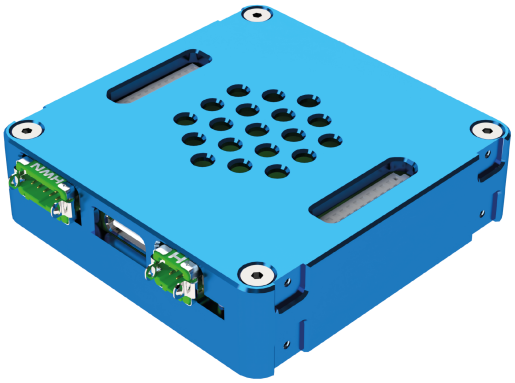
NanoDrive Series



Motion Controller
Reference Manual

micronix USA
PRECISION MOTION SOLUTIONS

Table of Contents

1. Introduction 1-3
1.1 Product Description 1-3
1.2 Features 1-4
1.3 Package Contents 1-4
2. Quick Start Guide 2-5
2.1 Quick Start Guide Overview 2-5
2.2 Quick Start Micronix Motion Controller Platform 2-8
2.3 Using the Micronix Motion Controller Platform 2-9
3. Technical Information 3-13
3.1 NanoDrive Specifications 3-13
3.2 USB Communication 3-13
3.3 RS-485 Communication 3-14
4. Operation 4-15
4.1 Axis Addressing 4-15
4.2 Motor Control Type 4-16
4.3 Feedback Control 4-16
4.4 HOM, MLN, and MLP 4-17
4.5 I/O Commands 4-17
5. Commands 5-21
5.1 Command Line Syntax 5-21
5.2 Command Line Format 5-22
5.3 Global Commands 5-22
5.4 Multiple Parameters 5-22
5.5 Synchronous Move 5-22
5.6 Internal Programming 5-23
5.7 Summary of Commands 5-24
5.8 Command Descriptions 5-27
5.9 Error Messages 5-116
6. Appendix 6-120
6.1 Power Input Pin-out 6-120
6.2 GPIO and Communication Pin-out 6-121
6.3 Motor & Encoder Pin-out 6-122
6.4 Firmware 6-123
6.5 3-Phase Brushless Motor Initialization 6-123
6.6 Closed Loop Operation 6-123
6.7 Motor Current Operation 6-124

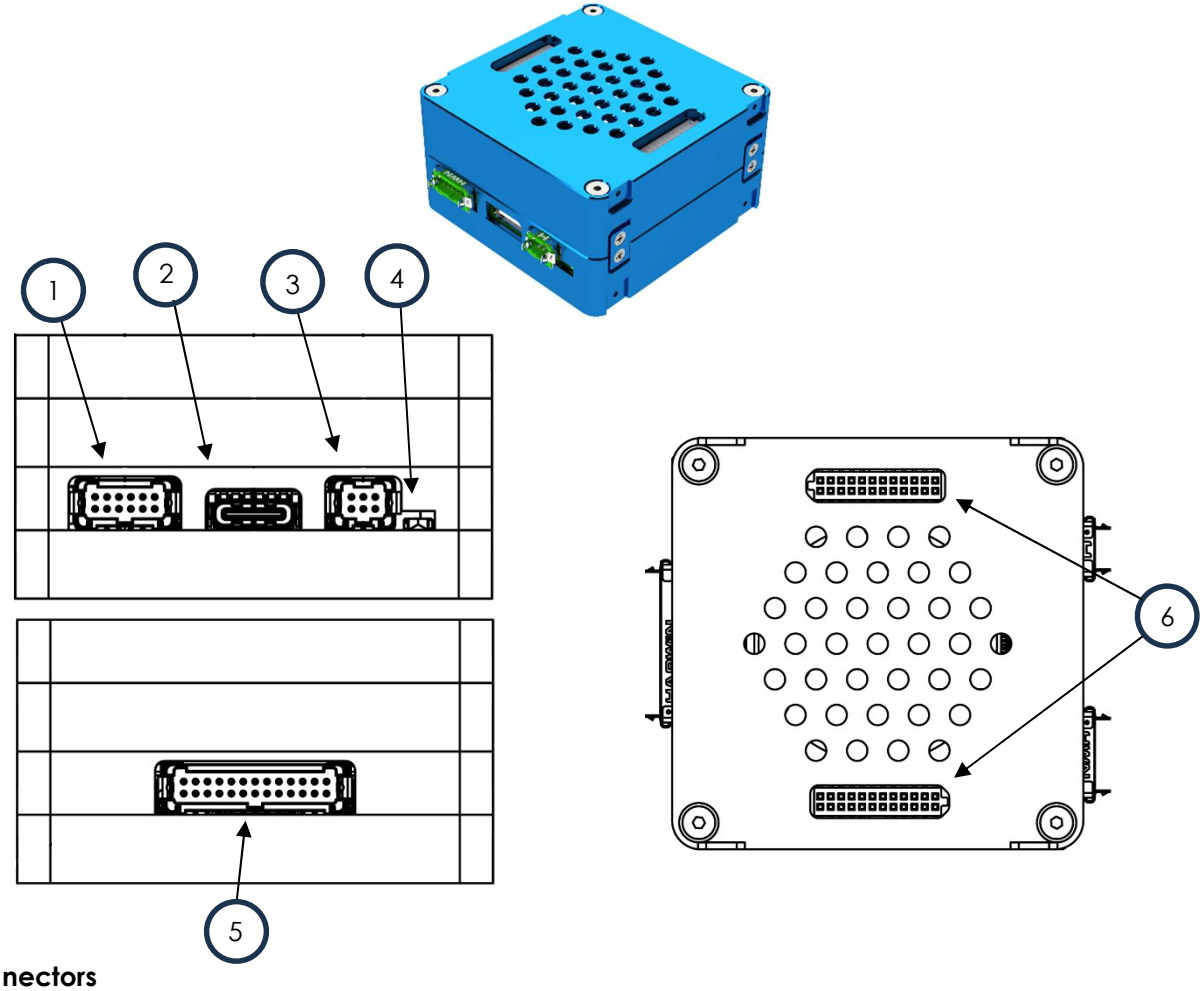
Command Index

Command	Description	Page	Command	Description	Page
ACC	Acceleration	27	LSP	Lead Screw Pitch	72
AMX	Maximum Acceleration	28	LST	Program List	73
ANR	Set Axis Number	29	MCM	Maximum Motor Current	74
CER	Clear Errors	30	MCR	Motor Current Reduction	75
CVG	Current Velocity Gain	31	MCS	Motor Current Setting	76
CVL	Correction Velocity	32	MFE	Max Following Error	77
CVP	Pulse at Constant Velocity	33	MLN	Move to Negative Limit	78
DAT	Trace Data Export	34	MLP	Move to Positive Limit	79
DBD	Closed Loop Deadband	35	MOT	Toggle Motor On/Off	80
DEC	Deceleration	36	MPL	Motor Polarity/Direction	81
DEF	Restore Defaults	37	MSA	Synchronous Move – Absolute	82
EAD	Set Encoder Type	38	MSR	Synchronous Move – Relative	83
EFF	Encoder Filter	39	MTC	Motor Control Type	84
EMF	Back EMF Constant	40	MTP	Motor Pole Pitch	85
ENC	Encoder Resolution	41	MTV	Motor Voltage	86
END	End Program Recording	42	MVA	Absolute Move	87
EPL	Encoder Polarity/Direction	43	MVR	Relative Move	88
ERA	Erase Program	44	PGL	Loop Program	89
ERR	Read Errors	45	PGM	Program Mode	90
EST	Emergency Stop	46	PGS	Run Program At Start-Up	91
EXC	Execute Program	47	PID	Set Feedback Constants	92
FBK	Set Control Loop Mode	48	PIP	Pulse at Position Interval	93
FFP	Feed Forward Parameter	49	POS	Position	94
FSR	Full Steps Per Revolution	50	PTP	Pulse at Target Position	95
HAC	Home Acceleration	51	PWM	PWM Operating Frequency	96
HCG	Home Configuration	52	REZ	Motor Resolution	97
HOM	Move to Home	53	RUN	Run Synchronous Move	98
HST	Hard Stop Detection	54	SAV	Save	99
HVL	Home Velocity	55	SCM	Stepper Control Method	100
IDN	Identification Name	56	STA	Status Byte	101
INI	Initialize BLDC Phasing	57	STP	Stop Motion	102
INP	In Position	58	SVP	Startup Position	103
IOD	I/O Direction	59	SYN	Sync	104
IOF	I/O Function	60	TLN	Negative Soft Limit Position	105
IOP	I/O Polarity	61	TLP	Positive Soft Limit Position	106
IOS	I/O Status	62	TRA	Trace	107
IWL	Integrator Windup Limit	63	VEL	Velocity	108
JAC	Jog Acceleration	64	VER	Firmware Version	109
JOG	Jog Mode	65	VMX	Maximum Velocity	110
LCG	Limit Configuration	66	VRT	Encoder Velocity	111
LDP	Load Parameters	67	WST	Wait For Stop	112
LDR	Limit Switch Direction	68	WSY	Wait For Sync	113
LPF	Low Pass Filter	69	WTM	Wait For Time Period	114
LPL	Limit Switch Polarity	70	ZRO	Zero Position	115
LRB	Limit Rebound	71			

1. Introduction

1.1 Product Description

The NanoDrive is a high-performance single axis integrated piezo, stepper, 3-phase brushless, and DC motor controller/driver designed to be used as a standalone single axis unit or stacked as a compact multi-axis module. The NanoDrive is capable of driving a motor with a step size resolution of less than 1 nm (motor dependent). The closed loop resolution is dependent on the resolution of the encoder.



