Start.....	44
Example 9 Two-Axis Out/Down, Wait for Run button, Rapid Return to Start.....	44
Example 10 Two-Axis Raster Scan With 0.1 Pulse at Each Stop & Return to Start.....	45
Example 11 Two-Axis With Two Different Raster Scans & Return to Start.....	45
Example 12 Three-Axis XY Matrix, Z up/dn at Each Position, & Return to Start.....	46
Example 13 Two-Axis Rectangle, Wait Input 1 at Each Corner, Loop Forever.....	46
Example 14 One-Axis Index/10ms Pulse, Looping, Using Smart Acel & Speed.....	46
Looping/Branching Commands.....	47
Loop.....	47
Jump.....	48
Pause.....	49
Ux Commands.....	49

Wait for Input.....	49
Input 1.....	49
Run Low/High.....	51
Conditional Branch on Input.....	51
Wait for Front Panel Button.....	51
Output.....	52
Output 1.....	52
Output 2.....	52
Output 3.....	52
Output 4.....	52
Outputs on Axes 2,3,4.....	53
Axis 2 Outputs.....	53
Axis 3 Outputs.....	53
Axis 4 Outputs.....	53
Jog While Waiting.....	54
Programmable Serial Prompt.....	54
Continuous Indexing.....	54
Operation Commands.....	56
Online/Off-Line.....	56
Clear Previous.....	56
Run/Start.....	56
Interrupt Motion.....	56
Status Request Commands.....	57
Help Menu.....	57
Help Menu in Jog Mode.....	57
Help Menu in OnLine Mode.....	57
Motor Position.....	57
Verify Status.....	58
Capture Position.....	58
Retrieve Captured Positions.....	58
Position When Decelerated.....	59
Read Analog Value.....	59
List All Settings.....	60
Backlash Compensation.....	61
Motor Type.....	61

Limit Mode	61
Firmware Status Request	61
Jog/Joystick Settings	61
Digital Jog	61
Analog Joystick	61
Pulse Output Settings.....	62
User Input Mode	62
Motor Jog Function	62
Troubleshooting.....	63
Technical Specifications	65
Environmental Requirements	65
Model VXC-1	65
Function	65
Physical.....	65
Cabling.....	65
Electrical Requirements	65
I/O	65
Serial Ports	65
Model VXC-2	66
Function	66
Physical.....	66
Cabling, Electrical Requirements, I/O, Serial Ports	66
Model VXC-3	66
Function	66
Physical.....	66
Cabling, Electrical Requirements, I/O, Serial Ports	66
Model VXC-4	66
Function	66
Physical.....	66
Cabling, Electrical Requirements, I/O, Serial Ports	66
Power Supply	67
Function	67
Physical.....	67
Electrical Requirements	67
Warranty	67

Appendix A (Motor Settings)	68
Setting Motor Type	68
Non-Standard Motors	69
Appendix B (Limits/Home/Stall Detect).....	70
Limit Switch Inputs.....	70
Home/Stall Detect Input	71
Limits, Home, Stall Detect Function Interaction	72
Device Compatibility with Limits, Home, Stall Detect	72
Using a Limit for Home Reference	73
Programming sequence for homing to a limit switch.....	73
Example 15 Move negative into limit switch, then back 400 steps, & zero position	73
Reasons to Use a Dedicated Home Switch	73
Using a Home Switch on Home Input	74
Homing to a home switch on rotary tables without limit switches, and when home switch is at limit end on devices with limit switches	74
Example 16 Move negative into home switch and zero position Example	74
Example 17 Move positive into home switch and zero position 1000 steps away	75
Programming sequence for homing to a centrally located home switch with limit switches.....	75
Example 18 Move negative into limit switch, then + to home switch, and zero position.....	75
Appendix C (Referencing/Feedback)	76
More Feedback & Precision	76
Limit or Home Switch for Initial Reference.....	76
Example 19 Typical Homing Routine (speed 500, move to +Limit, move back 200, zero position)76	
Backlash Compensation to Improve Accuracy.....	76
Backlash Compensation Set Examples.....	77
Indicate Limit Switch Encounter	77
Stall Detect Option to Verify Movement	77
Appendix D (Digital Jog)	78
Advanced Digital Jog Mode.....	78
Optional Digital Joystick.....	79
Jog Function Settings	80
Custom Jog Distances.....	81
Appendix E (Analog/Jog)	82
Analog Input.....	82
Analog Joystick Option	83

Analog Joystick Connection	85
Appendix F (Faults)	86
Fault Checking.....	86
Immediate Checks.....	86
Level 1 Faults.....	87
Level 2 Faults.....	87
Level 3 Faults.....	88
Level 3 Faults by Common Flash Code.....	89
Level 3 Faults Checked at Power-up Only.....	90
Index of All Faults.....	90
Appendix G (Serial Ports)	91
Serial Port Connections.....	91
USB Port	91
How to Find COM Port VXC is on in Microsoft Windows.....	91
How to Test USB Cable for a Reliable Connection.....	92
RS-422 Port	92
RS-422 Adapters.....	93
Appendix H (Modes)	94
Controller Mode.....	94
Appendix I (Inputs).....	96
Multifunction User Inputs.....	96
Appendix J (Dynamically Read Position)	98
Getting Motor Position When Moving	98
The Automatic Deceleration Capture	98
The Triggered Position Capture	98
Retrieve Captured Positions	98
Appendix K (Output Triggers)	99
Producing Trigger Outputs.....	99
Index, Stop, Output.....	99
Example 20 Index Motor one 400 Steps, Pause/Output	99
Continuous Indexing	99
U7 Continuous Indexing Command Examples.....	100
Pulse Every “n” Steps.....	101
Appendix L (Complex Motion)	103
Complex Profiles/Coordinated Motion.....	103

Complex Profiles	103
Coordinated Motion	103
Master/Slave Associated Programs	104
Appendix M (Math).....	105
Math Capability.....	105
Math Expressions.....	106
“if” Conditional Statement.....	107
“if” Expressions	107
“end” Statement	107
Setting & Proportioning Speed to Analog Input	108
Producing Custom Accelerations / Decelerations	109
Calculating Position in Pick & Place Applications.....	109
Self-determining Center of System Travel	110
Self-calculating Ratios of Absolute Position.....	110
Appendix N (Input Select Programs/Speed)	111
Stand-alone Methods to Select Program.....	111
External Selection of Programs.....	111
External Setting of Speed.....	111
Program Interruption.....	111
Conditional Branching.....	111
Skip Next Command If Input High/Low.....	112
Appendix O (Setting Baud Rate)	113
Baud Rate Setting Methods	113
Setting Baud with setBx Command.....	113
Setting Baud at Front Panel	113
Appendix P (VXC Versus VXM)	114
VXC Compatibility With VXM	114
VXC New Features Compared to Previous VXM	114
VXC versus VXM Settings Differences.....	115
VXC Versus VXM Command Differences.....	115
VXC New Commands.....	116
VXC New set/gets.....	116
VXC Different Values	116
Appendix Q (Microstepping).....	117
Dynamic & Static Microstepping	117

Appendix R (Dimensions).....	118
VXC Dimensions	118
VXC Models	118
Rack Mounting.....	119
Cleat Mounting	120
Power Supply Dimensions.....	121
Appendix S (I/O Specifications).....	122
I/O Electrical Specifications.....	122
Optional Auxiliary I/O Breakout Module	123
Appendix T (Motor Torque)	124
Motor Torque Curves.....	124
VML113-1.2-S (28CM013) Motor.....	124
VML173-2.5-D (42CM06-SZ) Motor	125
VML231-3.0-D (57CM06) Motor	127
VML232-4.0-D (57CM13) Motor	129
VML233-5.0-D (57CM22C) Motor	131
VML341-4.0-D (86M35) Motor	133
VMN231-4.2-D (ST5918X3008) Motor.....	135
VMN232-4.2-D (ST5918S3008-B).....	137
Cable Length Versus Torque	139
Example VM341-4.0-D-F, 2500 steps/sec, 100% power (S+2500), “TF with CLC” & 50 ft cable ..	139
Table 1 When VXC Will not be Factory Configured	18
Table 2 Settings for Limits, Home/Stall.....	19
Table 3 Velmex Product Advance per Turn & Advance per Step.....	22
Table 4 Motor Type Setting	68
Table 5 Motor Type Setting for Legacy 6 Wire Motors (Half Winding)	68
Table 6 “get Motor Type” Examples	69
Table 7 Limit Input Settings	70
Table 8 Home/Stall Input Settings	71
Table 9 Limit, Home, Stall Settings Interaction.....	72
Table 10 Device Combability with Limits, Home, Stall.....	72
Table 11 Jog Function Settings.....	80
Table 12 Analog Joystick Speed Ranges	84
Table 13 Joystick Y Cable Connections.....	85
Table 14 Index Listing of Faults.....	90
Table 15 Controller Mode Settings	94
Table 16 User Inputs Mode Settings.....	96
Table 17 Program Select Truth Table.....	96

Table 18 Program Associate Values	104
Table 19 Math Expressions	106
Table 20 "If" Expressions	107
Table 21 Analog to Speed Formulas	108
Table 22 Cable Lengths Torque Factors	139
Figure 1 VXC-2 Functional Diagram	15
Figure 2 Interactive Mode.....	32
Figure 3 Standalone Mode.....	34
Figure 4 Limit Switch Input Hardware	71
Figure 5 Limits Connection.....	71
Figure 6 Home Switch Input Hardware.....	71
Figure 7 Home Switch Active Area.....	74
Figure 8 Digital Joystick.....	79
Figure 9 Analog Input Connection to External Potentiometer	82
Figure 10 Analog Joystick	85
Figure 11 Wiggle Test on USB Cable	92
Figure 12 RS-422 Connection.....	92
Figure 13 RS-422 to Terminal Block Breakout	93
Figure 14 RS-422 to RS-232 Adapter/Converter ¹	93
Figure 15 VXC Dimensional Drawing.....	118
Figure 16 VXC Models	118
Figure 17 Rack Shelf Mounting Dimensions	119
Figure 18 Cleat Mounting Dimensions.....	120
Figure 19 Standard 24V Power Supply Dimensional Drawing	121
Figure 20 Optional 28V Power Supply Dimensional Drawing.....	121
Figure 21 Inputs and Outputs Circuit Diagram	122
Figure 22 Auxiliary I/O Breakout Module	123
Figure 23 VML113-1.2-S (28CM013) Motor Specifications	124
Figure 24 VML173-2.5-D (42CM06-SZ) Motor Specifications.....	126
Figure 25 VML231-3.0-D (57CM06-SZ) Motor Specifications.....	128
Figure 26 VML232-4.0-D (57CM13-SZ) Motor Specifications.....	130
Figure 27 VML233-5.0-D (57CM22C-SZ) Motor Specifications.....	132
Figure 28 VML341-4.0-D (86M35-SZ-19) Motor Specifications.....	134
Figure 29 VMN231-4.2-D (ST5918X3008-B) Motor Specifications.....	136
Figure 30 VMN232-4.2-D (ST5918S3008-B) Motor Specifications	138
Example 1 Enable On-Line Mode Echo on	43
Example 2 Single Axis Incremental Index Move 1 st Motor 400 Steps.....	43
Example 3 Single Axis Incremental Index Move 1 st Motor 400 Steps.....	43
Example 4 Clear all Commands from Current Program & Zero Motor Position.....	43
Example 5 Clear Previous & Index Move 2nd Motor 600 Steps.....	44
Example 6 Index Motor 1 Both Directions (Auto Reverse).....	44
Example 7 One-Axis Index Pausing 0.5 Sec, Loop, Return to Start	44
Example 8 One-Axis Index Using Index Absolute Zero to Return to Start	44
Example 9 Two-Axis Out/Down, Wait for Run button, Rapid Return to Start.....	44

Example 10 Two-Axis Raster Scan With 0.1 Pulse at Each Stop & Return to Start.....	45
Example 11 Two-Axis With Two Different Raster Scans & Return to Start	45
Example 12 Three-Axis XY Matrix, Z up/dn at Each Position, & Return to Start	46
Example 13 Two-Axis Rectangle, Wait Input 1 at Each Corner, Loop Forever	46
Example 14 One-Axis Index/10ms Pulse, Looping, Using Smart Acel & Speed	46
Example 15 Move negative into limit switch, then back 400 steps, & zero position	73
Example 16 Move negative into home switch and zero position Example	74
Example 17 Move positive into home switch and zero position 1000 steps away	75
Example 18 Move negative into limit switch, then + to home switch, and zero position.....	75
Example 19 Typical Homing Routine (speed 500, move to +Limit, move back 200, zero position)	76
Example 20 Index Motor one 400 Steps, Pause/Output	99
Example 21 Setting Speed Proportional to Analog Input	108
Example 22 Extra Long Accelerations	109
Example 23 Calculated Pick and Place	109
Example 24 Calculating Center of Travel	110
Example 25 Self-Calculating Ratios of Distance	110
Example 26 Change Speed Using U11 & U21	112
Example 27 Change Program Using U12.....	112
Torque Graph 1 for VML113-1.2-S (28CM013) Motor @ 24V	124
Torque Graph 2 for VML173-2.5-D (42CM06-SZ) Motor @ 24V With Damper.....	125
Torque Graph 3 for VML173-2.5-D (42CM06-SZ) Motor @ 24V	125
Torque Graph 4 for VML173-2.5-D (42CM06-SZ) Motor @ 28V, 100% Power, With/Without Damper..	126
Torque Graph 5 for VML231-3.0-D (57CM06) Motor @ 24V With Damper.....	127
Torque Graph 6 for VML231-3.0-D (57CM06) Motor @ 24V	127
Torque Graph 7 for VML231-3.0-D (57CM06) Motor @ 28V, 100% Power, With/Without Damper.....	128
Torque Graph 8 for VML232-4.0-D (57CM13) Motor @ 24V With Damper.....	129
Torque Graph 9 for VML232-4.0-D (57CM13) Motor @ 24V	129
Torque Graph 10 for VML232-4.0-D (57CM13) Motor @ 28V, 100% Power, With/Without Damper.....	130
Torque Graph 11 for VML233-5.0-D (57CM22C) Motor @ 24V With Damper	131
Torque Graph 12 for VML233-5.0-D (57CM22C) Motor @ 24V	131
Torque Graph 13 for VML233-5.0-D (57CM22C) Motor @ 28V, 100% Power, With/Without Damper ..	132
Torque Graph 14 for VML341-4.0-D (86M35) Motor @ 24V With Damper.....	133
Torque Graph 15 for VML341-4.0-D (86M35) Motor @ 24V	133
Torque Graph 16 for VML341-4.0-D (86M35) Motor @ 28V, 100% Power, With/Without Damper.....	134
Torque Graph 17 for VMN231-4.2-D (ST5918X3008) Motor @ 24V With Damper	135
Torque Graph 18 for VMN231-4.2-D (ST5918X3008) Motor @ 24V	135
Torque Graph 19 for VMN231-4.2-D (ST5918X3008) Motor @ 28V, 100% Power, With/Without Damper	136
Torque Graph 20 for VMN232-4.2-D (ST5918S3008-B) Motor @ 24V With Damper	137
Torque Graph 21 for VMN232-4.2-D (ST5918S3008-B) Motor @ 24V.....	137
Torque Graph 22 for VMN232-4.2-D (ST5918S3008-B) Motor @ 28V, 100% Power, With/Without Damper	138

Precautions

CAUTION:

Controller and AC power supply should be operating in a well-ventilated area. **DO NOT USE IN A WET, DIRTY, OR EXPLOSIVE ENVIRONMENTS.** The VXC has an IP30 rating, in industrial or outdoor environments repackaging into a higher IP rated enclosure is required. Do not disconnect motor while running. Keep Motor and Limit cables separated if possible. Only operate with designated motor. Do not alter cables in any way without first consulting Velmex

CAUTION:

Mismatched Motor Settings can Damage Motor & Controller

When Connecting Motor Set Controller to the Exact Motor Type Before Operating

THE VXC MUST BE SET TO THE EXACT TYPE MOTOR(S) BEFORE OPERATING. IMPROPER SETTINGS CAN CAUSE SEVERE DAMAGE TO MOTORS AND CONTROLLER.

Use Velmex VXC Utility App to configure VXC before use.

CAUTION:

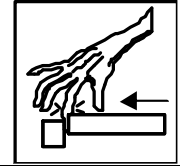
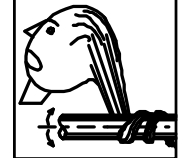


MOTOR(S) GET HOT WHEN RUNNING. Motor(s) must be mounted to a metal surface to dissipate internal heat.

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.)

! CAUTION:

PINCH and WRAP HAZARD. Rotating motors and related actuators create a potential risk of injury, keep fingers, hands, hair, jewelry, and clothing away from all rotating/moving parts.

	Pinch Point. Keep fingers clear during operation.
	Wrap Point. Do Not operate with exposed long hair, jewelry, or loose clothing.

! WARNING:

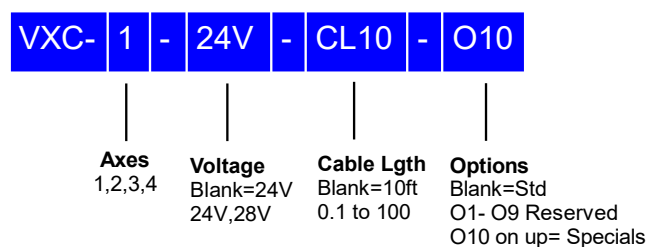
TO REDUCE THE RISK OF ELECTRICAL SHOCK, DO NOT ATTEMPT TO REMOVE COVERS ON POWER SUPPLY OR CONTROLLER. THERE ARE NO USER SERVICEABLE PARTS INSIDE. Any servicing should be done by Velmex qualified service personnel.

High Integration Stepping Motor Controller/Drive

- “Plug-in & Run” with Velmex motor driven products
- Complete with serial interface/Indexer/Bipolar Drive, AC Power Supply, Power cables, Motor cables, and Limit cables included
- 100-240 VAC input external power supply that is UL, CE, CSA, and TUV safety agency compliant
- Compatible with most NEMA size 11 to 34 four or eight wire hybrid step motors
- Voltage modulated PWM drive for low resonance and high motor torque
- Micro-stepping for smoother low speed operation and 10X higher resolution
- Energy saving automatic de-energizing motors at standstill, with settable holding torque and failsafe brake control output
- Interactive control or standalone
- Nonvolatile memory for user stored programs
- Standard USB 2.0 and RS-422/RS-232 serial ports
- 10-bit analog input for external sensor, setting speeds, and for analog joystick control
- Optically isolated limit switch inputs & home input
- Hall sensor interface for position verification (stall detection)
- Multipurpose inputs and outputs
- 5-volt user outputs sink/source 25 mA/25 mA
- Programmable output triggers to signal external devices
- Conditional branching commands
- Looping commands for raster scanning and matrix patterns
- FIFO buffer to capture motor positions on an input trigger
- Math capability for self-calculating custom acceleration profiles, self-centering, and calculating return-to-home in pick-and-place applications
- Coordinated motion to produce angles, arcs, circles, etc.

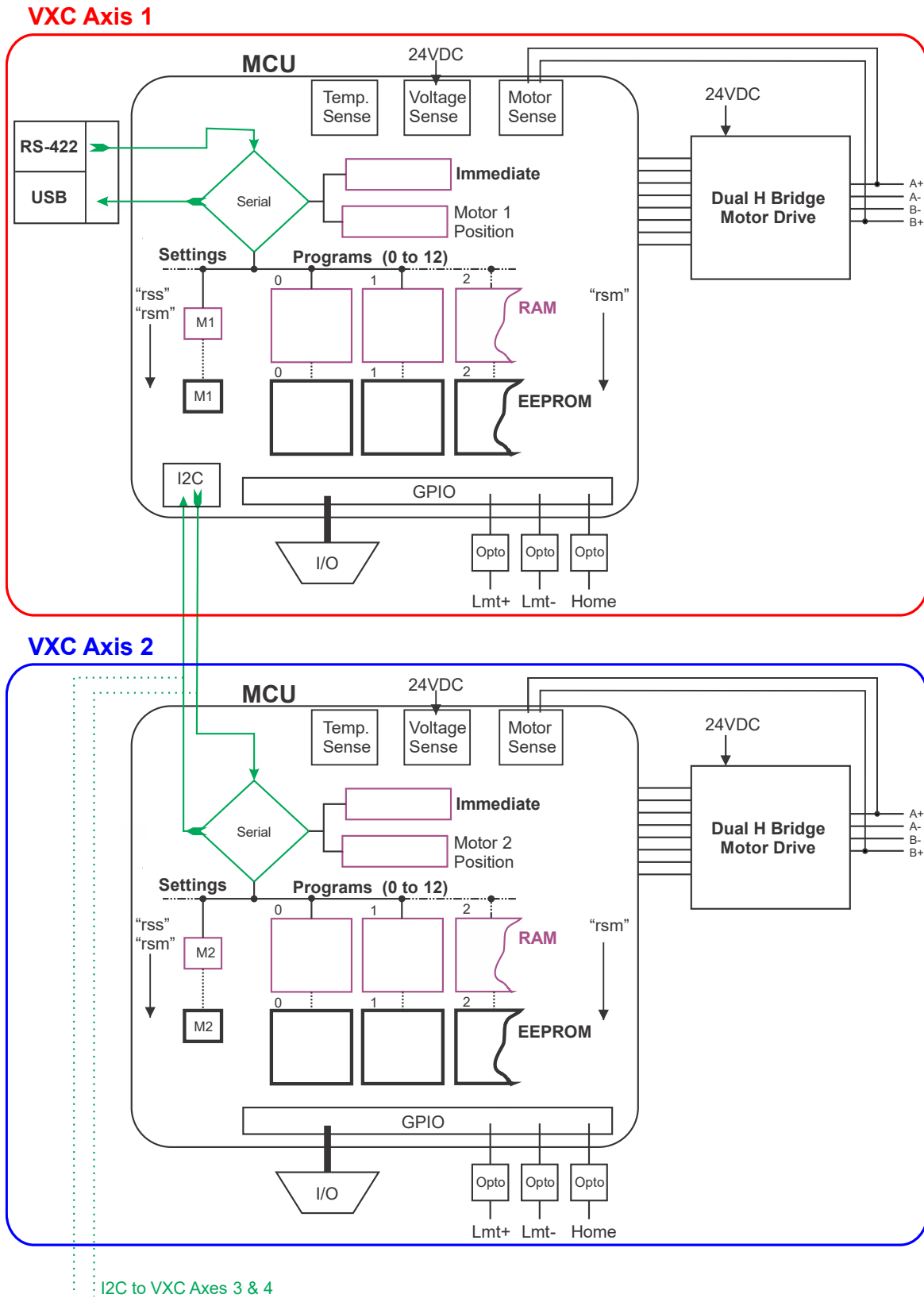
Model Numbers

The basic VXC model numbers are VXC-1, VXC-2, VXC-3, VXC-4 which include a 24V power supply and 10-foot motor and limit cables. Velmex will customize the VXC to your exact requirements. Different voltages, cable lengths, special labeling, optional connectors, custom remote jog controls, custom programming, and open chassis versions are some of the possibilities.



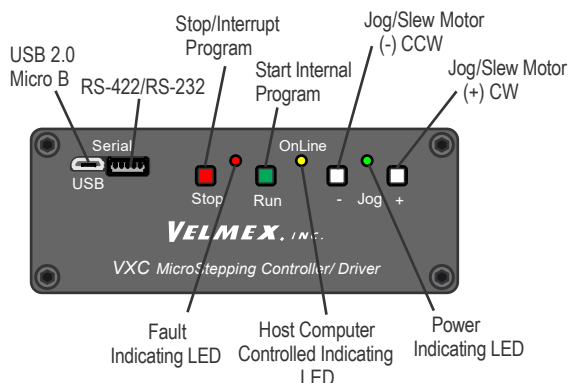
Function

Figure 1 VXC-2 Functional Diagram

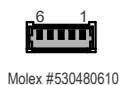


External Features

Front (Model VXC-1)



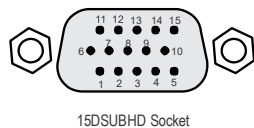
RS-422 Port



Pin# Name

- 1 Opt. +5V (out)
- 2 Tx+
- 3 Tx-
- 4 Gnd
- 5 Rx+
- 6 Rx-

Auxiliary I/O Connection

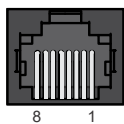


15DSUBHD Socket

Pin# Name

- 1 0V (Common Ground)
- 2 +5V Output
- 3 Ain (Analog In)
- 4 Run Input
- 5 I1 (Input 1)
- 6 I2 (Input 2)
- 7 I3 (Input 3)
- 8 I4 (Input 4/ Stop)
- 9 0V (Common Ground)
- 10 J- (Jog Mtr negative)
- 11 J+ (Jog Mtr positive)
- 12 O3 (Output 3)
- 13 O4 (Output 4)
- 14 O1 (Output 1)
- 15 O2 (Output 2)

VXC Limit Connection

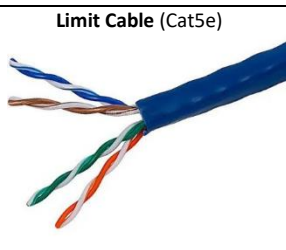


RJ45 Modular Jack

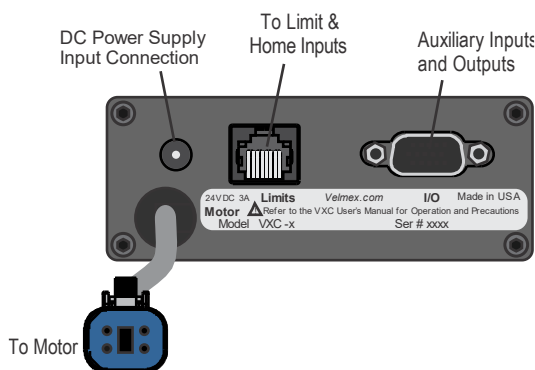
Pin# Name

- 1 Limit CW (+)
- 2 0V (Common Ground)
- 3 Limit CCW (-)
- 4 +10V Output
- 5 Chassis Gnd
- 6 0V (Common Ground)
- 7 Home Input
- 8 0V (Common Ground)

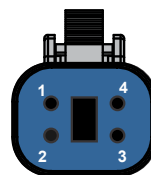
Pin#	Name	Wire Color
1	Limit CW (+)	Wh/Or
2	0V (Common)	Org
3	Limit CCW (-)	Wht/Grn
4	+10V Output	Blu
5	Chassis Gnd	Wht/Blu
6	0V (Common)	Grn
7	Home Input	Wht/Brn
8	0V (Common)	Brn



Rear (Model VXC-1)

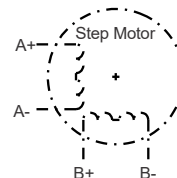


VXC Motor Cable Connector

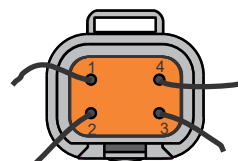


Amphenol #AT06-4S
(mates with: Amphenol #AT04-4P on Motor)

- | Pin# | Motor |
|------|-------|
| 1 | B- |
| 2 | B+ |
| 3 | A+ |
| 4 | A- |



VXC Motor Connector



Amphenol #AT04-4P
(mates with: Amphenol #AT06-4S on Cable)

Pin	Motor	Leadshine	Nanotec	Oriental	
1	B-	Blu	Blu	Grn & Wh/Blk	Blu
2	B+	Red	Yel	Blk & Wh/Grn	Red
3	A+	Blk	Red	Blu & Wh/Red	Blk
4	A-	Grn	Blk	Red & Wh/Blu	Grn

Setup

1. Connect the cables to motors and limit switches (if actuator has limit switches.) Connect a USB cable¹ from a computer to the micro B connector on the VXC's front panel labeled "USB" or make a RS-422/RS-232 connection to the VXC's front panel 6 pin serial connector (Refer to Appendix G for detailed serial port connection information.)

! CAUTION: Motor cables should not be bundled together with the Limit Switch, or any I/O cabling. Never put any of the VXC'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 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 VXC. Altered cables and resultant damage is not covered by the warranty. Refer to "Cable Length Versus Torque" when cable lengths are extended beyond standard.

IMPORTANT: The VXC will attempt to automatically detect limit switch inputs however, it is highly recommended that the limit inputs be set to the exact type of limit switches attached². Normally closed to move is the standard used for mechanical switches on Velmex products. Normally open to move is common for hall type limits and home switches. Velmex linear and rotary tables with the home switch option require that the switch inputs be configured correctly. Use the Velmex VXC Utility App to configure the VXC for the type of limit/home switches². If your computer is not a Windows based system, refer to Configuring VXC With Device ID #.

2. Connect cable from DC power supply to VXC
3. Plug the DC power supply into an outlet of an on/off switchable AC power strip
4. Turn on power strip switch, all three LEDs (red, green, yellow) will momentarily flash and the green LED will remain lit.
5. Use the VXC Utility App available at VelmexControls.com to set the VXC for the device attached². If your computer is not a Windows based system, use a terminal app to serially communicate with the VXC. Refer to Configuring VXC With Device ID #.

¹IMPORTANT: Use only the highest quality USB cables. Poor quality cables often cause intermittent connection issues. Refer to "How to Test USB Cable for a Reliable Connection" for procedure to test USB cables for proper connection.

² Refer to the "VXC Motor Controller Start Guide" sheet for any factory configured motor/limit/home settings that Velmex may have already set.

Configuring VXC With Device ID

Normally your VXC is factory set to match the motor(s) and other options of your electro-mechanical system. However, there are situations when Velmex does not factory configure the VXC as noted in Table 1 When VXC Will not be Factory Configured. The “VXC Motor Controller Start Guide” sheet will note if the VXC was factory configured¹ or not.

Table 1 When VXC Will not be Factory Configured

Multi-Axis when each axis is unique	When the system is multi-axis with different motor, limit switches, etc. on each axis, will require the end-user to decide which axis to connect to each component and then set the VXC accordingly.
Sold Separately	When the VXC is purchased without motors or actuators Velmex does not know what to set the VXC to operate with.

The three most critical settings are Motor Type, Limit Switch, and Home/Stall Detect. The VXC has the “setMLHmM=” command to set all three of these settings at once from the Device ID#.

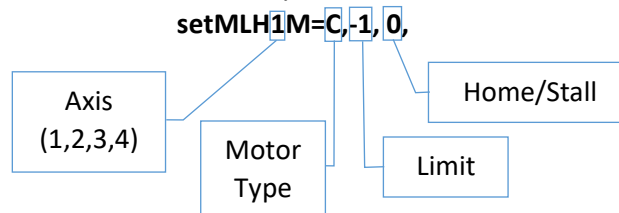
The Device ID label usually located near the motor connector. The Device ID# will start with a letter A to F followed by two other numbers separated by commas. Device ID example:

Device ID
D,-1,0,

! CAUTION: Failure to set the VXC for the Proper Device Connected Can Result in Damage to the VXC, Motor, Switches, and the Mechanical Assembly.

The easiest method to enter the Device ID# into the VXC is with the Velmex VXC Utility App. The other method is using a simple terminal program and directly type in the information. Put the VXC On-line first by sending an “E” or “F” then the “setMLHmM=” command and Device ID#.

This example sets axis one to work with an assembly with a Device ID# of **C,-1,0**,



This example sets axis two to work with an assembly with a Device ID# of **D,-1,0**,

setMLH2M=D,-1,0

This example sets axis three to work with an assembly with a Device ID# of **E,1,0**,

setMLH3M=E,1,0

NOTE: ¹ Only Needs to be set Once, the VXC Does an Autosave (rss) After This Setting. If Device ID is not Available Refer to Settings “setMT=” and “setL”

Device ID for Limit, Home/Stall Settings

The Device ID defines the system motor, limits, and home/stall hardware. Refer to Table 2 Settings for Limits, Home/Stall description of settings.

Table 2 Settings for Limits, Home/Stall

VELMEX, INC.		7550 State Routes 5A20 Bloomfield, NY 14489-9389 USA Tel: 985-657-6153 Fax: 985-657-6151 or 800-642-6446 Internet: VelmesControls.com Email: Support@velmes.com		4/29/2024	MLE							
VXC Label ID# Creator for "setMLHm M=Label ID" comma										NOTE: These are the basic settings, refer to VXC User's Manual, Appendix B for all options		
Motor Model	Type	C	Features	Devices	Limits	Home/Stall	Label ID	Limits	Home/Stall	Value	Value	
28CM013 (Size 11)	ALP		-	Linear Rotary	-	-	C,0,0,	0	0	0	0	
ST5981X3008 (Size 23 1x)	A		Snap Action Limits:	Linear Rotary	N/C (std)	-	C,1,0,	1	0	1	0	
ST5981S3008 (Size 23 2x)	B		+Home	Linear Rotary	N/C (std)	N/O Home	C,1,16,	1	16	1	16	
42CM06 (Size 17)	C		+Stall	Linear Rotary	N/C (std)	Stall	C,1,128,	1	128	1	128	
57CM06 (Size 23 1x)	D											
57CM13, 57CM22C (Size 23)	E		Magnetic Hall Limits:	Linear Rotary	N/O	-	C,-1,0,	-1	0	-1	0	
86CM35 (Size 34)	F		+Home	Linear Rotary	N/O	N/O Home	C,-1,16,	-1	16	-1	16	
			+Stall	Linear Rotary	N/O	Stall	C,-1,128,	-1	128	-1	128	
			Home + Stall*	-	Rotary	Home*	C,-2,128,	-2	128	-2	128	
* Wire N/O Home Switch to both limit inputs												
NOTE: Limits + Home + Stall is not possible since Home/Stall is a common input												

See Appendix A (Motor Settings) and Appendix B (Limits/Home/Stall Detect) for more information on reading and setting for motor type, limits, and home/stall.

Jog Mode

When the On-Line (yellow) light is not on, the VXC is in the Local/Jog mode. Using the front panel jog buttons, each motor can be jogged a single step¹ or slewed to 2000 sps² (5 revs/sec.) in either direction.

When a Jog button is pressed the motor moves 1 step (1/400 rev. ¹) If the button is held for >0.3 second¹ the motor will accelerate to 2000 sps². Pressing Stop or the other Jog button while jogging will hold the speed at 63 sps.

¹ Value is settable, see "Jog Function Settings" for configuring jog settings.

² Refer to Appendix D for details on the "setj" and "setJ" commands for setting jog speeds to a different value

Optional Joysticks

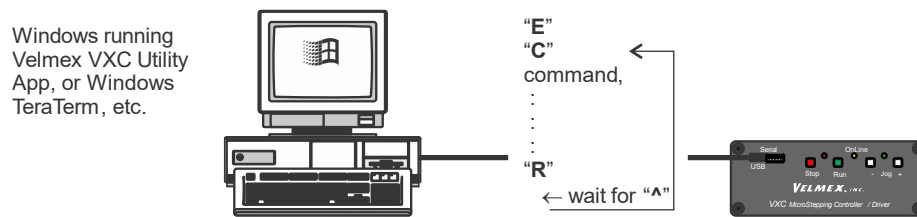
There are two types of external joysticks available for the VXC. One is digital that functions like the front panel jog buttons, and the second is an analog proportional speed type that can operate 1,2, or more motors. For more information/configuration refer to Appendix D and Appendix E.

Serial Communication

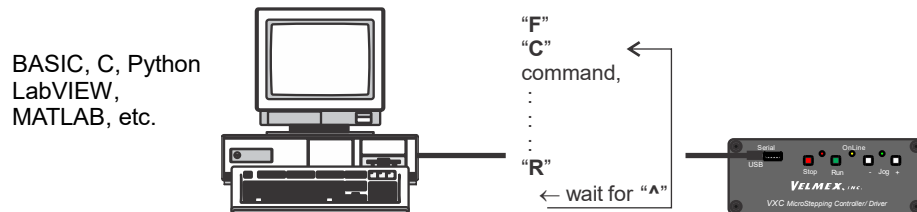
NOTE: All command characters are case sensitive

Programming of the VXC is accomplished by sending commands (ASCII characters) to the VXC through the USB or RS-422/RS-232 interface. Default serial port parameters: 8 Data, No Parity, 1 Stop, 57600 baud. To put the VXC 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 VXC on-line with echo "on" (echoes all characters received back to the host). The "F" puts the VXC on-line with echo "off". If you are using a terminal program to communicate to the VXC 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 VXC. Use "C" to clear previous commands from any previous command sending.

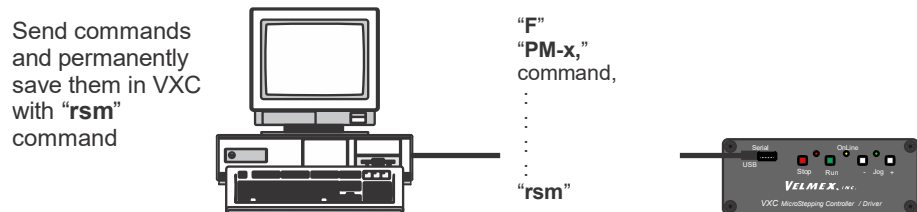
The simplest method to send commands is with the Velmex VXC Utility App, or with a terminal program like TeraTerm, CoolTerm, Termite, etc.



Another common method to send commands is with commercially available languages such as BASIC, C, LabVIEW, MATLAB, Python, etc.



Download/Standalone method allows the VXC to run without interacting with a host.



VXC holds program(s) that can be activated with the Run input. The default program to run is program 0. Inputs 1,2, and 3 can be configured to binary select and run programs 0 to 7.

👁 See Appendix I and Appendix N for more information.



Units & Direction

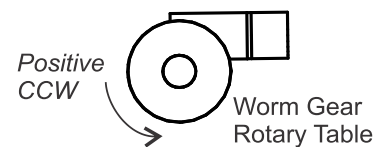
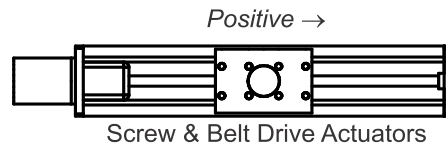
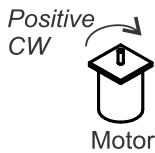
The VXC uses step units for Index and Speed parameters. One half-step is $1/400^1$ of a motor revolution. Step units for distance are used with the Index commands ("I" command.)