Connectors

1. I/O and Communication - Male 12-Pin Harwin Gecko Connector
2. Communication and Power - USB-C Connector (PD Compatible – 100W Max)
3. Power - Male 6-Pin Harwin Gecko Connector
4. LED Power/Error Indicator
    a. Teal – Performing boot-up sequence
    b. Blue – Initializing
    c. Red – Error
    d. Green – Normal Operation
5. Motor, Encoders, and Limits - Male 26-Pin Harwin Gecko Connector
6. Multi-axis Connectors – Female 24-Pin Connector

- See Appendix for connector pinouts and suggested mating connectors

1.2 Features

- Integrated controller/driver for various motor types
  - 2-Phase Stick-Slip Piezo
  - Bipolar Stepper
  - 3-Phase Brushless
  - DC Motors
- Support for the following encoder types
  - Incremental (A-B Quadrature and Z Index)
  - Absolute (BiSS-C Protocol)
- Compact, modular design
- Configurable as a standalone unit or stackable up to 16 axes
- Open loop/closed loop operation (motor dependent)
- Open loop resolution of less than 1 nm (motor dependent)
- Closed loop resolution dependent on the encoder (typically 2nm)
- USB interface (one interface for all axis)
- USB Power Delivery capable of supplying 100W (20V/5A)
- Windows GUI, and LabVIEW VI

1.3 Package Contents

If the product is damaged or if there are missing components, please contact MICRONIX USA immediately. Do not discard product packaging in case of return shipment.

Package Contents:

- NanoDrive Controller
- Quick Start Guide
- Power Supply
- USB-C to USB-A Cable
- Adapter Cable(s)

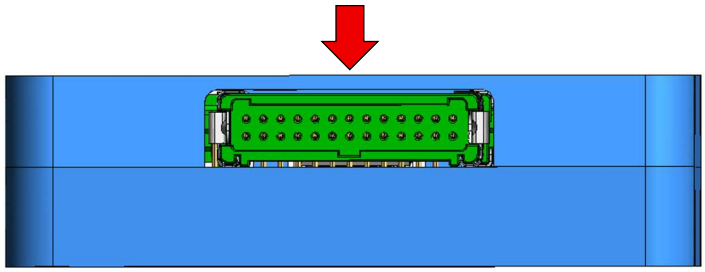
2. Quick Start Guide

2.1 Quick Start Guide Overview

The following Quick Start Guide is intended to provide a basic set-up of the NanoDrive in the least amount of time. The following paragraphs will provide a walkthrough of the steps needed to set up the controller and verify that the system is working correctly.

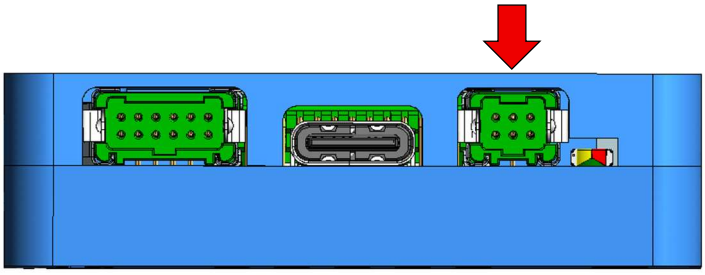
1. Install USB Drivers
    a. For Windows 10 and beyond, the native inbox driver will automatically be installed once the NanoDrive device is connected to the PC upon first-time USB connection.
    b. For OS versions prior to Windows 10 (e.g. Windows 7 and Windows 8), users will need to manually install ST's STM32 Virtual COM Port Driver (STSW-STM32102). The software/drivers can be downloaded from ST website: https://www.st.com/en/development-tools/stsw-stm32102.html

2. Connect Motion Device
    a. A single NanoDrive controller is capable of driving a piezo, stepper, 3-phase brushless, or DC motor. It supports both sr incremental and absolute BiSS-C encoders.
    b. The NanoDrive is typically shipped with pre-configured parameters for a particular stage & encoder type. There is a label on the side of the NanoDrive that lists out the stage part number.
    c. Due to the different modes of operation for each motor and encoder type, users should pair the NanoDrive to the specified stage and encoder.
        i. Warning: Failure to pair the NanoDrive configuration to the correct stage motor type and encoder type can result in damage to both stage and controller.
        ii. Consult the respective stage and controller datasheets for compatibility
    d. Once the stage is verified to be matched with the corresponding NanoDrive, connect the female 26-pin motor cable from the stage to the Motor/Encoder connector on the NanoDrive. Note that the Motor/Encoder connector should only be connected or disconnected while the NanoDrive is NOT powered.



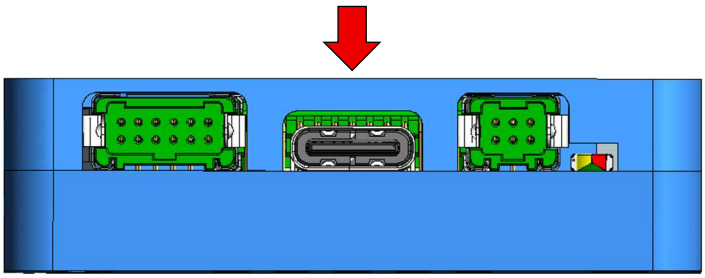
    e. Ensure that the latches are properly secure to the NanoDrive after connecting the 26-pin motor cable. The 26-pin connection will have a retention force holding the connectors in place.

3. Powering On the Controller
    a. A 6-Pin Gecko connector to 2.1mm Plug Barrel Plug adapter cable is provided with each NanoDrive.
    b. Connect the 6-pin adapter cable to the NanoDrive. Confirm that the connection is securely fastened by checking if the metal latches are visible on both sides of the connection.



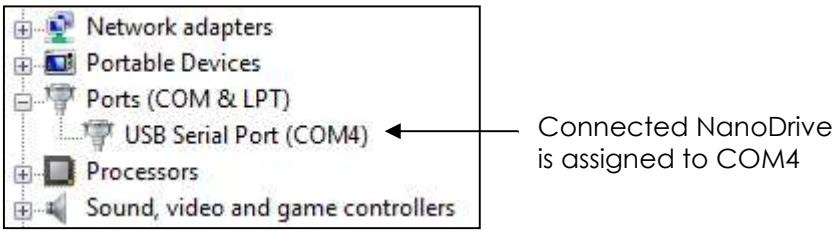
    c. Connect the controller to its included power supply.
        i. The NanoDrive can also be powered with a 12V – 55V regulated power supply with the correct amperage rating.
    d. Each NanoDrive requires up to 5A depending on the motor settings. If powering a stack; add up the current requirements of the individual controllers to determine the necessary power supply for the stack.
    e. Once powered on, the NanoDrive will start its bootup process with the status indicated by the LED color. This process takes a few seconds; once the LED is green, the NanoDrive is ready for use..

4. Connect Module/Stack to PC
    a. Use the supplied USB-C cable to connect the NanoDrive controller to the communicating PC. Only one USB cable is required per module/stack.



5. Check COM Port
    a. It is necessary to note the COM Port assigned to the NanoDrive when connecting to a PC. In Windows, this can be done by checking Device Manager.

    b. After powering up the controller (Step 3), note the USB Serial Port assigned. See the figure below showing a snapshot of the Device Manager window:



6. COM Port Drivers (Window OS before Windows 10 only)
    a. For operating systems before Windows 10, the STM32 Virtual COM Port Driver is required for the system to recognize and communicate with the NanoDrive.
    b. Install the STSW-STM32102 driver from ST website.
    https://www.st.com/en/development-tools/stsw-stm32102.html

7. Continue to Quick Start NanoDrive Motion Controller Platform
    a. The following section will help you get running with the Motion Controller Platform program.

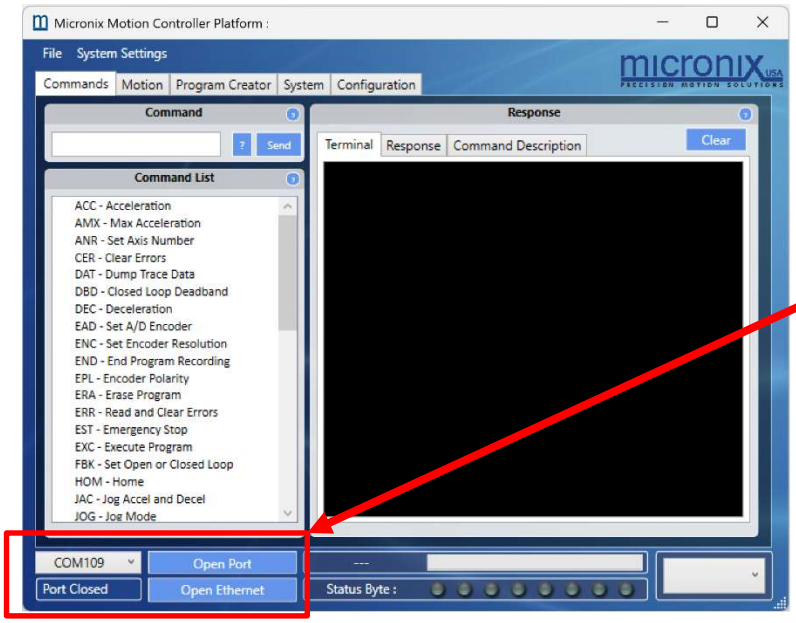
2.2 Quick Start Micronix Motion Controller Platform

The following Quick Start Guide is intended to provide a basic set-up of the Micronix GUI program. The following details will provide a walkthrough of the steps needed to install the program and verify that the system is working correctly.