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/sec² units. Refer to Application Note AN106C 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 UniSlide®, XSlide™ and BiSlide® Assemblies, 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 direction, refer to "Controller Mode"



¹ Indexing 4000 microstep steps per rev resolution is achieved by entering a decimal value (i.x where x= 1 to 9)

Units for Velmex Positioners

Table 3 Velmex Product Advance per Turn & Advance per Step

Lead Screw Models		BiSlide ²				Speed
UniSlide ¹		XSlide ³	Advance per turn	Advance per step	@ 1000 sps (2.5 rps)	
x	x	y	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	
		M10 ⁴	100 mm	0.250 mm	250 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	

¹ Typical UniSlide model (where x is from Table 3): MA4012x-S4

² Typical BiSlide model (where y is from Table 3): MN10-0100-y-21

³ Typical XSlide model (where y is from Table 3): XN10-0040-y-71

⁴ BiSlide Belt drive units usually use a gearbox, divide Advance/turn and Advance/step by gear ratio.

Unit Conversion

To convert from "real" units to steps, divide the distance desired to move by the Advance per step.

$$\text{Distance} \div \text{Advance per step} = \text{Steps}$$

Example #1: To move 3.000 inches with the BiSlide E04 lead screw ($3.000 \div 0.001 = 3000$) requires a 3000 step index.

Example #2: To move 90 degrees with the B5990 rotary table ($90 \div 0.01 = 9000$) requires a 9000 step index.

Example #3: To move 4.000 inches with the UniSlide W1 lead screw ($4.000 \div 0.00025 = 16000$) requires a 16000 step index.

Other Formulas

$$\text{One Motor rev} = 400 \text{ steps}$$

$$\text{Linear Speed} = \text{Advance per step} \times \text{pulses per second}$$

$$\text{Rotary Speed} = \text{Advance per step} \times \text{pulses per second}$$

$$\text{Pulses per second} \div 400 = \text{rev/sec}$$

Motion Commands

This section gives detailed explanations of the VXC motion commands. For the advanced user, refer to the Appendices for more information.

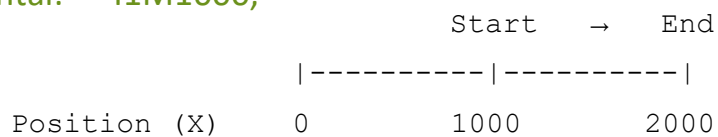
Most commands with variables (except "set" and "get" commands) use the VXC's program memory space. The required memory needed per command is specified in the command description. The VXC has 256 bytes of program memory allocated for each program. There are 13 (0,1,2,3,4,5,6,7,8,9,10,11,12) programs.

A program can be cleared by a "C" and selected by the "PMx" command. The default program when the VXC is powered up is #0. Using different programs is only relevant to users who will be operating the VXC in a stand-alone mode. Using the VXC in a serial interactive mode would only require that the default program be Cleared ("C") after the "R" (Run program) 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.

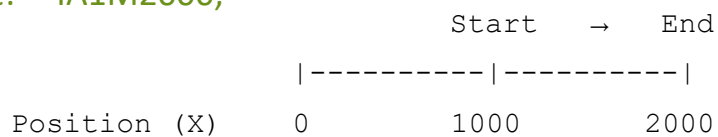
Incremental: I1M1000,



An absolute Index is, a move relative to absolute zero position, a distance and direction from the present position calculated by the VXC based on absolute zero position.

Absolute zero is established when the VXC is powered-up, by use of the "N", or the "IAmM-0" command.

Absolute: IA1M2000,



Sending Commands to the VXC

The standard serial port communication settings on the VXC are 57600 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) or a comma.

Incremental Index (Move an Increment)

ImMx Set steps (400 steps/rev.) to incremental¹ Index (move) motor, m =motor# (1,2,3,4)²
Motor moves CW³ (positive, Slider/Carriage will move away from motor end, Rotary Table will rotate CCW³) when $x=1$ to 16,777,215. Motor moves CCW³ (negative, Slider/Carriage will move toward motor end, Rotary Table will rotate CW³) when $x= -1$ to -16,777,215

Memory usage = 4 bytes.

Examples:

This example sets motor 1 to index 1200 steps CW:

```
I1M1200<cr>
```

This example sets motor 2 to index -9200 steps CCW:

```
I2M-9200<cr>
```

ImMx.y Set steps (4000 steps/rev.) to incremental¹ Index (move) motor, m =motor# (1,2,3,4)²
Motor moves CW³ (positive, Slider/Carriage will move away from motor end, Rotary Table will rotate CCW³), when $x.y= 0.1$ to 1,048,575.9. Motor moves CCW³ (negative, Slider/Carriage will move toward motor end, Rotary Table will rotate CW³), when $x.y= -0.1$ to -1,048,575.9.

Memory usage = 4 bytes.

Examples:

This example sets motor 1 to index 0.1 steps⁴ CW:

```
I1M0.1<cr>
```

This example sets motor 2 to index 9200.2 steps⁴ CW:

```
I2M9200.2<cr>
```

¹ An incremental Index is, a move relative to the present position, a distance and direction specified by the Index from the present position.

² The default motor is the last motor selected when using the “mM” designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the “mM” in a command is not required.

³ Motor direction can be inverted from the standard direction designated here with the “setDMx” command.

⁴ The digit (1 to 9) to the right of the decimal point is 1 step of a 4000 steps/rev microstep which is 1/10th of 400 step /rev. half step.

Absolute Index (Move to Position)

IAmMx Set steps (400 steps/rev.) to Absolute¹ Index (move), m =motor# (1,2,3,4)², $x=\pm 1$ to $\pm 16,777,215$ ³

NOTE: The absolute position registers have a range of -8,388,608 to 8,388,607 steps therefore to maintain a correct absolute position 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 12200:

IA1M12200<cr>

This example sets motor 2 to index to absolute position -9900:

IA2M-9900<cr>

IAmMx.y Set steps (4000 steps/rev.) to Absolute¹ Index (move), m =motor# (1,2,3,4)², $x.y=\pm 0.1$ to $\pm 1,048,575.9$ ^{3,4}

Memory usage = 4 bytes.

Examples:

This example sets motor 1 to index to absolute position 1000.1:

IA1M1000.1<cr>

This example sets motor 2 to index to absolute position 1900.2:

IA2M1900.2<cr>

¹ An Absolute Index is, a move relative to absolute zero position, a distance and direction from the present position calculated by the VXC based on absolute zero position. Absolute zero is established when the VXC is powered-up, by use of the "N", or the "IAmM-0" command. Refer to the "X", "Y", "Z", and "T" commands for determining the current position value.

² The default motor is the last motor selected when using the "mM" designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the "mM" in a command is not required.

³ Motor direction can be inverted from the standard direction designated here with the "setDMx" command.

⁴ The digit (1 to 9) to the right of the decimal point is 1 step of a 4000 steps/rev microstep which is 1/10th of 400 step /rev. half step.

Index to Absolute Zero (Move to Zero)

IAmM0 Set an Index to Absolute zero position, m =motor# (1,2,3,4)² When this command is used the VXC 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), "IAmM-0" command was used, or when the VXC was powered up.

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>

Zero Motor Position

IAmM-0 Zero motor position, m =motor# (1,2,3,4)² This command clears the position register for motor m , making this position absolute zero.

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>

 See also "N" the immediate Null Absolute Position Register command.

² The default motor is the last motor selected when using the "mM" designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the "mM" in a command is not required.

Move to Positive to Home/Limit

ImM0 Move positive until Home or positive limit switch is encountered, m =motor# (1,2,3,4)² Refer to Appendix B (Limits/Home/Stall Detect) for limit switch and home settings. The Index will end if the switch is not encountered after 16,777,215 steps.

Memory usage = 4 bytes.

Example:

This example sets motor 1 to seek the positive home switch:

```
I1M0<cr>
```

Move to Negative to Home/Limit

ImM-0 Move negative until Home or negative limit switch is encountered, m =motor# (1,2,3,4)² Refer to Appendix B (Limits/Home/Stall Detect) for limit switch and home settings. The Index will end if the switch is not encountered after 16,777,215 steps.

Memory usage = 4 bytes.

Example:

This example sets motor 1 to seek the negative limit switch:

```
I1M-0<cr>
```

² The default motor is the last motor selected when using the “mM” designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the “mM” in a command is not required.

Refer to Appendix B (Limits/Home/Stall Detect) for more information on using limit and home inputs.

Speed

SmMpx Set Speed of motor, m = motor# (1,2,3,4)¹, x = 1 to 61.9 in 0.1 step²/sec intervals, 62 to 6000 steps²/sec. in 1 step²/sec. intervals, p = motor power (default = 70%, + =100%, - = 40%) If never set, the default speed will be 2000 steps/sec. at 70% power.

Memory usage = 3 bytes.

Examples:

This example sets the speed of motor 1 to 1000 steps/sec. and 70% power:

S1M1000<cr>

This example sets the speed of motor 2 to 4500 steps/sec at 100% power:

S2M+4500<cr>

This example sets the speed of motor 3 to 500 steps/sec at 40% power:

S3M-500<cr>

This example sets the speed of motor 4 to 3.3 steps/sec at 40% power:

S4M-3.3<cr>

This example sets the speed of the default¹ motor to 100 steps/sec at 40% power:

S-100<cr>

This example sets the speed of the default¹ motor to 2500 steps/sec at 70% power:

S2500<cr>

This example sets the speed of the default¹ motor to 5000 steps/sec at 100% power:

S+5000<cr>

Motor running power by default is 70% of the maximum possible output. For most applications 70% will be adequate. For moving heavy loads vertically upwards the "+" speed (100% power) may be needed. For very light loads, low speeds, and vertical moving down 40% power should be used to prevent excess heating and motor resonance.

NOTE: Motor power will always be zero when the motor is stationary (motors are normally un-energized at a standstill. Contact the Velmex Engineering department for more information on fail-safe brake and holding torque options.

NOTE: Motor torque decreases as speed increases, and some motors have limited useful torque above 3000 steps/sec. If the motor torque is below the needed torque to move the load, the motor will stall (lose synchronism and proper position.)

! CAUTION: Motor and Controller surface temperatures become hot when running motors continuously. Only use 100% ("+") motor power if maximum torque is required. For maximum efficiency when lifting heavy loads vertically, use the "+" speed for traversing upwards, and the "-" speed for the moving downwards and for low speeds.

¹ The default motor is the last motor selected when using the "mM" designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the "mM" in a command is not required.

² The speed is based on 400 steps/rev.

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.

SMART SPEEDS

The Smart Speed command is a quick way to set motor to a maximum reliable speed.¹²

- SmM+X** Set Maximum Speed of motor at 100% load (heavy load moving up, extremely heavy load moving horizontally) $m = \text{motor\# (1,2,3,4)}$
- SmMX** Set Maximum Speed of motor at partial load (moderate load moving up, heavy load moving horizontally) $m = \text{motor\# (1,2,3,4)}$
- SmM-X** Set Maximum Speed of motor at no load (moving down, very light loads, small slides, and most rotary tables) $m = \text{motor\# (1,2,3,4)}$

The VXC determines a speed and power setting based on the current motor attached and which one of three Smart Speed commands above is used. To see the actual speed value the VXC calculated use the "lst" command to list out the current program.

See Table 4 Motor Type Setting" for speeds that are used for each type motor.

NOTE: Smart Speeds values are determined based on the current motor setting at the time the Smart Speed command is used. If the motor determined (Smart Speed) result is saved in a program the speed will be fixed and will not change even if a different motor is selected.

NOTE: If no motor is set and Smart Speed is used a "?" will be sent back and a fault #30 (Value entered out of range) will be logged.

¹Maximum speed is limited by actual load, motor size, lead screw pitch/mechanism ratio, external forces, wear/friction, acceleration values, frictional and inertia damping. Smart Speed values are based on a maximum speed that has approximately 50% of the starting torque of the motor. Contact the Velmex Engineering department for help on determining what system parameters to consider for your particular application.

² **! IMPORTANT:** Smart Speeds are only applicable for the standard and correctly selected motors per Table 4 Motor Type Setting in Appendix A

 See also "AmMx" Acceleration/deceleration command

Acceleration

AmMx Set Acceleration/deceleration of motor, m = motor# (1,2,3,4)¹, x = 1 to 127. If never set the default value will be 2. The higher the number 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 too high.

Memory usage = 2 bytes.

Examples:

This example sets the acceleration/deceleration of motor 1 to 3:

A1M3<cr>

This example sets the acceleration/deceleration of motor 4 to 10:

A4M10<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.

SMART ACCELERATION

The Smart Acceleration command is a quick way to set motor acceleration for different types of loads and devices.¹²

AmM+X Set Acceleration of motor for a heavy load moving up, extremely heavy load moving horizontally, Direct drive belt drives, and coarse pitch leadscrews. m = motor# (1,2,3,4)¹

AmMX Set Acceleration of motor for a medium load, moderate load moving up, heavy load moving horizontally with geared drive belt drives, and medium pitch leadscrews. m = motor# (1,2,3,4)¹

AmM-X Set Acceleration of motor for a light load, fine pitch leadscrews with medium loads, moving down or horizontal with very light loads. m = motor# (1,2,3,4)¹

To see the actual acceleration value the VXC calculated use the "**lst**" command to list out the current program.

¹ The default motor is the last motor selected when using the " mM " designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the " mM " in a command is not required.

² **! IMPORTANT:** Smart Accelerations are only applicable for the standard and correctly selected motors per Table 4 Motor Type Setting in Appendix A

Program Coding 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 VXC (Model VXC-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 VXC:

A2 , S4000 , I400 ,

For running a particular motor of a multi-axis VXC, only the first command needs the motor number.

For example, all of these commands would be for motor 2:

I2M200 , I-200 , S2000 , IA0 ,

Interactive Mode

The VXC can be used both interactively and standalone. Interactive mode is the most common method when a computer is always attached to the VXC. Interactive control is accomplished with the following steps¹.

1. Establish a serial connection to the VXC's USB or RS-422 port (send "E" to put VXC in On-line Mode when using terminal programs like COSMOS or TeraTerm, send "F" to put VXC in On-line Mode when communicated using program languages)
2. Clear any previous commands in the VXC (send "C")
3. Send motion commands to the VXC ("Ax", "Sx", "Ix")
4. Run the commands (send "R")
5. Wait for the VXC to finish (wait for "A")
6. Output, take a measurement, or wait for an external event, etc.
7. Repeat from Step #2

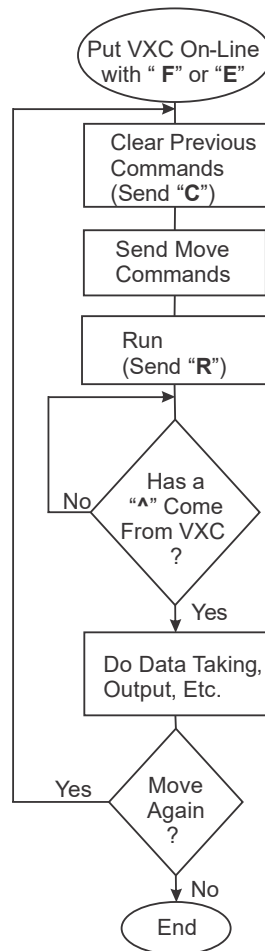


Figure 2 Interactive Mode

¹ Application software such as Velmex VXC Utility App TeraTerm, C, BASIC, Python, LabVIEW, MATLAB, etc. are commonly used to send, receive, wait, and repeat in Interactive Mode. Refer to examples at VelmexControls.com

Interactive Examples

The interactive examples below are shown keyed directly into Velmex VXC Utility App terminal window.

NOTE: Spaces between commands are optional and commands with a value should end with a comma or carriage return (enter).

Example that puts the VXC On-line with echo on, Clears memory, Indexes motor one 4000 steps

```
E C I1M4000,R
```

After VXC finishes previous Index

```
E C I1M4000,R^
```

Example that clears out last commands, sets speed to 3000, Indexes negative 2000

```
C S1M3000,I1M-2000,R
```

Read motor 1 position

```
X
```

Returns current motor 1 position

```
X7000
```

Example that puts VXC into Jog Mode (Off-line)

```
Q
```

Example that puts VXC On-line with echo off, clears, sets motor 2 speed and Indexes 1000

```
FC S2M4000,I2M1000,R
```

Standalone Mode

Standalone mode allows the VXC to run programs without interaction with a host computer. After initial downloading and saving programs in the VXC, running a program² is accomplished by the Run button or Run input. Standalone control is accomplished with the following steps¹.

1. Establish a serial connection to the VXC's USB or RS-422 port (send "E" to put VXC in On-line Mode when using terminal programs like VXC Utility App, TeraTerm, CoolTerm, Termit, etc.)
2. Select and clear program in the VXC
3. Send all motion commands to the VXC ("Ax", "Sx", "Ix", etc.)
4. If saving multiple programs² repeat from Step #2
5. Send the Run Save Programs command (send "rsm")
6. Disconnect host and power down VXC
7. Powerup VXC, program 0 will be the default
8. Use the Run button or Run input to start the program²

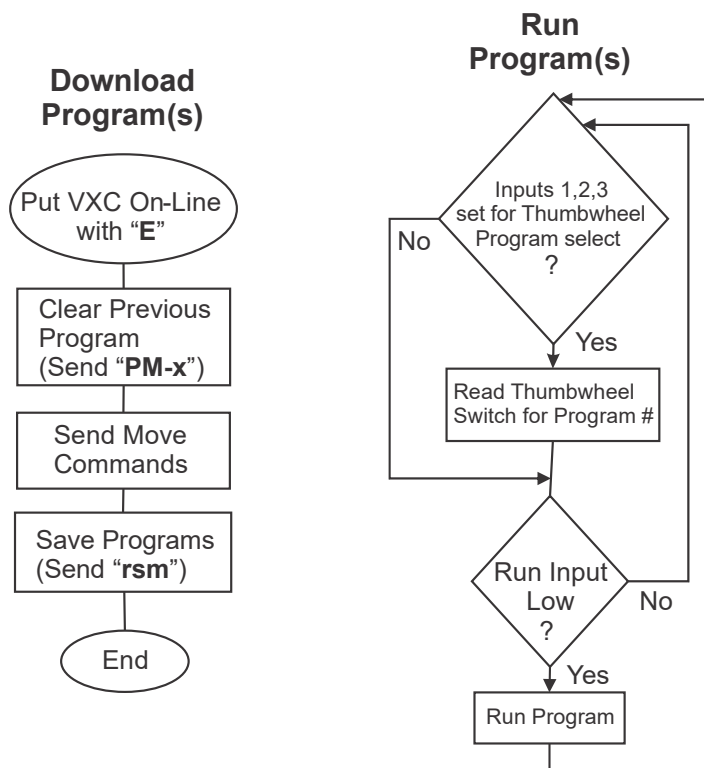


Figure 3 Standalone Mode

¹ Application software such as Velmex VXC Utility App, TeraTerm, CoolTerm, Termit, etc. are commonly used to send/download commands. Refer to examples at VelmexControls.com

²When running Standalone, multiple programs (0 to 7) can be selected and run with a thumbwheel selector switch connected to Inputs 1,2,3, refer to Multifunction User Inputs and Application Note for selecting programs with a thumbwheel switch.

Standalone Examples

The Standalone examples below are shown keyed directly into Velmex VXC Utility App terminal window.
NOTE: Spaces between commands are optional and commands with a value should end with a comma or carriage return (enter).

Example that saves an Index for motor one of 4000 steps in Program #0

```
F PM-0, I1M4000, rsm
```

Example that saves an Index, Pause 1 second, Index back, for motor one in Program #0

```
F PM-0, I1M4000, P1, I-4000, rsm
```

Waits for Run to go low/high, Index, Wait for Run, Index back, for motor two in Program #0

```
F PM-0, U90, I2M400, U90, I-400, rsm
```

Waits for Run to go low/high, Index, High .1 sec. on Output 1, Wait for Input 1 low, Index back

```
F PM-0, U90, I1M400, PA.1, U0, I-400, rsm
```

Eight programs to be selected with a Thumbwheel, moves out, .1 sec. Output 1, move to absolute 0

```
F  
PM-0, A1M4, S800, I1000, PA.1, IA0,  
PM-1, A1M4, S1000, I1500, PA.1, IA0,  
PM-2, A1M4, S1200, I2000, PA.1, IA0,  
PM-3, A1M4, S1400, I2500, PA.1, IA0,  
PM-4, A1M4, S1600, I3000, PA.1, IA0,  
PM-5, A1M4, S1800, I3500, PA.1, IA0,  
PM-6, A1M4, S2000, I4000, PA.1, IA0,  
PM-7, A1M4, S2200, I4500, PA.1, IA0  
rsm
```

Command Summary (Common & Advanced)

The following are the abbreviated descriptions for the most common commands, refer to the "Command Reference" section for detailed command information.

Operation Commands

- E** Enable On-Line mode with echo "on"
- F** Enable On-Line mode with echo "off"
- C** Clear all commands from current program
- Q** Quit On-Line mode (return to Local/Jog 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
- D** Decelerate to a stop (interrupts current index/ program in progress)
- G** Enable On-Line mode with echo off Grouping a <cr> with "^", ":", and other character responses

Editing/Debugging Tools

- ;** Ignore all characters from here to end of the line (<cr>)
- :** Put Controller on Hold (stop after each command and wait for go)
- del** Delete last command
- res** Software reset controller
- #** Request the number of the currently selected motor

Setup Commands

- setMLHmM=x** Set axis m for Motor, Limits, Home, x= Device ID# on unit
- setMTmM=x** Set axis m for motor type x
- setLmMx** Set Limit Switch mode for axis m, m= motor# (1,2,3,4)
- setHSmMy** Set Home/Stall input for axis m, m= motor# (1,2,3,4)
- rss** Run save settings (saves setup values to nonvolatile memory)
- setDAmMx** Set Joystick Deadband value for motor m.
- setjmMx** Set first range Jog Speed for motor m. setjAmM for Joystick range setting
- setJmMx** Set second range Jog Speed for motor m. setJAmM for Joystick range setting
- setKmMx** Set Backlash Compensation
- setPmMn** Set "Pulse Every n # Steps" on output 2 for axis m
- setPAx** Set Pulse width used by setPmMx and U7, x=1 to 255 (10 msec increments)
- setIx** Set operating mode of User Inputs. The value for x is a number between 0 and 255
- setDMx** Set operating mode of VXC. The value for x is a number between 0 and 255
- setMJmMx** Set Motor Jog function decimal number x between 0-255.
- setBx** Set RS-232 Baud rate (1=9600, 2=19200, 3=38400, 4=57600)

IMPORTANT: Always use "rss" command after set to permanently save

Status Request Commands

Help	Display context sensitive help screen (for use with terminal program interfacing)
X	Send current position of motor 1 to host (Motor can be in motion)
Y	Send current position of motor 2 to host (Motor must be stationary)
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)
V	Verify Controller's status, VXC sends "F" for in fault, "B" if busy, "R" if ready, "J" if in the Jog/slew mode, or "b" if Jogging
!	Capture motor positions for later recall with "x", "y", "z", "t" commands
x	Send last 4 positions of motor 1 to host that were captured
y	Send last 4 positions of motor 2 to host that were captured
z	Send last 4 positions of motor 3 to host that were captured
t	Send last 4 positions of motor 4 to host that were captured
*	Request the position when the last motor started decelerating (position when "D" command or Stop/User input 4 used)
@	Read user analog input value
lss,	List all "getx" settings for a single axis VXC
lssmM	List all "getx" setting from axis m
getKmM	Read Backlash compensation setting
getMTmM=	Read motor type set for motor/axis m
getLmM	Read mode of limits for motor/axis m
getF	get current fault
getFA	get all current faults (VXC buffers a maximum of 10 faults)
getF-	get last fault copied to EEPROM memory
getDx	Read mode/version/date code/serial number
getDAmM	Read Joystick Deadband setting for motor m.
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
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
getMJ	Get Motor Jog function

Motion Commands

ImMx	Set steps to incremental Index motor, m= motor# (1,2,3,4), x=±1 to ±16,777,215 (± for CW/CCW)
ImMx.y	Set steps (1/10 th μsteps) to incremental Index motor m, x.y= ±0.1 to ±1,048,575.9
IAMMx	Set Absolute Index distance, m=motor# (1,2,3,4), x= ±1 to ±16,777,215 steps
IAMMx.y	Set steps (1/10 th μsteps) to Absolute Index motor m, x.y= ±0.1 to ±1,048,575.9
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 home or positive limit is encountered, m=motor# (1,2,3,4)
ImM-0	Index motor until home or negative limit is encountered, m=motor# (1,2,3,4)
SmMpx	Set Speed of motor m (70% power, ±p for 100%/40% power), x=1 to 6000 (1 to 61.9 in 0.1 steps/sec. intervals)
AmMx	Acceleration/deceleration, m= motor# (1,2,3,4), x=1 to 127.
(ImMx,I1Mx,)	Run Index moves for axes simultaneously, m = 2,3,4 Axis (Axis 1 must be last)

Program Management Commands

- PMx** Select Program number x, x= 0 to 12
- PM-x** Select and clear all commands from Program number x, x= 0 to 12
- PM** Request the number of the current Program
- Mem** Request Memory available for currently selected program
- lst,** List the current program for a single axis VXC
- lstmMx** List program x of axis m
- rsm** Run save memory (saves setup & program values to nonvolatile memory)
- [m[d,d1,d2,...]** Pass data d to Axis m, m=Axis 2,3,4, d= program stored commands
- PMAx,y** Program Associate Master/Slave(s) x to program number y (Multi-axis VXC's start the same time) x= 0 (disable)
- PMA** Request the current program associate number

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
- Jx** Jump to the beginning of program number x, x= 0 to 12
- JMx** Jump to the beginning of program number x and come back for More after program x ends, x= 0 to 12

Pausing, Input/Output Commands

- Px** Pause x seconds, (x=0.0001 to 5.9999 & 6.0 to 6553.5 sec.)
- PAX** Pause x seconds (x=0.0001 to 5.9999 & 6.0 to 6553.5 sec, 10 µsec when x=0) Altering output 1 high for duration of the pause
- 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
- U90** Wait for a low to high on the Run button or connection I/O,4 with debouncing for a mechanical push-button switch
- 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 auto-decel to stop
- U10** Synchronize Master and Slave Axis
- 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
- U21** Skip next command if input 1 is low
- U22** Skip next command if input 2 is low
- U13** Wait for a front panel button to jump to a program or continue
- U14** User output 2 low (reset state)
- U15** User output 2 high
- U16** User output 3 low (reset state)
- U17** User output 3 high
- U18** User output 4 low (reset state)
- U19** User output 4 high
- U23** Wait for a front panel button to jump to a program and come back, or continue
- 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, holding user output 1 high
- U65** Stop Slave Axes
- U66** Kill Slave Axes
- U104** Axis 2 User output 1 "low" (reset state)
- U105** Axis 2 User output 1 high
- U114** Axis 2 User output 2 low (reset state)
- U115** Axis 2 User output 2 high
- U116** Axis 2 User output 3 low (reset state)
- U117** Axis 2 User output 3 high
- U118** Axis 2 User output 4 low (reset state)
- U119** Axis 2 User output 4 high
- U154** Axis 3 User output 1 "low" (reset state)
- U155** Axis 3 User output 1 high
- U164** Axis 3 User output 2 low (reset state)
- U165** Axis 3 User output 2 high
- U166** Axis 3 User output 3 low (reset state)
- U167** Axis 3 User output 3 high
- U168** Axis 3 User output 4 low (reset state)
- U169** Axis 3 User output 4 high
- U204** Axis 4 User output 1 "low" (reset state)
- U205** Axis 4 User output 1 high
- U214** Axis 4 User output 2 low (reset state)
- U215** Axis 4 User output 2 high
- U216** Axis 4 User output 3 low (reset state)
- U217** Axis 4 User output 3 high
- U218** Axis 4 User output 4 low (reset state)

Program Management Commands

Most commands with variables (except set commands) use the VXC's program memory space. The required memory needed per command is specified in this section. The VXC has 256 bytes of program memory for each program. There are 13 (0,1,2,3,4,5,6,7,8,9,10,11,12) programs. A program can be cleared by a "C" and selected by the "PMx" command. The default program when the VXC is powered up is #0. Using different programs is only relevant to users who will be operating the VXC in a stand-alone mode. Using the VXC in a USB or RS-422/RS-232 interactive mode. would only require that the default program be cleared after the R command.

To select a specific program, use the following command.

PMx Select Program number x as the current program, $x = 0$ to 12. Each program can contain up to 256 bytes¹ of commands.

Memory usage = 0 bytes. Immediate command (not stored)

Example:

This example selects program #1 for the current program:

PM1<cr>

C Clear all commands from the current program. All setup values, motor position values, and the state of user outputs will not be altered.

Memory usage = 0 bytes. Immediate command (not stored)

PM-x Select and clear commands from Program number x , $x = 0$ to 12. This command will select program x as the current program and delete all commands from this program. All setup values, motor position values, and the state of user outputs will not be altered.

Memory usage = 0 bytes. Immediate command (not stored)

Example:

This example selects program #2 and erases all commands within it:

PM-2<cr>

PM Request the number of the current program. the VXM will send a value between 0 and 12 indicating the current program number selected.

Example:

PM<cr>

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

3<cr>

Mem Request Memory available for currently selected program. Returns remaining memory in current program (0 to 256)

¹ 256 bytes of memory is equivalent to 64 individual index moves. However, with the "Lx" looping commands the number of moves possible are virtually unlimited if they have a common repeating pattern.

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

Use this command to:

1. To permanently save setup/special function (commands) values that have been modified.
2. To save programs/commands and settings 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 keep a program in the VXC for use without a host computer (stand-alone use.)

lst, List commands in current program to host of a single axis VXC. The VXC will return program number and the current memory remaining in the program before listing the commands.

Example listing when current the program is 0 and there is one Index command in the program:

```
lst
PM0 M252
I1M400
```

NOTE: The VXC also sends the ASCII character <EOT> (End-Of-Transmission, decimal value 4) at end of listing a program.

lstmMx, List commands in program x to host from axis m, m= motor# (1,2,3,4)² and program x, x= program# (0,1,2,3,4,5,6,7,8,9,10,11,12.) The VXC will return program number and the current memory remaining in the program before listing the commands.

NOTE: The VXC also sends the ASCII character <EOT> (End-Of-Transmission, decimal value 4) at end of listing a program.

² The default motor is the last motor selected when using the “mM” designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the “mM” in a command is not required.

Editing/Debugging Tools


- ;** Ignore all characters from here to end of the line (<cr>) Used to add comments in script files to send to the VXC. Example:

```
F          ;Put VXC On-Line with Echo Off
C          ;Clear Out All Previous Commands
I1M400,   ;Index Motor 1 400 steps+ (1 rev)
```

The VXC only interprets the "FCI1M400," commands from the above script.

- :** Put the Controller on Hold¹ (single step through program). When the VXC receives the single character ":" 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 ":" 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.
- 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 VXC 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 VXC 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.)

¹ Refer to Appendix I for another method that uses Input 1 as a Stop/Hold

 See also "Ist", "X", "Y", "Z", "T", "V" commands

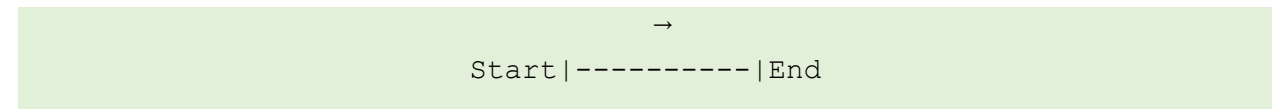
Program Examples

The following examples can be keyed into the VXC with a terminal program like TeraTerm, or the Velmex VXC Utility App. Another method to send commands is with commercially available languages such as BASIC, C, LabVIEW, Matlab, etc.

The "<cr>" is a carriage return character (<Enter> key on most keyboards). Command characters are shown 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 1 Enable On-Line Mode Echo on

```
E
```

Bytes: -

Example 2 Single Axis Incremental Index Move 1st Motor 400 Steps

```
I1M400,R
```

Bytes: 4

Another method to enter the same command as above:

Example 3 Single Axis Incremental Index Move 1st Motor 400 Steps

```
I1M400<cr> R
```

Bytes: 4

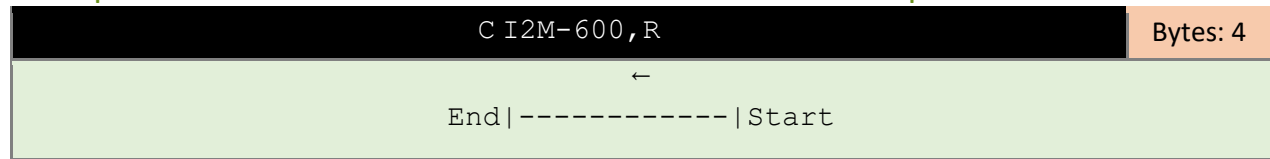
Example 4 Clear all Commands from Current Program & Zero Motor Position

```
C N
```

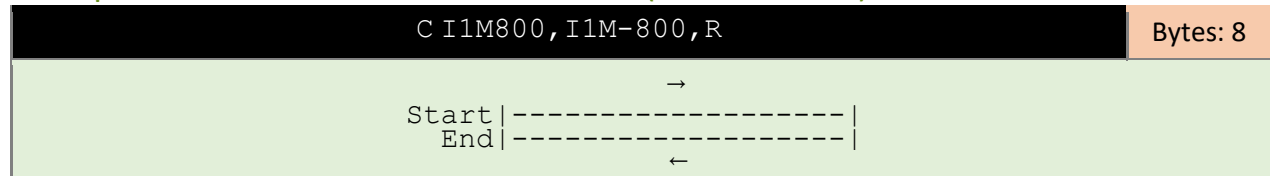
Bytes: -

NOTE: Running Multi-Axis examples on a VXC that lacks an axis will result in Fault # 31 "Axis Does Not Exist"

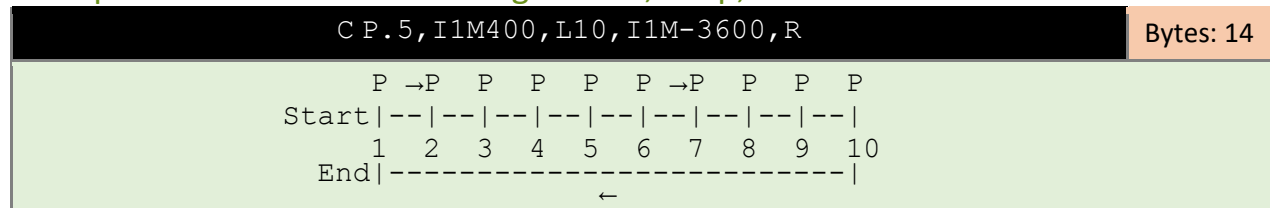
Example 5 Clear Previous & Index Move 2nd Motor 600 Steps



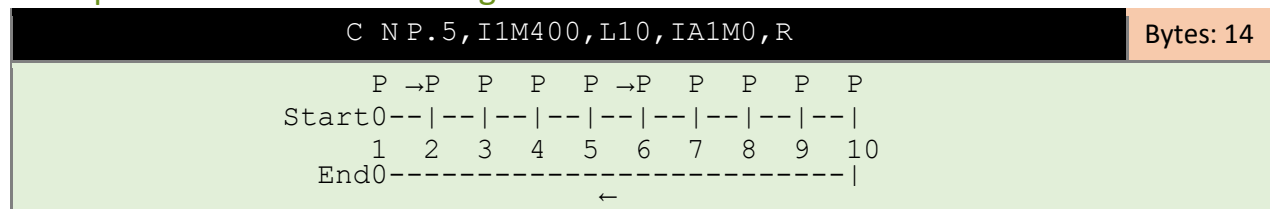
Example 6 Index Motor 1 Both Directions (Auto Reverse)



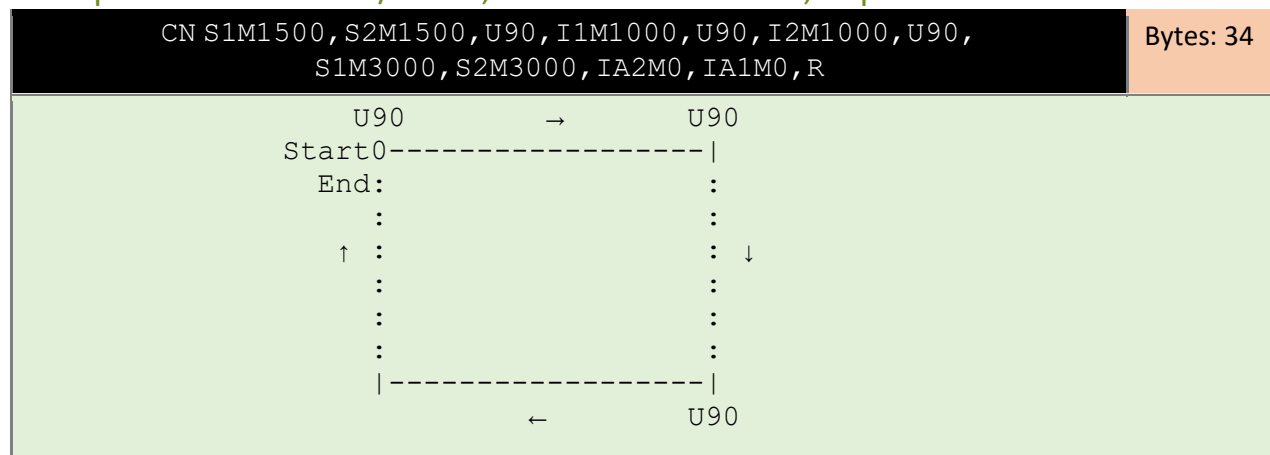
Example 7 One-Axis Index Pausing 0.5 Sec, Loop, Return to Start



Example 8 One-Axis Index Using Index Absolute Zero to Return to Start



Example 9 Two-Axis Out/Down, Wait for Run button, Rapid Return to Start

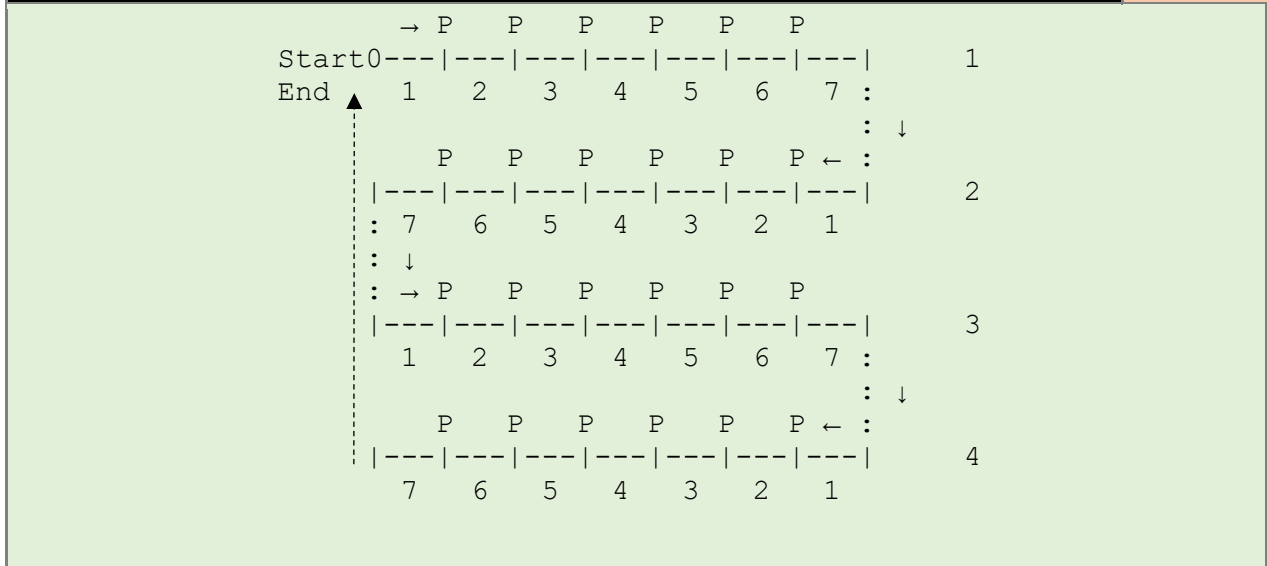


NOTE: Running Multi-Axis examples on a VXC that lacks an axis will result in Fault # 31 "Axis Does Not Exist"

Example 10 Two-Axis Raster Scan With 0.1 Pulse at Each Stop & Return to Start

N C I1M300, PA.1, L7, I2M400, L-4, IA2M0, R

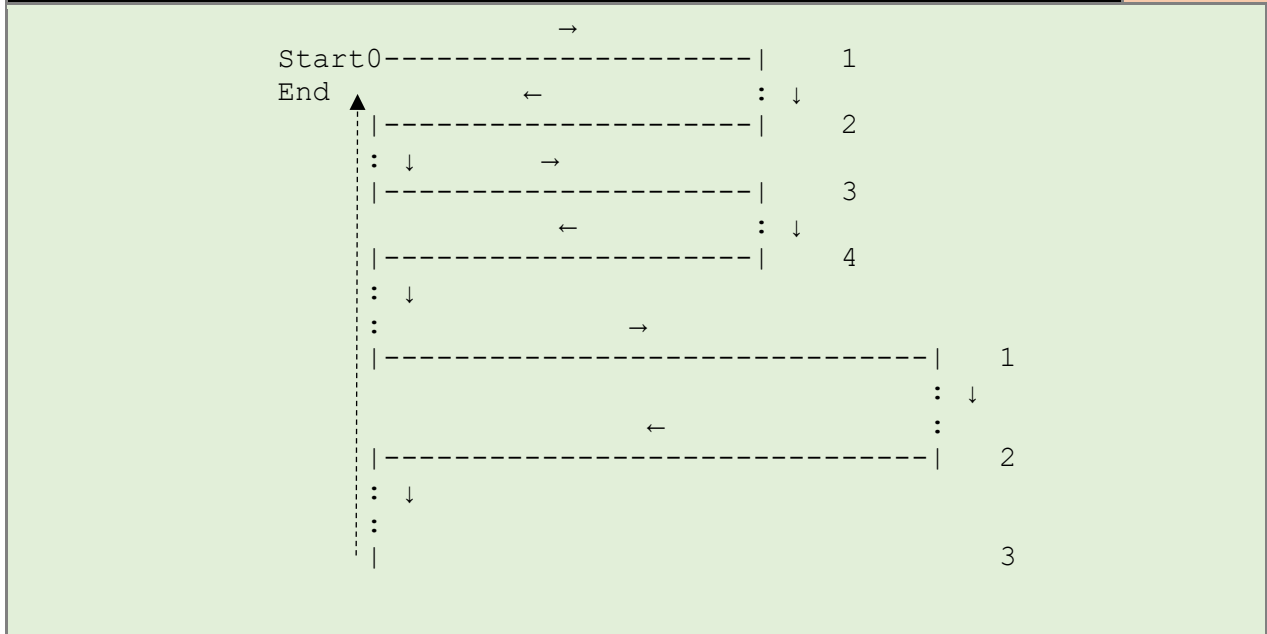
Bytes: 21



Example 11 Two-Axis With Two Different Raster Scans & Return to Start

CN I1M2000, I2M300, L-4, LM0, I2M600, I1M3000, L-3, IA2M0, R

Bytes: 27



👁 See Appendix B (Limits/Home/Stall Detect) for Homing Examples.