1. Pre-Installation
    a. This guide assumes you have already completed the previous Quick Start guide and that the controller is on and connected to a COM port on your computer.
2. Install
    a. To install the Micronix Motion Controller Platform, run the installation file from the supplied USB and follow the onscreen instructions.
3. Run
    a. The installer will place a shortcut to the Micronix GUI program on your desktop. Make sure that your NanoDrive is powered on and connected to a valid COM port as discussed in section 2.1
    b. Open the start menu
    c. Open the 'All Programs' tab
    d. Open the MICRONIX USA folder
    e. Run the Micronix MCP program

2.3 Using the Micronix Motion Controller Platform

1. Connect to the NanoDrive via USB or Ethernet

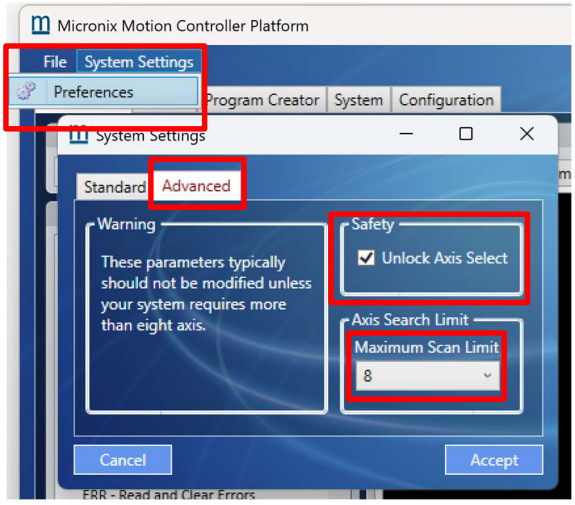


User should select the appropriate communication that controller is connected to.

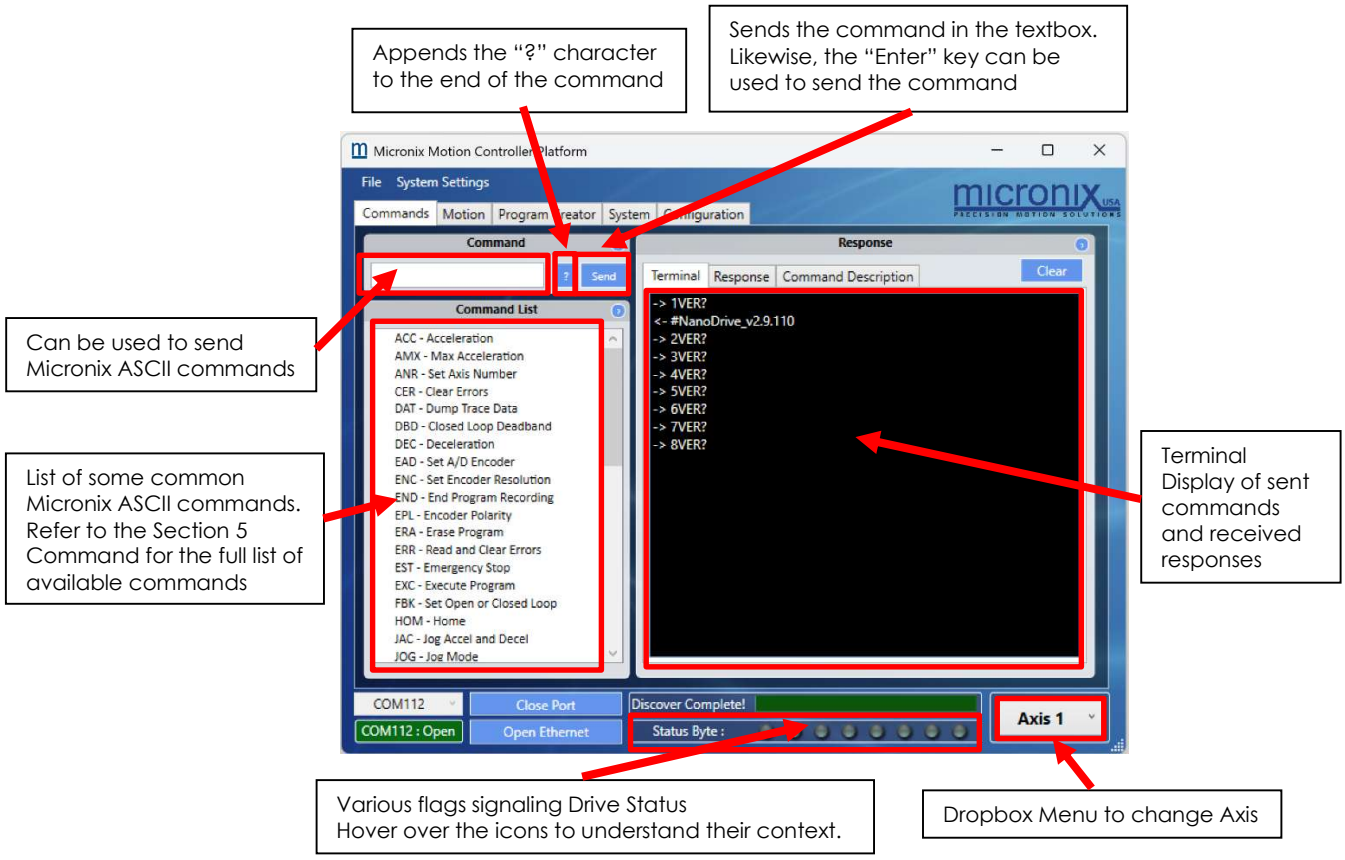
Once the communication is established, the terminal will populate with controller information. Once completed, the system is ready for use.

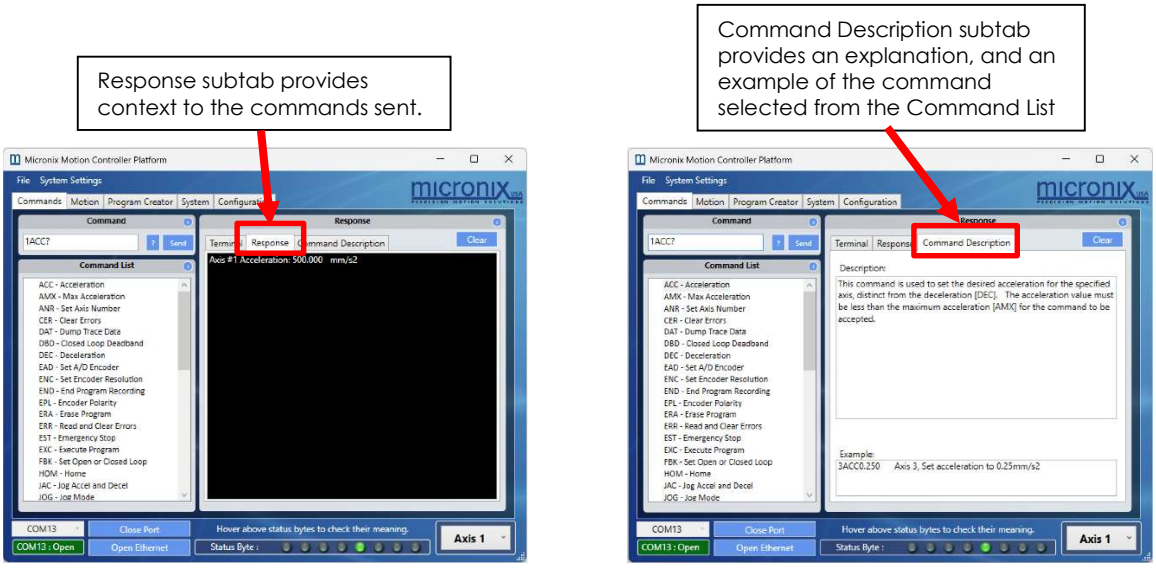
   a. For IP connection over Ethernet, use the additional NanoDrive-EXP add-on board. Refer to the NanoDrive-EXP reference manual for details.

2. Once connected to the NanoDrive via USB or Ethernet, the MCP program will poll the number of available axes(up to 8 by default)
   a. To bypass the 8-axis limit, navigate to the System Settings menu → Preferences → Advanced tab → Check the "Unlock Axis Select" and adjust the Maximum Scan Limit to the desired number of axes.

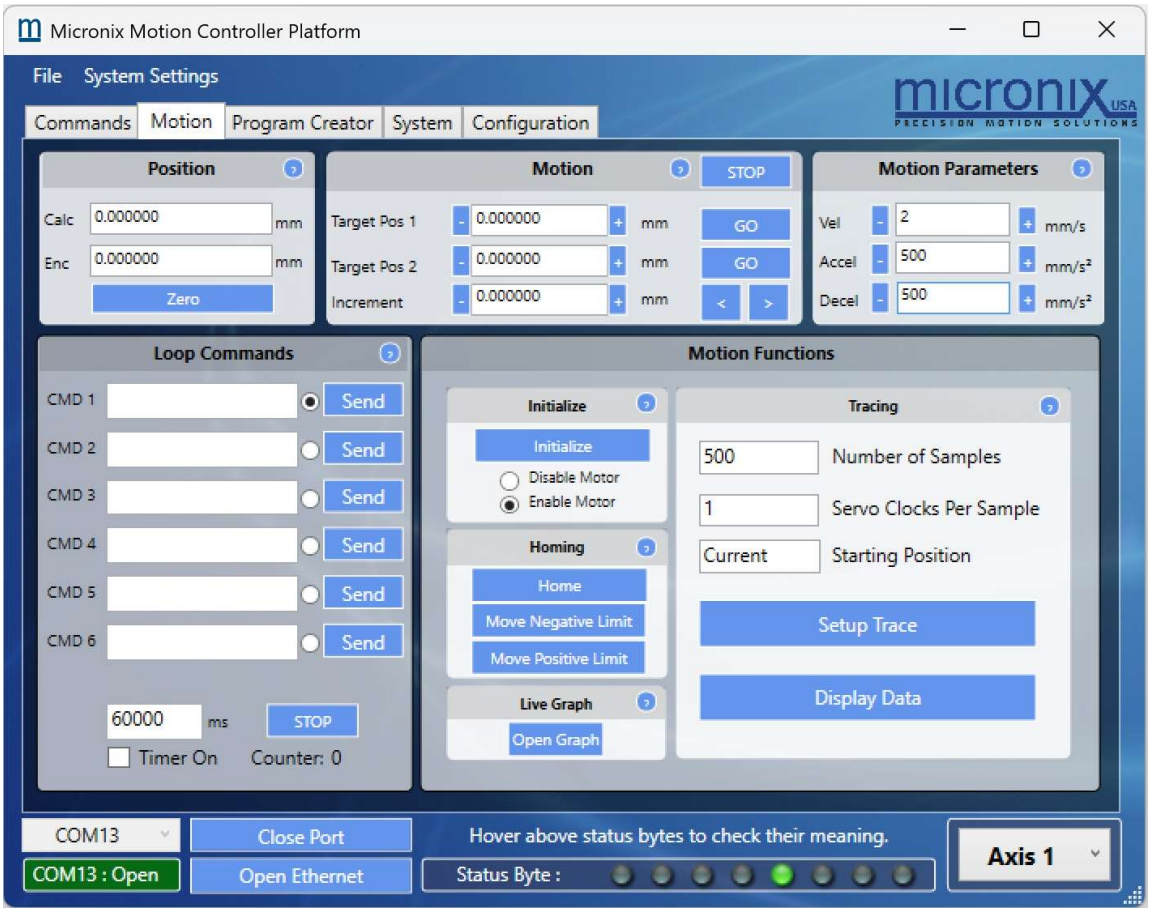


3. The Commands tab allows the user to manually send MMC commands, query NanoDrive parameters, and see command descriptions.

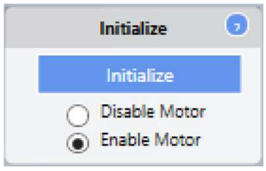




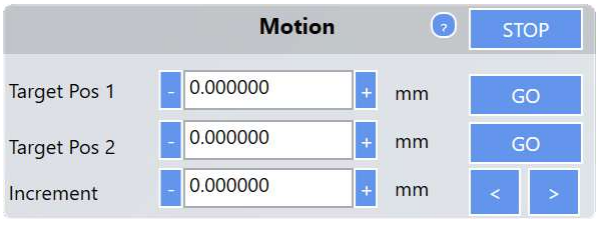
4. The Motion tab can be used to initialize and perform various types of motion.



a. Piezo motors require no initialization routine. They only need to be enabled by using the MOT command or by enabling the motor as shown below.

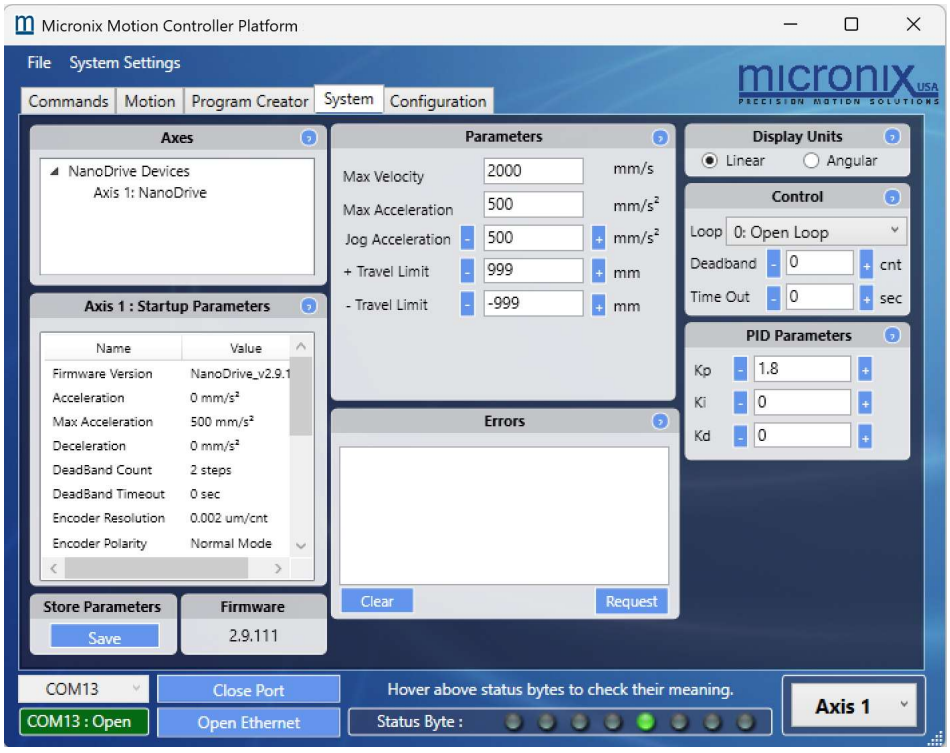


b. The initialization process will vary depending on motor type. See the Appendix 6.5 for the initialization routine of other motor types.
c. Once enabled, the motor can perform any of the position moves or homing routines in the Motion tab.



d. The startup configuration of the NanoDrive will differ depending on the motor type.
   i. Piezo & Stepper Motors – Start in open loop mode as factory default. An encoder is not required for this mode. Use the FBK command to put the NanoDrive in a closed loop configuration. Note that all closed loop configurations will require encoder feedback. These changes can be made in the System tab. Reference section 5.8 for details on the FBK command.
   ii. 3-Phase Brushless & Brushed DC Motors – Start in closed loop mode. Both motors require encoder feedback for proper operation. They cannot operate in open loop mode.

5. The System tab can be used to change the configurations settings of the NanoDrive.



3. Technical Information

3.1 NanoDrive Specifications

Parameter	Description
Axes	1 (stackable up to 16 axes)
Motor Type	2-Phase Stick-Slip Piezo Motor Bipolar Stepper Motor 3-Phase Brushless Motor DC Motor
Encoder Interface	Incremental - A Quad B (RS-422) Absolute - BiSS-C Protocol
Interface	RS-485, CAN Bus, and USB 2.0
Commands	ASCII Commands
Trajectory Mode	Trapezoidal velocity profile
Servo Clock	10 kHz
Trajectory Update	1 kHz
Power Supply	Regulated 12V to 55V DC (5A max operation per axis) USB PD (up to 20VDC/5A)
Enclosure Dimensions	50 x 50 x 28 (mm)
Software Interface	MCP-GUI, LabVIEW VI's

*A single power supply may be used per stack. Each module/axis requires 1A, therefore add up individual module amperages to determine the power supply amperage requirement.

3.2 USB Communication

The required USB drivers for the NanoDrive are automatically built into Windows systems. When connected, the USB will create a virtual COM port to allow for communication with the NanoDrive. Below are the virtual COM port configuration settings necessary for correct communication setup:

Software Parameter	Setting
Data Bits	8
Stop Bits	1
Parity	No
Handshake	No
Baud rate	Any

If the NanoDrive is not automatically recognized by your computer, contact Micronix for support.

3.3 RS-485 Communication

RS-485 can be used to communicate with the NanoDrive. Below are the RS-485 port configuration settings necessary for correct communication setup.

Software Parameter	Setting
Data Bits	8
Stop Bits	1
Parity	No
Handshake	No
Baud rate	115200

4. Operation

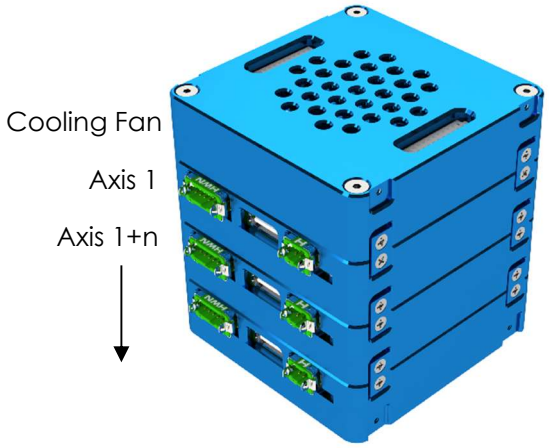
4.1 Axis Addressing

Auto Addressing is the default method of assigning axis numbers on start-up. Controllers are automatically assigned axis numbers on every power up, starting with axis 1 and increasing consecutively until reaching the last axis.