NOTE: Running Multi-Axis examples on a VXC that lacks an axis will result in Fault # 31 "Axis Does Not Exist"

Example 12 Three-Axis XY Matrix, Z up/dn at Each Position, & Return to Start

```

NC I3M1000,I3M-1000,I1M600,L5,I2M400,L-3,(IA2M0,IA1M0,)R Bytes: 32

```

Example 13 Two-Axis Rectangle, Wait Input 1 at Each Corner, Loop Forever

```

C I1M2000,U1,I2M1000,U1,I1M-2000,U1,I2M-1000,U1,L0,R Bytes: 25

```

Example 14 One-Axis Index/10ms Pulse, Looping, Using Smart Acel & Speed

```

C A1MX,S1MX,PA.01,I1M400,L10,L-2,R Bytes: 18

```

Looping/Branching Commands

Loop

- LO** 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 Loop-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 Loop-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
- Lx** Loop 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** Loop 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>**
- LAx** Loop 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 600-1 times: **LA600<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 Loop Always from beginning or Loop-to-marker of the current program $x-1$ times alternating direction of motor 1 indexes ($x=2$ 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 100-1 times: **LA100**<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): **A50000,LA50000**<cr>

Jump

Jx Jump to the beginning of program number x , $x=0$ to 12. Program number x will temporarily be the current program and all commands will be executed starting from the first one that was previously entered into program x . If there are not any commands in program x , or after executing the last command, the program will end, and the VXC will send the ready prompt to the host ("^"). The current program number will still be the program that was originally selected with a "PM x " or "PM- x " command. Linking multiple programs (maximum of 13) together is possible by using a jump command, as the last command, to make a jump to a different program.

Memory usage = 2 bytes

Example:

This example will jump to program #1: **J1**<cr>


JMx Jump to the beginning of program number x and come back for More after program ends, $x=0$ to 12. Program number x will temporarily be the current program and all commands will be executed starting from the first one that was previously entered into program x . If there are not any commands in program x , or after executing the last command, control will be transferred back to the program that initiated the Jump, followed by the next command in the initiating program to be executed. The maximum commands active at a time is 12. This command is used to make programming more modular, easier to maintain, and edit by having a main program that jumps to other programs (modules) and returns. All looping commands in program x will be local to this program.

! CAUTION: Motor reverse-direction-flags are set by "L- x " looping commands. If a **JMx** command is used "inside" one of these loops motor 1 may be reversed in program x .

Memory usage = 2 bytes

Example:

This example will jump to program #3 and return: **JM3**<cr>

 See also "U11", "U12", "U21", "U22" commands for branching on state of an input.

Pause

Px Pause *x* seconds, (*x*=0.0001 to 5.9999 & 6.0 to 6553.5 seconds)

Memory usage = 3 bytes

Examples:

This example pauses for 50 milliseconds: **P.05**<cr>

This example pauses for 0.6 seconds: **P.6**<cr>

This example pauses for 10 seconds: **P10**<cr>

This example pauses for 1.25 seconds: **1.25**<cr>

This example pauses for 1 hour: **P3600**<cr>

PAx Pause *x* seconds (*x*=0.0001 to 5.9999 & 6.0 to 6553.5 seconds, 10 microseconds when *x*=0)

Altering output 1 (I/O,14) high (+5V) for duration of the pause.

Memory usage = 3 bytes

Examples:

This example pauses for 15 milliseconds holding output 1 high: **PA.015**<cr>

This example pauses produces a 10 microsecond pulse on output 1: **PA0**<cr>

This example pauses for 1.5 seconds holding output 1 high: **PA1.5**<cr>

Ux Commands

Wait for Input

Input 1

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 as the preferred pushbutton switch input command.

Memory usage = 2 bytes

U1 Wait for a "low" on the user input1 holding 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 "**U51**" command as the preferred pushbutton switch input 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 1ms, 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

U50 Wait for a low to high transition on the user input 1 with 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 VXC 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 VXC will filter out the electrical bounces associated with mechanical switches.

Memory usage = 2 bytes

Run Low/High

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 “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 VXC is in an idle state either On-Line or in Local Jog/slew mode.

Conditional Branch on Input

U11 Skip next command if input 1 (I/O,5) is high

U21 Skip next command if input 1 (I/O,5) is low

U12 Skip next command if input 2 (I/O,6) is high

U22 Skip next command if input 2 (I/O,6) is low

Wait for Front Panel Button

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

U32 Wait for "Motor 1 Jog -" button to be pressed on front panel with debouncing
Memory usage = 2 bytes

U33 Wait for "Motor 1 Jog +" button to be pressed on front panel with debouncing
Memory usage = 2 bytes

Output

Output 1

- 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

Output 2

- 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

Output 3

- U16** User output 3 "low". The user output 3 (I/O,12) 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 (I/O,12) will go to +5V. This command is used in conjunction with the "**U16**" command.
Memory usage = 2 bytes

Output 4

- U18** User output 4 "low". The user output 4 (I/O,13) 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 (I/O,13) will go to +5V. This command is used in conjunction with the "**U18**" command.
Memory usage = 2 bytes

Outputs on Axes 2,3,4

The I/O outputs on a second, third, or fourth Slave VXC (in a Master/ Slave bussed configuration) are available through the Master. To access the user I/O on the Axis 2 from the Master use the standard "Ux" commands +100. For Axis 3 use the standard "Ux" commands +150. For Axis 4 use the standard "Ux" commands +200.

The following are valid commands for I/O addressing on a multi-axis VXC.

Axis 2 Outputs

- U104** Axis 2 User output 1 "low" (reset state)
- U105** Axis 2 User output 1 high
- U114** Axis 2 User output 2 low (reset state)
- U115** Axis 2 User output 2 high
- U116** Axis 2 User output 3 low (reset state)
- U117** Axis 2 User output 3 high
- U118** Axis 2 User output 4 low (reset state)
- U119** Axis 2 User output 4 high

Axis 3 Outputs

- U154** Axis 3 User output 1 "low" (reset state)
- U155** Axis 3 User output 1 high
- U164** Axis 3 User output 2 low (reset state)
- U165** Axis 3 User output 2 high
- U166** Axis 3 User output 3 low (reset state)
- U167** Axis 3 User output 3 high
- U168** Axis 3 User output 4 low (reset state)
- U169** Axis 3 User output 4 high

Axis 4 Outputs

- U204** Axis 4 User output 1 "low" (reset state)
- U205** Axis 4 User output 1 high
- U214** Axis 4 User output 2 low (reset state)
- U215** Axis 4 User output 2 high
- U216** Axis 4 User output 3 low (reset state)
- U217** Axis 4 User output 3 high
- U218** Axis 4 User output 4 low (reset state)
- U219** Axis 4 User output 4 high

Jog While Waiting

- 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.

Programmable Serial Prompt

- U6** Send "**W**" to the host and wait for a "**G**" to continue. The VXM sends the single character "W" to the host when this command is executed. The VXC will wait until a "**G**" is received from the host before proceeding in the program.
Memory usage = 2 bytes

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 (default width is 10 sec.) This pulse would be used to trigger data acquisition 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
- 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
- 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 VXC 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 VXC 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.
- 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 VXC executes this command, to prevent an excessively hard stop that may cause a mechanical overshoot of intended position.
- U10** Synchronize master and slave axes. Used in associated programs for multi-axis coordinated motion. When used in a Slave (axes 2,3,4) the slave axis will wait for Master to send a start trigger. When used in a Master (axis 1) the Master will send a trigger to slave axes.
Memory usage = 2 bytes
- U65** Decelerate motor on Slave to a stop¹.
Memory usage = 2 bytes
- U66** Kill operation on Slave¹. 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.
Memory usage = 2 bytes
! CAUTION: The motor speed should be below 800 steps/second, when the VXC executes this command, to prevent an excessively hard stop that may cause a mechanical overshoot of intended position.

¹With the U65 and U66 commands (Master Stop/Kill) it is possible to accomplish a coordinated stop over long distances and time without calculating distances at speeds to exact ratios.

Operation Commands

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

Online/Off-Line

- E** Enable On-Line mode with echo on. The single character "E" is used to put the VXC in the On-Line mode after power-up. All characters the VXC receives will be echoed back to the host.
- F** Enable On-Line mode with echo off. The single character "F" is used to put the VXC in the On-Line mode after power-up. No characters will be echoed back to the host. The VXC will still respond to motor position and status requests.
- Q** Quit On-Line mode (return to Local Jog/slew mode.) The "Q" command is used to get back to the power-up state, where the VXC is in the Local Jog/slew mode (On-Line light is off.)

Clear Previous

- C** Clear all commands from the currently selected program. All setup values, motor position values, and the state of user outputs will not be altered.
- N** Null (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).

Run/Start

- R** Run currently selected program. The "R" command will start execution of commands stored (of current program) in the VXC's memory. At the end of the "run" the single character "^" (no carriage return or line feed follows the "^" unless put unless the VXC was put On-Line with the "G" command) will be transmitted to the host. Additional "R" commands received by the VXC will repeat the same program. Refer to the "C" and "PM-x" 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.

Interrupt Motion

- K** Kill 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 800 steps/sec. when the interrupt occurs, the motor may lose position due to mechanical overshoot (see the "D" command for a less abrupt method to interrupt indexes). The VXC will transmit the "^" to the host after receiving the "K" command.
- D** Decelerate to a stop (interrupts current index in progress, default function of Stop button too). When the VXC 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 VXC will then proceed to the next command in the program.

Status Request Commands

Help Menu

Help Display context sensitive help screen (for use with terminal program interfacing) Sending "Help" to the VXC will list the most basic commands pertinent to the current mode the Controller is in.

Help Menu in Jog Mode

```
VXC Jog Mode Commands
V  =Status: J=Jog,b=Busy
E  =Enable OnLine echo on
F  =Enable OnLine echo off
X  =Read Motor 1 Position
For more info go to VelmexControls.com
```

Help Menu in OnLine Mode

```
VXC Information Commands
V  =Status: R=Ready,B=Busy,F=Fault
X  =Read Motor 1 Position
lss =list current settings
lst =list commands in program
getFAC =get current Faults & details
getF- =get last Fault stored
indexF =index list of all Faults
VXC Mode Commands
Q  =Quit OnLine return to Jog Mode
VXC Motion Commands
Ix =Index (move) x steps
Sx =Speed x steps/sec (1-6000)
VXC Program Commands
R  =Run current program
C  =Clear program commands
For more info go to VelmexControls.com
```

Motor Position

- X** Send position of motor 1 to the host. When the VXC receives the single character "X" it will transmit the value from its motor 1 Absolute Position Register. This is what the host would receive if motor 1 is at negative 1200: **-0001200**<cr>
This command can be used when the motor is indexing. Refer to the "N" command for how to zero the Absolute Position Registers.
- Y** Send position of motor 2 to the host. When the VXC receives the single character "Y" it will transmit the value from its motor 2 Absolute Position Register. This is what the host would receive if motor 2 is at negative 200: **-0000200**<cr>
Motor must be stationary. Refer "N" command to zero the Absolute Position Registers.
- Z** Send position of motor 3 to the host. When the VXC receives the single character "Z" it will transmit the value from its motor 3 Absolute Position Register. This is what the host would receive if motor 3 is at positive 30000: **0030000**<cr>
Motor must be stationary. Refer "N" command to zero the Absolute Position Registers

- T** Send position of motor 4 to the host. When the VXC receives the single character "T" it will transmit the value from its motor 4 Absolute Position Register. This is what the host would receive if motor 4 is at positive 930005: **0930005<cr>**
Motor must be stationary. Refer "N" command to zero the Absolute Position Registers

Verify Status

- V** Verify Controller's status, when On-Line the VXC sends a "B" to the host if it is busy, or an "R" if it is ready. If the VXC is in a level 3 fault (flashing fault) "F"¹ will be returned. The "V" command is used to poll the VXC to see if it is busy running a program, or ready to receive more commands.

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


¹ "getF" can be used to read the fault or "K" can be used to kill the flashing fault.

NOTE: The following commands are only available in On-Line mode

Capture Position

- !** Capture motor positions for later recall with "x", "y", "z", "t" commands. Captures motor 1,2,3, & 4 motor positions into a FIFO buffer (4 positions per axis maximum)

NOTE: buffered data is automatically zeroed at the start of every run.

 See Appendix J for setting Capture Motor Position on Input 4 Trigger

Retrieve Captured Positions

- x** Send last 4 positions of motor 1 to host that were captured by the "!" command.

NOTE: buffered data is automatically zeroed at the start of every run.

This example shows the values the VXC returned when the "!" was sent at positions 521, 919, and 1149 while motor 1 was moving.

+0000521

+0000919

+0001149

+0000000

- y** Send last 4 positions of motor 2 to host that were captured by the "!" command.

NOTE: buffered data is automatically zeroed at the start of every run.

- z** Send last 4 positions of motor 3 to host that were captured by the "!" command.

NOTE: buffered data is automatically zeroed at the start of every run.


- t** Send last 4 positions of motor 4 to host that were captured by the "!" command.

NOTE: buffered data is automatically zeroed at the start of every run.

Position When Decelerated


- * (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 its deceleration at position negative 14901.

-14901<cr>

 See Appendix J for more information.

Read Analog Value

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

 See Appendix E for more information.

List All Settings

lssmM, List all the VXC current settings from axis *m*, *m*= motor# (1,2,3,4)¹
Example to list settings from axis 1: **lss1M,**

Example VXC output listing from above command:

```
Firmware Version:  getD0  5.00
Firmware Date:    getD1  05-01-24
Serial Number:    getD4  2405-1
Last Fault:       getF   40
NV Stored Fault:  getF-   44
Axes Found:       getAF   12
Axes Set:         getAS   12
CONFIG Mtr,Lmt,Hm: set/getMLH=  C,-1,0,
Motor Type:       set/getMT=  C
Limit Function:   set/getL   -1
Home & Stall:     set/getHS   0
Backlash:        set/getK   0
Operating Mode:   set/getDM   1
  getDMc for list
Run,Stop,Input3: set/getI   7
  getIc for list
Motor Hold:       set/getMH   0
Jog Settings:    set/getMJ   0
  getMJc for list
Outputs at Pwr-Up: set/getOUT  0
  getOUTc for list
Steps/Pulse:     set/getP   0
Pulse Width:     set/getPA   1
Jog Speed D Low: set/getj   2000
Jog Speed D High: set/getJ   2000
Jog Range A Low: set/getjA   0
Jog Range A High: set/getJA   0
Jog A Deadband:  set/getDA   50
```

NOTE: The VXC also sends the ASCII character <EOT> (End-Of-Transmission, decimal value 4) at end of listing the settings.

Example listing all settings on a single axis VXC (VXC-1): **lss,**

¹ The default motor is the last motor selected when using the “mM” designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the “mM” in a command is not required.

Backlash Compensation

getKmM Read Backlash Compensation setting for axis m, off when value returned=0 (default) The number of “comp” steps will be returned when set.

👁 See “**setKmM**” command for more information

Motor Type

getMTmM= Read motor type set for motor/axis m

👁 See “**setMTmM=**” command for more information

Limit Mode

getLmM Read mode of limits for motor/axis m

👁 See “**setLmM**” command for more information

Firmware Status Request

There are two special commands that get additional status from the VXC.

getDx Read mode/version/date code/serial number

getD0 Gets the VXC’s firmware version in the format X.XX

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

Jog/Joystick Settings

Digital Jog

getjmM Read first range Jog Speed for motor m.

👁 See “**setjmM**” command for more information

getJmM Read second range Jog Speed for motor m.

👁 See “**setJmM**” command for more information

Analog Joystick

getDAmM Read Joystick Deadband setting for motor m.

👁 See “**setDAmM**” command for more information


getjAmM Read first range analog joystick Speed for motor m.


👁 See “**setjAmM**” command for more information

getJAmM Read second range analog joystick Speed for motor m.


👁 See “**setJAmM**” command for more information

Pulse Output Settings


getPmM Read “Pulse Every x # Steps” value for axis m
 See “**setPmM**” command for more information

getPA Read Pulse width used by setPmMx and U7
 See “**setPAmM**” command for more information

User Input Mode

getI Read operating mode of user inputs
 See “**setImM**” command for more information

Motor Jog Function

getMJ Get Motor Jog function
 See “**setMJmM**” command for more information

Troubleshooting

Symptom	Cause/Remedy
At Power-up, Green LED flashes rapidly	Jog, Run, or Stop button is down (low). Release button or input.
At Power-up, Red/Green LEDs alternate flash constantly	Wrong power supply voltage, verify power supply is 24VDC
At Power-up, Red+Yellow/Green LEDs alternate flash constantly	Internal fuse issue. See Level 3 Faults for more information
VXC sends a “?” and Red LED is on or is flashing	A Fault has occurred. See Appendix F for information about faults
When motor is commanded to move, the Red LED flashes several times instead	Motor is not connected or VXC is not set for a motor type, see “ setMTmM= ” and “ setMLH1M= ” commands
When pressing a Jog button, the Red LED stays on while button is pressed and motor does not move	VXC is set to have limit switches connected but they not plugged-in or setting is incorrect, see “ setLmMx ” command
Red LED blinks on & Green LED blinks off for ½ second when motor is moving	VXC is near maximum output current. Check that motor setting is correct for motor attached
Motor stops unexpectedly and Red+Green LEDs flash together constantly	Overcurrent/voltage drop occurred, check that VXC is set to correct motor attached and power connector is tight
Motor has low torque, stalls, makes noise but does not move	Check that VXC is set to correct motor attached, motor connector is fully mated, and cables where not altered/lengthened
Motor gets very hot (too hot to touch)	Check that VXC is set to correct motor attached and use the 70% or 40% power option when setting the speed. Motor not mounted to a metal surface to dissipate heat.

Symptom	Cause/Remedy
Motor vibrates excessively between 100 and 2000 steps per second	Wrong motor setting or power is too high for load. Use 40% power (S-x for speed) if light to no load application.
Motor vibrates excessively between 2000 and 3000 steps per second	Wrong motor setting or power is too high for load. Use 70% or 40% power (Sx or S-x for speed) if light to no load application.
Motor vibrates excessively between 3000 and 6000 steps per second	Wrong motor setting or power is too high for load or motor needs dampening. Add motor damper or use 70% or 40% power (Sx or S-x for speed) if light to no load application.
Motor stops turning (stalls) when commanded to run at high speed and just makes noise.	Wrong motor setting, power is too high or too low for load, motor torque inadequate for load (see Motor Torque Curves), or motor damper needed.
Computer has intermittent connection issues with VXC over USB	Cable or connectors on USB ends are faulty. Replace USB cable.
Unable to communicate with VXC	Wrong COM port selected on PC, Wrong baud rate, data bits, or stop bits. Missing driver for virtual COM port. Verify COM port by unplugging/plugging in to see what ports changed on PC, Set port to 57600 baud, 8 data, No Parity, 1 Stop. Run VXC Utility App to diagnosis connection.

Technical Specifications

Environmental Requirements

Ambient Operating Temperature: 35°-95° F (2°-35° C)

Relative Humidity: 10%-90% (non-condensing)

Model VXC-1

Function

One Axis Motor Controller with Half-step/Micro-step 31.25 kHz PWM motor drive for size 11 to 34 hybrid stepping motors.

Physical

Weight: 1.3 lbs (0.57 kg)

Height (without feet): 1.5" (38.1 mm)

Width: 3.90" (99.1 mm)

Length: 4.42" (112.3 mm)

Cabling

Integrated 10 ft (3 meter) long Motor Cable, 10 ft (3 meter) Detachable Limit Cable

Electrical Requirements

24VDC 3 Amps

I/O

All inputs & outputs impedance and shunt diode protected

Outputs: 5V TTL 25mA maximum

Inputs: 5V TTL compatible active low (4.7k ohm internal pull-up)

Limits and Home input opto-isolated (powered by onboard user 10V)

Serial Ports

USB: USB 2.0 standard, micro B connector, electrostatic discharge (ESD) protection > 4 kV

Host controlled/programed with USB-2.0 and RS-422 Interfaces, 8 Data, No Parity, 1 Stop, 57600 (default), 9600, 19200, 38400 baud rate settable.

RS-422 (Configurable to RS-232): Four wire full duplex transceiver with ±18-kV IEC ESD protection

Signals: Tx-, Tx+, Rx-, Rx+

Biasing Resistors on Rx: 4.7k Ohms

Termination: None

Model VXC-2

Function

Two Axis Motor Controller with Half-step/Micro-step 31.25 kHz PWM motor drive for size 11 to 34 hybrid stepping motors.

Physical

Weight: 2.6 lbs (1.14 kg)

Height (without feet): 3.0" (76.2 mm)

Width: 3.90" (99.1 mm)

Length: 4.42" (112.3 mm)

Cabling, Electrical Requirements, I/O, Serial Ports

Same as VXC-1 Except 2x Power and Cabling

Model VXC-3

Function

Three Axis Motor Controller with Half-step/Micro-step 31.25 kHz PWM motor drive for size 11 to 34 hybrid stepping motors.

Physical

Weight: 3.9 lbs (1.71 kg)

Height (without feet): 4.5" (114.3 mm)

Width: 3.90" (99.1 mm)

Length: 4.42" (112.3 mm)

Cabling, Electrical Requirements, I/O, Serial Ports

Same as VXC-1 Except 3x Power and Cabling

Model VXC-4

Function

Four Axis Motor Controller with Half-step/Micro-step 31.25 kHz PWM motor drive for size 11 to 34 hybrid stepping motors.

Physical

Weight: 5.2 lbs (2.28 kg)

Height (without feet): 6.0" (152.4 mm)

Width: 3.90" (99.1 mm)

Length: 4.42" (112.3 mm)

Cabling, Electrical Requirements, I/O, Serial Ports

Same as VXC-1 Except 4x Power and Cabling

Power Supply

Model: Cincon #TRH70A240-11E03-Level-VI¹

Function

Switch Mode Desktop Power Supply

Complies with IEC/EN/UL 62368-1, EN55032 and CISPR/FCC Class B, CoC Tier 2 and DOE Level VI

Physical

Weight: 1.0 lbs (0.45kg)

Height: 1.22" (31.0 mm)

Width: 2.05" (52.0 mm)

Length: 4.72" (120.0 mm)

Output Cable: Integrated 1.8 meter (71in)

AC Cord included: NEMA 5-15P plug, 0.9 meter (3ft) long (other cords available on request)

Electrical Requirements

100-240VAC 1.5A 47-63Hz

Output (to VXC) 24VDC 3A

¹ One Power Supply per axis.

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 gives you specific legal rights which vary from State/Region to State/Region.

Appendix A (Motor Settings)

Setting Motor Type

The VXC must be set for the exact type motor is connected to it. The motor type is based on a specific combination of current, voltage, induction rating of the motor. A motor should be connected before powering the VXC to allow motor auto-connected detection. **NOTE: The red LED will Flash 6 Times When Attempting to run Without a Motor Connected.**

Use the **setMTmM=a** to set the motor(s) for each axis m to value “a” in Table 4 Motor Type Setting.

IMPORTANT: Always use “**rss**” command after set to permanently save

Table 4 Motor Type Setting

a	Motor Model (base number less suffixes)	Velmex Catalog Number	Smart Speed (SmMX)
-	No Motor	-	-
ALP	28CM013 ¹	VML113-1.2-S	6000
A	ST5918X3008	VMN231-4.2-D	6000
B	ST5918S3008	VMN232-4.2-D	4500
C	42CM06 ¹	VML173-2.5-D	5000
D	57CM06 ¹	VML231-3.0-D	6000
E	57CM13, 57CM22C	VML232-4.0-D, VML233-5.0-D	4000
F	86CM35	VML341-4.0-D	2500

NOTE: The red LED will Flash 3 Times When Attempting to run a Motor When the VXC has not Been set for a Motor.

Table 5 Motor Type Setting for Legacy 6 Wire Motors (Half Winding)

a	Motor Model (base number less suffixes)	Velmex Catalog Number	Smart Speed (SmMX)
DMN	PK245-01	-	- ²
DIJ	PK264-03	-	5500
DMN	PK266-03	-	- ²
DPQ	PK268-03	-	- ²
F	PK296-03	-	2500

¹ These motors are restricted to 70% maximum power when operating at 28V.

² Do not use Smart Speed with this motor, it is not applicable to this legacy motor

To read the type motor set use the “getMT=” command.

getMTmM= get motor type for axis *m* (*m*= axis# 1,2,3,4.)

The value returned will indicate if a motor is connected along with the motor type. The “-“ before the value indicates there is not a motor connected. A “- -” means there is no motor connected and there is not a motor type set. There will be an additional suffix “&x” after the value if the VXC has been set to compensate for different cable length²

Table 6 “get Motor Type” Examples

Value returned by getMT=	Meaning
--	Motor not connected and VXC not set to a motor
-	Motor is connected but VXC not set to a motor
-A	Motor is not connected, VXC set for motor type A
C	Motor is connected, VXC set for motor type C
B&3	VXC set for motor type B with compensation for 30 ft cables ²

²NOTE: Different cable lengths require a special factory setting to compensate for loss/gain in the added/shortened cable. Torque losses in long cables can be significant resulting in motor stalling.

Refer to “Cable Length Versus Torque” for more information on long cables.

Non-Standard Motors

The VXC is designed to run motors with the following characteristics.

Motor Type: **4-Wire, 6-Wire, or 8-Wire 2-Phase Permanent Magnet Stepper Motors**

Maximum Induction (L)= **4 Mh**

Motor Rated Current (I)= **0.4A to 5A**

Maximum product of Inductance & Amps: Maximum L (Mh) x I (Amps) = **12**

! CAUTION: Incorrect Motor Settings can Damage VXC and Motor

Contact Velmex technical support at Support@Velmex.com for correct setting for motors not listed in Table 4 Motor Type Setting & Table 5 Motor Type Setting for Legacy 6 Wire Motors (Half Winding)

Appendix B (Limits/Home/Stall Detect)

Limit Switch Inputs

The VXC by default recognizes both normally closed (N/C to run) and normally open (N/O to run) limit switches, this is the auto-detect mode for limit switches.

! CAUTION: In auto-detect mode, If the device is on (actuating) one of the limits at power-up the VXC will not know for certainty if the switches are N/C or N/O type. **Limit switch mode should be set to the type of switches attached rather than the default auto-detect.**

If the VXC limit inputs are used with normally open (hall effect type), or with a home switch, the limit switch mode must be set for reliable operation. The “setLx” command is used to set the operating mode of the limit switch inputs. When a limit input is triggered, motion will stop in the direction of travel, the red LED will light, and a number 42 fault (Hit Limit Switch) will be logged. Refer to

Table 7 Limit Input Settings for optional settings for what occurs when a limit switch is encountered.

setLmMx Set Limit Switch mode for axis m, m= motor# (1,2,3,4)

IMPORTANT: Always use “rss” command after set to permanently save

getLmM Get Limit Switch mode setting for axis m, m= motor# (1,2,3,4) value returned will be value x in Table 7 Limit Input Settings.

Table 7 Limit Input Settings

Description	X					
	- Limit Function ²	+ Limit Function ²	Base Value ³	End Program & Send “?” Fault	End Program & do Flashing Fault	End Program & Run Program 11
Auto-detect N/C to run (default) ¹	N/C	N/C	0 ⁴	8 ⁴	16 ⁴	32 ⁴
Enabled –Limit N/C, +Limit N/C to run	N/C	N/C	1	9	17	33
Enabled –Limit N/O, +Limit N/O to run	N/O	N/O	-1	-9	-17	-33
When the +Limit and -Limit switches are a different output type						
Enabled –Limit N/C, +Limit N/O to run	N/C	N/O	1A	9A	17A	33A
Enabled –Limit N/O, +Limit N/C to run	N/O	N/C	-1A	-9A	-17A	-33A
When limit inputs are used as a home input (NOTE: setSHx command will override these settings)						
Disabled N/C for Home Switch use ⁵	N/C	N/C	2			
Disabled N/O for Home Switch use ⁵	N/O	N/O	-2			

¹ The VXC will attempt to determine function of limits by state of inputs at power-up. If both inputs are high, mode will be N/O. If both inputs are low, mode will be N/C. If one input is low and other is high the last mode previously determined will be used.

² N/C = Normally Closed contact is the state of the switch when not actuated and state when the motor can move.

N/O = Normally Open contact is the state of the switch when not actuated and state when the motor can move.

³ When set to base value, a #42 fault will be logged, & movement will stop in limit direction and program will continue at the next command.

⁴ These values will return as negative when using “getLmM” if N/O limits have been detected.

⁵ **ONLY USE WHEN A HOME SWITCH IS CONNECTED TO BOTH +LIMIT & -LIMIT INPUTS! (Do Not Use if home switch is connected to the Home Input) NOTE:** “setSHx” command will override home settings set by “setLx” (Refer to the “setSHx” command for more information).

Figure 4 Limit Switch Input Hardware



Home/Stall Detect Input

The VXC has a dedicated Home/Stall input that can be configured several different ways. When used as a dedicated home input it allows the limit switches to coexist with a home input. Verification of motor position is possible if the input is used with the Velmex stall detect option on the motor² The “**setHSx**” command is used to set the operating mode of the home input.

setHSmMy Set Stall/Home input for axis m, m= motor# (1,2,3,4)

IMPORTANT: Always use “**rss**” command after set to permanently save

getHSmM Get Stall/Home mode setting for axis m, m= motor# (1,2,3,4) value returned will be **y** value in Table 8 Home/Stall Input Settings.

Table 8 Home/Stall Input Settings

Description	y
Disable Home Input	0
Enable Home for N/O switch ¹	16
Enable Home for N/C switch ¹	17
Enable Home for N/O switch with decelerate to a stop ¹	20
Enable Home for N/C switch with decelerate to a stop ¹	21
Enable Stall Detect with Send “?” Fault ²	128
Enable Stall Detect with Send “?” Fault and Run Program 11 ²	129
Enable Stall Detect with Send “?” Fault and Do Flashing Fault ²	130

¹ These settings will override the “**setLx**” command “Disabled for Home Switch” setting, see **ImMO** index homing command.

² Advance Option that requires additional sensor on motor, contact Velmex for more information.

Figure 6 Home Switch Input Hardware

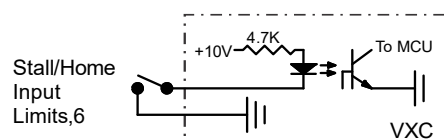
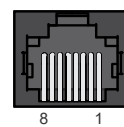


Figure 5 Limits Connection

VXC Limit Connection



RJ45 Modular Jack

Pin#	Name
1	Limit CW (+)
2	0V (Common Ground)
3	Limit CCW (-)
4	+10V Output
5	Chassis Ground
6	0V (Common Ground)
7	Home Input
8	0V (Common Ground)

Limits, Home, Stall Detect Function Interaction

Since the VXC inputs are multifunctional, some functions conflict with each other and home settings will override limit settings. There are two different places to connect a home switch, to limit switch inputs tied together or to the dedicated Home Input. Table 9 Limit, Home, Stall Settings Interaction shows the interaction of the limit, home, and stall functions.

Table 9 Limit, Home, Stall Settings Interaction

setLmMx	Limit	setHSmMy	Home	Stall
x = 0,8,16,32; ±1,9,17,33	✓ ²	y = 0		
x = 0,8,16,32; ±1,9,17,33	✓ ²	y = 16,17,20,21	✓ ⁵	
x = 0,8,16,32; ±1,9,17,33	✓ ²	y = 128,129,130	7	✓ ⁴
x = 2,-2	3	y = 0	✓ ⁶	
x = 2,-2	3	y = 128,129,130	✓ ⁶	✓ ⁴
x = 2,-2	3 ^A	y = 16,17,20,21	✓ ⁵	

¹ Requires optional sensor hardware on motor

² Limits enabled

³ Limits disabled & Limit inputs available to use with home switch

^{3A} Limits disabled, Home Input used with home switch

⁴ Stall detect enabled

⁵ Home enabled, home switch connected to Home Input

⁶ Limits disabled, home switch connected to both +Limit and -Limit inputs

⁷ Either limit can be used as a home reference

⁸ Limit switch will stop motion immediately and a hit limit (#42) fault will be logged

Device Compatibility with Limits, Home, Stall Detect

Table 10 Device Combability with Limits, Home, Stall and Table 2 Settings for Limits, Home/Stall shows that a dedicated home switch is not compatible with both limits and stall detect. If limits are not needed, then the limits inputs can be set for dedicated use with a home switch. When stall detect, limits, and a method for homing are required, either limit would be used as a home reference.

Table 10 Device Combability with Limits, Home, Stall

Linear Actuators/Rotary Tables	Limits	Home ⁵	
Linear Actuators/Rotary Tables	Limits (Use a limit for home ⁸)		Stall Detect
Rotary Tables		Home ⁶	Stall Detect

Using a Limit for Home Reference

When a dedicated home switch is not possible or necessary, like when the optional stall detect is needed, the limit switches can be utilized as a home reference.

Programming sequence for homing to a limit switch

1. Set a speed for homing (maximum of 1000.) Always use this selected speed for homing to maintain repeatability.
2. Set move to home/limit command (**ImM0**, or **ImM-0**)
3. Move a fixed distance from the limit and zero motor position at this position.

This example homes motor 1 moving negative direction into limit switch, then moves positive 400 steps, and zeros position.

Example 15 Move negative into limit switch, then back 400 steps, & zero position

C S1M800,I1M-0, I1M400,IA1M-0,R	Bytes: 15
←	
Stopped by -Limit Switch ----- Start , Index to "Home"	
→	
Index 400 --- End	
0	

NOTE: A limit switch will stop motion immediately and a hit limit (#42) fault will be logged

Reasons to Use a Dedicated Home Switch

There are advantages to using dedicated home switches versus the limit switches as a home reference.

1. A dedicated home allows a home reference to be anywhere along travel length.
2. Limit switches will always be for indicating an overtravel fault when there is a dedicated home.
3. Limit switches would indicate a home switch failure if a limit was encountered while homing to a dedicate home switch.

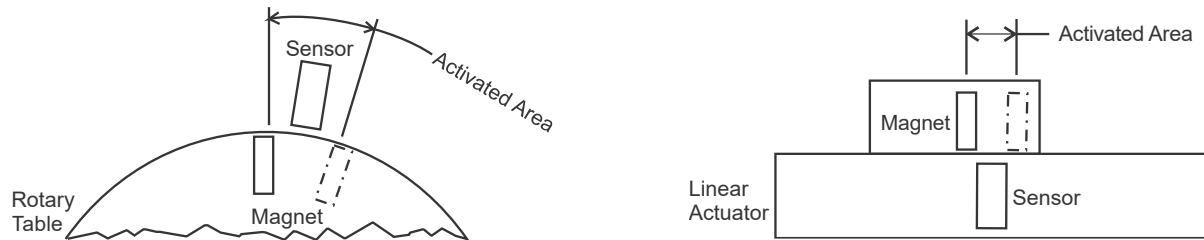
Using a Home Switch on Home Input

When properly applied, a home switch provides a high degree of precision and accuracy as an absolute starting reference. Repeatability of 1 motor step is achievable if the proper procedures are followed when referencing to a home switch.

For a hall effect or reed type switch use “Enable Home for N/O switch” setting.

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

Figure 7 Home Switch Active Area



Homing to a home switch on rotary tables without limit switches, and when home switch is at limit end on devices with limit switches

1. Set a speed for homing (maximum of 1000.) Always use this selected speed for homing to maintain repeatability.
2. Set move to home command (**ImM0**, or **ImM-0**) Pick a direction to move to the home switch. Always use this direction to maintain accuracy. If the slide/table is already in the active area of the switch the slide/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 ensure slide/table will be totally out of activated area of the switch before doing final homing. Typically, 4000 steps should be adequate to move beyond the active switch area. Refer to “Home Input for Measuring Hysteresis” to determine exact active area of switch.
4. Set move to limit command (**ImM0**, or **ImM-0**) and zero motor position at this position if desired.

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

Example 16 Move negative into home switch and zero position Example

C S1M600,I1M-0,I1M4000,I1M-0,IA1M-0,R	Bytes: 19
<pre> Stopped by Home Switch -----←----- Start, Index to Home -----→----- Index 4000 ----- -----←----- Stopped by Home Switch ----- Index to Home End 0 </pre>	

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

Example 17 Move positive into home switch and zero position 1000 steps away

C S2M800,I2M0,I2M-4000,I2M0,I2M-1000,IA2M-0,R	Bytes: 23
<pre> Start, Index to Home -----→----- Stopped by Home Switch ← ----- Index -4000 → Index to Home ----- Stopped by Home Switch End -- Index -1000 0 ← </pre>	

Special consideration must be made when the home switch is located near center of travel. Using a home switch when there is available travel on both sides of the switch requires referencing a limit switch first. Moving to a limit first guarantees the home move will start on the intended side of the home switch.

Programming sequence for homing to a centrally located home switch with limit switches

1. Set a speed for homing (maximum of 1000.) Always use this selected speed for homing to maintain repeatability.
2. Set a move which is at least as long as the overall travel distance in the opposite direction that will be used for homing.
3. Set move to home command (**ImM0**, or **ImM-0**) Pick a direction to move to the home switch. Always use this direction to maintain accuracy. If the slide/table is already in the active area of the switch the slide/table will not move. Steps 3 and 4 below will compensate for this situation.
4. Set move to limit command (**ImM0**, or **ImM-0**) and zero motor position at this position if desired.

This example homes motor 1 moving negative direction into limit switch, then positive to home switch, and zeros position.

Example 18 Move negative into limit switch, then + to home switch, and zero position

C S1M700,I1M-50000,I1M0,IA1M-0,R	Bytes: 15
<pre> Stopped by -Limit Switch -----←----- Start, Index 50000 → Index to Home ----- Stopped by Home Switch 0 End </pre>	

Appendix C (Referencing/Feedback)

More Feedback & Precision

There are four ways to improve system precision and accuracy with the VXC.

1. Use the home switch or limit switch to establish a home position
2. Enable backlash compensation feature for higher accuracy when changing direction
3. Set the limit switch settings to indicate to the host computer that a positioner has inadvertently hit a limit switch
4. Use the stall detect option to verify motion

Limit or Home Switch for Initial Reference

The limit switches or an optional home switch on a typical Velmex assembly can provide an absolute position reference with a repeatability of better than 0.0004" (0.010 mm.) Referencing a limit or home switch after initial power-up is the simplest method to insure consistent repeatability from a precision positioning system. See Appendix B (Limits/Home/Stall Detect) for home input configuration, for configuring limit switch inputs, and setting program 11 to do homing routines at power-up.

Example 19 Typical Homing Routine (speed 500, move to +Limit, move back 200, zero position)

<code>S1M500, I1M0, I1M-200, IA1M-0, R</code>	Bytes: 15

Backlash Compensation to Improve Accuracy

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

setK m M x Set backlash compensation for motor, m = motor# (1,2,3,4)¹, x = 1 to 255 steps. Backlash compensation is 20 steps when =1, off when =0 (default). Desired number of “comp” steps can be input directly (x = 2 to 255.) The VXC 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. **NOTE:** The VXC does 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. Immediate command (not stored)

IMPORTANT: Always use “rss” command after set to permanently save

Backlash Compensation Set Examples

This example sets the backlash compensation on motor 2 to 1 (20 steps):

```
setK2M1<cr>
```

This example sets the backlash compensation on motor 1 to 30 steps:

```
setK1M30<cr>
```

This example disables backlash compensation on motor 3:

```
setK3M0<cr>
```

IMPORTANT: Always use “rss” command after set to permanently save

getK m M Read current backlash compensation setting for motor, m = motor# (1,2,3,4), off when value returned= 0 (default) The number of “comp” steps will be returned when set.

Indicate Limit Switch Encounter

A properly connected and configured limit switch will always stop a motor immediately. If the VXC is commanded to always run within the range of the limits then the limit switches will never interfere with the movement. If the motor receives an erroneous move command or jams in one direction then it is very likely that a limit switch will be encountered. Normally the host computer would not know if a limit switch stopped a move. With the “setLmM x ” command the VXC can be set to notify the host computer or do a flashing fault that it has hit a limit switch. Refer to Appendix B (Limits/Home/Stall Detect) for more information on the “setLmM x ” command.

Stall Detect Option to Verify Movement

Stepping motors by default are operated in open loop mode where the motor runs synchronously to its commanded position. Repeatably in stepping motor systems is extremely reliable when sound wiring practices are followed and motors are loaded below their maximum torque output. If the motor is overloaded enough to cause loss of synchronism (stall), or if there is a break in the motor wiring, then the actual motor position is not guaranteed. The VXC has a stall detection circuit that in conjunction with an optional motor mounted sensor will verify that a motor has or has not stalled. Refer to Appendix B (Limits/Home/Stall Detect) for more information on the stall detect option.

Appendix D (Digital Jog)

Advanced Digital Jog Mode

When the On-Line (yellow) light is not lit, the VXC 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 step¹ (1/400 rev. default or 1/4000 rev. option) If the button is held for >0.3 second¹ the motor will accelerate¹ to a default speed of 2000 sps.

Pressing/holding the Stop button or the opposite Jog button while using Jog will hold the speed at 63 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

setjmMx Set primary maximum jog speed x , m = motor# (1,2,3,4), x = 1 to 6000 sps (default=2000 at 70% power, + x for 100% power, - x for 40% power)

setjmM0 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)

setJmMx Set secondary maximum jog speed x , m = motor# (1,2,3,4) x = 63 to 6000 sps (default=2000 at 70% power, + x for 100% power, - x for 40% power.) When input 2 is low (I/O,6) this speed will be the maximum jog speed.

IMPORTANT: Always use “rss” command after set to permanently save

getjmM 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)

getJmM Get the secondary maximum jog speed. The value returned will be a number between 63 and 6000 (default=2000) m = motor# (1,2,3,4)

¹ Refer to Table 11 Jog Function Settings for changing these values

Optional Digital Joystick

The optional digital joystick allows remote jog control of a one or two axis VXC 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.)

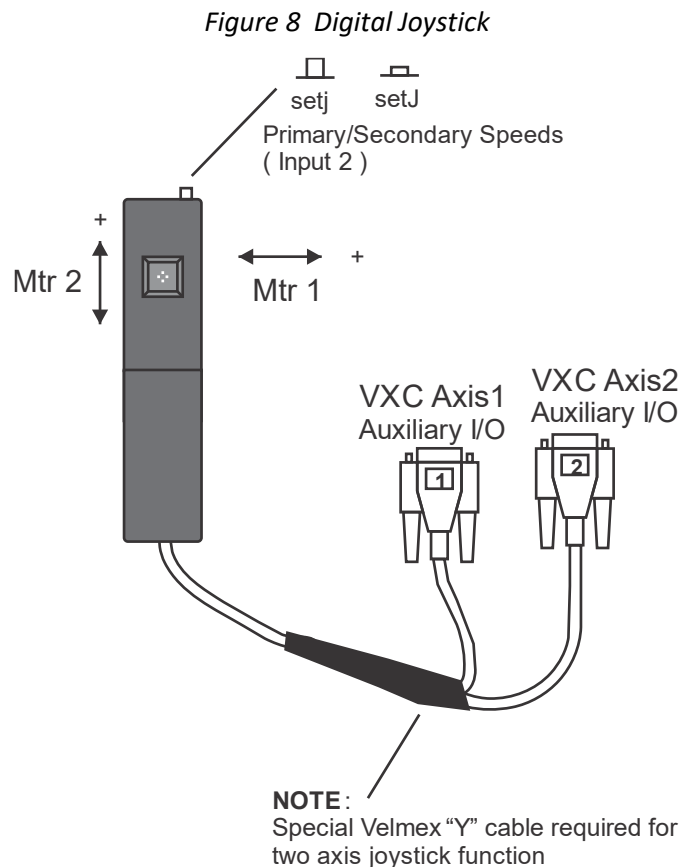
See Advanced Digital Jog Mode for information on jog function and speed settings.

Heavy duty and joysticks without the enclosure are available for OEM applications.

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 cable's connector.



Refer to Table 13 Joystick Y Cable Connections for joystick cable requirements

Jog Function Settings

Optional Jog settings are available for how the digital jog responds to the jog buttons/inputs. There is an 8-bit value for the settings (see Table 11 Jog Function Settings) to read and write with the following commands.

setMJmMx Set Motor Jog function decimal number x between 0-255, m= motor# (1,2,3,4)
setMJmMb,x Set Motor Jog function 8 bit binary number x between 0000000-11111111.
setMJmMb,x,y Set Motor Jog function bit number x of 0-7, state value y of 0 or 1.

getMJmM Get Motor Jog function as decimal 0-255 number.
getMJmMb Get Motor Jog function as binary 0000000-11111111 number.
getMJmMb,x Get Motor Jog function state 0 or 1 of bit x (x=0-7)
getMJmMc Get Motor Jog function complete detail list of settings.

Table 11 Jog Function Settings

Bit#	7	6	5	4	3	2	1	0
Decimal	128	64	32	16	8	4	2	1
	Slower ¹ Acceleration 1=set	Faster ¹ Acceleration 1=set	Slower ² Deceleration 1=set	Faster ² Deceleration 1=set	Longer Pause ³ after 1 st step 1=set	Shorter Pause ³ after 1 st step 1=set	-Jog runs program #7 +Jog runs program #8 1=set	0= half-step ⁴ first step 1= μ step first step
Default	0	0	0	0	0	0	0	0

¹ Default Acceleration= 2000 sps², Slower= 1333 sps², Faster= 4000 sps²

² Default Deceleration= 16000 sps², Slower= 8000 sps², Faster= 32000 sps²

³ Default Pause= 0.4 seconds, Longer Pause= 0.7 seconds, Shorter Pause= 0.2 seconds

⁴ Default half-step= 0.9 degrees of rotation, μ step= 0.09 degrees of rotation

This Example will return current setting in decimal format for motor 1.

getMJ1M<cr>

VXC will send: 0<cr>

This Example will set bit 0 which is “ μ step first step” for motor 2.

setMJ2Mb0,1<cr>

This Example will return current setting in binary format for motor 2.

getMJ2Mb<cr>

VXC will send: 00000001<cr>

This Example will set bit 6 which is “Faster Jog Acceleration” for motor 1.

setMJ1Mb6,1<cr>

“**getMJ1Mb<cr>**” will return “10000001<cr>”

“**getMJ1M<cr>**” will return “129<cr>”

“**getMJ1Mb7<cr>**” will return “1<cr>”

This Example will set Jog functions on motor 1 back to default.

setMJ1Mb,00000000<cr>

IMPORTANT: Always use **rss** command after set to permanently save

This Example will list the complete detailed Jog settings.
getMJc

Below is what the VXC will send back.

```

* 76543210
# 00000000
(Default Setting)
  7,#1: Slower Accel
*7,#0:(-----)
  6,#1: Faster Accel
*6,#0:(-----)
  5,#1: Slower Decel
*5,#0:(-----)
  4,#1: Faster Decel
*4,#0:(-----)
  3,#1: Longer Pause
*3,#0:(-----)
  2,#1: Shorter Pause
*2,#0:(-----)
  1,#1: Jog-/+ = Pgms 7/8
*1,#0:(-----)
  0,#1: uStep 1st Step
*0,#0:(HalfStp 1st Step)
To change: setMJb*,#
  
```

Note: Factory default settings are indicted by ()
 The current setting is prefixed by "*"
 The Indented setting is the alternate of the current setting
 Bit #1 is set which is "-Jog runs program #7, +Jog runs program #8"

Custom Jog Distances

Normally the jog function is initially one step with slew until jog button is released. The "setMJ" command provides an option for custom increments and also a return to zero function. Setting bit 1 enables "-Jog runs program #7, +Jog runs program #8". This will set bit 1 to enable Custom Jog.

```
setMJb1,1<cr>
```

Both programs 7 and 8 must have index commands in them for the jog distances. This example will set jog distance to 400 steps (1 revolution) and the default program 0 to move to zero position when the Run button is pressed.

```
PM-7,I-400,PM-8,I400,PM-0,IA0,rsm,
```

Custom Jog additional features:

- a) Stop button zeros the motor position register.
- b) Holding Stop while pressing a Jog button reverts to the default jog function.

Appendix E (Analog/Jog)

Analog Input

The VXC has a 10-bit analog to digital converter for general use, motor speed setting, or for use with the Analog Joystick Option

The analog reference voltage is the internal +5VDC which is also used for the VXC'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 +5VDC or damage may occur to the analog input.

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 analog converter and the I/O Ain.

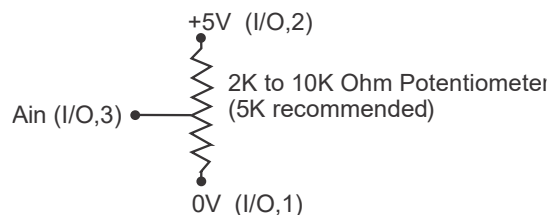
By default, the voltage at the analog input Ain (I/O,3) is +2.5VDC. The digital value of the Ain can be read directly with the "@" command. Since this is a 10-bit converter, the 2.5VDC would be equal to 512 ($\frac{1}{2}$ of 1024.) Connecting Ain to the +5VDC would return a value of 1024. If Ain is connected 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. The analog value can be assigned to a speed or an index value, see Setting & Proportioning Speed to Analog Input.

Figure 9 Analog Input Connection to External Potentiometer



Analog Joystick Option

The VXC Proportional Speed Joystick provides a precise efficient one, two, three, or four axis variable speed positioning system when used with VXC Stepping Motor Controllers.

The joystick has a button switch for a primary/secondary speed range selection.

Heavy duty and Joysticks without the enclosure are available for OEM applications.

setjAmMx Set primary joystick speed range x , m = motor# (1,2,3,4), x = 1 to 24 (default 70% power, + x for 100% power, - x for 40% power.) See Table 12 Analog Joystick Speed Ranges.

setjAmM0 Disable analog joystick for motor m . This command will deactivate the joystick for motor m (default.) m = motor# (1,2,3,4.)

setJAmMx Set secondary joystick speed range x , m = motor# (1,2,3,4), x = 1 to 24 (default 70% power, + x for 100% power, - x for 40% power.) See Table 12 Analog Joystick Speed Ranges. When input 2 is low (I/O,6) this speed range is used for joystick input.

getjAmM 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.)

getJAmM 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.)

setDAmMx Set joystick Deadband value, x = 20 to 100 (default=50) m = motor# (1,2,3,4.)

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.

getDAmM Get joystick Deadband value. Value returned is a number between 20 and 100 (default=50) m = motor# (1,2,3,4.)

IMPORTANT: Always run "rss" command to permanently save settings!

! CAUTION: The joystick must be at its self-centered position (middle) at power-up. The VXC 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.

Table 12 Analog Joystick Speed Ranges

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

IMPORTANT: Use the "rss" command to save new joystick settings

Analog Joystick Connection

The joystick provides analog outputs for two VXC axes. A connection to each axis is accomplished with a special Velmex “Y” cable.

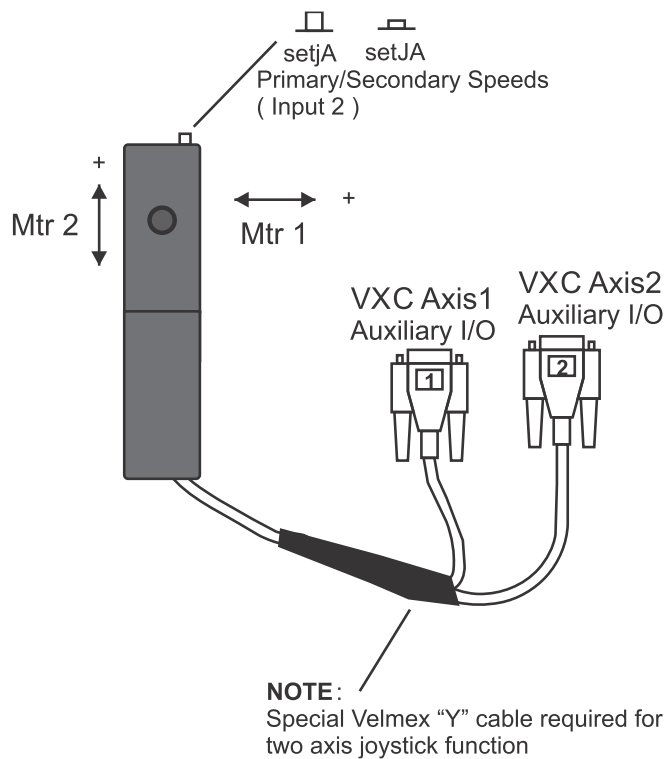
The analog joystick has a button switch connected to Input 2 for toggling between the primary and secondary settable jog speeds.

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 is are needed for another function. 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 cable’s connectors.

NOTE: The joystick will not operate Axis2 if the “Y” cable is not connected to Axis1 or if Axis1 is off. The common +5V reference voltage for the joystick comes from Axis1 (pin 2.)

Figure 10 Analog Joystick



Joystick	Function	Axis 1	Axis 2
1	Gnd	1	-
2	+5V	2	-
3	Ain Axis 1	3	-
4	-	-	-
5	Ain Axis 2	-	3
6	In2 (Spd Shift)	6	6
7	-	-	-
8	-	-	-
9	Gnd	-	9
10	J1-	10	-
11	J1+	11	-
12	J2-	-	10
13	J2+	-	11
14	-	-	-
15	-	-	-

Table 13 Joystick Y Cable Connections

! CAUTION: If Axis1 is turned off with Axis2 on, motor(s) on Axis2 will run! Use a common power strip to turn on/off all VXCs

Appendix F (Faults)

Fault Checking

The VXC does extensive fault checking with three levels of alarms dependent on the significance of the fault. Limit switches and the optional Stall detect allow user settable alarm levels. Additionally, Output 4 can be set as an alarm output for level 3 fatal faults.

Immediate Checks

At power-up the VXC checks for Run, Stop, Jog+ & Jog- being “low”. If a Jog is “low” the green LED will flash rapidly, yellow LED flashes for Run “low”, red LED flashes for Stop “low”, until the input(s) go “high”.

When a limit switch is encountered the red LED will light. The next move will turn off the red LED if a limit is not encountered.

NOTE: The red LED will flash 6 times when attempting to run a motor that was not connected at power-up. If the motor is connected but the axis is not set (refer to setMTmM=a command) for a motor type the VXC will flash the red LED 3 times when attempting to run the motor.

Faults other than immediate ones are saved/logged by the VXC. Faults will be cleared on every power-down. To read the current logged faults use the following commands.

getFmM Get last logged fault for axis *m*, *m*= motor# (1,2,3,4.)¹ See the following tables for description of values returned. A value of zero is no fault. The VXC stores up to 10 faults in a first in first out buffer. When read the value will be removed from the buffer/log.

getFAmM Get all the logged faults for axis *m*, *m*= motor# (1,2,3,4.)¹ See the following tables for description of values returned. The VXC will list out all the fault numbers currently logged and clear the fault buffer/log.

getFmMc Get last logged fault with complete description for axis *m*, *m*= motor# (1,2,3,4.)¹

Example of fault request from a VXC axis 1 after power-up.

getF1Mc

VXC will send: 40 Power Failed/Reset <cr>

getFAmMc Get all the logged faults with complete descriptions for axis *m*, *m*= motor# (1,2,3,4.)¹

getFmM- Get the last level 3 fault (flashing fault) saved for axis *m*, *m*= motor# (1,2,3,4.)¹ The VXC automatically saves the last level 3 fault in EEPROM memory. This command is useful for determining a fault after powering down the VXC to recover from the fault.

getFmM-c Get the last level 3 fault (flashing fault) with complete description saved for axis *m*, *m*= motor# (1,2,3,4.)¹

¹ The default motor is the last motor selected when using the “mM” designator in a command. At power up the VXC sets the default motor to 1. If the VXC is a one axis version, using the “mM” in a command is never required.

Level 1 Faults

Level 1 faults are change of state faults. Level one faults log the fault only.

Fault #	Description	Type
21	Motor detection failure ¹	hardware
40	Power failed (default every power up) ¹	Interrupt
41	Stop pressed (input 4 low)	Interrupt
42	Hit limit switch	Interrupt
44	First power up after factory default ¹	Interrupt

Level 2 Faults

Level 2 faults are error indicating faults. Level 2 faults log the fault, send “?” , and turn on the red LED.

Fault #	Description	LED ²	Type
11	Motor Wiring Fail ¹ (A broken connection was detected in motor wiring)	-	hardware
30	Value entered out of range	R:1	software
31	Axis does not exist	R:1	software
32	Program memory is full (see “ Mem ” and “ C ” commands for program memory)	R:1	software
36	Math result is greater than 8388607 or less than -8388607	R:1	software
42	Hit limit switch (only if enabled, see “ setLmMx ” command)	R:1	Interrupt
43	Stall detected (only if enabled, see “ setHSmM ” command)	R:1	Interrupt
45	Slave Axis 2 Fault	R:1	Interrupt
46	Slave Axis 3 Fault	R:1	Interrupt
47	Slave Axis 4 Fault	R:1	Interrupt

¹ Only at power-up will these faults occur.

² R= Red LED, 1= on

Level 3 Faults

Level 3 faults are fatal faults that stop/interrupt the VXC immediately.

Level 3 faults log the fault, send "?", flash the LEDs in a unique code pattern relative to the fault, and output 4 if "setDMb6" is set.

Fault #	Description	Flashing LEDs ²	Type
12	Voltage drop/over current	R:10 Y:00 G:10	hardware
13	Fuse blown ¹³	R:10 Y:10 G:01	hardware
14	Input voltage too low ¹	R:10 Y:00 G:01	hardware
15	Over temperature	R:10 Y:01 G:01	hardware
16	ROM memory error	R:10 Y:10 G:00	hardware
17	EEPROM memory error	R:10 Y:10 G:00	hardware
18	RS-422 overrun	R:00 Y:10 G:00	hardware
19	USB overrun	R:00 Y:10 G:00	hardware
20	Input voltage too high ¹	R:10 Y:00 G:01	hardware
22	Lost communication with Axis 2	R:10 Y:01 G:00	hardware
23	Lost communication with Axis 3	R:10 Y:01 G:00	hardware
24	Lost communication with Axis 4	R:10 Y:01 G:00	hardware
25	Slave Axis version not up to date	R:10 Y:01 G:00	hardware
26	Slave Axis 0 not valid found	R:10 Y:01 G:00	hardware
27	Slave Axis is off	R:10 Y:01 G:00	hardware
33	Continuous Indexing missing ending command	R:10 Y:10 G:10	software
34	More than 20 nested Loops	R:10 Y:10 G:10	software
35	More than 13 nested "JMx"s	R:10 Y:10 G:10	software
42	Hit limit switch (only if enabled, see "setLmMx" command)	R:10 Y:00 G:00	Interrupt
43	Stall detected (only if enabled, see "setHSmM" command)	R:10 Y:00 G:00	Interrupt
45	Slave Axis 2 Fatal Fault	R:10 Y:01 G:00	Interrupt
46	Slave Axis 3 Fatal Fault	R:10 Y:01 G:00	Interrupt
47	Slave Axis 4 Fatal Fault	R:10 Y:01 G:00	Interrupt

¹ Only at power-up will these faults occur.

² R= Red, Y= Yellow, G= Green, 1= LED on, 0= LED off, 10= on/off, 01= off/on.

Examples: "R:10 Y:00 G:00" is flash just Red with Yellow & Green off.

"R:10 Y:00 G:10" is Red + Green flash together with Yellow off.

"R:10 Y:01 G:00" is alternate flashing Red/Yellow with Green off.

Fuse Failed: "R:10 Y:10 G:01" is alternate flashing Red + Yellow with Green.

³ A blown Fuse can be verified by measuring the lack of 10V on pin 4 of Limits RJ45 connector, refer to Figure 5 Limits Connection. The removable fuse is an automotive 4-amp Mini AT size accessible by separating the top and bottom of the enclosure by first removing the screws from the front and rear panels. Before attempting fuse replacement contact Velmex technical support at Support@Velmex.com for help and the proper procedure to check or replace fuse.

Level 3 Faults by Common Flash Code

Master/Slave Communication Faults

Fault #	Description	Flashing LEDs	Type
22	Lost communication with Axis 2	R:10 Y:01 G:00	hardware
23	Lost communication with Axis 3	R:10 Y:01 G:00	hardware
24	Lost communication with Axis 4	R:10 Y:01 G:00	hardware
25	Slave Axis version not up to date	R:10 Y:01 G:00	hardware
26	Slave Axis 0 not valid found	R:10 Y:01 G:00	hardware
27	Slave Axis is off	R:10 Y:01 G:00	hardware
45	Slave Axis 2 Fatal Fault	R:10 Y:01 G:00	Interrupt
46	Slave Axis 3 Fatal Fault	R:10 Y:01 G:00	Interrupt
47	Slave Axis 4 Fatal Fault	R:10 Y:01 G:00	Interrupt

Voltage Input Faults

Fault #	Description	Flashing LEDs	Type
13	Fuse blown	R:10 Y:00 G:01	hardware
14	Input voltage low	R:10 Y:00 G:01	hardware
20	Input voltage high	R:10 Y:00 G:01	hardware

Software Range Faults

Fault #	Description	Flashing LEDs	Type
33	Continuous Indexing missing ending command	R:10 Y:10 G:10	software
34	More than 20 nested Loops	R:10 Y:10 G:10	software
35	More than 13 nested "JMX"s	R:10 Y:10 G:10	software

Hardware Failure Faults

Fault #	Description	Flashing LEDs	Type
16	ROM memory error	R:10 Y:10 G:00	hardware
17	EEPROM memory error	R:10 Y:10 G:00	hardware

Serial Port Faults

Fault #	Description	Flashing LEDs	Type
18	RS-422 overrun	R:00 Y:10 G:00	hardware
19	USB overrun	R:00 Y:10 G:00	hardware

Overcurrent Fault

Fault #	Description	Flashing LEDs	Type
12	Voltage drop/over current	R:10 Y:00 G:10	hardware

Over Temperature

Fault #	Description	Flashing LEDs	Type
15	Over temperature	R:10 Y:01 G:01	hardware

Level 3 Faults Checked at Power-up Only

Fault #	Description	Flashing LEDs	Type
13	Fuse blown	R:10 Y:10 G:01	hardware
14	Input voltage too low	R:10 Y:00 G:01	hardware
20	Input voltage too high	R:10 Y:00 G:01	hardware
27	Slave Axis is off	R:10 Y:01 G:00	hardware

Index of All Faults

To see a listing of all the possible faults with their description use the indexF command.

indexF Will list the following index of all the faults. See Table 14 Index Listing of Faults

Table 14 Index Listing of Faults

```

11 Motor Wiring Fail
12 Volt Drop/Over Amps
13 Internal Fuse Blown
14 Input Voltage Low
15 Over Temperature
16 Flash ROM Error
17 EEPROM Error
18 RS-422 Overrun
19 USB Overrun
20 Input Voltage High
21 Motor Detect Fail
22 Lost Comm Axis 2
23 Lost Comm Axis 3
24 Lost Comm Axis 4
25 Slave Version Obs
26 Bus Axis0 Not Valid
27 Slave Axis Not On
30 Value Out Of Range
31 Axis Does Not Exist
32 Program Memory Full
33 Cont Index Error
34 > 20 Nested Loops
35 > 13 Nested Jumps
36 Result > +/-8388607
40 Power Failed/Reset
41 Stop Input Occurred
42 Hit Limit Switch
43 Motor Stall Detect
44 First Power Up
45 Fault On Axis 2
46 Fault On Axis 3
47 Fault On Axis 4

```

Appendix G (Serial Ports)

Serial Port Connections

The VXC has two independent serial ports for communication with a host computer or PLC. One port is a USB standard port and the other is a full duplex RS-422 port that can be configured as a RS-232 compatible port. In Jog mode (power-up state) the VXC scans each port to see if data is being received, when data is discovered on a port, the VXC will lock on to that port making it the default communication port when On-Line.

USB Port

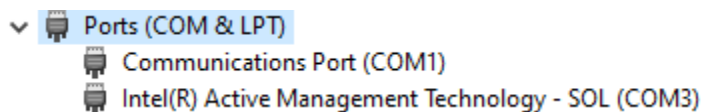
The USB port is a USB 2.0 compliant port implemented with a Microchip MCP2221A protocol converter to the VXC's MCU.

USB port specifications:

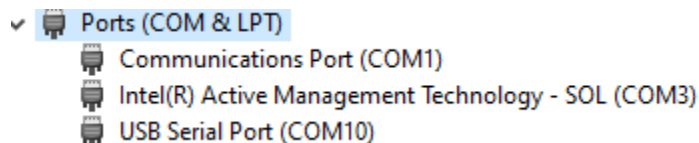
- Bus powered (10ma typical, 15ma maximum current requirement)
- Enumerates as a Composite USB Device (CDC and HID) using standard drivers for Virtual Com Port
- Compatible with: Windows XP (SP3), Vista, 7, 8, 8.1, 10, and 11. Linux® (Any distribution with support for CDC and HID classes.) Mac OS®
- 57600 default baud rate, 8 Data, No Parity, 1 Stop
- Connector: USB Micro-B Receptacle

How to Find COM Port VXC is on in Microsoft Windows

Search on "Device Manager" and select "Ports (COM & LPT)"



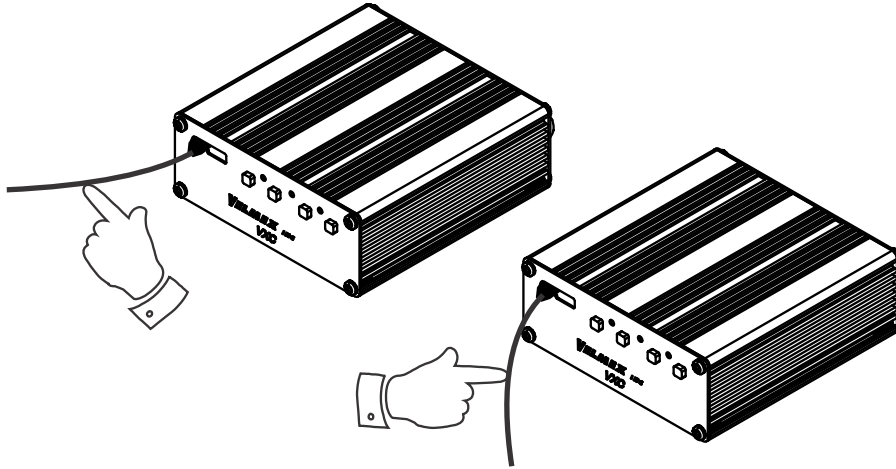
Connect USB cable from computer to VXC and observe what new COM port shows on the list.



How to Test USB Cable for a Reliable Connection

Old and low-quality USB cables can cause intermittent connection problems. By doing a simple bend the cable side to side (wiggle) test will usually identify if there is a connection issue. If your computer indicates a port disconnect/reconnect while doing this test, replace your USB cable with a new high-quality cable¹.

Figure 11 Wiggle Test on USB Cable



¹Velmex sells USB cables that have been evaluated and tested to provide error free communication with the VXC. Since the USB receptacle in the VXC has a durability rating of 10,000 cycles, any connection failures are most likely due to the cable. Contact Velmex, Inc. if you are still experiencing communication problems after replacing the USB cable.

RS-422 Port

The RS-422 port utilizes a RS-485/RS-422 transceiver that meets or exceeds the requirements of the TIA/EIA-485A Standard.

RS-422 port specifications:

- Four wire, full duplex
- Rx Pullup/Pulldown bias resistors: 4700 ohms
- Termination resistors: None
- 57600 default baud rate, 8 Data, No Parity, 1 Stop
- Connector: Molex #530480610

Figure 12 RS-422 Connection



Pin#	Name
1	Opt. +5V (out)
2	Tx+
3	Tx-
4	Gnd
5	Rx+
6	Rx-

RS-422 Adapters

There are optional RS-422 Breakout cables and a RS-422 to RS-232 Adapter/Converter¹ cables available from Velmex.

Figure 13 RS-422 to Terminal Block Breakout

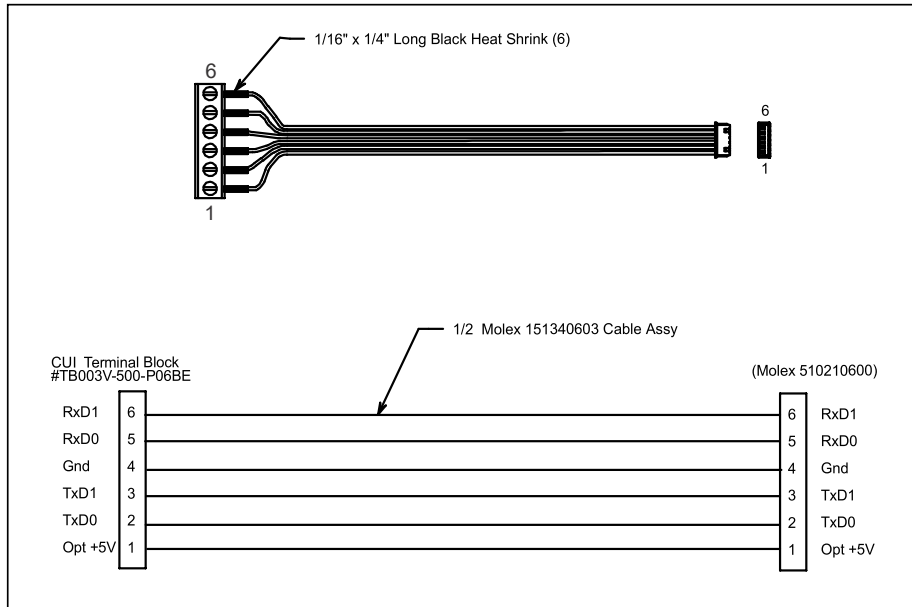
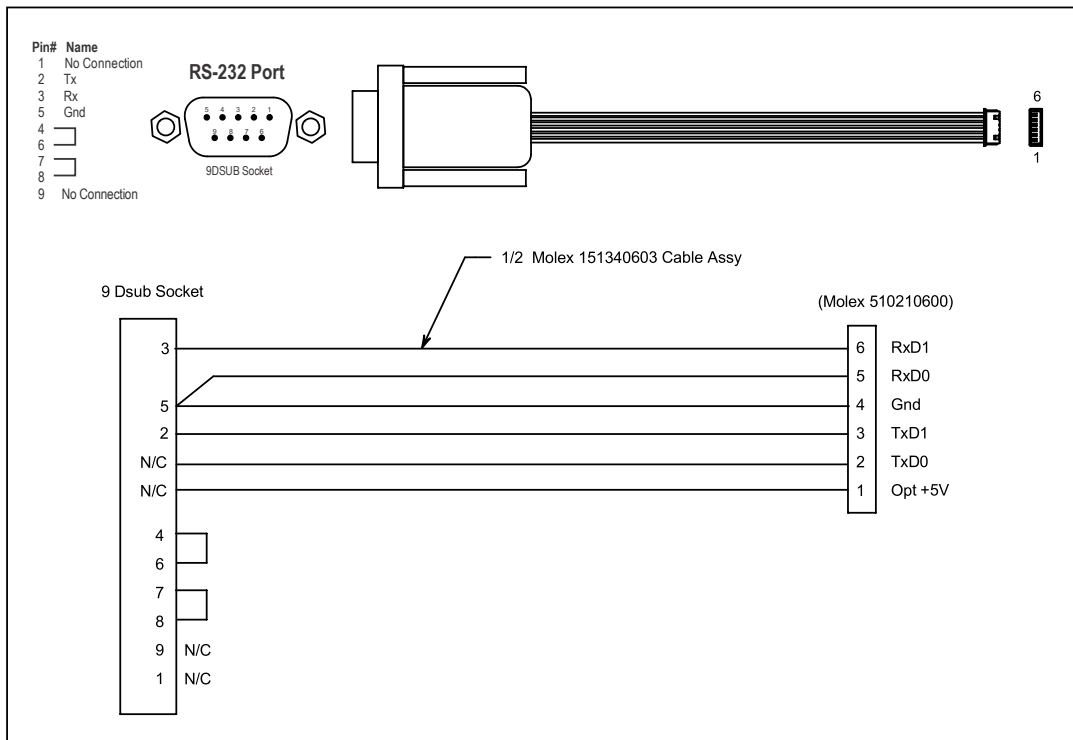


Figure 14 RS-422 to RS-232 Adapter/Converter¹



¹ Only RS-232 receivers that accept 0-5VDC on Rx as valid data are compatible with this adapter

Appendix H (Modes)

Controller Mode

The VXC has a main mode register that can be set with the “setDMx” command.