Manual axis numbers may be assigned to a controller using the ANR command. After an ANR assignment, a SAV command should be issued to save the configuration to flash. When manually assigning the axis number, care should be taken to ensure each axis number is unique to avoid communication conflicts. This overrides Auto Addressing, as the controller stores the axis number until reassigned or reset back to Auto Addressing. In the case of having a mix of manually assigned and auto addressed controllers, the Auto Addressed axis numbers increase consecutively after each manually assigned axis in the stack. For example, in a stack of 5 controllers with the third controller manually assigned to axis 10, the axis numbers will read: 1, 2, 10, 11, 12

If two controllers are accidentally assigned the same axis number, use the global command "0ANR0" to reset all controllers back to Auto Addressing.

The figure shown below illustrates axis numbers for a three module stack with Auto Addressing assigned. Axis 1 is located on the top of the stack.



Vertical stack
multiple axis may include a cooling fan which is on the very top, but is not considered as Axis 1

4.2 Motor Control Type

The NanoDrive can control four different types of motors; stick-slip piezo, bi-polar stepper, three phase brushless, and DC motors. The Motor Control Type command (nMTCx) is used to select the type of motor connected to the NanoDrive. See the table below for the value assignment of the MTC command.

Motor Control Type (MTC)	Motor
1	Stick-Slip Piezo
2	Bi-Polar Stepper
3	Three Phase Brushless DC
4	DC

Some commands, settings, and routines are only applicable to certain motor types. The NanoDrive will typically be shipped with pre-configured settings for the connected motor and encoder. Changing the motor control type parameter from the pre-configured value may impact performance. If further configuration is needed, contact Micronix for support.

4.3 Feedback Control

The NanoDrive has three different movement modes of operation. When executing a move command, the controller will drive a stage differently when set to different modes. The FBK command is used to switch between these modes.

The first mode (nFBK0) is a traditional Open Loop. It follows a standard trapezoidal velocity characteristic (acceleration phase, constant velocity phase, and deceleration phase). When operating a piezo, motion transitions between acceleration, constant velocity and deceleration on the resolution settings (nREZx) or the distance it travels in one pulse. See Appendix 6 for the operation of other motor types.

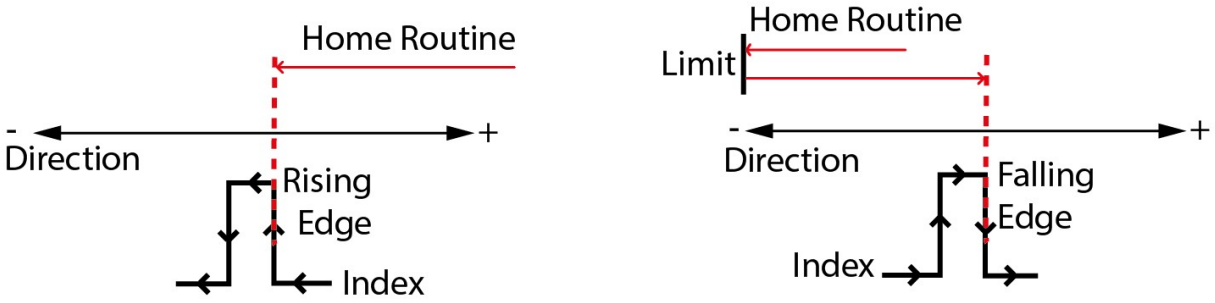
The second mode (nFBK2) is a version of closed loop,, taking position data from an attached encoder and using it to ensure that it stops at the desired position. In this mode, the controller runs in open loop mode until it reaches its deceleration phase. During the deceleration phase, the position constantly reads from the encoder and corrects its position to arrive at the correct target position. This guarantees the target position within the specified deadband range which can be adjusted using the Deadband command (DBD). However, this mode does not operate in closed loop during motion. Feedback mode 2 is only available for piezo and stepper motor operation and not available for brushless motor and DC motor.

The third mode (nFBK3) is a traditional closed loop. The controller will constantly try to match the real trajectory to the commanded trapezoidal velocity characteristic. Like the previous mode (nFBK2), traditional closed loop uses positional data from an attached encoder and guarantees final position within the specified deadband. The closed loop algorithm in the third feedback mode will utilize a PID control loop (PID command) along with a feedforward parameter (FFP command). See Appendix section 6.6 for more details on the closed loop operation.

4.4 HOM, MLN, and MLP

The HOM command requires that the attached stage has an encoder with an Index signal. Most incremental encoders have an index signal while absolute encoders do not have an index signal.

The NanoDrive has improved the homing routine to simplify the required motion. The HOM command will initially move in the direction dictated by the Home Configuration command (HCG) in search of the index location of the encoder. If the stage is moving towards the index, it will move until it reaches the index and then latch the position at the rising edge of the index pulse. If the stage is moving away from index, it will move until it reaches a hard limit, the maximum travel, or a limit switch (configured through the LCG command). It then reverses direction and proceeds until it reaches the falling edge of the index pulse and latch the position.



The MLN and MLP commands requires either an encoder or limit switches and moves in the positive or negative direction in search of the hard stop, soft limit, or limit switch locations. The LCG command can be used to determine which type of limit condition is used to signal the end of travel. Once the hard stop location is found, motion will reverse direction and move the distance defined by the limit rebound (LRB command).

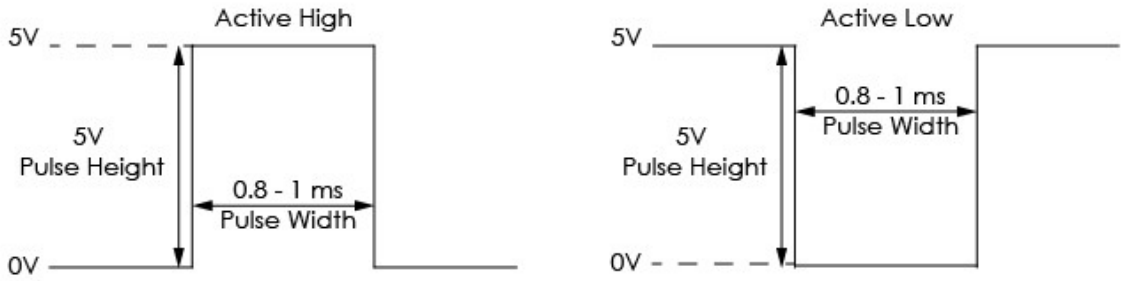
4.5 I/O Commands

The NanoDrive has I/O that can be used to output pulse signals when the stage has completed certain tasks. The I/O are located on the 12-pin Gecko connector, and its pinout is as described in section 6.2 of the Appendix.

Each I/O pin on the NanoDrive can be assigned as either an input or output using the I/O direction command (IOD). The available I/O functions will only operate on I/O pins with the required direction. For example, function 1 of IOF (pg. 5-60) is available for defined inputs and functions 2 to 6 are available for defined outputs

The I/O Function (IOF) command (pg. 5-60) assigns functions to the corresponding I/O pins. The I/O function will either output a signal via the I/O pin when certain conditions are met or use an incoming signal from an I/O pin to perform an operation.

The I/O Polarity (IOP) command (pg. 5-61) is used to change the polarity of the pulse output of an I/O pin. The I/O polarity sets active high or active low pulse output for the following commands (CVP, PIP, and PTP).



The I/O Status (IOS) command (pg. 5-62) can be used to manually read and write I/O pins. Note that the I/O pin can only be set high or low on defined outputs. The I/O polarity setting (IOP) will also affect the read and write operation of the I/O status.

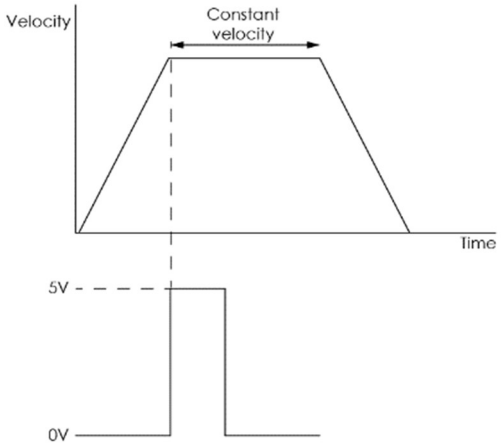
CVP – Pulse at Constant Velocity

The CVP(Pulse at Constant Velocity) command sends a pulse trigger once the stage reaches constant velocity. All I/O pins assigned to the I/O function (IOF) #4 will output a pulse at constant velocity. The CVP command requires an enable (1)/disable (0) parameter.

The CVP operation is available on any defined output.