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

setDMb,x Set operating mode as an 8 bit binary number x between 0000000-11111111.

setDMbx,y Set operating mode bit number x of 0-7, state value y of 0 or 1.

Example:

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

```
setDMb1,1<cr>
```

IMPORTANT: Always use “rss” command after set to permanently save

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

getDMb Get operating mode as binary 0000000-11111111 number.

getDMbx Get operating mode state 0 or 1 of bit x (x=0-7)

getDMc Get operating mode complete detail list of settings.

Table 15 Controller Mode Settings

Bit#	7	6	5	4	3	2	1	0
Decimal	128	64	32	16	8	4	2	1
	Insert/ Combine Program #12 ¹ on 1 st Run ² 1=set	Output 4 ³ When Fatal Fault occurs 1=set	Invert Motor 4 Direction 1=set	Invert Motor 3 Direction 1=set	Emulate VXM Mode ⁴ 1=set	Invert Motor 2 Direction 1=set	Invert Motor 1 Direction 1=set	1= Dynamic µstepping on ⁵ 0= Always half- stepping ⁵
Default	0	0	0	0	0	0	0	1

¹ Program will be #4 if in Emulate VXM Mode (bit 3 set)

² Use this feature for homing VXC every power-up in stand-alone applications. Put home routine in program #12. First run after power up will run program #12 first.

³ Output 4 will go “high” when a Fatal flashing fault occurs, on multi-axis systems each axis output 4 will need to be monitored.

⁴ In VXM Mode: Insert/Combine Program #4, Program select with inputs 2,3 only, and Pauses = 10ths of seconds.

⁵ Dynamic µstepping greatly reduces vibration and noise at low speed. Do not use ½ stepping unless the goal is to produce higher vibration.

CAUTION: WITH µSTEPPING OFF, OPERATING AT 100% POWER BELOW 1000 SPS AN OVERCURRENT FAULT CAN OCCUR

This Example will return current setting in decimal format.

```
getDM<cr>
```

VXC will send: 1<cr>

This Example will return current setting in binary format.

```
getDMb<cr>
```

VXC will send: 00000001<cr>

This Example will set bit 7 which will run program #12 on first Run after being powered up.

setDMb7,1<cr>

"getDMb<cr>" will return "10000001<cr>"

"getDM<cr>" will return "129<cr>"

"getDMb7<cr>" will return "1<cr>"

This Example will set Mode settings back to default.

setDMb,00000001<cr>

This Example will list the complete detailed Mode settings.

getDMc

Below is what the VXC will list back.

```
* 76543210
# 00000001
(Default Setting)
    7,#1: Do Pgm 12 1st Run
*7,#0:(-----)
    6,#1: Out4 on F Faults
*6,#0:(-----)
    5,#1: Mtr4 Invert Dir
*5,#0:(Mtr4 Std Dir)
    4,#1: Mtr3 Invert Dir
*4,#0:(Mtr3 Std Dir)
    3,#1: VXM Emulate Mode
*3,#0:(VXC Std Mode)
    2,#1: Mtr2 Invert Dir
*2,#0:(Mtr2 Std Dir)
    1,#1: Mtr1 Invert Dir
*1,#0:(Mtr1 Std Dir)
*0,#1:(Dyn uStepping On)
    0,#0: Dyn uStepping Off
To change: setDMb*,#
```

Note: Factory default settings are indicated by ()
The current setting is prefixed by "*"
The Indented setting is the alternate of the current setting

Appendix I (Inputs)

Multifunction User Inputs

setlx set operating mode of User Inputs. The value for x is a number between 0 and 255 that can be derived from Table 16 User Inputs Mode Settings.

Example:

This example enables binary selection of programs 0 to 7 with user inputs 1,2, and 3.

```
setIb6,1<cr>
```

IMPORTANT: Always use “rss” command after set to permanently save

getl Get operating mode of User Inputs. The value returned is a number between 0 and 255 (see table below.) default=7

getlb Get operating mode of User Inputs as binary 0000000-11111111 number.

getlbx Get operating mode of User Inputs state 0 or 1 of bit x (x=0-7)

getlc Get operating mode of User Inputs complete detail list of settings.

Table 16 User Inputs Mode Settings

Bit#	7	6	5	4	3	2	1	0
Decimal	128	64	32	16	8	4	2	1
	Capture Motor Position on Input 4 Trigger ¹²⁷ 1=set	Program # ⁶ Select with Inputs 1 ³ ,3,2 1=set	Jump to Program # 10 after Stop (Input 4) 1=set	Stop and Hold on Input 1 “low” ⁴⁵ 1=set	Low Valid time for Run & Input 1 ² 1= 100 µsec. 0= 1 msec.	Input 3 Interrupt Waits and Jumps to Program # 9 1=set	Stop Enable/Disable 1= Enabled 0= Disabled	Run Enable/Disable 1= Enabled 0= Disabled
Default	0	0	0	0	0	1	1	1

¹ When set, Stop will be disabled.

² Slave axes 2,3,4 will also be set, for multi-axis position capture all Input 4s should be connected together as a common trigger.

³ Inputs 2,3 only in VXM mode. See **setDMx** command

⁴ When Input 1 is “low” program will stop before every command and resume when Input 1 goes “high”


⁵ Not useable if bit 6 set too.

Table 17 Program Select Truth Table

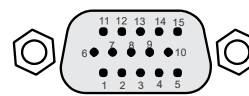
Program #	Input 1	Input 3	Input 2
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0

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

⁶The program select feature of inputs 1,2, and 3 can be used to select programs 0 to 7 for stand-alone applications. A rotary type binary switch would be connected to inputs 1,2, and 3 for program selection. After program selection, pressing the Run button will start the selected program. See Table 17 Program Select Truth Table for function of each input.

 ⁷See also “x”, “y”, “z”, “t”, “!” commands

Auxiliary I/O Connection



15DSUBHD Socket

- Pin# Name
- 1 0V (Common Ground)
 - 2 +5V Output
 - 3 Ain (Analog In)
 - 4 Run Input
 - 5 I1 (Input 1)
 - 6 I2 (Input 2)
 - 7 I3 (Input 3)
 - 8 I4 (Input 4/ Stop)
 - 9 0V (Common Ground)
 - 10 J- (Jog Mtr negative)
 - 11 J+ (Jog Mtr positive)
 - 12 O3 (Output 3)
 - 13 O4 (Output 4)
 - 14 O1 (Output 1)
 - 15 O2 (Output 2)

This Example will return current setting in binary format.

getIb<cr>

VXC will send: 00000111<cr>

This Example will return Run enable/disable bit setting.

getIb0<cr>

VXC will send: 1<cr>

This Example will list the complete detailed Mode settings.

getIc

Below is what the VXC will list back.

The diagram illustrates the output of the `getIc` command. It shows a list of mode settings with annotations:

- Bit #**: A blue arrow points to the bit number (e.g., 7, 6, 5, 4, 3, 2, 1, 0) in the first column of the output.
- Current Bit Values**: A blue arrow points to the binary values (e.g., 1, 1, 1, 1, 1, 1, 1, 1) in the second column of the output.
- Current Setting**: A blue arrow points to the function descriptions (e.g., Pos Capture In4, Pgm Sel In 1,2,3, Stop=Jump Pgm 10, In1= Stop/Hold, Valid InLow=100us, (Valid InLow=1ms), (In3=IrqWt/JmpPg9), (Stop Enabled), Stop Disabled, (Run Enabled), Run Disabled) in the third column of the output.
- Bit#, Value: Function**: A blue box with a bracket on the right side of the output, indicating that the bit number, value, and function are grouped together.

```
* 76543210
# 00000111
(Default Setting)
 7,#1: Pos Capture In4
*7,#0: (-----)
 6,#1: Pgm Sel In 1,2,3
*6,#0: (-----)
 5,#1: Stop=Jump Pgm 10
*5,#0: (-----)
 4,#1: In1= Stop/Hold
*4,#0: (-----)
 3,#1: Valid InLow=100us
*3,#0: (Valid InLow=1ms)
*2,#1: (In3=IrqWt/JmpPg9)
 2,#0: -----
*1,#1: (Stop Enabled)
 1,#0: Stop Disabled
*0,#1: (Run Enabled)
 0,#0: Run Disabled
To change: setIb*,#
```

Appendix J (Dynamically Read Position)

Getting Motor Position When Moving

Motor position can be directly read while motor is in motion on Axis 1 using the “X” command. At high motor speeds serial port latency will result in miss reading the actual position. Another approach is to use an external trigger to tell the VXC to capture motor position(s) for later retrieval (after motion has ended.)

The Automatic Deceleration Capture

When the motor starts a deceleration 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 “^”) reading the position where deceleration started.

* (Asterisk) Request motor position of last motor run when 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 its deceleration at position negative 14901.

-0014901<cr>

The Triggered Position Capture

The VXC can capture motor position(s) either by the host sending a “!” or from a trigger pulse on input 4. Up to 4 positions will be recorded (one for each trigger input.) After moving the recorded positions can be requested with the “x”, “y”, “z”, and “t” commands.

! Capture motor positions for later recall with “x”, “y”, “z”, “t” commands. Captures motor 1,2,3, & 4 motor positions into a FIFO buffer (4 positions per axis maximum)

NOTE: buffered data is automatically zeroed at the start of every run.

 See Appendix I for setting Capture Motor Position on Input 4 Trigger

Retrieve Captured Positions

x Send last 4 positions of motor 1 to host that were captured by the “!” command.

NOTE: buffered data is automatically zeroed at the start of every run.

This example shows the values the VXC returned when the “!” was sent at positions 521, 919, and 1149 while motor 1 was moving.

+0000521

+0000919

+0001149

+0000000

y Send last 4 positions of motor 2 to host that were captured by the “!” command.

NOTE: buffered data is automatically zeroed at the start of every run.

z Send last 4 positions of motor 3 to host that were captured by the “!” command.

NOTE: buffered data is automatically zeroed at the start of every run.

t Send last 4 positions of motor 4 to host that were captured by the “!” command.

NOTE: buffered data is automatically zeroed at the start of every run.

Appendix K (Output Triggers)

Producing Trigger Outputs

There are three different methods to produce a trigger pulse, at exact positions, commonly used for signaling data acquisition equipment.

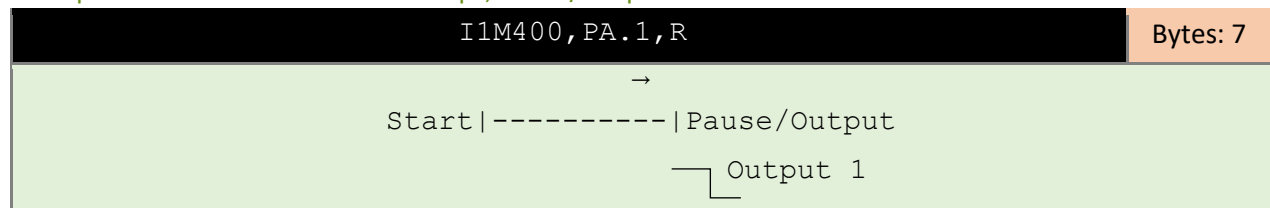
1. Index to a position, then while stopped use “PA”, “U1”, or “U5”/” U4” command.
2. Use the Continuous indexing method with the “U7” command to produce a pulse out instead of stopping at the end of indexes.
3. Set VXC to Pulse every “n” (“ n” is a number from 0 to 32,767) number of steps on output 2

Index, Stop, Output

Example:

This example Indexes 400 steps and makes a pulse on output 1 for 0.1 seconds

Example 20 Index Motor one 400 Steps, Pause/Output



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 (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) The acceleration value set before the “U7” command will be used in the continuous index.
- c) Speed settings, Jumps, Loops, and Output commands are allowed between indexes but Pauses and Wait commands are not allowed.
- d) Only Axis 1 can be directly run, all other axes need to be pass through program loaded using “[A[d,d1,d2...]]” and linked to Axis 1 with the “PMAx,y” program associate command.

U7 Continuous Indexing 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 1 and pulse 100 times:

```
U7,I1M400,LA100,U9<cr>
```

This example makes an index on motor 1, producing a pulse with speed changes between each index:

```
S1M1500,U7,I1M2000,S1M3000,I1M4000,S1M500,I1M800,U9,S1M3000,IA1M0<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 VXC 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 VXC 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, Appendix L

Pulse Every “n” Steps

The following commands make it possible to produce a pulse at a fixed interval of motor steps from 1 to 32,767.

setPmMn 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.)

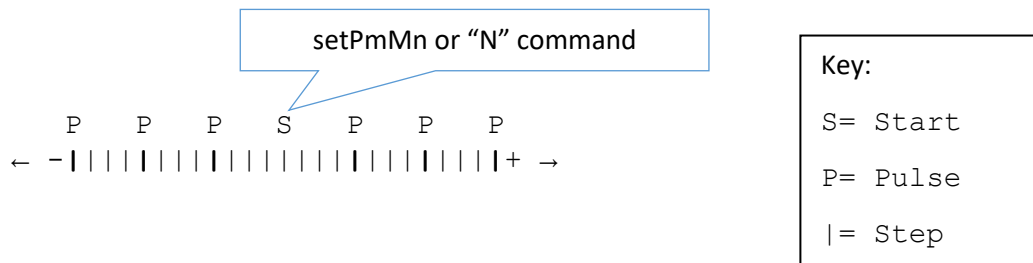
NOTE: Step count to next pulse is reset when **setPmMn** command is set or when motor position is zeroed (“N” command)

NOTE: Pulse will be generated both when indexing and when jogging.

Example:

This example will pulse Output 2 for every 4 steps of Motor 1

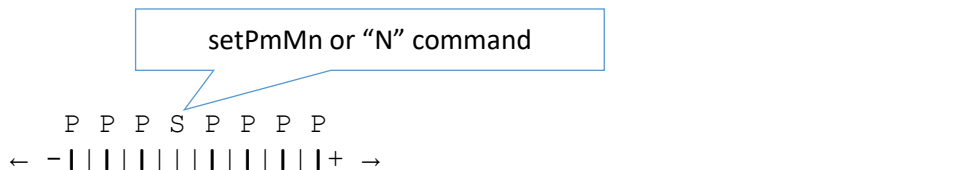
setP1M4



Example:

This example will pulse Output 2 for every 2 steps of Motor 1

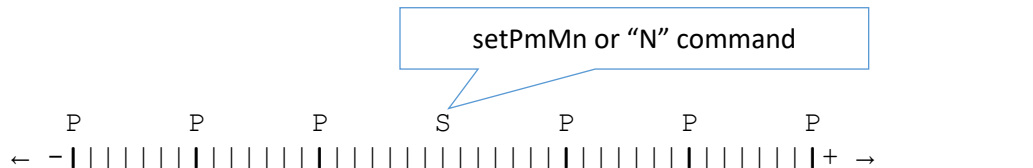
setP1M2



Example:

This example will pulse (Axis 2) Output 2 for every 7 steps of Motor 2

setP2M7



IMPORTANT: Always use “rss” command after set to permanently save

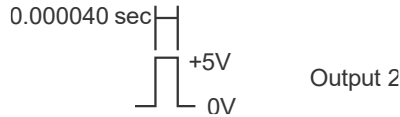
getPmM Get value for “Pulse Every n # Steps” for axis *m*. *m*= motor# (1,2,3,4)
 Value returned will between 0 and 32,767 (0= disabled/default)

setPAx Set Pulse width used by setPmMn and U7 commands. *x*= 1 to 255 (1=default.) Units are 10µsec increments (10 x 10⁻⁶ seconds)

Example:

This example will set the pulse width to 40 microseconds

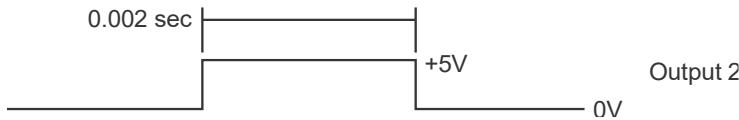
setPA4



Example:

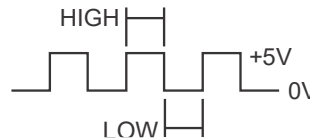
This example will set the pulse width to 2 milliseconds

setPA200



Exceeding this value will truncate pulse width

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



n = value from setPmM*n* or ImM*n* following a U7
 HIGH = Pulse width value *x* from setPA*x* command
 LOW = Low time needed between pulses

IMPORTANT: Always use “rss” command after set to permanently save

getPA Get Pulse width value used for setPmMn and U7 commands.
 Value returned will between 1 and 255, 1= 10µsec (default)

Appendix L (Complex Motion)

Complex Profiles/Coordinated Motion

Complex Profiles

The VXC automatically does a simple profile move every time it is required to perform an index. This profile consists of an acceleration segment, slew segment, and a deceleration segment. Using the VXC's Continuous indexing feature more complex motion profiles are possible. In addition to the Continuous Index commands described in Appendix K there are three 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 VXC executes this command, to prevent an excessively hard stop that may cause a mechanical overshoot of intended position.
- U10** Synchronize Master and Slave Axes. Used in Master and Slave(s), Master sends a trigger to Slave(s) waiting with/for a U10 command trigger.

 See also, Continuous Indexing

Coordinated Motion

The most common method to move is one axis at a time. All multi-axis VXC models have the capability to move each axis simultaneously producing complex coordinated motion profiles.

This is a typical example of running two motors sequentially with a model VXC-3:

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.

(ImMx,I1Mx,) Combine Index commands to run simultaneously on a model VXC-3. *m* = index commands for slave axes 2,3,4. Indexes for Axis 1 must be entered last. Memory usage = 2 bytes.

Example: **(I3M800,I1M400,)R**

NOTE: Motors 2,3, or 4 index commands must be before a motor 1 for simultaneous operation to occur.

Example that will not run motors simultaneously:

~~**(I1M400,I3M800,)R**~~

Master/Slave Associated Programs

Multi-axis VXC models can be set to run programs, residing in each axis, simultaneously. The procedure to coordinate VXC axis programs to VXC axis programs is as follows.

1. Transfer a program(s) to a Slave Axes using “[m[...]]”
2. Program associate the Master to the Slave(s) with the “PMAx,y” command

[m[d,d1,d2,...,] Send data to Slave Axis *m* through the Master, *m*= motor# (2,3,4), *d*= local commands for axis “m”. **NOTE: Status requests, “R”, and “Q” commands are not allowed. Do Not use “mM” in commands (internally each axis is a “1M”)**

Example to load axis 2, program 0, directly with speed and two index moves:

[2[PM-0,S1000,I400,I-400,]

PMAx,y Program Associate Master/Slave(s) *x* to program number *y*. Program *y* in the Slave(s) will run the same time when program *y* in the Master is run. *x*= 0,1,2,3,4

Table 18 Program Associate Values

x	Function	y	Function	y	Function
0	Disabled(default)	0	Pgm 0	-0	All Pgms except 0
12	Axis 1+2	1	Pgm 1	-1	All Pgms except 1
13	Axis 1+3	2	Pgm 2	-2	All Pgms except 2
14	Axis 1+4	3	Pgm 3	-3	All Pgms except 3
123	Axis 1+2+3	4	Pgm 4	-4	All Pgms except 4
124	Axis 1+2+4	5	Pgm 5	-5	All Pgms except 5
1234	Axis 1+2+3+4	6	Pgm 6	-6	All Pgms except 6
		7	Pgm 7	-7	All Pgms except 7
		8	Pgm 8	-8	All Pgms except 8
		9	Pgm 9	-9	All Pgms except 9
		10	Pgm 10	-10	All Pgms except 10
		11	Pgm 11	-11	All Pgms except 11
		12	Pgm 12	-12	All Pgms except 12

NOTE: Whenever an Associated program is Cleared in Axis 1 (Master) PMAx will be set to 0 (disabled)

Example: To run program 0 simultaneously in Axes 1 & 2: **PMA12,0**

Example: To run program 1 simultaneously in Axes 1,2,3 & 4: **PMA1234,1**

Example: Set to run all programs except 0 simultaneously in Axes 1 & 3: **PMA13,-0**

Example: Disable Program Associate (default): **PMA0**

Example: To read current Program Associate value: **PMA,**

Visit VelmexContols.com for special spreadsheets to calculate/create VXC commands for sinusoidal motion, triangles, rectangles with radius & chamfered corners, and circles.

 See also, Program Management Commands

Appendix M (Math)

Math Capability

The VXC can perform basic arithmetic operations of addition, subtraction, multiplication and division.

Arithmetic capability allows for:

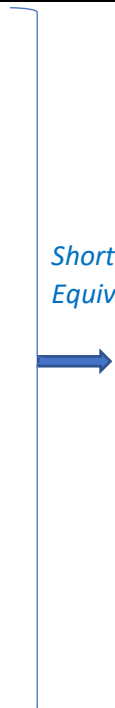
- 1) Setting & proportioning speed to analog input value
- 2) Producing custom asymmetrical accelerations / decelerations
- 3) Calculating position in pick & place applications
- 4) Self-determining center of system travel.
- 5) Self-calculating ratios of absolute position "X"

Operator	Meaning
+	Add
-	Subtract
*	Multiply
/	Divide

- Division of two integer values will give an integer result (any fractional remainder is discarded.)
Example: $a=1000$, $a=a/6$, will result in "a" equal to 166.
- The Maximum result of an operation is +/- 8388607. The red fault LED will light and software fault #30 will be logged if result is out of range (read fault with "**getFmM**" command)
- Multiplier and divisor range is +/- 255
- Operations are limited to two values at a time (see Table 19 Math Expressions)
- There are three general purpose integer (whole numbers) variables a,b,c.
- Variables can be assigned to a literal (constant), Absolute motor position "X", analog input value ("@") or another variable.
- Index & Speed commands can be assigned to variables a,b,c.

Math Expressions

Table 19 Math Expressions

a=<literal>	a=@	ImMx=a	a=a+<literal>	<i>Shorthand Equivalent</i> 	a+<literal>
b=<literal>	b=@	ImMx=b	b=b+<literal>		b+<literal>
c=<literal>	c=@	ImMx=c	c=c+<literal>		c+<literal>
a=b	a=-a	IAmMx=a	a=a-<literal>		a-<literal>
a=c	b=-b	IAmMx=b	b=b-<literal>		b-<literal>
b=a	c=-c	IAmMx=c	c=c-<literal>		c-<literal>
b=c	a=-b		a=a*<literal>		a*<literal>
c=a	a=-c	SmMx=a	b=b*<literal>		b*<literal>
c=b	b=-a	SmMx=b	c=c*<literal>		c*<literal>
a=X	b=-c	SmMx=c	a=a/<literal>		a/<literal>
b=X	c=-a		b=b/<literal>		b/<literal>
c=X	c=-b		c=c/<literal>		c/<literal>

Memory usage = 4 bytes.

<literal> range is +/- 8388607

Symbol	Meaning
=	Equal
>	Greater Than
<	Less Than
~	Not Equal

“if” Conditional Statement

The VXC can perform the statement “if” in the following format.

if <expression> True=do next/False=skip next

When “if” is encountered in a program the next command will be skipped if the result is False (performs next command if True)

“if” Expressions

Table 20 “If” Expressions

if a=b	if a>b	if a<b	if a~b	if a= <literal>	if a< <literal>
if a=c	if a>c	if a<c	if a~c	if b= <literal>	if b< <literal>
if b=a	if b>a	if b<a	if b~a	if c= <literal>	if c< <literal>
if b=c	if b>c	if b<c	if b~c	if a> <literal>	if a~ <literal>
if c=a	if c>a	if c<a	if c~a	if b> <literal>	if b~ <literal>
if c=b	if c>b	if c<b	if c~b	if c> <literal>	if c~ <literal>

“~” (tilde) is not equal

<literal> range is +/- 4194303

Memory usage = 4 bytes

“end” Statement

When “end” is used with “if” it is possible to stop the program on the result of the “if”

Memory usage = 1 bytes

This example will index motor one 400, 800, 1200, 1600, 2000 steps pausing 1.5 seconds after indexes and ending when “a” is > 2000:

```
a=0, LM0, a=a+400, if a>2000, end, I1M=a, P1.5, L0,
```

Setting & Proportioning Speed to Analog Input

The VXC has a 10-bit analog to digital converter for general use, motor speed setting, or for use with the optional Analog Joystick.

Example 21 Setting Speed Proportional to Analog Input

Description	Motors Run	Function
Analog Speed	1	Set speed 1 to 4000 proportional to the value of the analog input

$a=@, a*254, a/65, a+1, S=a,$

Refer to Table 21 Analog to Speed Formulas for other speed ranges. Refer to Excel spreadsheet "VXC Analog Value to Speed Calculator" to create commands for any speed range.

Table 21 Analog to Speed Formulas

Range		VXC Commands		Low=	High=	Actual Range	
Low	High					Low	High
1	1000	$a=@, a* 83, a/ 85, a+ 1, S=a,$		1	1024	1	1000
1	2000	$a=@, a* 127, a/ 65, a+ 1, S=a,$		1	1024	1	2001
1	3000	$a=@, a* 85, a/ 29, a+ 1, S=a,$		1	1024	1	3002
1	4000	$a=@, a* 254, a/ 65, a+ 1, S=a,$		1	1024	1	4002
1	5000	$a=@, a* 127, a/ 26, a+ 1, S=a,$		1	1024	1	5002
1	6000	$a=@, a* 252, a/ 43, a+ 1, S=a,$		1	1024	1	6002
2000	3000	$a=@, a* 83, a/ 85, a+ 2000, S=a,$		2000	1024	2000	2999
2000	4000	$a=@, a* 127, a/ 65, a+ 2000, S=a,$		2000	1024	2000	4000
2000	5000	$a=@, a* 85, a/ 29, a+ 2000, S=a,$		2000	1024	2000	5001
2000	6000	$a=@, a* 254, a/ 65, a+ 2000, S=a,$		2000	1024	2000	6001
3000	4000	$a=@, a* 83, a/ 85, a+ 3000, S=a,$		3000	1024	3000	3999
3000	5000	$a=@, a* 127, a/ 65, a+ 3000, S=a,$		3000	1024	3000	5000
3000	6000	$a=@, a* 85, a/ 29, a+ 3000, S=a,$		3000	1024	3000	6001
4000	5000	$a=@, a* 83, a/ 85, a+ 4000, S=a,$		4000	1024	4000	4999
4000	6000	$a=@, a* 127, a/ 65, a+ 4000, S=a,$		4000	1024	4000	6000
5000	6000	$a=@, a* 83, a/ 85, a+ 5000, S=a,$		5000	1024	5000	5999
400	2000	$a=@, a* 75, a/ 48, a+ 400, S=a,$		400	1024	400	2000
400	1200	$a=@, a* 148, a/ 189, a+ 400, S=a,$		400	1024	400	1201
100	400	$a=@, a* 56, a/ 191, a+ 100, S=a,$		100	1024	100	400
1900	2100	$a=@, a* 10, a/ 51, a+ 1900, S=a,$		1900	1024	1900	2100
1500	2500	$a=@, a* 83, a/ 85, a+ 1500, S=a,$		1500	1024	1500	2499
400	2000	$a=@, a* 255, a/ 163, a+ 400, S=a,$		400	1024	400	2001

Producing Custom Accelerations / Decelerations

Very long accelerations and decelerations are possible using a variable and adder between moves of a continuous index.

Example 22 Extra Long Accelerations

Description	Motors Run	Function
Long Acceleration	1	Move 60K steps taking one-minute to ramp up/down from 10 to 6000 to 10 steps/sec

```
PM-2, a=9, U77, LM0, a+1, S=a, I5, LA6000, LM0, a-1, S=a, I5, LA6000, U99,
```

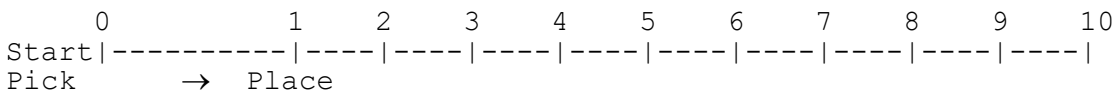
Calculating Position in Pick & Place Applications

Utilizing absolute index moves and incrementing a variable makes it possible to do standalone pick and place applications.

Example 23 Calculated Pick and Place

Description	Motors Run	Function
Pick & Place	1	Move from a common pick point to populate a row with 10 places 400 steps apart

Graphic Representation:



```
PM-0, ;select and clear program #0
A1M5, ;set accel
S1M4000, ;set speed
IA1M-0, ;Zero motor position at this point (start location)
a=2000, ;set distance = 1st place in row
LM0, ;set loop-to marker here
U5, ;Output 1 on to grip/lift part
P.5, ;Pause 1/2 second
IA1M=a, ;move to place part (absolute position "a")
U4, ;Output 1 off/Drop part
P.5, ;Pause 1/2 second
IA1M0, ;Move to 0 motor position to pick up another part
a+400, ;calculate next place to place part
L10, ;Loop to loop marker (do 10 parts in the row)
```

Self-determining Center of System Travel

Traversing a linear slide from its limit to limit will reveal the travel distance.

Equating travel distance to a variable and dividing it by 2 will be the midpoint of travel.

Example 24 Calculating Center of Travel

Description	Motors Run	Function
Self-Centering	1	Run limit to limit to determine max travel and then move to center

```
PM-0, ;select and clear program #0

S1M1000, ;set speed low enough for immediate stop on limits
I1M-0, ;Move negative until the negative limit switch
IA1M-0, ;Zero motor position at this point
I1M0, ;Move positive until the positive limit switch
a=X, ;set a = travel distance
a/2, ;calculate midpoint
S1M3000, ;set speed higher
IA1M=a, ;Move to center of travel
```

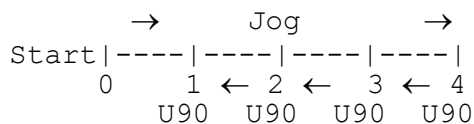
Self-calculating Ratios of Absolute Position

When jogging between two points the VXC keeps track of the distance in its “X” absolute position.

An efficient method to measure and divide items into equal segments can be accomplished using a variable and division in conjunction with traversing in both jog mode and programmed mode.

This example is to cut an item into 4 equal lengths using a programmable saw stop/linear slide. The operator would start/power up the VXC at the zero distance from the saw. An item of unknown length would be placed with one end at the zero position. The operator would jog/line-up the slide to the opposite edge of the item to measure the overall length. Repeatedly pressing Run will advance the slide so it can be cut into 4 equal parts. The last run will return to the initial jogged to position:

Graphic Representation:



Example 25 Self-Calculating Ratios of Distance

```
PM-0, ;select and clear program #0
U90, ;wait for release of Run button
a=X, ;set a = Jog distance from start/zero
b=X, ;save X in b too for later return position
a/4, ;calculate ¼ equal parts
LM0, ;set loop-to marker here
I1M=-a, ;Move back ¼ increments
U90, ;wait for press and release of Run button
LA3, ;do total of 3x
IA1M=b, ;Move absolute distance back to point #4
rsm,
```

Appendix N (Input Select Programs/Speed)

Stand-alone Methods to Select Program

Sometimes it is desirable for the user to interact directly with the VXC 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 VXC in standalone applications.

External Selection of Programs

The default program the VXC uses is program #0. By pressing the Run button or a button connected to the Run input on the I/O program #0 will be started. If a binary switch is connected, to Inputs 1,2, and 3, programs #0 to #8 can be selected. See Appendix I and Application Note #AN103C for more information.

External Setting of Speed

The "@" command can assign analog input value to motor speed and be used to vary speeds based on an external potentiometer setting. Refer to Appendix E and Application Note #AN102C 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 VXC will decelerate a running motor to a stop, and end the program. The VXC can be optionally set to run program #10 after stopping. See Appendix I for more information and other stop settings.

! CAUTION: THE STOP BUTTON/INPUT 4 IS NOT AN EMERGENCY STOP. FOR EMERGENCY STOP, POWER TO VXC 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 VXC 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.
2. Input 3 (I/O,7) is a special interrupt of a wait for input command. Example: the U0 command will wait for a low on input 1 (I/O,5). When the VXC is waiting for Input 1 and Input 3 (I/O,7) is pulled low the program will immediately branch to program #9. See Multifunction User Inputs 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.

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 26 Change Speed Using U11 & U21

Description	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 27 Change Program Using U12

Description	Motors Run	Function
Change Program	1	Changes 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 next command if Input 2 high
J1        ;Jump to program 1
S1M2000   ;Speed to 2000
I1M8000   ;Index
    
```


Appendix O (Setting Baud Rate)

Baud Rate Setting Methods

Baud rate for the serial ports¹ can be changed by two different methods, “setBx” command or through the front panel buttons.

Setting Baud with setBx Command

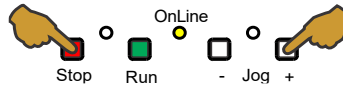
Baud rate can be changed with the “setBx” command. This method is only possible if the current rate is known and the host has multiple baud rate options.

setBx Set Serial Port¹ Baud rate, x=1,2,3,4 (1=9600, 2=19,200, 3=38,400, 4=57,600 baud)
IMPORTANT: Immediately after changing this setting, the host must be changed to the new baud rate and run “rss” (run save settings) in the VXC to save the new setting!

Setting Baud at Front Panel

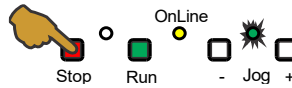
The baud setting can be changed using the Run, Stop, Jog buttons.

1. While VXC is powered off, hold both Jog+ & Stop buttons down at the same time



2. Power-up VXC while holding Jog+ & Stop buttons until On-Line LED blinks & stays on.

3. Press & release Stop button to see current setting. On-Line will blink once and green LED will flash x times to indicate the current setting: **x=1** for 9600, **x=2** for 19,200, **x=3** for 38,400, **x=4** for 57,600



4. Press and release Run button x times to toggle in new baud setting: **x=1** for 9600, **x=2** for 19,200, **x=3** for 38,400, **x=4** for 57,600

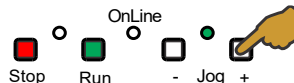


5. Press & release Stop button to see current setting. On-Line will blink once and green LED will flash x times to indicate the current setting: **x=1** for 9600, **x=2** for 19,200, **x=3** for 38,400, **x=4** for 57,600



6. Repeat step 4 if setting is not correct

7. To save setting² and exit, Press either Jog button



¹ Both (USB,RS-422) serial ports share the same baud rate setting

² If no Run button entries are made the baud rate will not change (stay as previously set)

Appendix P (VXC Versus VXM)

VXC Compatibility With VXM

Users transitioning to the VXC from the previous Velmex VXM Stepping Motor Controller need to be aware of the fundamental differences outlined here.

VXC New Features Compared to Previous VXM

- Autodetect motor connected
- More compact size
- 1/10th microstepping for smoother motion and higher resolution
- Runs common 4 wire hybrid stepper motors
- Bipolar motor drive produces less heat and has more torque
- Better electromagnetic compatibility
- Dedicated Home input
- Outputs 3,4, and 10VDC power for hall sensors are on standard external connectors
- Fault status LED with advanced fault checking and logging
- Dual serial ports (USB 2.0 and RS-422/RS-232)
- Voltage/current and temperature sensing
- Holding torque and failsafe brake output is standard
- 13 user programs (0-12)
- Stall detect option using simple hall effect sensors
- Higher 57.6K baud rate
- Simultaneous motion on all axes synchronized without needing a hardware coordination cable
- 10x higher speed resolution in range of 1.1 to 61.9 (0.1 resolution)
- Three power levels settable through speed command
- Stop and hold program function using user Input 1
- Smart speed and acceleration commands set value based on motor
- Math and logical operations capability
- True pause units (0.0001 to 5.9999 seconds & 6.0 to 6553.5 seconds)
- Custom digital jog settings
- Power up state of outputs is configurable
- Save only settings command (“rss”)
- Direct descriptive listing of settings, bit settable settings, and built-in help

VXC versus VXM Settings Differences

Function	VXC	VXM
Default Serial Port Baud Rate	57600 ¹	9600
Motor Setting	setMT=x ²	setMx

¹ Can be set to match the VXM default of 9600, see Appendix O for method to change baud rate

² Requires settings “A” to “F” and new 4 wire stepping motors

VXC Versus VXM Command Differences

Function	VXC	VXM
Speed setting for 100% power	S+x	SAx
Assign Analog value to Speed	S=@ ³	S-x
Backlash Compensation	setKx	Bx
Indicate limit switch Over-travel	setLx ⁴	Ox
Put Controller on Hold	:	H
List Programs	lst,	lst
Request Memory available	Mem	M
Send data to Slave through Master	[A[d, d1, d2...]	[d, d1, d2 . . .]
Pause units	Seconds ⁵	Tenths of seconds
Jog Mode: Read all motor positions (Digitize)	x ⁶	D
Loop once from beginning or Loop-to-marker reversing index direction of motor 2	-	LM-2
Loop once from beginning or Loop-to-marker reversing index direction of motor 1 & 2	-	LM-3
Pick and Place	- ⁷	JM-x
Program Associate	PMAx, y	PMAx
Program # for “Insert/Combine on 1st Run”	12 ⁸	4

³ Refer to Setting & Proportioning Speed to Analog Input (Appendix M)

⁴ Refer to Appendix B (Limits/Home/Stall Detect) for proper value for x

⁵ Setting VXC to VXM mode (setDMb3,1) makes Pause units tenths (the same as VXM)

⁶ Also use Y,Z,T if these axes exist

⁷ Refer to Calculating Position in Pick & Place Applications (Appendix M)

⁸ Setting VXC to VXM mode (setDMb3,1) makes 1st run program 4 (the same as VXM)

VXC New Commands

Command	Function
S-x ¹	Speed with Power at 40%
SX	Smart Speed, sets motor to maximum speed based on motor
AX	Smart Acceleration, sets motor to maximum acceleration based on motor
Ix.y	Micro-Incremental Index motor
rss	Run save settings (saves setup values only)
lssmM,	List all settings of axis "m"
lstmMx	List program "x" of axis "m"
U10	Synchronize Master and Slave Axis
+, -, *, /	Add, Subtract, Multiply, Divide
=, >, <, ~	Equal, Greater Than, Less Than, Not Equal
if	if <expression> True=do next/False=skip next
end	End the program on the result of an "if"

¹ Was analog equate to speed in VXM

VXC New set/gets

Command	Function
setMLH=x	Set for Motor, Limits, & Home to Device ID
setHSx	Set Home/Stall input
setMJx	Set Motor Jog function
getF	Read Fault(s)

VXC Different Values

Parameter	VXC Value	VXM Value
Minimum Slew Hold-at Speed in Jog Mode	63 steps/second	39 steps/second
Analog Jog Deadband default	50	40
Steps/rev	400/4000	400
Pulse Every "x" # Steps" on Output 2	Starts "x" Steps away	Starts "x/2" Steps away
Verify Controller's status	B,R,J,b,F(Fault)	B,R,J,b
Program Number Range	0-12	0-4
setIx (bit #4)	Stop & Hold on Input 1	Decel/Hard Stop on Input 4
setDMx	Bits 7,6,3,0 are different	Bits 7,6,3,0 are different

Appendix Q (Microstepping)

Dynamic & Static Microstepping

Dynamic microstepping is built into the VXC to greatly reduce motor vibration at the lower speeds. The VXC always uses dynamic microstepping regardless if the index position value is a half-step or a microstep distance. For users that prefer high vibration at low speed, dynamic microstepping can be turned off. Refer to the Controller Mode command “setDMx” for how to disable dynamic microstepping.