An example of setting up an I/O pin to output at constant velocity is as follows:

```
1IOD1,0     //IO1 set to an output
1IOP1,1     //IO1 pulse trigger as active high
1IOF1,4     //IO1 pulse trigger with CVP
1CVP1       //CVP is enabled.
```



PTP – Pulse at Target Position

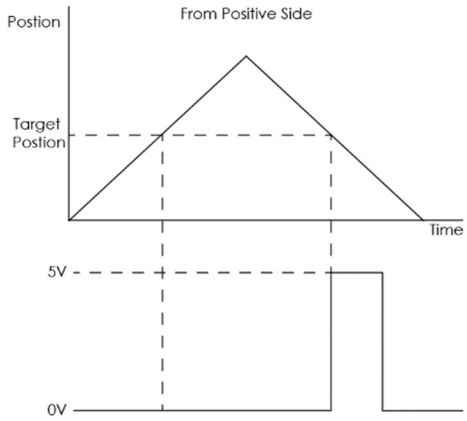
The PTP operation will send an output pulse to the assigned I/O pins once the NanoDrive has reached the specified target position. Note that the PTP function will require encoder feedback for the target position. All I/O pins assigned to the I/O Function (IOF) #5 will output a pulse.

The PTP command accepts two parameters. The first parameter defines the target position and the second parameter determines the required direction of motion to trigger an output pulse. Depending on the direction chosen for the PTP command, the pulse trigger may activate when moving negative (0), positive (1), or from either direction (2).

The PTP function is only available on I/O defined as outputs.

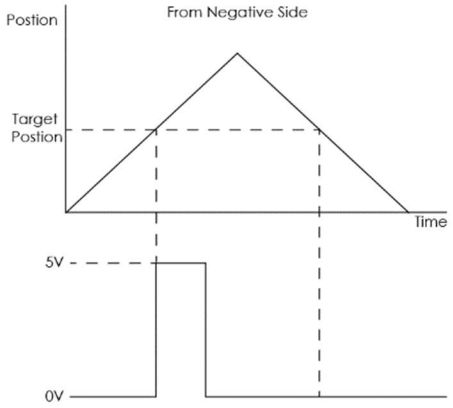
Example 1: Output pulse to IO #4 at target position 3.0mm in the negative direction

```
1IOD4,0     //Define IO4 as an output
1IOF4,5     //Assign PTP to IO4
1PTP4,0     //Pulse trigger (-) motion
```



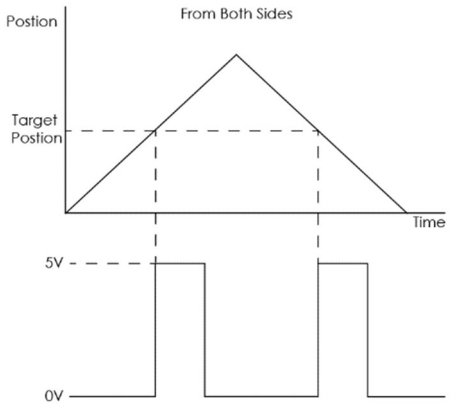
Example 2: Output pulse to IO #5 at target position 3.0mm in the positive direction

```
2IOD3,0     //Define IO3 as an output
2IOF3,5     //Assign PTP to IO3
2PTP3,1     //pulse trigger (+) motion
```



Example 3: Output pulse to IO #1 at target position 3.0mm in either direction.

```
3IOD1,0     //Define IO1 as an output
3IOF1,5     //Assign PTP to IO1
3PTP3,2     //pulse trigger from either direction
```



PIP – Pulse in Regular Intervals

The position interval pulse (PIP command) function will send pulses at a defined position interval. Note that the PIP function will require encoder feedback for the position interval. All I/O pins assigned to the I/O function (IOF) #6 will output a pulse during the defined position intervals.

The syntax for PIP is as follows: *PIPx,i,y* where "x" is the starting position, "i" is the interval for each pulse trigger, and "y" is the ending position.

The interval, "i", can be either positive or negative interval depending on the difference between the ending position and starting position.

If y – x > 0 , then interval i has to be positive.
If y – x < 0 , then interval i has to be negative.

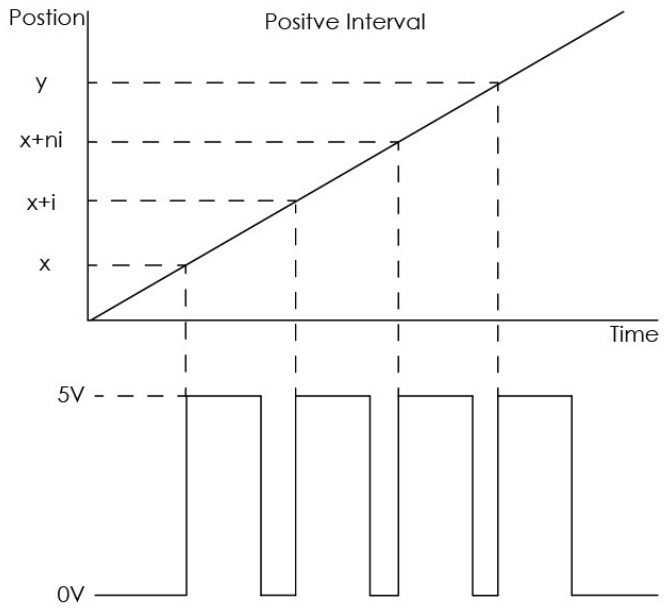
The total number of pulses expected is calculated as

Total pulses = int[(y-x) / i] + 1 if y-x ≥ i
int[(y-x) / i] + 2 if y-x < i

The PTP operation is only available on I/O pins defined as outputs.

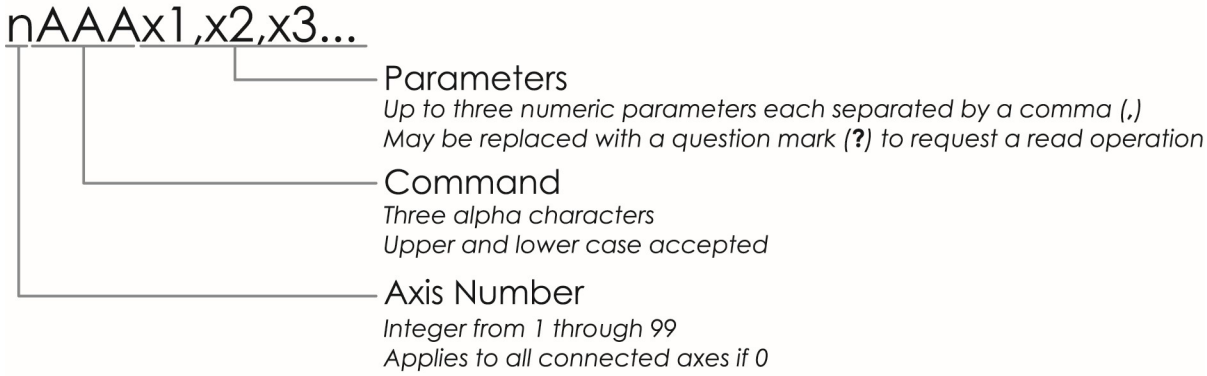
An example of setting up an IO pin to output pulses with PIP is as follows:

```
1IOD4,0     //Define IO4 as an output
1IOP4,1     //IO4 pulse trigger as active high
1IOF4,6     //IO4 pulse trigger with PIP
1PIP1,1,3   //Pulse triggers at 1mm, 2mm, and 3mm.
```



5. Commands

5.1 Command Line Syntax



There are three components to every command. The first is the ***Axis Number*** which designates which controller or axis will receive the command. If the ***Axis Number*** is 0, then the command will be sent globally to all connected controllers. It is possible to connect up to 16 controllers; therefore, the ***Axis Number*** will be an integer value from 0 through 16.

The second component is the ***Command***, which is always comprised of three letters. Each command is outlined, along with its corresponding parameters, in section 5.8, Command Descriptions.

The third and final component is the ***Parameter***. This portion is command-dependent, meaning that the parameter value will change depending on the specific requirements of the ***Command***. Where applicable, a question mark (?) may be substituted to initiate a read operation which will return information regarding the particular command. There may be up to three separate parameters for a particular command, each of which is separated by a comma (,).

All white spaces (blank spaces) are ignored in the command format. The following are examples of equivalent commands:

```
1VEL2
1 VEL 2
```

When communicating with the controller, it is necessary to note the terminating characters involved in transmitting and receiving data. To send data to the controller, enter the desired commands in the command line followed by the new line and carriage return terminating characters [\n\r], or just the carriage return terminating character [\r]. When receiving, each line of data will be followed by the new line terminating character [\n] and the final line will end in the new line and carriage return terminating characters [\n\r]. The ASCII value for new line [\n] is 0X0A and for carriage return [\r] is 0X0D. The following is an example of data transmission:

```
1VEL0.005 \n\r      | Axis 1, Set velocity to .005 mm/s [degrees/s] [New line, Carriage Return]
```

5.2 Command Line Format

Commands are first executed in the order of input, then line-by-line. This means that two commands on the same line are executed significantly closer to each other than if they were on two separate lines. Each command is separated by a semicolon (;) and every command line ends in a terminator (i.e. carriage return). The following is an example of a command line entry:

```
1MVR16;3MVR12        | Axis 1, Move 16 mm [16 degrees]; Axis 3, Move 12 mm [12 degrees]
```

Using multiple commands on the same command line allows for closer synchronization of different commands to different axes.