Static microstep positioning with the VXC is a step size that is 1/10th the increment of a traditional half-step and it has 1/10th the strength of a half-step which makes positioning to microstep resolution a challenge. When a load is applied to the motor, the load can be greater than the force produced by a microstep increment resulting in deviation from the intended position. Even though it is difficult to achieve high resolution microstep positioning there are applications where it is useful as noted below.

Microstepping for extremely low speeds

This example indexes 1 microstep, Pauses 9 seconds & repeats 4000 times:

I1M0.1 , P9 , L4000 ,

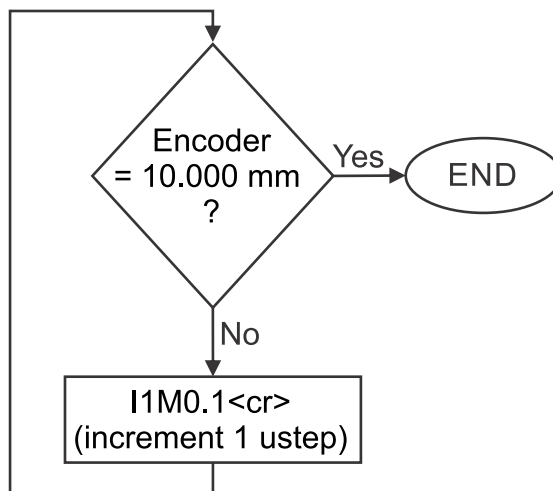
The above commands will rotate one motor revolution in 10 hours. (4000 μ steps/rev x 9 seconds / μ step = 36000 seconds / rev = 600 minutes = 10 hours)

Microstepping for high resolution positioning with position feedback

When a control system has a high-resolution encoder or optical interferometer feedback, microstepping provides a higher resolution means to “nudge” into the desired position.

For example:

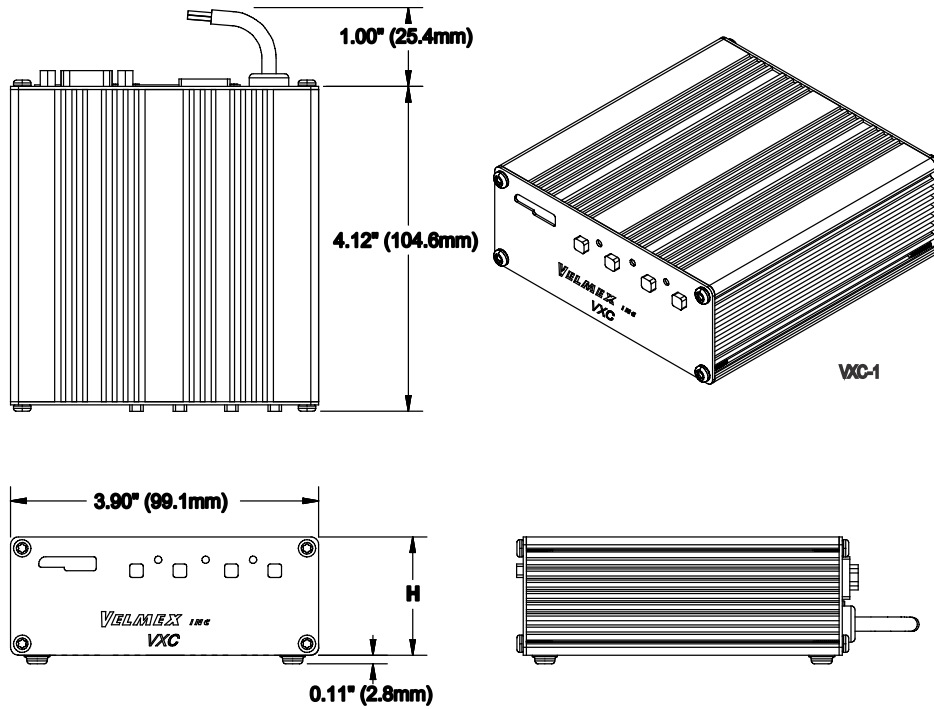
The target encoder position is 10.000 mm, step 1 μ step until the encoder readout has the desired position.



Appendix R (Dimensions)

VXC Dimensions

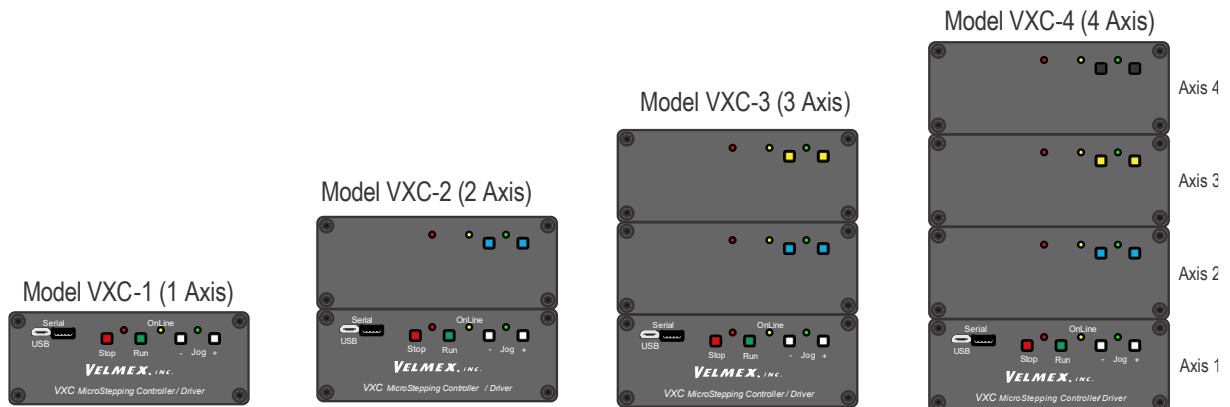
Figure 15 VXC Dimensional Drawing



Model	H
VXC-1	1.50" (38.1 mm)
VXC-2	3.00" (76.2 mm)
VXC-3	4.50" (114.3 mm)
VXC-4	6.00" (152.4 mm)

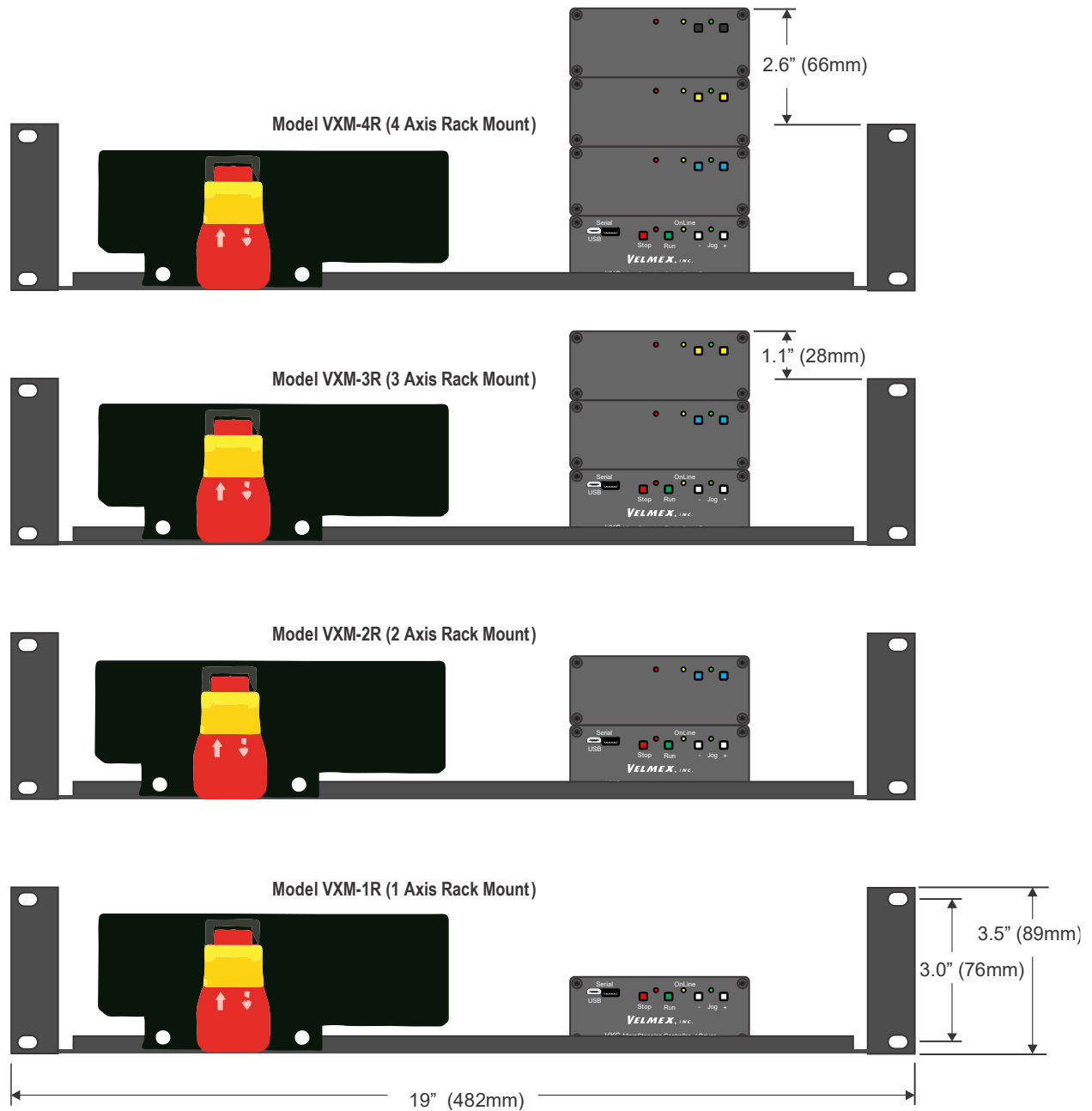
VXC Models

Figure 16 VXC Models



Rack Mounting

Figure 17 Rack Shelf Mounting Dimensions

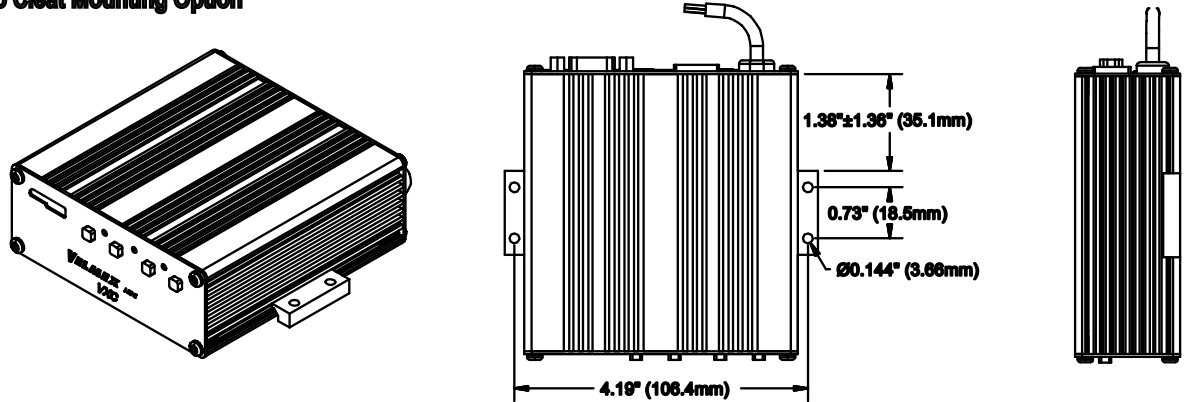


Depth = 10" (254mm)

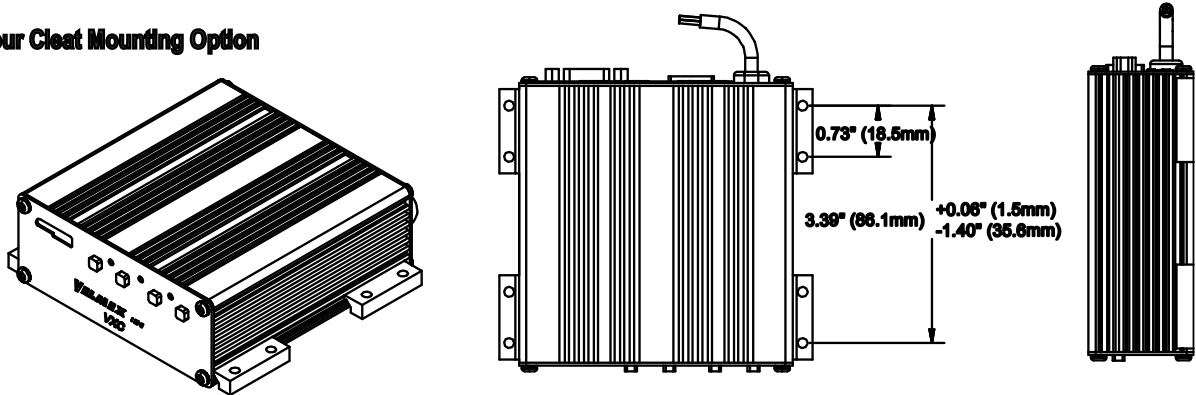
Cleat Mounting

Figure 18 Cleat Mounting Dimensions

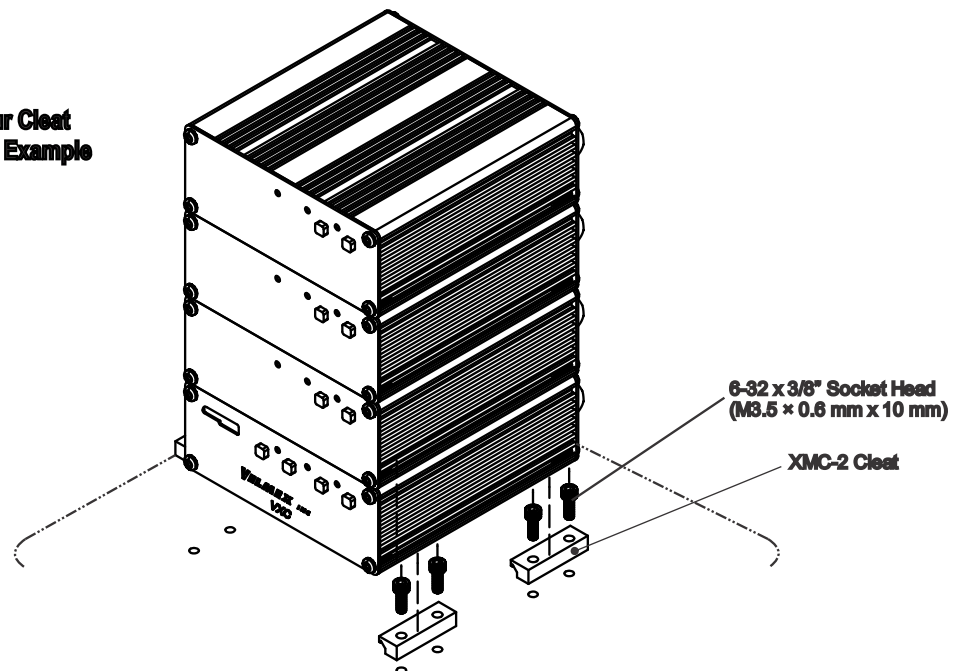
Two Cleat Mounting Option



Four Cleat Mounting Option



VXC-4 Four Cleat Assembly Example



Power Supply Dimensions

Figure 19 Standard 24V Power Supply Dimensional Drawing

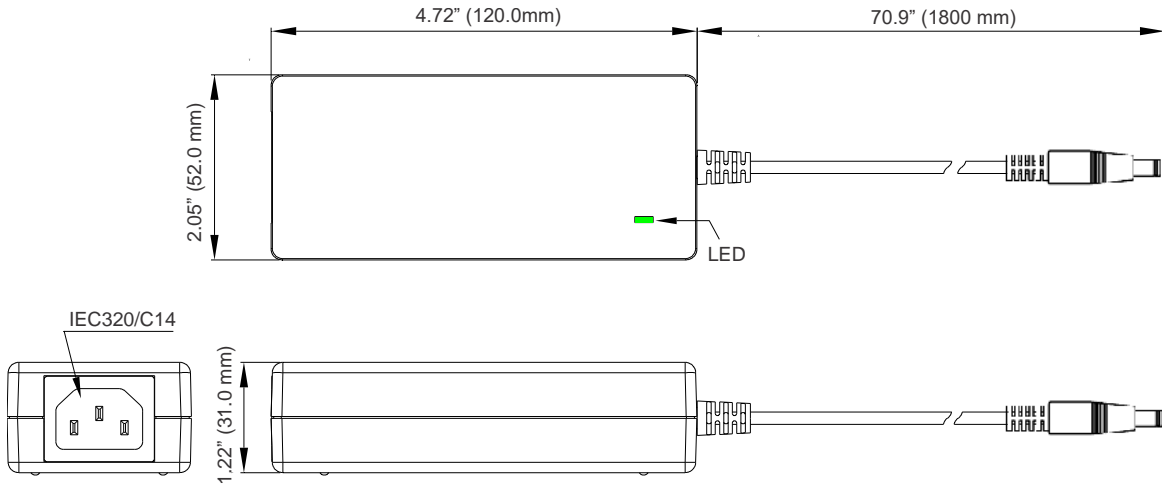
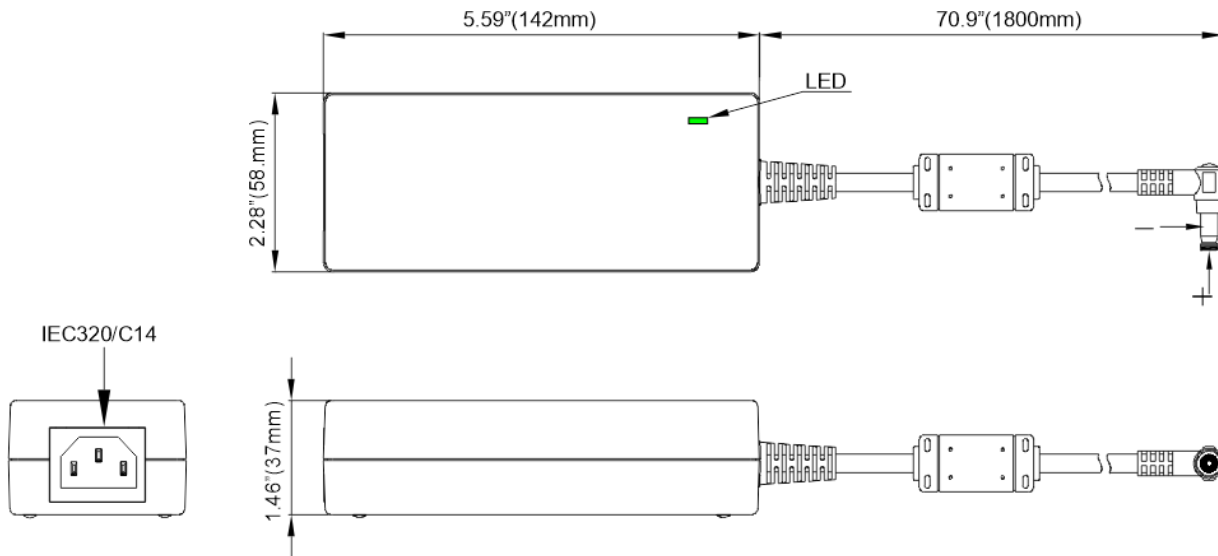


Figure 20 Optional 28V Power Supply Dimensional Drawing



Appendix S (I/O Specifications)

I/O Electrical Specifications

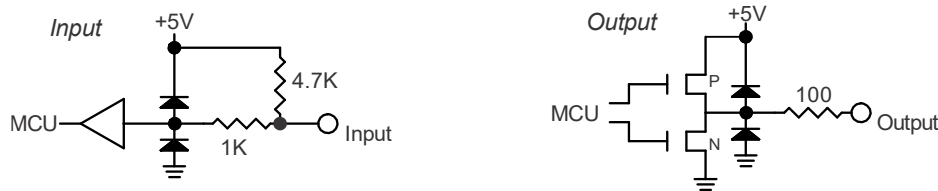
All User I/O inputs and outputs (except limit switch inputs) are TTL levels (0 to +5VDC.)

Inputs have a 4700 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.

Figure 21 Inputs and Outputs Circuit Diagram



Limit switch inputs and Home are optically isolated. These inputs are internally connected to 10VDC through a 4.7K ohm resistor to power the LED in the optical isolator (see Appendix B (Limits/Home/Stall Detect) 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 VXC I/O to an inductive load, any device that is not within 10 feet of the VXC, or anything not powered at the same AC source.

Damage due to improperly interfacing VXC 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 VXC I/O.
4. If no other protective earth ground is available, use the earth ground on the VXC's Auxiliary I/O connector shell or connector shell on shielded cable.

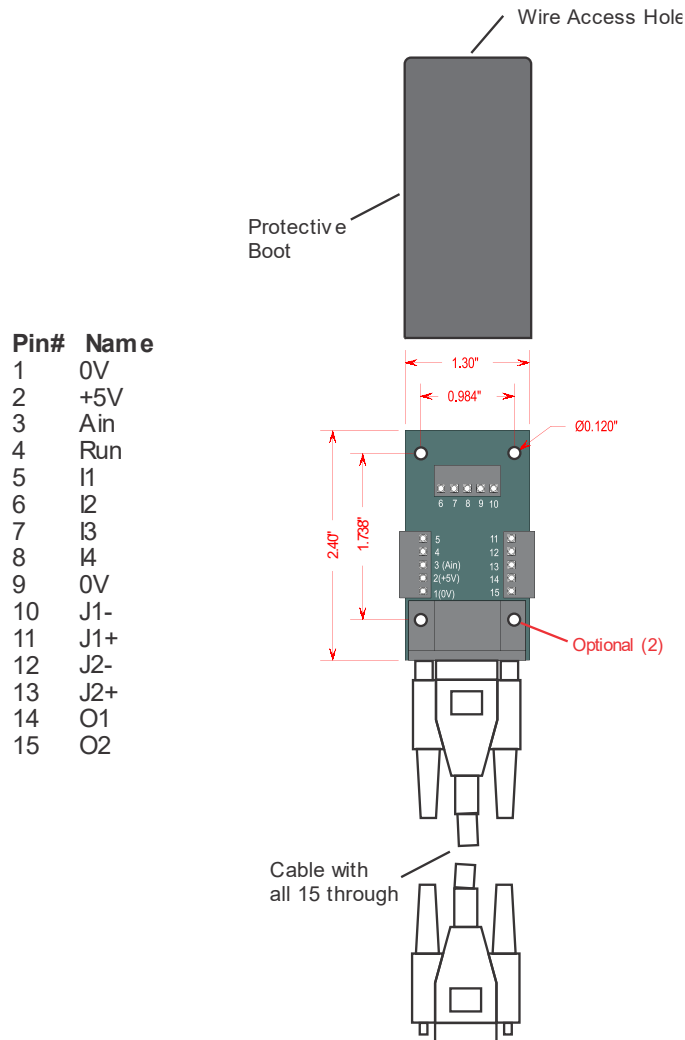
Optional Auxiliary I/O Breakout Module

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

Specifications

Wire size: 26 to 18 AWG
 Boot material: PVC
 Boot dielectric strength: 700 V/mil

Figure 22 Auxiliary I/O Breakout Module



See also, External Features

Appendix T (Motor Torque)

Motor Torque Curves

VML113-1.2-S (28CM013) Motor

Torque Graph 1 for VML113-1.2-S (28CM013) Motor @ 24V

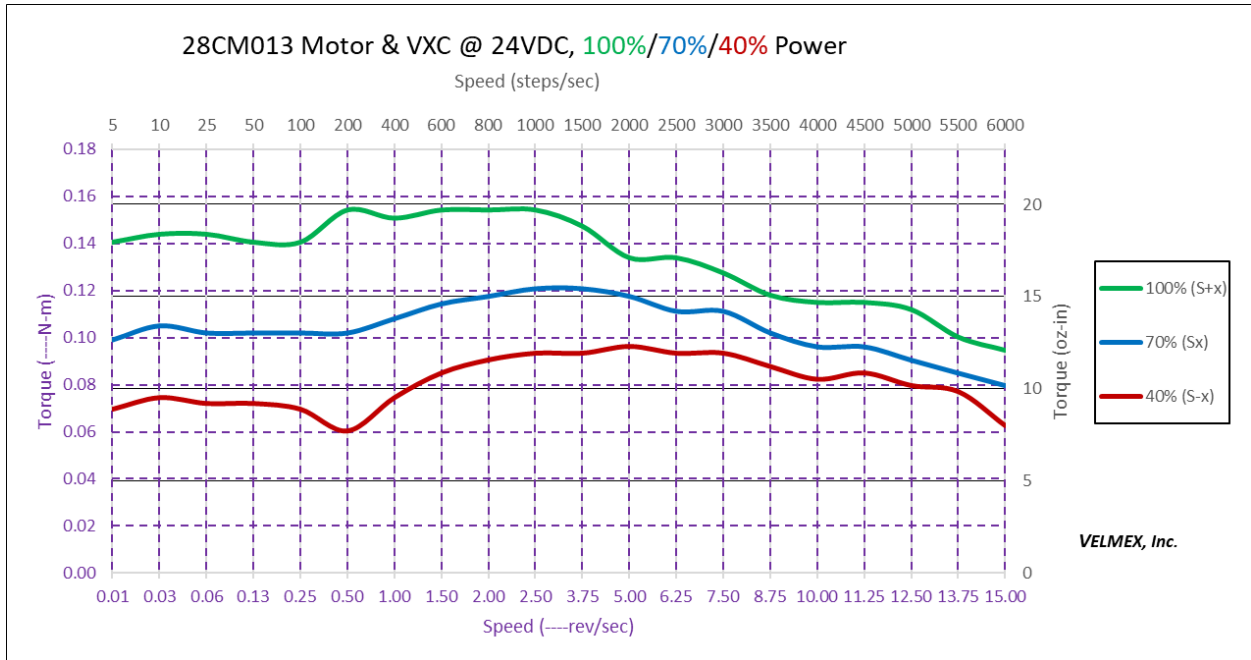
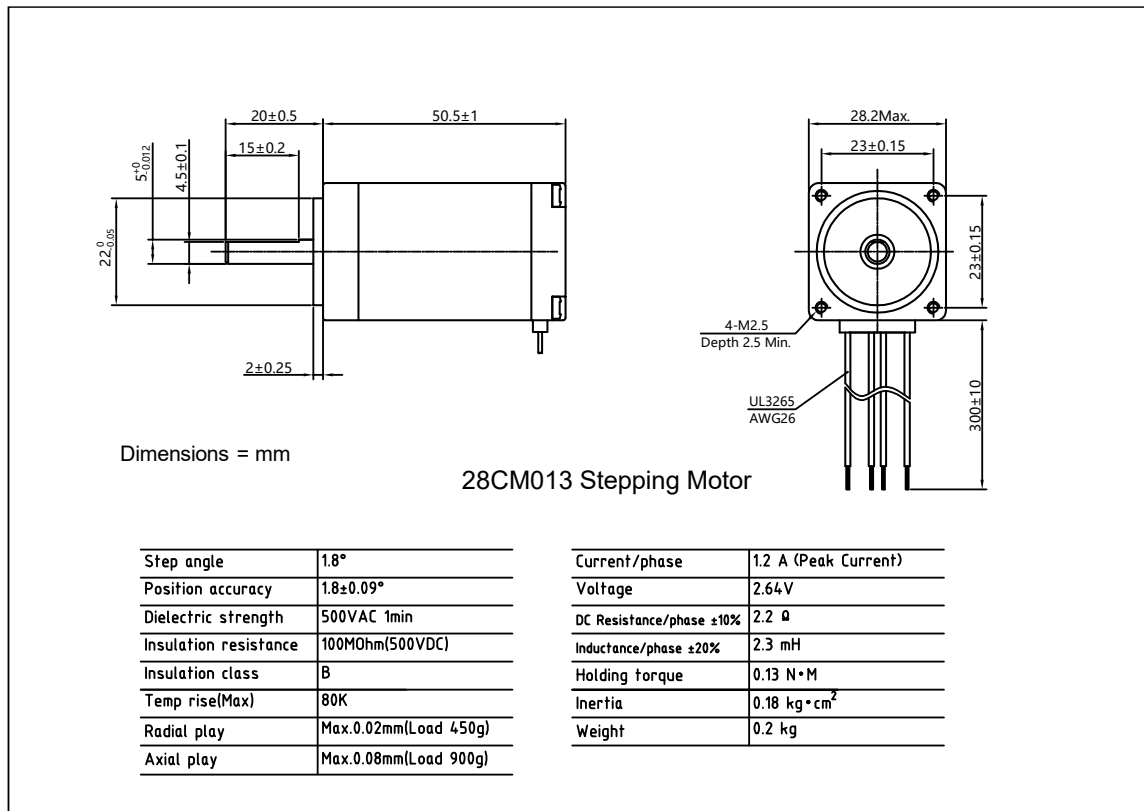
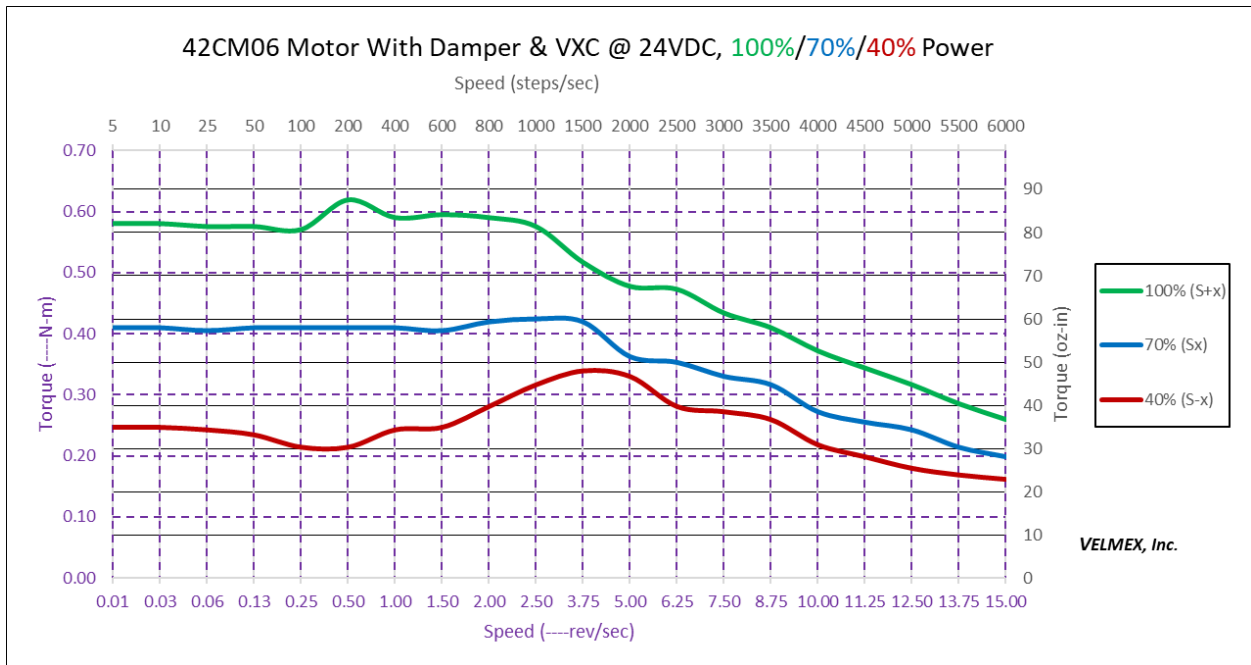


Figure 23 VML113-1.2-S (28CM013) Motor Specifications

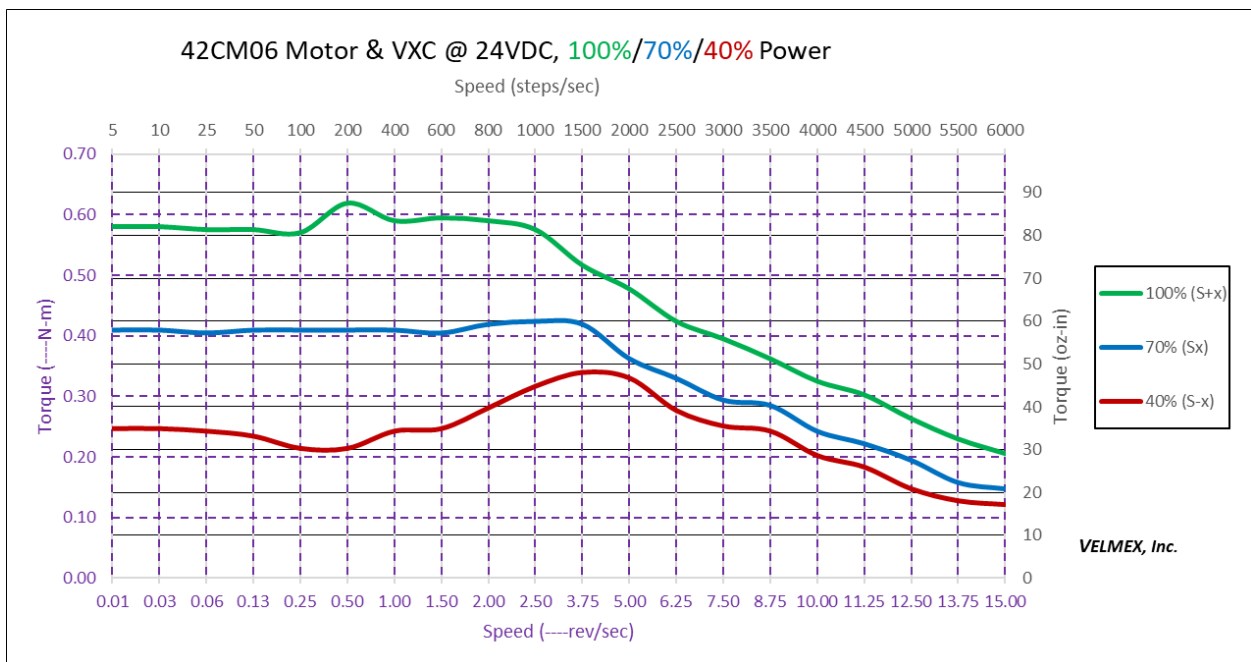


VML173-2.5-D (42CM06-SZ) Motor

Torque Graph 2 for VML173-2.5-D (42CM06-SZ) Motor @ 24V With Damper

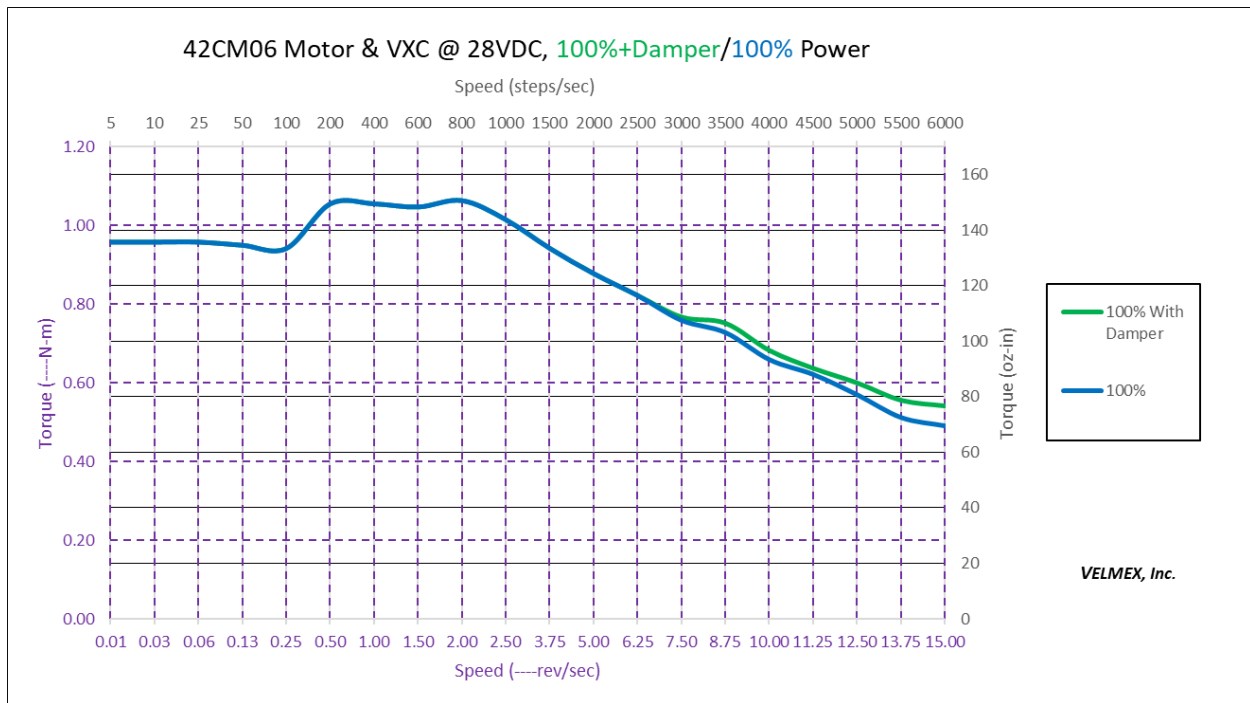


Torque Graph 3 for VML173-2.5-D (42CM06-SZ) Motor @ 24V



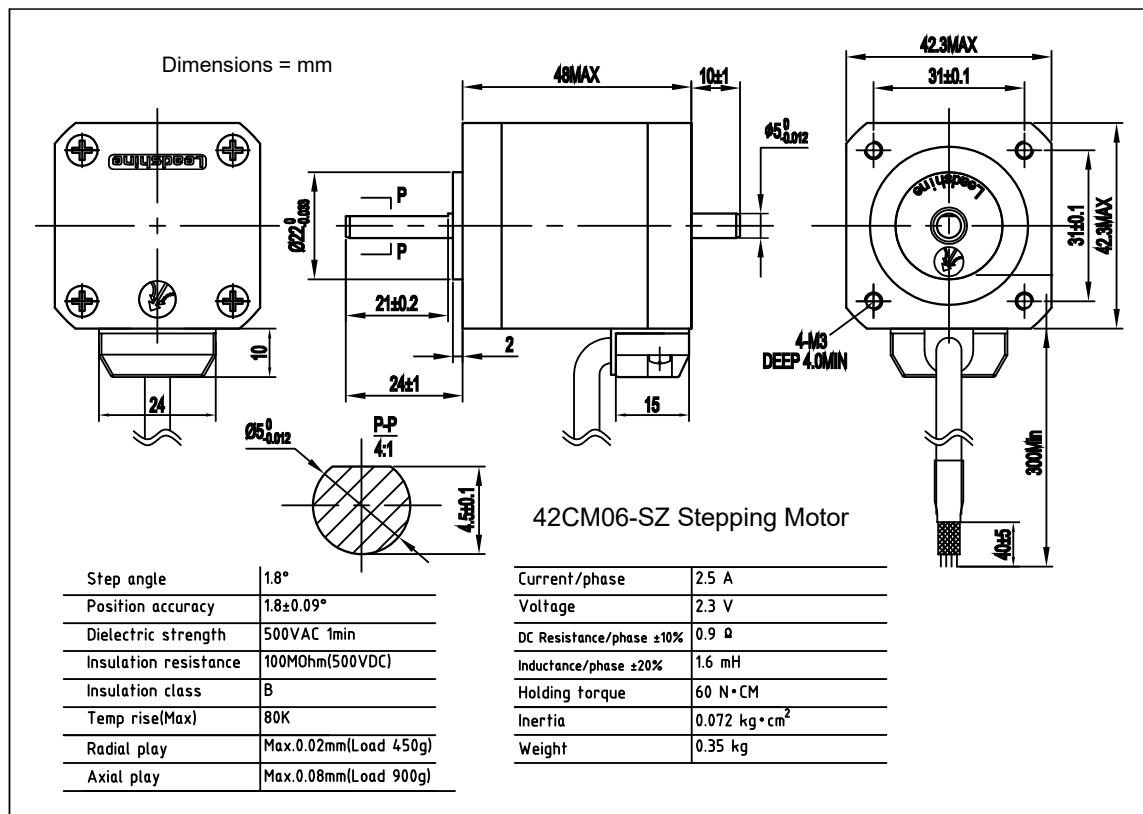
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance.

Torque Graph 4 for VML173-2.5-D (42CM06-SZ) Motor @ 28V, 100% Power, With/Without Damper



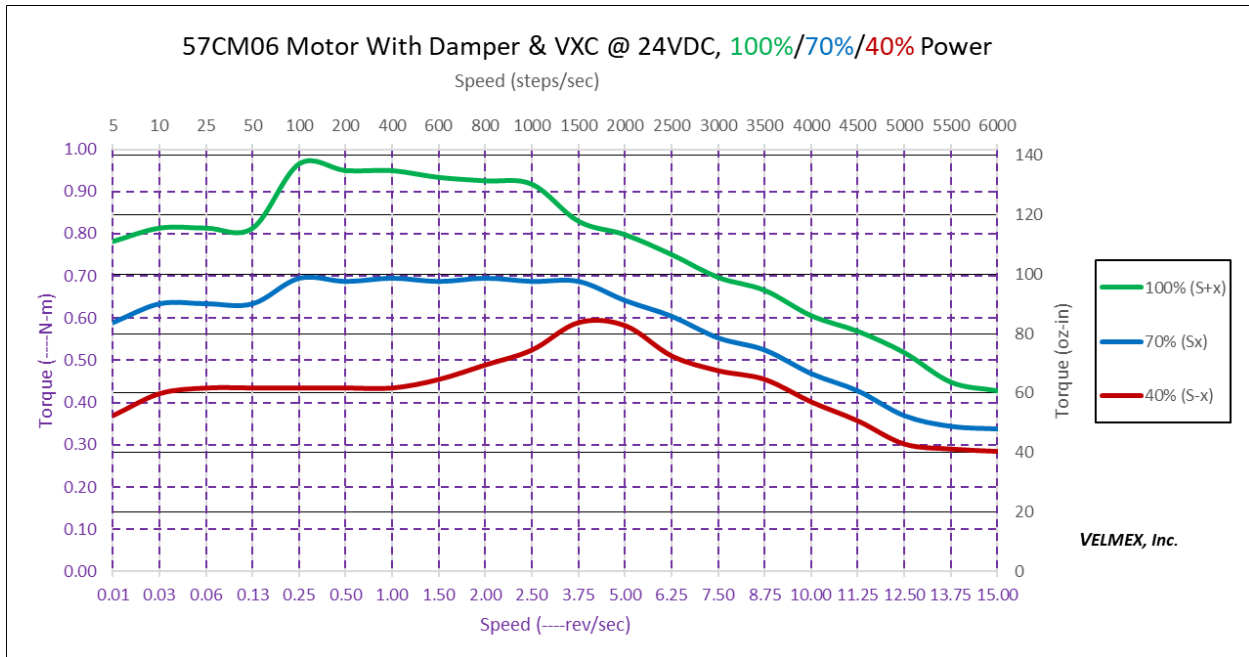
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance

Figure 24 VML173-2.5-D (42CM06-SZ) Motor Specifications

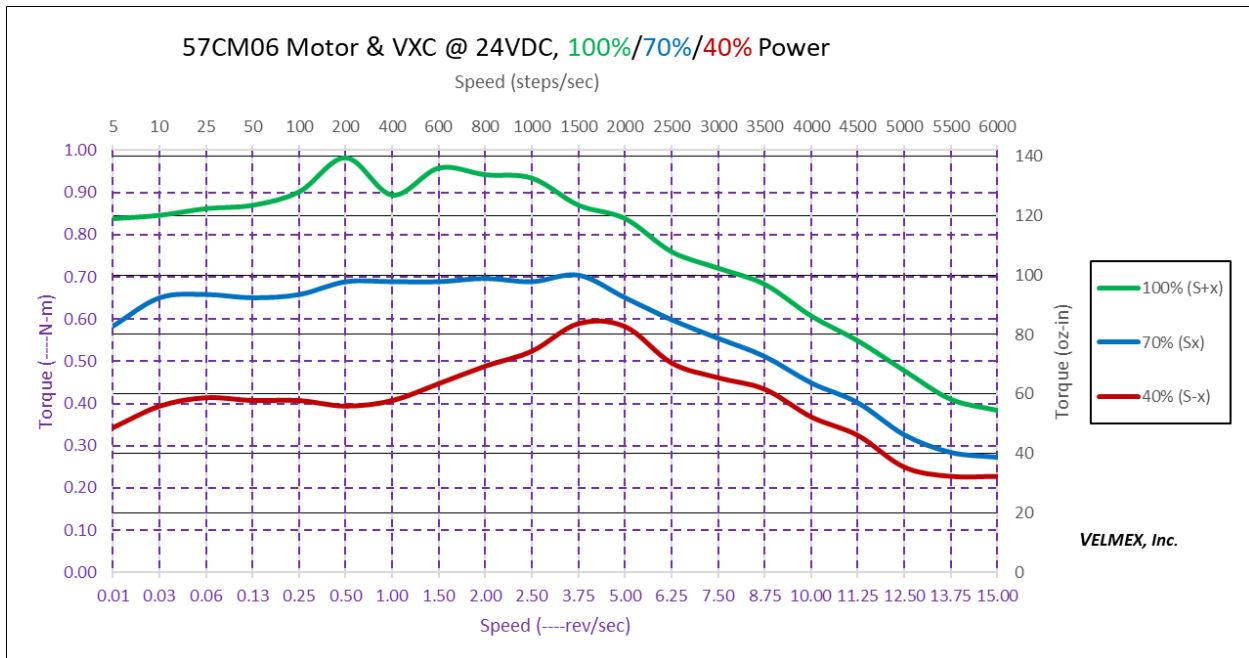


VML231-3.0-D (57CM06) Motor

Torque Graph 5 for VML231-3.0-D (57CM06) Motor @ 24V With Damper

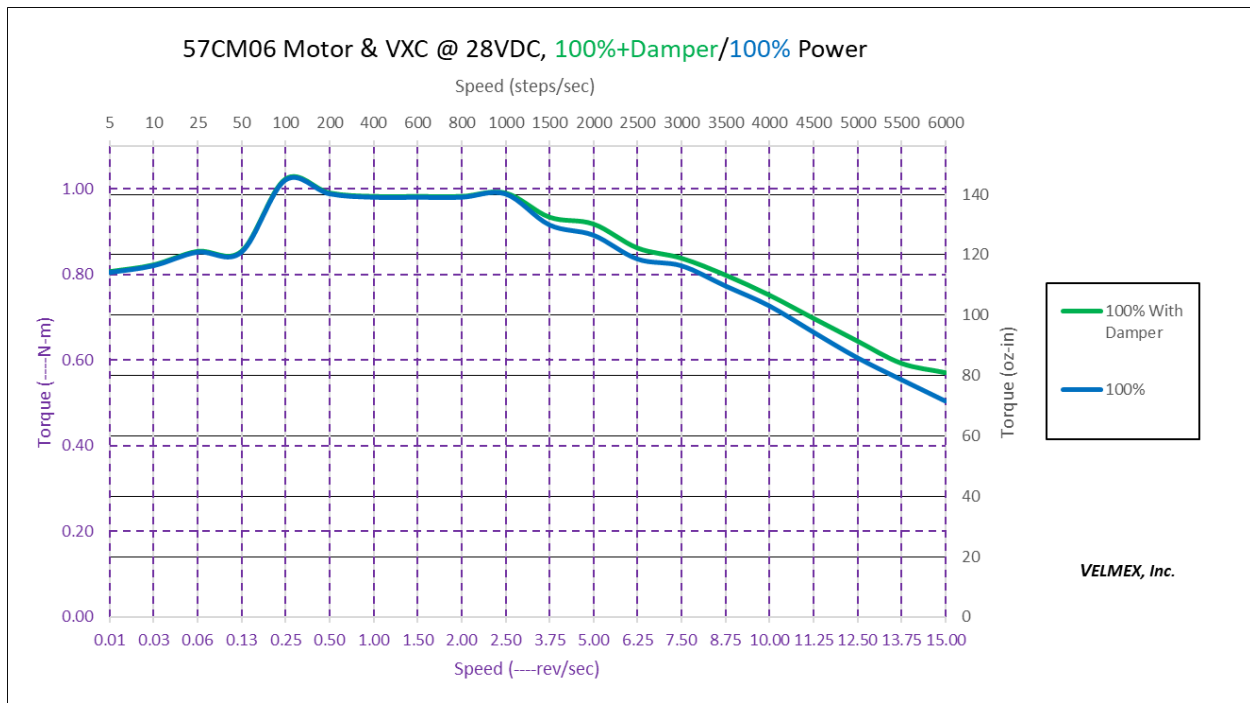


Torque Graph 6 for VML231-3.0-D (57CM06) Motor @ 24V



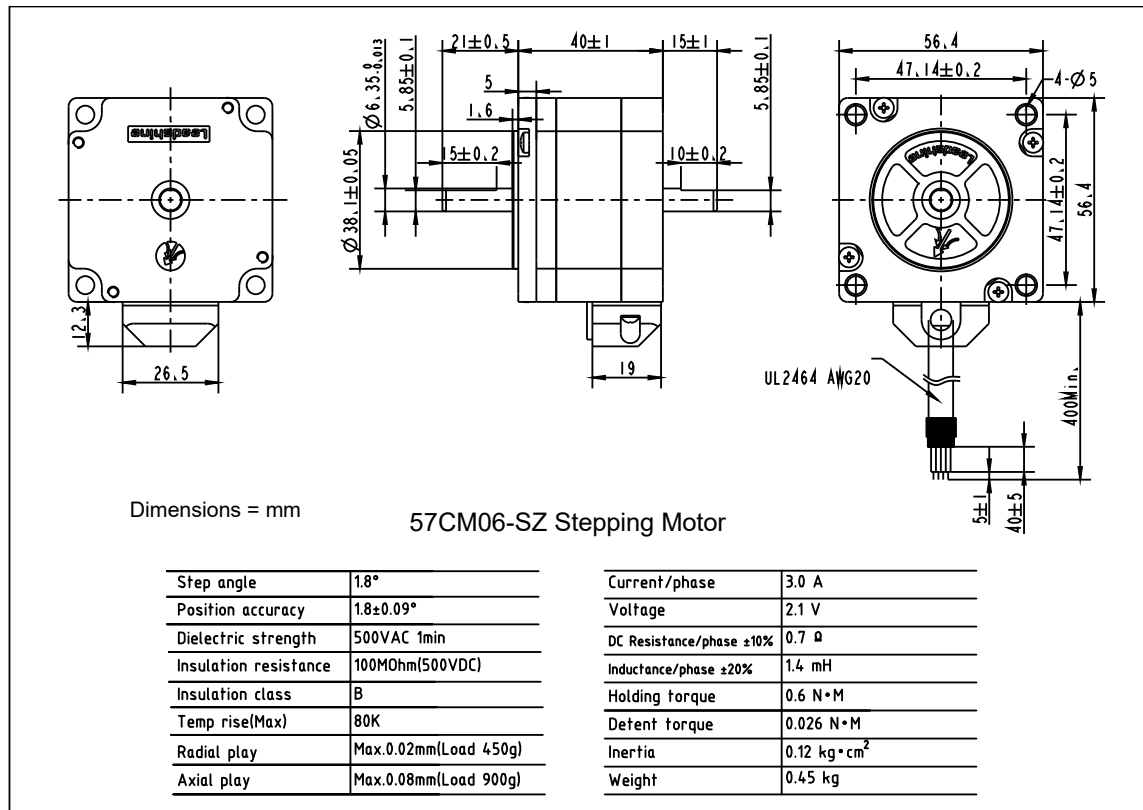
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance.

Torque Graph 7 for VML231-3.0-D (57CM06) Motor @ 28V, 100% Power, With/Without Damper



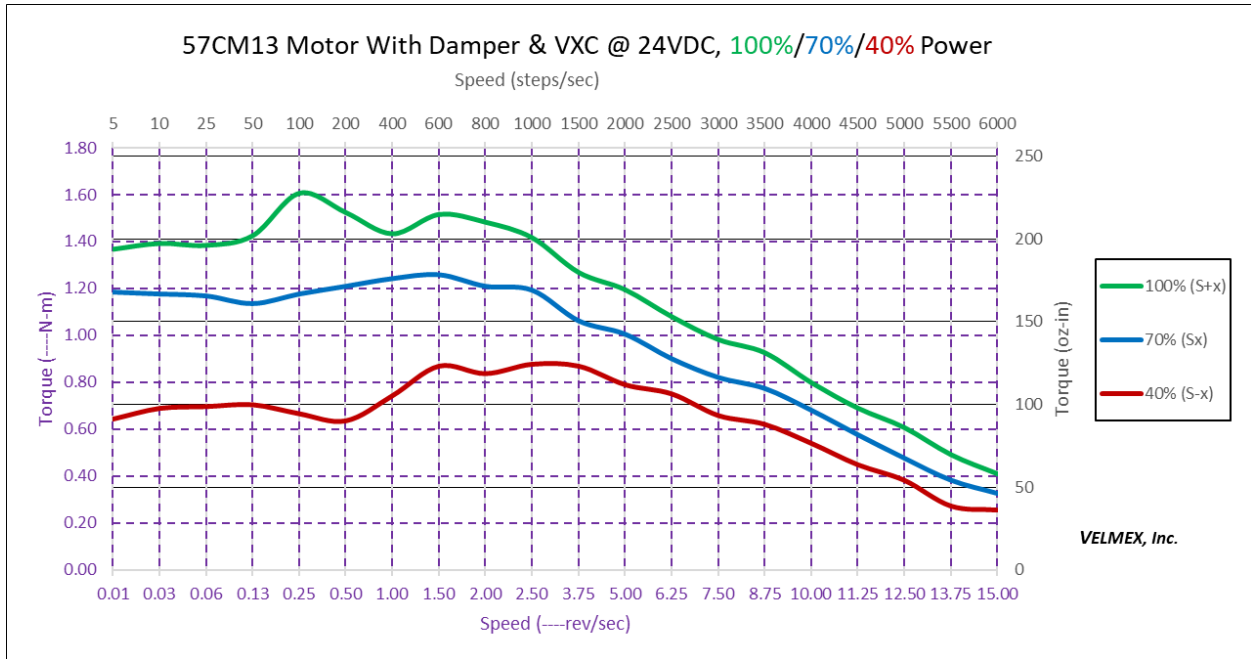
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance.

Figure 25 VML231-3.0-D (57CM06-SZ) Motor Specifications

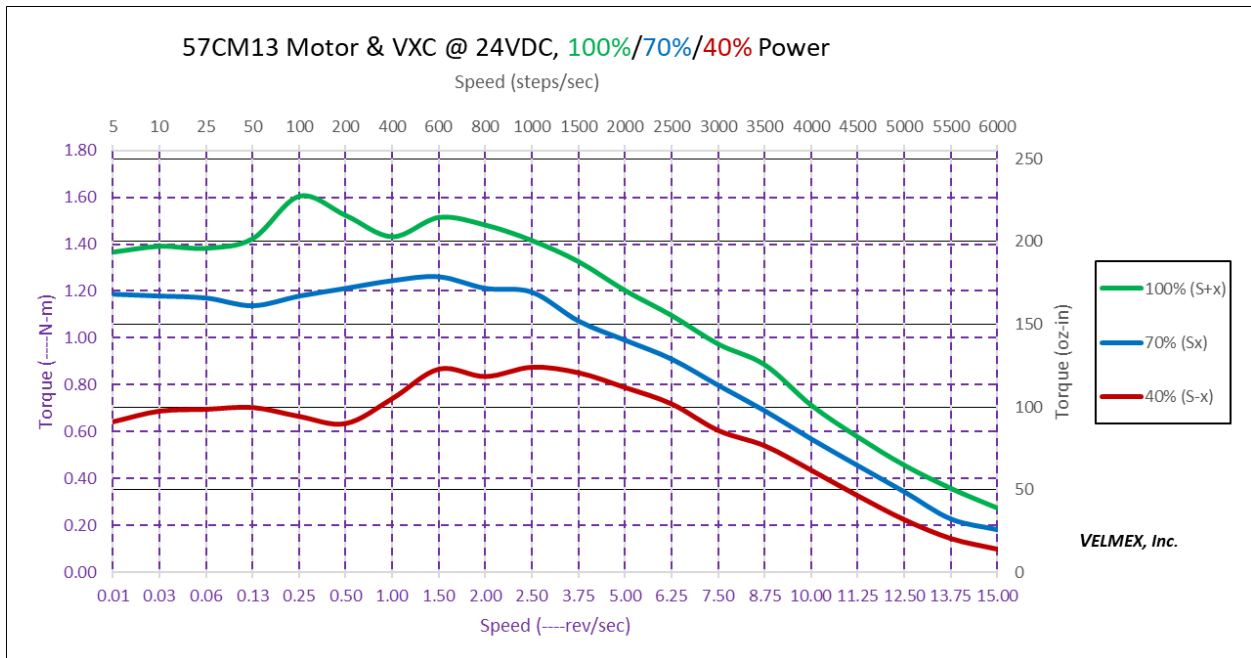


VML232-4.0-D (57CM13) Motor

Torque Graph 8 for VML232-4.0-D (57CM13) Motor @ 24V With Damper

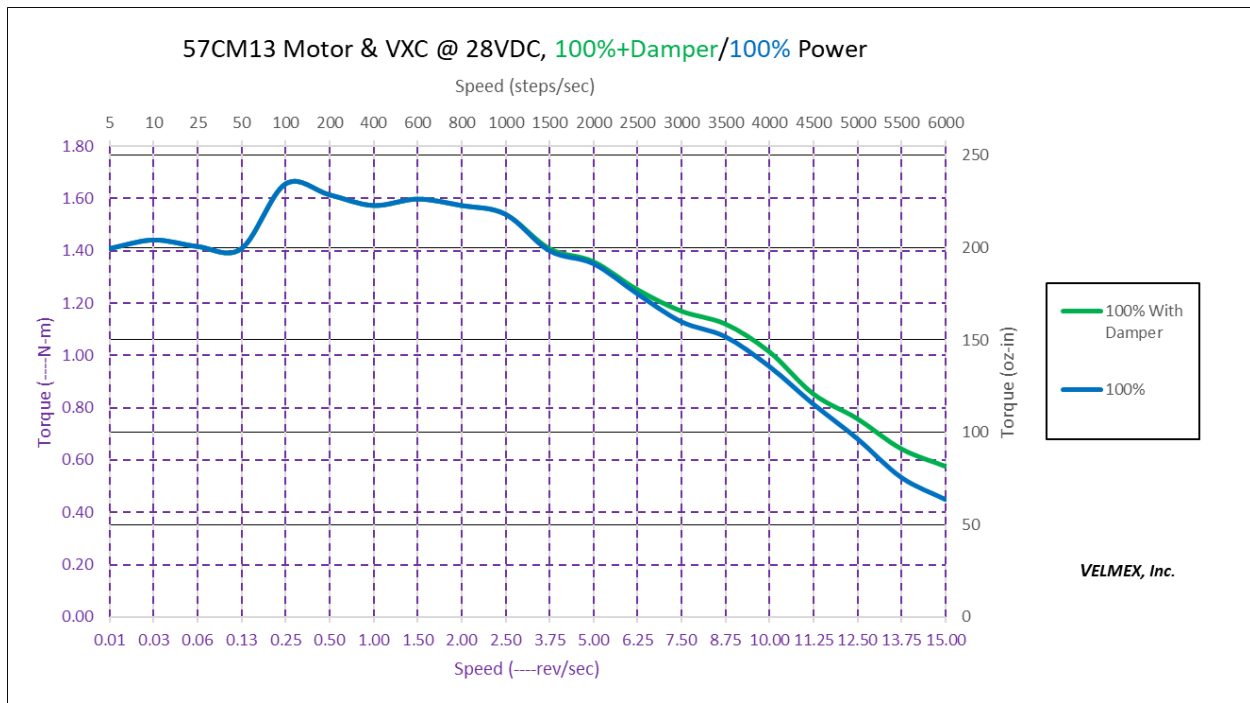


Torque Graph 9 for VML232-4.0-D (57CM13) Motor @ 24V



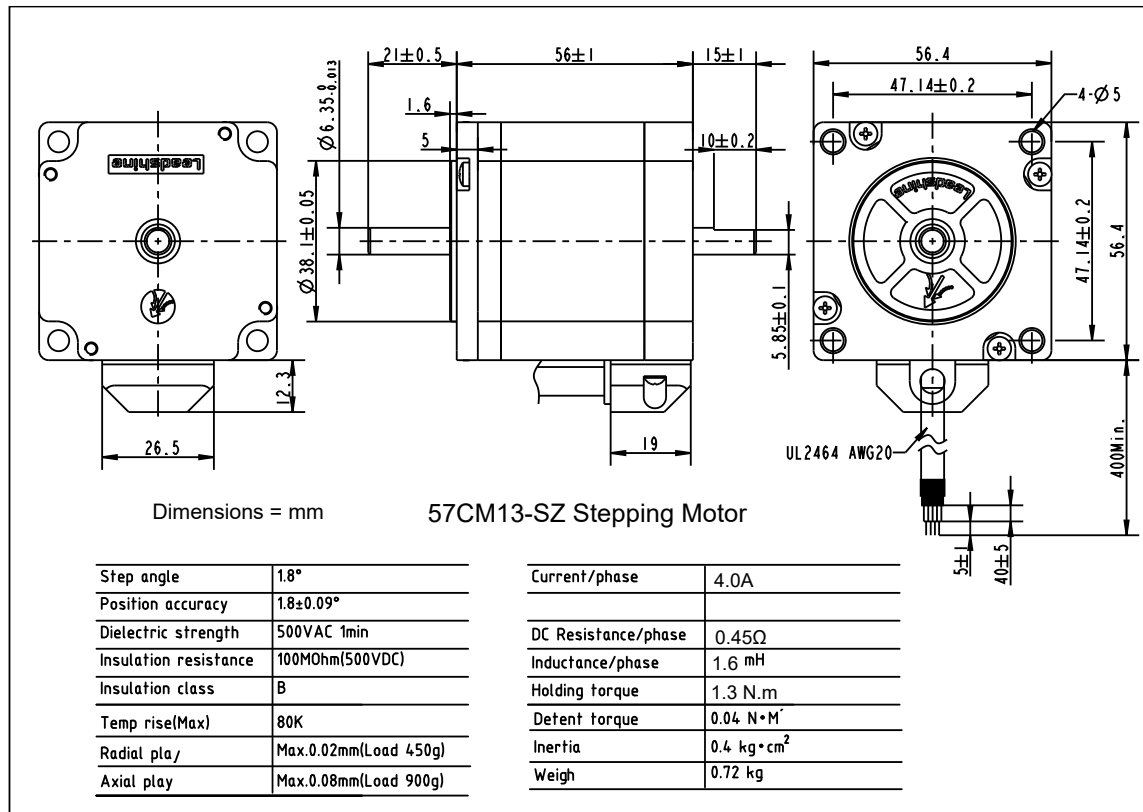
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance.

Torque Graph 10 for VML232-4.0-D (57CM13) Motor @ 28V, 100% Power, With/Without Damper



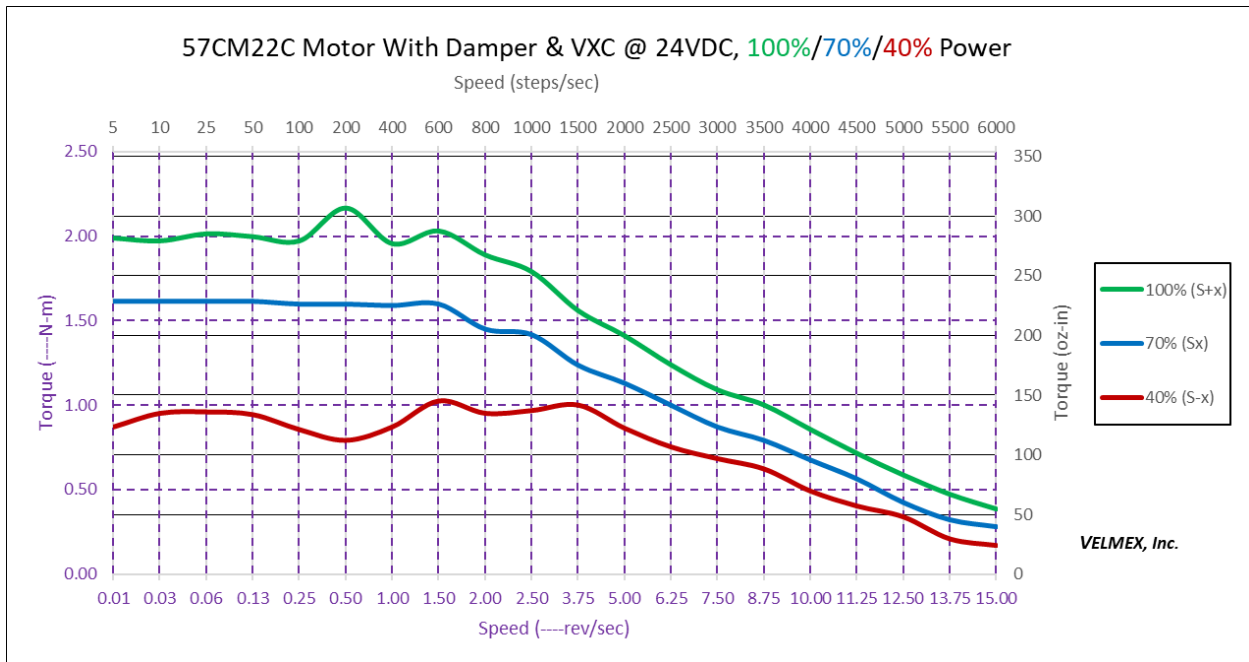
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance.

Figure 26 VML232-4.0-D (57CM13-SZ) Motor Specifications

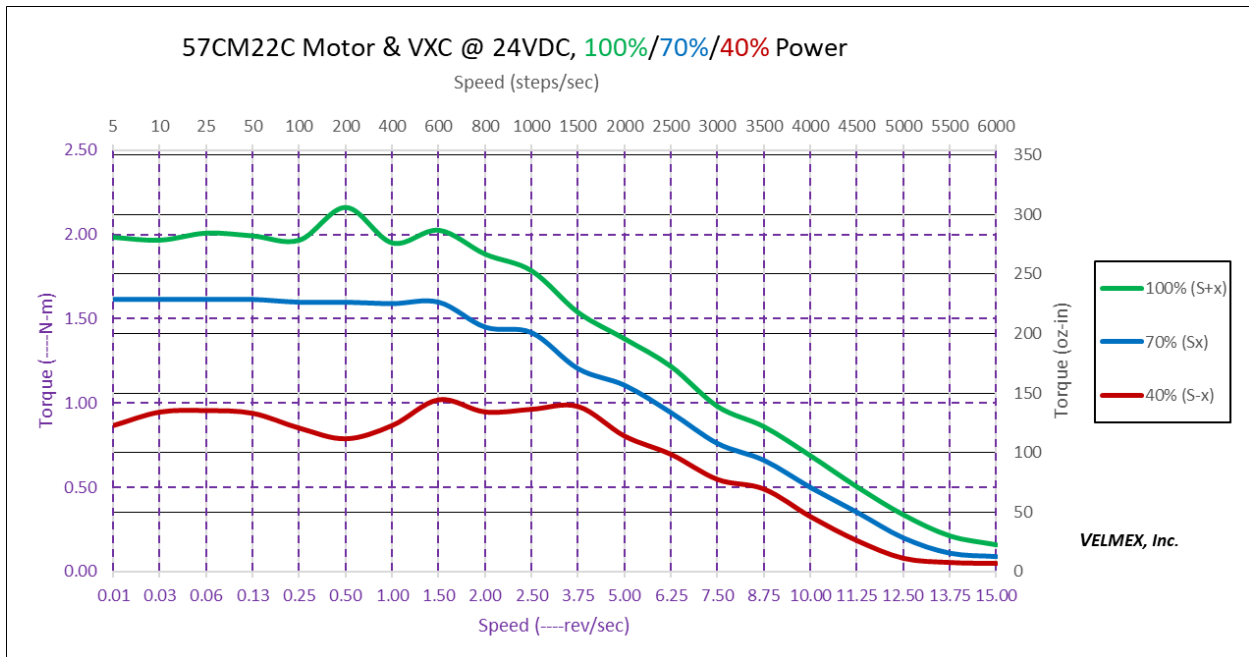


VML233-5.0-D (57CM22C) Motor

Torque Graph 11 for VML233-5.0-D (57CM22C) Motor @ 24V With Damper

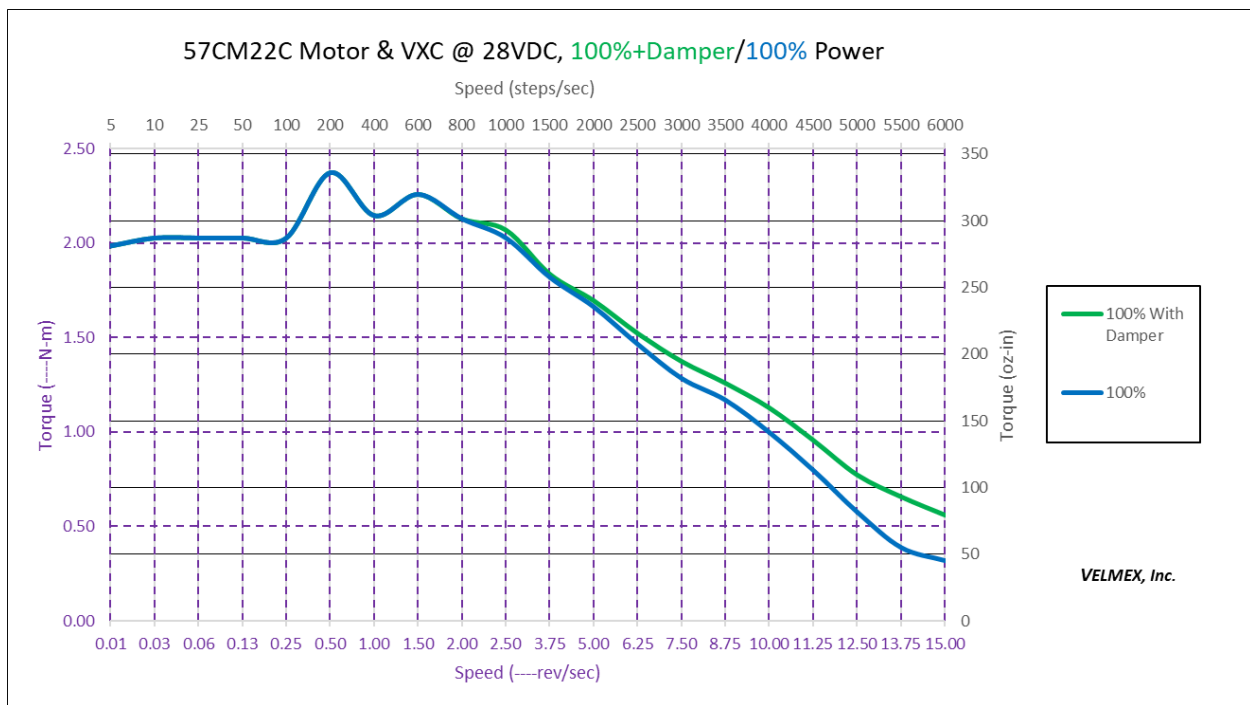


Torque Graph 12 for VML233-5.0-D (57CM22C) Motor @ 24V



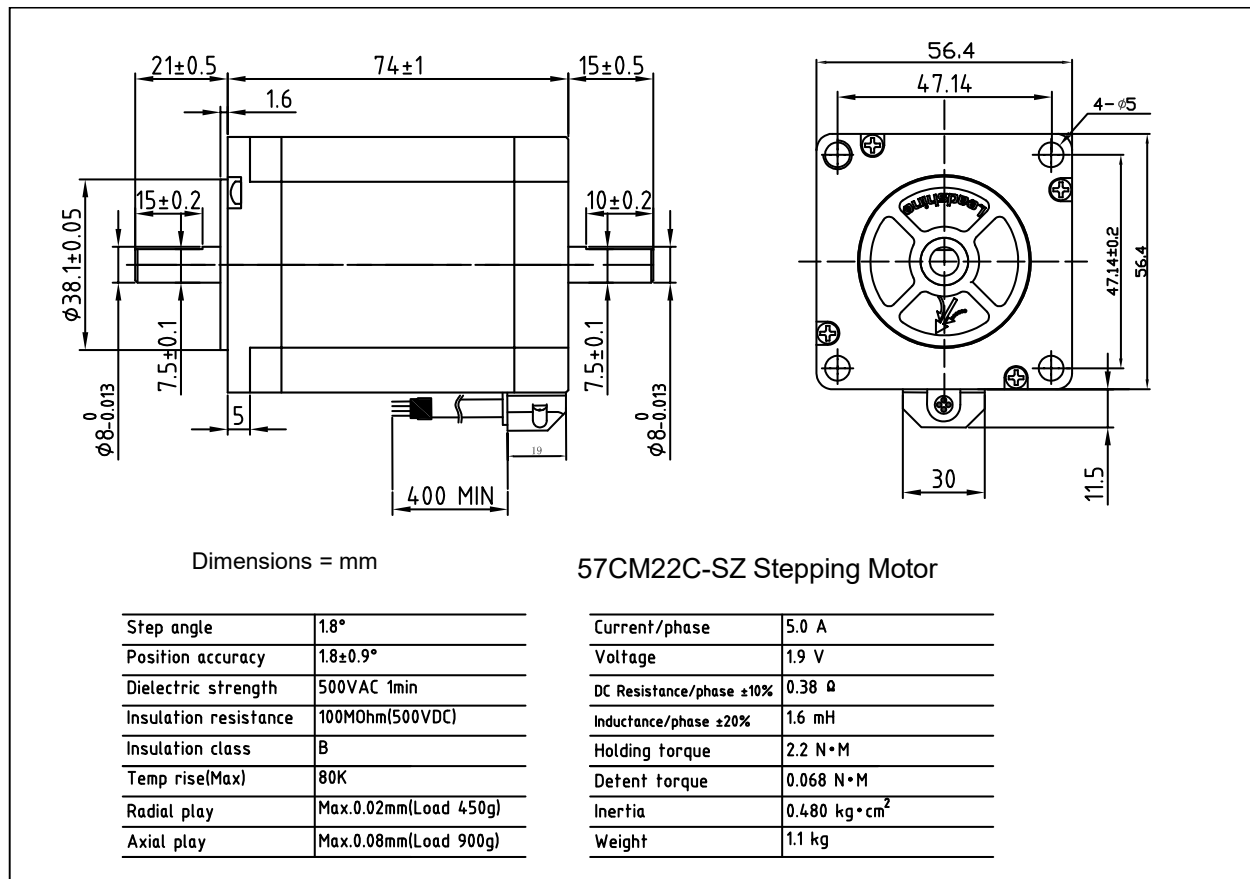
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance.