5.3 Global Commands

Some commands have the option of being executed globally. This means that you can send the same command to all available axes. To do this, replace the axis number of a global command with a '0'. For example, 0ACC 50 will set the acceleration of all available axes to 50 mm/s^2 [degrees/s^2].

5.4 Multiple Parameters

When dealing with a command that has multiple parameters, it is possible to change a single parameter by omitting numbers for the parameters that will remain unchanged. For example, `4DBD,0.3` will only change the second parameter to a new value, "0.3".

5.5 Synchronous Move

It is possible to execute multiple moves at the same time by setting up and executing a synchronous move. To set up a synchronous move, use the MSA and MSR commands. These commands can be written on the same command line (up to 8 allowed) or on separate lines followed by a line terminator. To execute the move, use the RUN command on the proceeding command line followed by a line terminator. For example,

```
1MSA4;2MSA4;3MSA4      | Axis 1, Move 4mm; Axis 2, Move 4mm; Axis 3 Move 4mm
0RUN                   | Run Synchronous Move
```

Or

```
1MSA4                  | Axis 1, Move 4mm
2MSA2                  | Axis 2, Move 2mm
3MSA3                  | Axis 3 Move 3mm
0RUN                   | Run Synchronous Move
```

5.6 Internal Programming

An internally stored program may be used to save time when repeatedly using a sequence of commands. Each controller or axis must be programmed individually; however, multiple controllers may execute the same program at the same time.

To record a program sequence, enter the PGM command on a unique line followed by a line terminator. End a program sequence by entering the END command on a unique line followed by a line terminator. When you want to execute this program, use the EXC command. See the *Summary of Commands* page for a list of program compatible commands and more information about the PGM, END and EXC commands.

Commands intended to be stored in a program must be preceded with the '%' character, or else they will be executed immediately.

Program Examples:

```
1pgm1
%1mvr2
%1vel1
1mvr2       //This will happen immediately and will not be part of the program.
%1end
```

A command stored into a program with a hard-coded axis (ex. '1mvr5'), will no longer execute after a change in the axis number (ANR command). Programs can contain a special 'me' character ('*'), this allows programs to function properly if the address for a controller is changed.

```
1pgm2
%*mvr2
%*vel1
%*end
```

Existing program numbers cannot be overwritten unless previously erased using the ERA command. nPGM? will return a binary representation of which program slots are already programmed. nLST1 will return a list of the commands that are written in program 1.

5.7 Summary of Commands

Command	Description	During Motion		Idle		Program Mode		Global	Page
		Set	Read	Set	Read	Set	Read	Set	
ACC	Acceleration		✓	✓	✓	✓	✓	✓	27
AMX	Maximum Allowable Acceleration		✓	✓	✓		✓	✓	28
ANR	Set Axis Number		✓	✓	✓		✓	✓	29
CER	Clear Errors	✓		✓		✓		✓	30
CVG	Current Velocity Gain		✓	✓	✓			✓	31
CVL	Correction Velocity		✓	✓	✓			✓	32
CVP	Constant Velocity Pusle	✓	✓	✓	✓	✓	✓	✓	33
DAT	Dump Trace Data				✓				34
DBD	Closed Loop Deadband		✓	✓	✓		✓	✓	35
DEC	Deceleration		✓	✓	✓	✓	✓	✓	36
DEF	Restore Factory Defaults			✓				✓	37
EAD	Set Analog or Digital Encoder		✓	✓	✓		✓	✓	38
EFF	Encoder Filter								39
EMF	Back EMF Constant		✓	✓	✓		✓	✓	40
ENC	Select Encoder Resolution		✓	✓	✓		✓	✓	41
END	End Program Recording			✓		✓			42
EPL	Encoder Polarity		✓	✓	✓		✓	✓	43
ERA	Erase Program	✓		✓				✓	44
ERR	Read and Clear Errors		✓		✓				45
EST	Emergency Stop	✓		✓		✓		✓	46
EXC	Execute Program			✓				✓	47
FBK	Set Open or Closed Loop Mode		✓	✓	✓		✓		48
FFP	Feed Forward Parameter		✓	✓	✓	✓	✓	✓	49
FSR	Full Steps Per Revolution		✓	✓	✓	✓	✓	✓	50
HAC	Home Acceleration								51
HCG	Home Configuration		✓	✓	✓	✓	✓	✓	52
HOM	Home		✓	✓	✓	✓	✓	✓	53
HST	Hard Stop Detection		✓	✓	✓	✓	✓	✓	54
HVL	Home Velocity		✓	✓	✓	✓	✓	✓	55
IDN	Identification Name	✓	✓	✓	✓	✓	✓	✓	56
INI	Initialize BLDC Phasing			✓		✓		✓	57
INP	In Position		✓	✓	✓			✓	58
IOD	I/O Direction	✓	✓	✓	✓	✓	✓	✓	59
IOF	I/O Function	✓	✓	✓	✓	✓	✓	✓	60
IOP	I/O Polarity	✓	✓	✓	✓	✓	✓	✓	61
IOS	I/O Status	✓	✓	✓	✓	✓	✓	✓	62
IWL	Integrator Windup Limit		✓	✓	✓	✓	✓	✓	63
JAC	Jog Acceleration and Deceleration		✓	✓	✓	✓	✓	✓	64
JOG	Jog Mode	✓		✓	✓	✓		✓	65

Continued...

Command	Description	During Motion		Idle		Program Mode		Global	Page
		Set	Read	Set	Read	Set	Read	Set	
LCG	Limit Configuration		✓	✓	✓	✓	✓	✓	66
LDP	Load Parameters			✓				✓	67
LDR	Limit Switch Direction		✓	✓	✓	✓	✓	✓	68
LPF	Low Pass Filter	✓	✓	✓	✓	✓	✓	✓	69
LPL	Limit Switch Polarity		✓	✓	✓	✓	✓	✓	70
LRB	Limit Rebound	✓	✓	✓	✓	✓	✓	✓	71
LSP	Lead Screw Pitch		✓	✓	✓	✓	✓	✓	72
LST	Program List				✓				73
MCM	Motor Current Maximum		✓	✓	✓	✓	✓	✓	74
MCR	Motor Current Reduction		✓	✓	✓	✓	✓	✓	75
MCS	Motor Current Setting		✓	✓	✓	✓	✓	✓	76
MFE	Maximum Following Error		✓	✓	✓	✓	✓	✓	77
MLN	Move to Negative Limit			✓		✓		✓	78
MLP	Move to Positive Limit			✓		✓		✓	79
MOT	Toggle Motor On/Off		✓	✓	✓	✓	✓	✓	80
MPL	Motor Polarity		✓	✓	✓	✓	✓	✓	81
MSA	Synchronous Move – Absolute	✓		✓		✓		✓	82
MSR	Synchronous Move – Relative	✓		✓		✓		✓	83
MTC	Motor Control Method		✓	✓	✓	✓	✓	✓	84
MTP	Motor Pitch		✓	✓	✓	✓	✓	✓	85
MTV	Motor Voltage		✓	✓	✓	✓	✓	✓	86
MVA	Move Absolute	✓		✓		✓		✓	87
MVR	Move Relative	✓		✓		✓		✓	88
PGL	Loop Program		✓	✓	✓	✓	✓	✓	89
PGM	Begin Program Recording		✓	✓	✓		✓		90
PGS	Run Program At Start-Up		✓	✓	✓		✓	✓	91
PID	Set Feedback Constants		✓	✓	✓	✓	✓	✓	92
PIP	Position Interval Pulse		✓	✓	✓	✓		✓	93
POS	Position		✓		✓		✓		94
PTP	Pulse Target Position		✓	✓	✓	✓		✓	95
PWM	PWM Operating Frequency		✓	✓	✓	✓	✓	✓	96
REZ	Set Resolution		✓	✓	✓	✓	✓	✓	97
RUN	Start Synchronous Move			✓		✓		✓	98
SAV	Save Axis Settings			✓				✓	99
SCM	Stepper Control Method		✓	✓	✓	✓	✓	✓	100
STA	Status Byte		✓		✓		✓		101
STP	Stop Motion	✓		✓		✓		✓	102
SVP	Save Startup Position		✓	✓	✓	✓	✓	✓	103
SYN	Sync					✓		✓	104
TLN	Negative Soft Limit Position		✓	✓	✓	✓	✓	✓	105

Continued...