Torque Graph 13 for VML233-5.0-D (57CM22C) Motor @ 28V, 100% Power, With/Without Damper



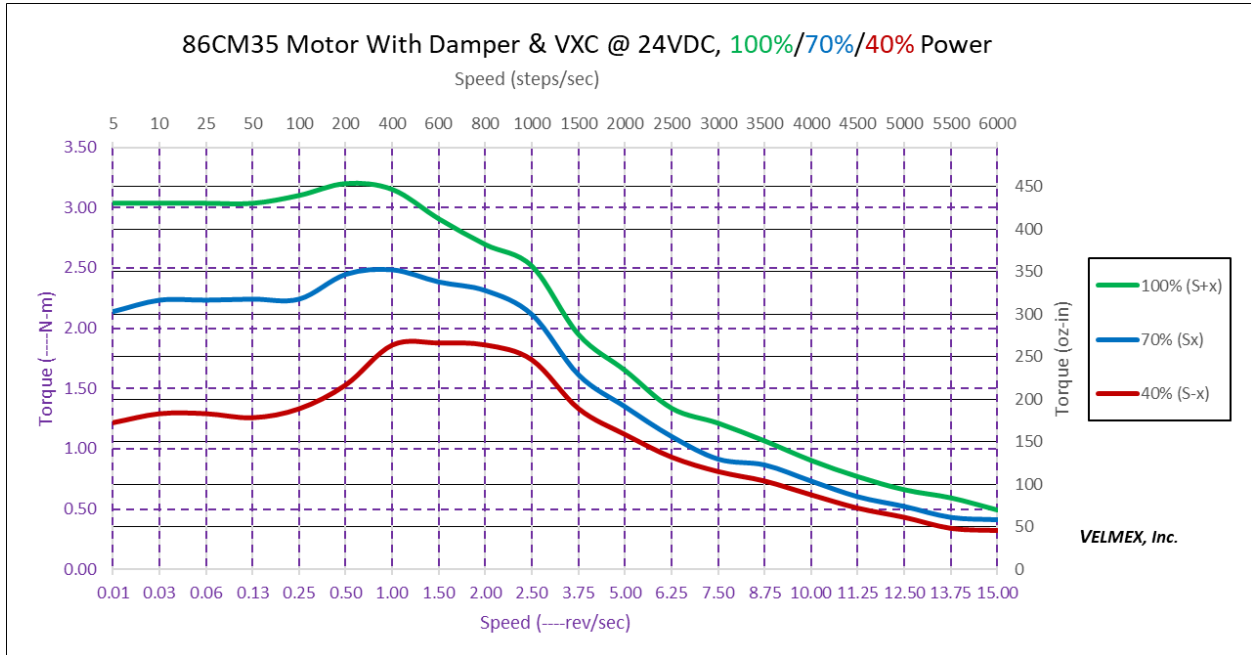
NOTE @ 100% Power Without Damper: 2500-6000 sps the motor requires > 25% torque load to dampen resonance.

Figure 27 VML233-5.0-D (57CM22C-SZ) Motor Specifications

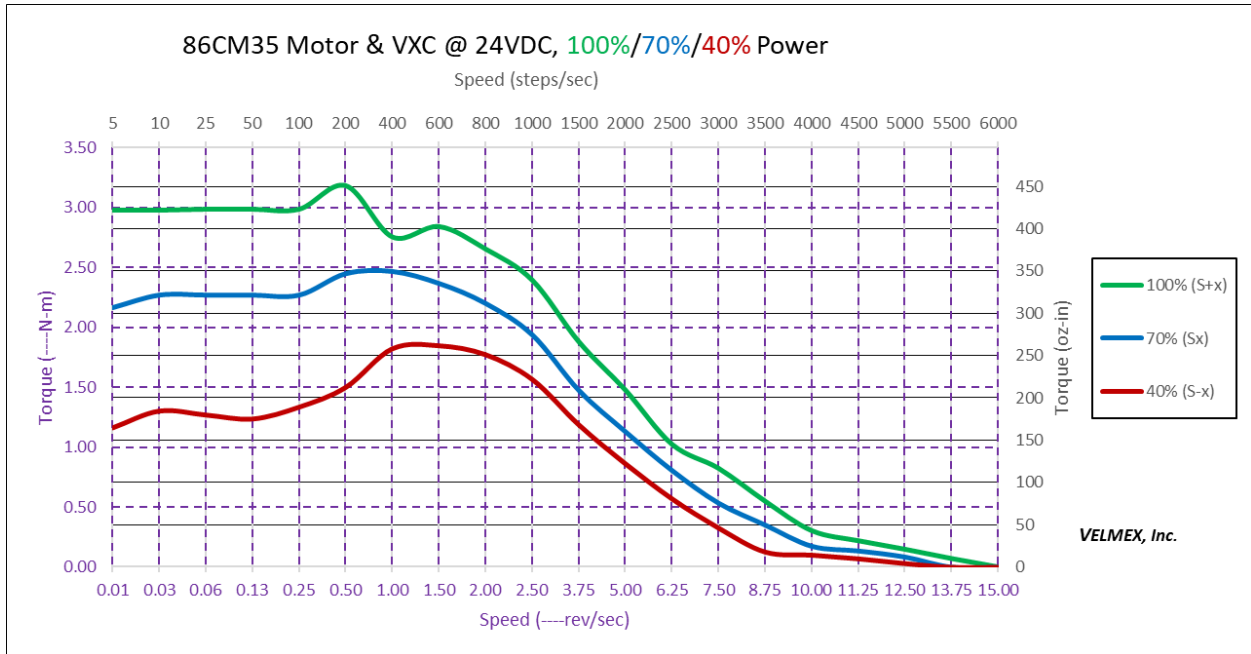


VML341-4.0-D (86M35) Motor

Torque Graph 14 for VML341-4.0-D (86M35) Motor @ 24V With Damper

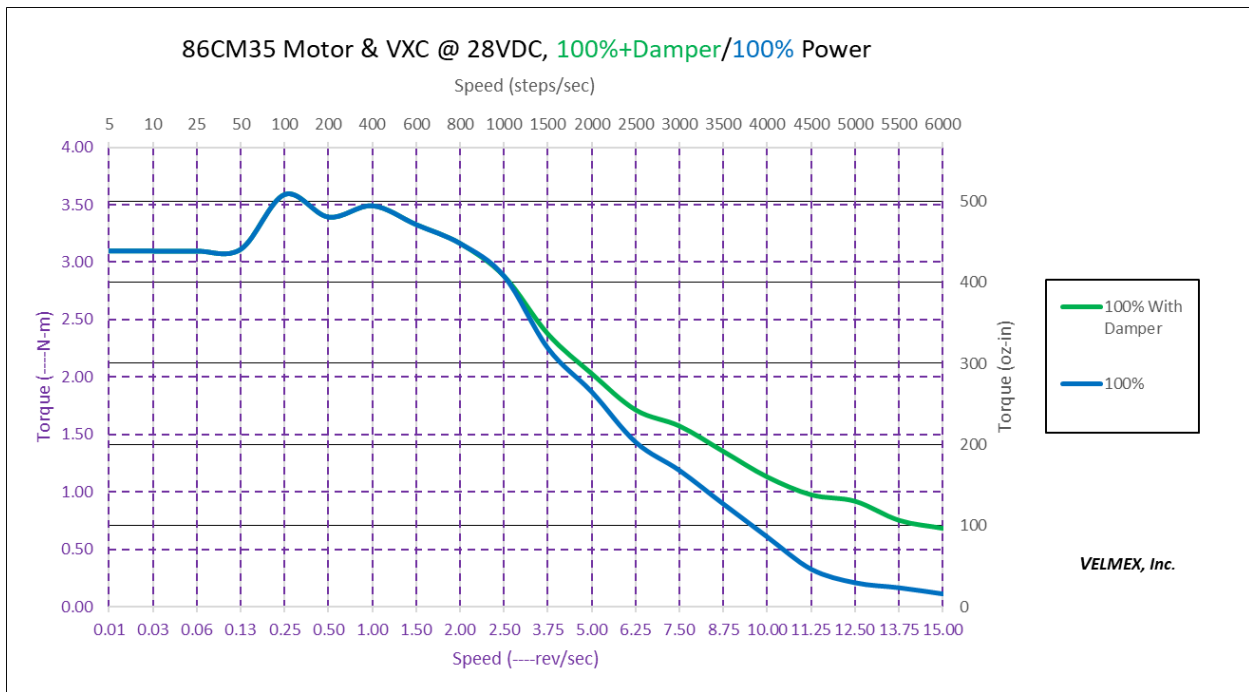


Torque Graph 15 for VML341-4.0-D (86CM35) Motor @ 24V



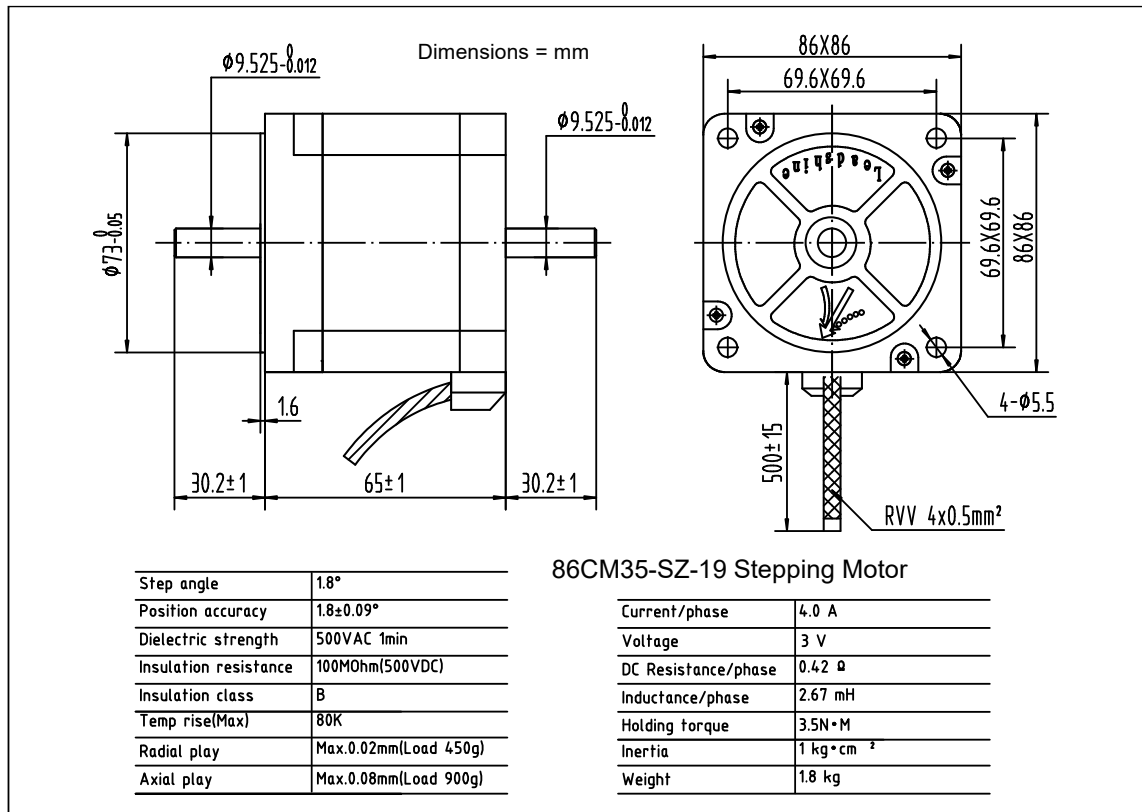
NOTE @ 100% Power Without Damper: 1500-5000 sps the motor requires > 25% torque load to dampen resonance, 5000 sps and higher the motor is unstable.

Torque Graph 16 for VML341-4.0-D (86CM35) Motor @ 28V, 100% Power, With/Without Damper



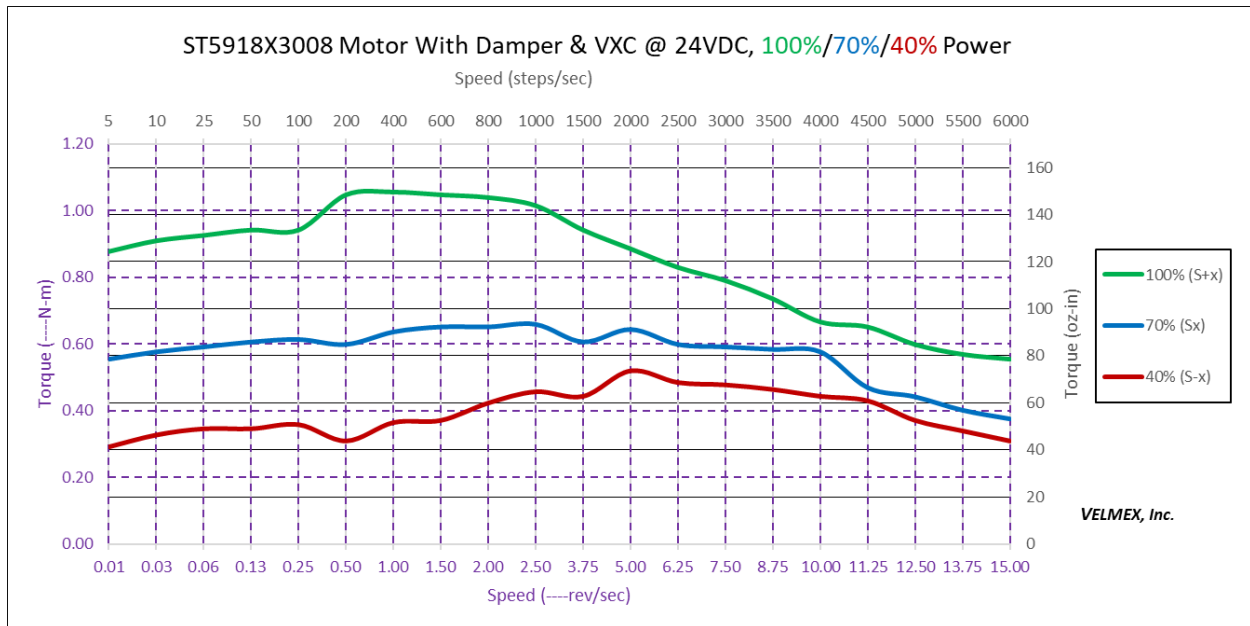
NOTE @ 100% Power Without Damper: 1500-5000 sps the motor requires > 25% torque load to dampen resonance, 5000 sps and higher the motor is unstable.

Figure 28 VML341-4.0-D (86M35-SZ-19) Motor Specifications

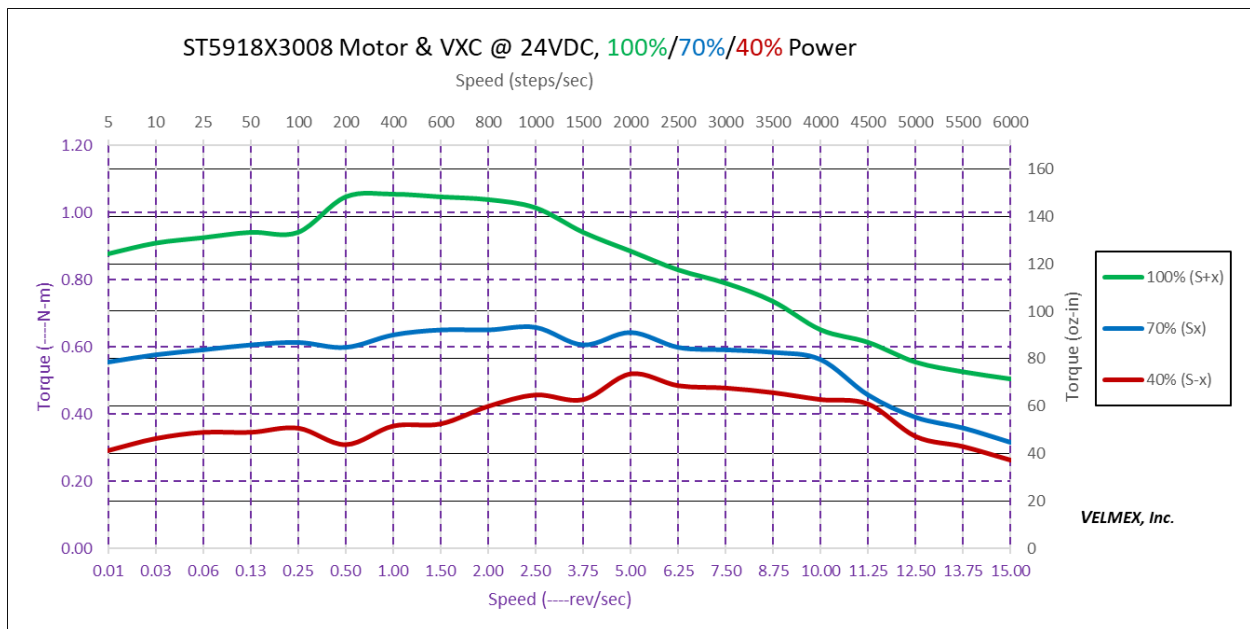


VMN231-4.2-D (ST5918X3008) Motor

Torque Graph 17 for VMN231-4.2-D (ST5918X3008) Motor @ 24V With Damper

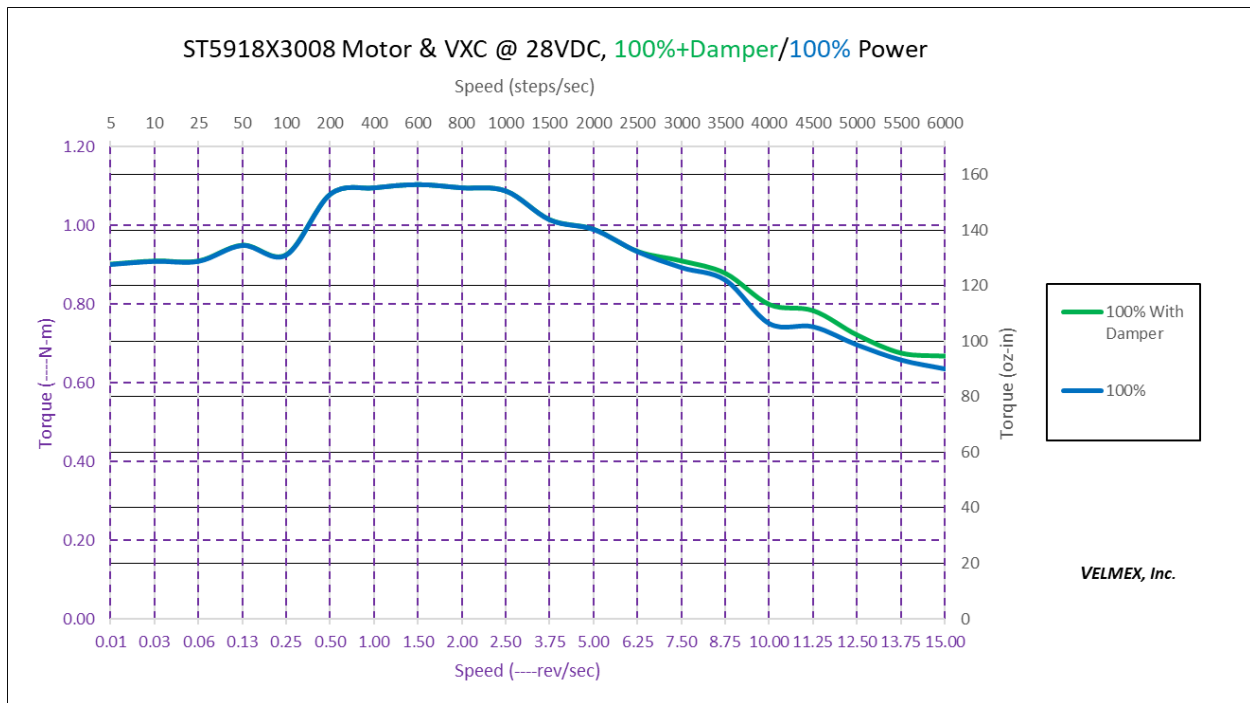


Torque Graph 18 for VMN231-4.2-D (ST5918X3008) Motor @ 24V



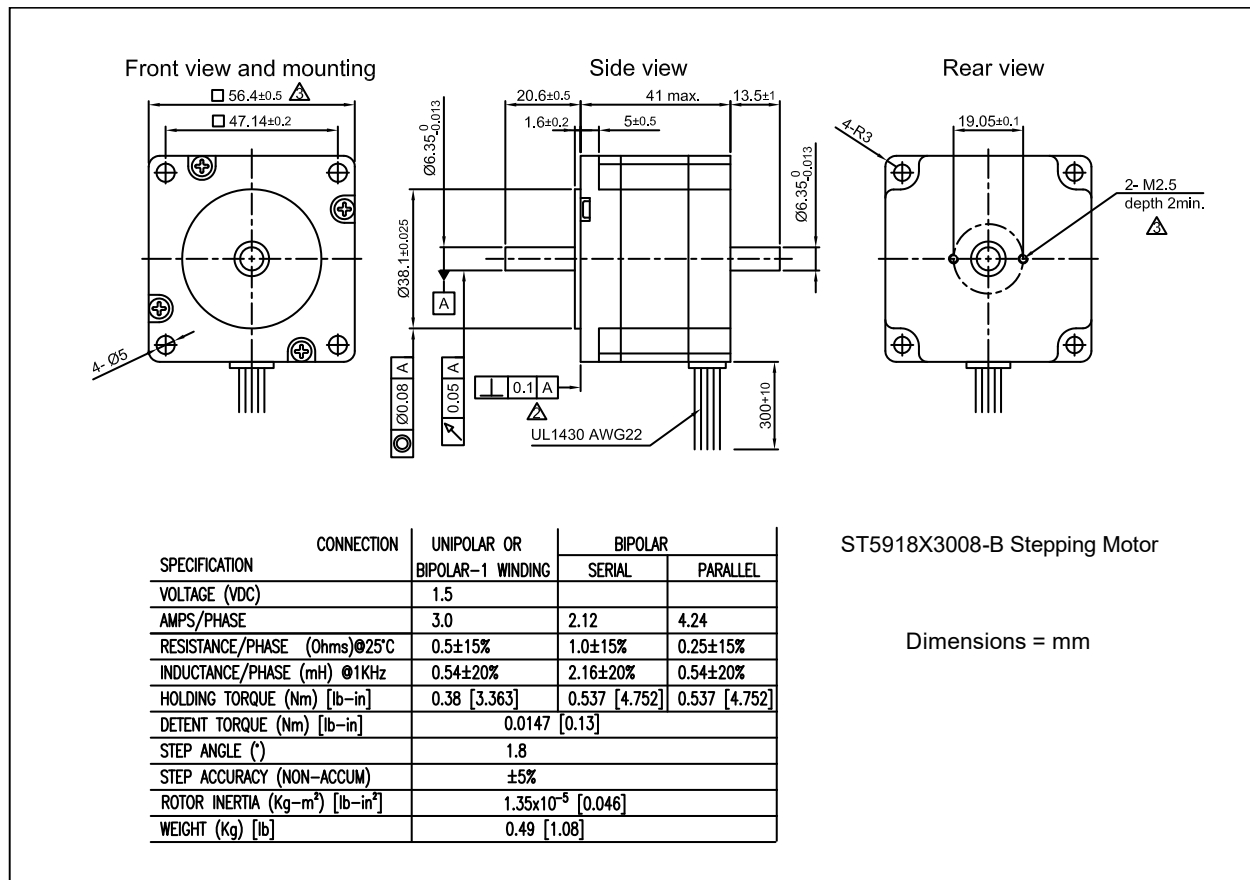
NOTE @ 100% Power Without Damper: 4000-6000 sps the motor requires > 25% torque load to dampen resonance.

Torque Graph 19 for VMN231-4.2-D (ST5918X3008) Motor @ 28V, 100% Power, With/Without Damper



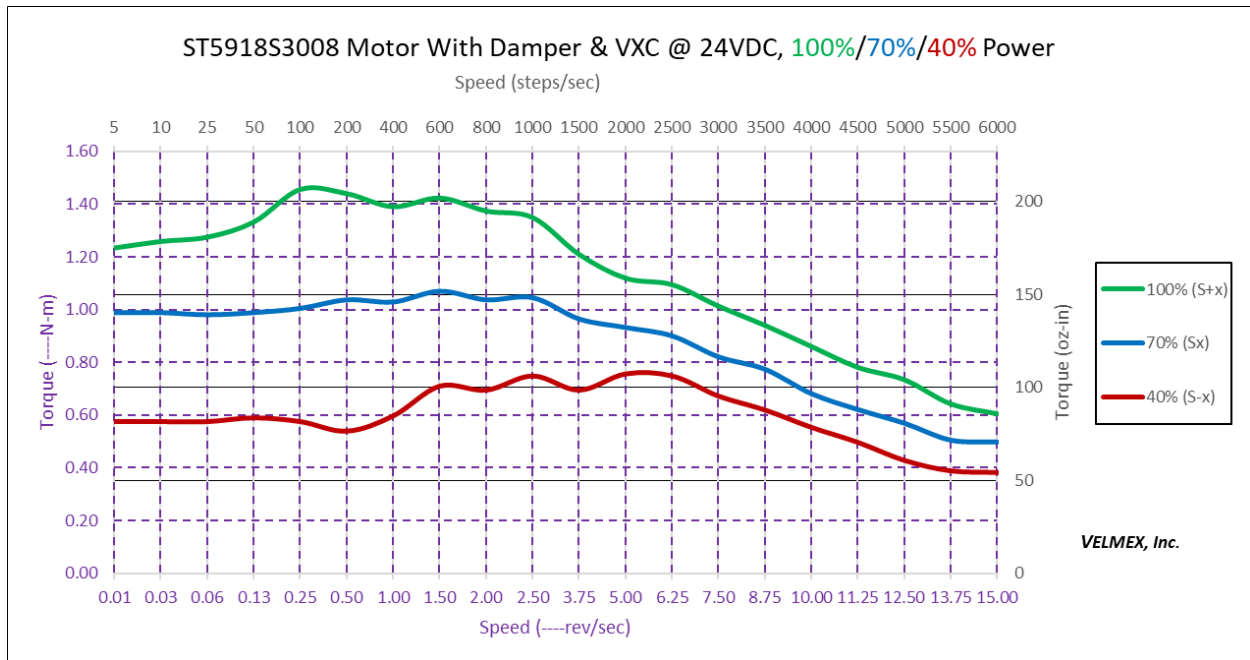
NOTE @ 100% Power Without Damper: 4000-6000 sps the motor requires > 25% torque load to dampen resonance.

Figure 29 VMN231-4.2-D (ST5918X3008-B) Motor Specifications.

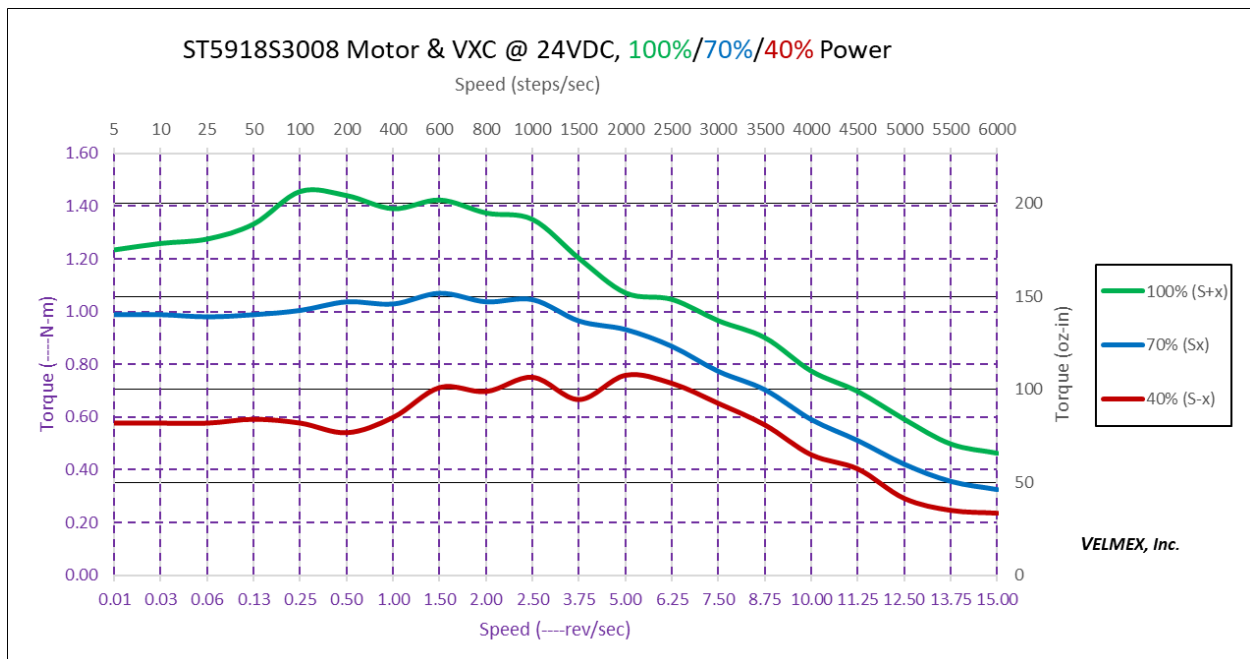


VMN232-4.2-D (ST5918S3008-B)

Torque Graph 20 for VMN232-4.2-D (ST5918S3008-B) Motor @ 24V With Damper

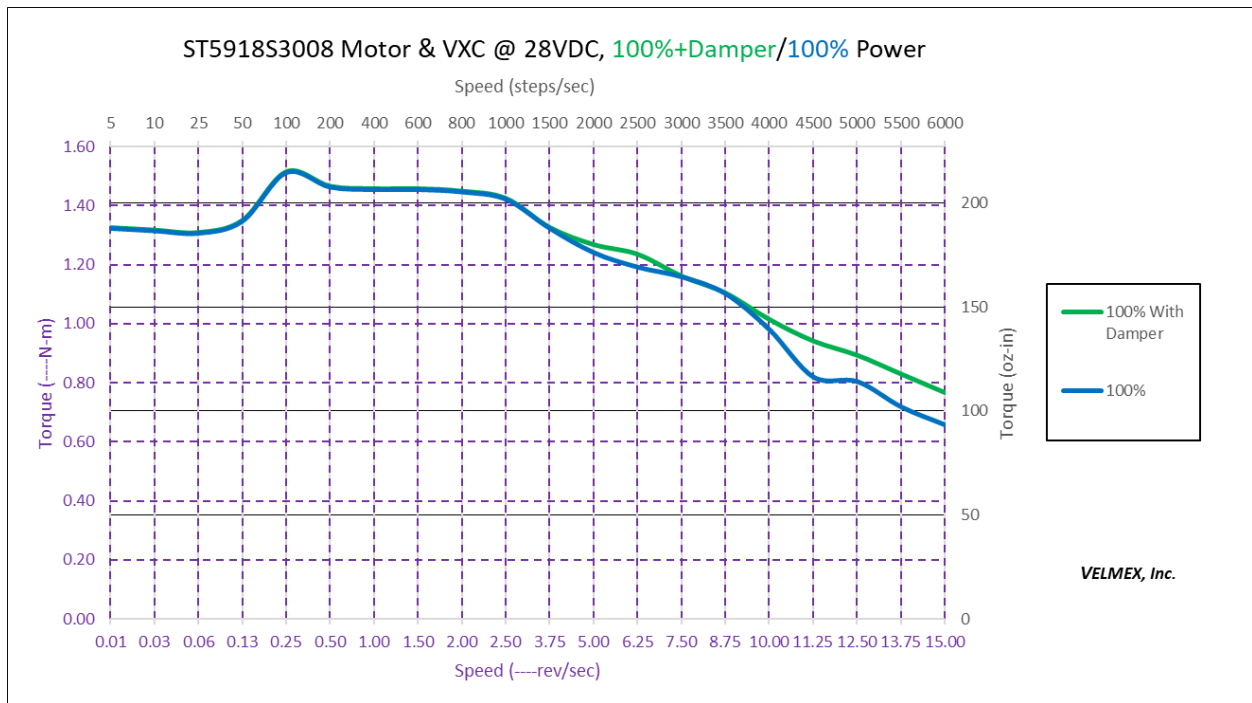


Torque Graph 21 for VMN232-4.2-D (ST5918S3008-B) Motor @ 24V



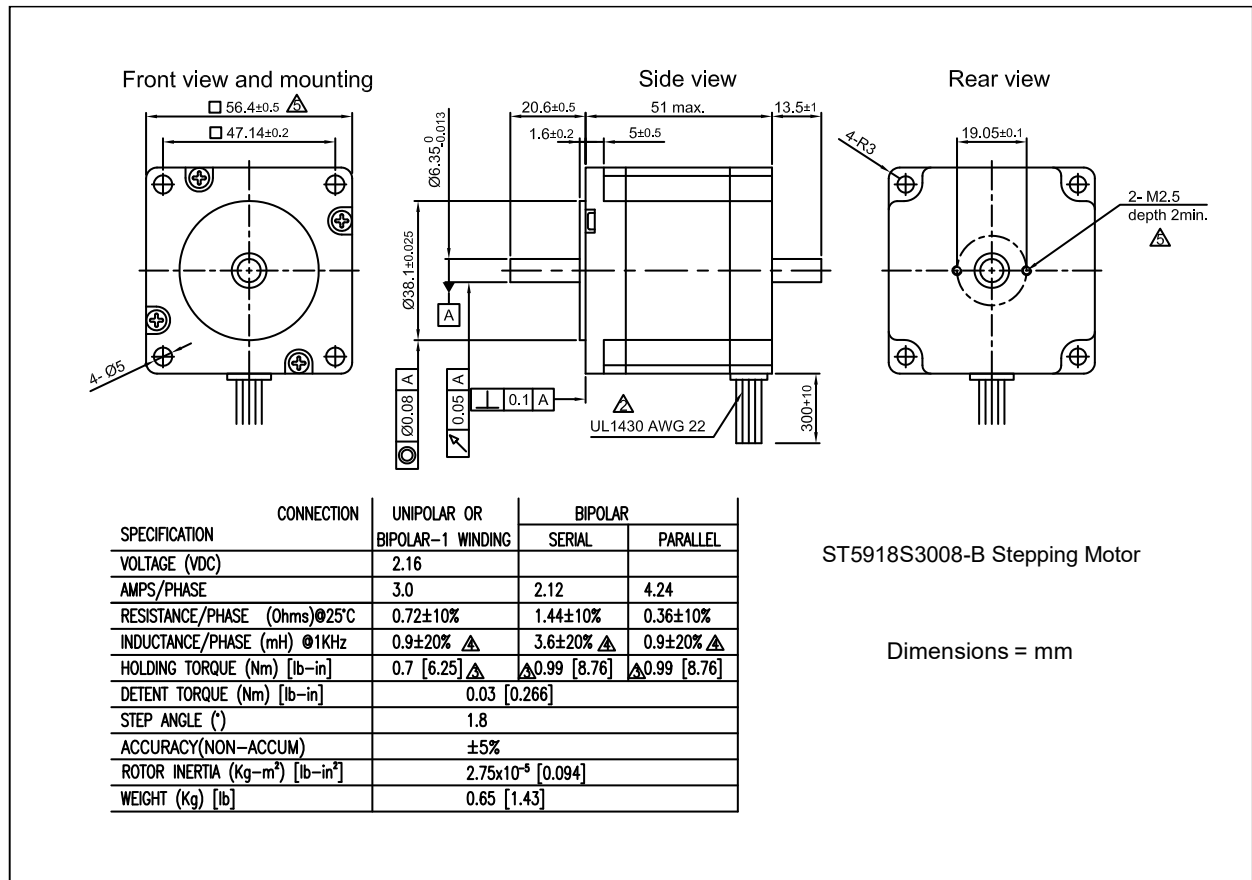
NOTE @ 100% Power Without Damper: 3000-6000 sps the motor requires > 15% torque load to dampen resonance

Torque Graph 22 for VMN232-4.2-D (ST5918S3008-B) Motor @ 28V, 100% Power, With/Without Damper



NOTE @ 100% Power Without Damper: 3000-6000 sps the motor requires > 25% torque load to dampen resonance.

Figure 30 VMN232-4.2-D (ST5918S3008-B) Motor Specifications



Cable Length Versus Torque

The previous torque graphs are based on a VXC with standard 10 ft (3 m) motor cable. For longer cable lengths there will be reduced motor torque due to energy loss in the cable. Table 22 Cable Lengths Torque Factors shows the difference from a standard VXC at 24V operation. “TF” is the multiplier to calculate torque for longer cables.

NOTE: Without factory CLC (Circulating Loss Compensation) the torque loss can be significant at the lower speeds. If the VXC is ordered with a longer cable option Velmex will set CLC.

Table 22 Cable Lengths Torque Factors

Cable Length(ft)	Cable Length(m)	TF no CLC	TF with CLC	TF with CLC @ 28V
20	6.1	0.88	0.94	1.10
30	9.1	0.82	0.90	1.06
40	12.2	0.76	0.87	1.03
50	15.2	0.70	0.84	1.00
60	18.3	0.64	0.81	0.97
70	21.3	0.58	0.77	0.93
80	24.4	0.52	0.74	0.90
90	27.4	0.46	0.71	0.87
100	30.5	0.40	0.68	0.84

$$\text{Torque} = [\text{Graph torque}] \times \text{TF}$$

TF no CLC = Torque multiplier with no factory CLC (Circulating Loss Compensation)

TF with CLC = Torque multiplier with factory CLC

TF with CLC @28V = Torque multiplier with factory CLC with 28V option

Example VM341-4.0-D-F, 2500 steps/sec, 100% power (S+2500), “TF with CLC” & 50 ft cable

From Table 22 Cable Lengths Torque Factors:

Cable length= 50 ft (15.2m)

TF with CLC= 0.84

From Torque Graph 15 for VML341-4.0-D (86CM35) Motor @ 24V:

[Graph torque] = 1.00 N-m

$$\text{Torque} = [\text{Graph torque}] \times \text{TF}$$

$$\text{Torque} = 1.00 \text{ N-m} \times 0.84$$

$$\text{Torque} = 0.84 \text{ N-m} = 119 \text{ oz-in}$$

Summary:

With standard length cables the torque is 1 N-m versus 0.84 N-m with 50 ft cables.