Command	Description	During Motion		Idle		Program Mode		Global	Page
		Set	Read	Set	Read	Set	Read	Set	
TLP	Positive Soft Limit Position		✓	✓	✓	✓	✓	✓	106
TRA	Perform Trace		✓	✓	✓			✓	107
VEL	Velocity	✓	✓	✓	✓	✓	✓	✓	108
VER	Firmware Version		✓		✓		✓		109
VMX	Max. Allowable Velocity		✓	✓	✓	✓	✓	✓	110
VRT	Encoder Velocity		✓		✓		✓		111
WST	Wait For Stop					✓			112
WSY	Wait For Sync					✓			113
WTM	Wait For Time Period					✓			114
ZRO	Zero Position			✓		✓		✓	115

5.8 Command Descriptions

ACC

Acceleration

During Motion		Idle		Program Mode		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired acceleration for the specified axis, distinct from the deceleration [DEC]. The acceleration value must be less than the maximum allowable acceleration [AMX] for the command to be accepted. The SAV command can be used to save the acceleration value to flash.
Returns:	A read operation returns the acceleration value in mm/s^2 for the specified axis.
Syntax:	nACCx – Standard syntax nACC? – Read acceleration value 0ACCx –All axes set acceleration value Error [#]: ACC? – Read operation with missing axis number [27] nACC – Missing acceleration parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Acceleration [mm/s^2 or degrees/s^2] ? – Read acceleration value
Parameter Range:	n – 0 to 99 x – 000.001 to AMX [mm/s^2 or degrees/s^2]
Related Commands:	DEC, VEL, JAC, AMX, HAC, SAV
Example:	`3ACC100` | Axis 3, Set acceleration to 100 [mm/s^2 or degrees/s^2] – `4ACC?` | Axis 4, Read acceleration value

AMX — Maximum Allowable Acceleration

During Motion		Idle		Program Mode		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to set the maximum allowable acceleration for the specified axis.
Returns:	A read operation returns the maximum allowable acceleration value in mm/s^2 or degrees/s^2 for the specified axis.
Syntax:	nAMXx – Standard syntax nAMX? – Read maximum allowable acceleration value 0AMXx – All axes set maximum allowable acceleration value Error [#]: AMX? – Read operation with missing axis number [27] nAMX – Missing maximum acceleration parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Maximum acceleration ? – Read maximum allowable acceleration value
Parameter Range:	n – 0 to 99 x – 000.001 to 20000.000 mm/s^2 [or degrees/s^2]
Related Commands:	DEC, VEL, JAC, VMX, ACC, SAV
Example:	`2AMX500` | Axis 2, Set max acceleration to 500 mm/s^2 [or degrees/s^2] – `6AMX?` | Axis 6, Read max acceleration value

ANR

Set Axis Number

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓*

Command Description:	This command is used to override Auto Addressing by manually assigning an axis number to a controller. Auto Addressing is the default method of assigning axis numbers on power up and may be reassigned to an axis by substituting a "0" for the parameter value. Simultaneous axis swapping is possible by using multiple ANR commands on the same command line. The SAV command can be used to save the axis number setting to flash. *This command can be called globally by specifying a '0' for the axis number; however, it will only work if the new axis number parameter is set to '0' for auto-addressing.
Returns:	A read operation returns the following axis number values for the specified axis: 0 – Auto Addressing assigned (default) 1-99 – Manually assigned, axis number displayed
Syntax:	nANRx – Standard syntax nANR? – Read axis number value Error [#]: ANR? – Read operation with missing axis number [27] nANR – Missing new axis number parameter [28] ANRx – Missing axis number [30]
Parameter Description:	n[int] – Axis number x[int] – New axis number, 0 for Auto Addressing ? – Read axis number value
Parameter Range:	n – 0 to 99 x – 0 to 99
Related Commands:	SAV
Example:	`5ANR1;1ANR5` | Simultaneous axis swapping: Axis 5, Set to axis Axis 1, Set to axis 5 – `4ANR0` | Axis 4 Set to Auto Addressing. However, it will remain axis 4 until reset

CER

Clear Errors

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to clear all error messages without reading them. Sending this command will set Bit 7 of the STA command to 0 and clear the Error LED indicator.
Returns:	A read operation cannot be used with this command.
Syntax:	nCER – Standard syntax 0CER – All axes clear error messages
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 0 to 99
Related Commands:	ERR, STA
Example:	`1CER` | Axis 1, clear error messages – `0CER` | All axes, clear error messages

CVG Current Velocity Gain

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓			✓

Command Description:	This command is used to change the current velocity gain setting which can increase the target current of the motor as the velocity increases. This setting is only applicable for stepper motor operation. The SAV command can be used to save the current velocity gain setting to flash.
Returns:	The value for the current velocity gain
Syntax:	nCVG – Standard syntax Error [#]: CVGx – Missing axis number [30]
Parameter Description:	n[int] – Axis number x[float] – Current velocity gain ? – Read current velocity gain
Parameter Range:	n – 1 to 99 x – 0.001 to 20000.000
Related Commands:	MTC, SCM, SAV
Example:	`2MTC2` | Axis 2, Set the motor type to stepper operation `2CVG3` | Axis 2, Set current velocity gain to 3

CVL Correction Velocity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓			✓

Command Description:	This command is used to set the correction velocity. This term is only used FBK modes 2 and 3. If the trajectory is outside of the deadband range, then the trajectory will switch the commanded velocity [VEL] to the correction velocity [CVL]. For FBK2, correction velocity will be used when the encoder position is outside of the deadband range of the target/commanded position. Correction velocity will not be applied during calculated trajectory. For FBK3, correction velocity is used when the encoder position is outside of the deadband range at the command position and during the calculated trajectory. Generally, CVL should be set higher than VEL.
Returns:	The value for the correction velocity [mm/s or degrees/s]
Syntax:	nCVL – Standard syntax Error [#]: CVLx – Missing axis number [30]
Parameter Description:	n[int] – Axis number x[float] – Correction velocity ? – Read correction velocity
Parameter Range:	n – 1 to 99 x – 000.001 to VMX [mm/s or degrees/s]
Related Commands:	FBK
Example:	`2CVL10` | Axis 2, Set correction velocity to 10mm/s

CVP

Pulse at Constant Velocity

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
	✓	✓	✓			✓	

Command Description:	This command is used in conjunction with IOD, IOF, and IOP to command the controller to emit a pulse once the constant velocity state of motion is reached. CVP operation is available on any I/O defines as outputs
Returns:	A read operation returns the values assigned to the enable value.
Syntax:	nCVPx – Standard syntax 0CVPx – All axes execute enable value nCVP? – Read enable value Error [#]: 1CVP2 Invalid Input Parameter (enable must be a "1" or a "0").
Parameter Description:	n[int] – Axis number x[int] – Enable value
Parameter Range:	n – 0 to 99 x – 0 CVP Disabled 1 CVP Enabled
Related Commands:	IOD, IOF, IOP
Example:	`4IOP1,1` | Axis 4, Set IO Pin 1's pulse as active high `4IOF1,4` | Axis 4, Set IO Pin 1 for CVP function `4CVP1` | Axis 4, Pulse trigger at IO pins with CVP when constant velocity is reached.

DAT

Dump Trace Data

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
			✓			

Command Description:	This command is used to read trace data from a specified axis initially recorded by the trace command [TRA]. The retrieved trace data set is dumped from the controller, consequently allowing the data to be retrieved only once.
Returns:	A read operation returns the trace data values for the specified axis in the following format: [Theoretical Position (.5nm per count)],[Actual Position(.5nm per count)], [DAC Value], [Not Used]
Syntax:	nDAT? – Read trace data values Error [#]: DAT? – Read operation with missing axis number [27] nDAT – Missing read operation parameter [28]
Parameter Description:	n[int] – Axis number ? – Read trace data values
Parameter Range:	n – 1 to 99
Related Commands:	TRA
Example:	`1TRA100,1,` | Axis 1, Set trace to record 100 data values total. 1 set of data values every 1 servo cycle `1MVR2` | Axis 1, Move to start trace recording … | Wait until motion is completed `1TRA?` | Axis 1, Check number of samples ready to output `//#100` | Axis 1, returns 100 out of 100 trace data ready to output `1DAT?` | Axis 1, Read trace data values

DBD

Closed Loop Deadband

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to set the acceptable deadband and deadband timeout values. Deadband refers to the number of encoder counts (±) from the target that is considered acceptable. If the parameter (x1) is set to "0", the controller will continuously oscillate around the target. Deadband timeout refers to the amount of time that the controller will try to move into the deadband area. If the parameter (x2) is set to "0", the controller will seek continuously. The SAV command can be used to save the deadband parameters to flash.
Returns:	A read operation returns the deadband and deadband timeout values for the specified axis.
Syntax:	nDBDx1,x2 – Standard syntax nDBD? – Read deadband and deadband timeout values 0DBDx1,x2 – All axes set deadband and deadband timeout values Error [#]: DBD? – Read operation with missing axis number [27] nDBD – Missing deadband and deadband timeout parameter values [28]
Parameter Description:	n[int] – Axis number x1[int] – Deadband x2[float] – Deadband timeout ? – Read deadband and deadband timeout values
Parameter Range:	n – 0 to 99 x1 – Encoder dependent, 0 for continuous, Encoder Counts x2 – Encoder dependent, 0 for infinite, Seconds (default 0)
Related Commands:	ENC, EPL, FBK, INP, SAV
Example:	`1DBD10,1` | Axis 1, Set deadband to 10 encoder counts & deadband timeout to 1 second – `4DBD5,0` | Axis 4, Set deadband to 5 encoder counts & deadband timeout to infinite

DEC Deceleration

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired deceleration for the specified axis, distinct from the acceleration [ACC]. The deceleration value must be less than the maximum acceleration value [AMX] for the command to be accepted. The SAV command can be used to save the deceleration setting to flash.
Returns:	A read operation returns the deceleration value in mm/s^2 or degrees/s^2 for the specified axis.
Syntax:	nDECx – Standard syntax nDEC? – Read deceleration value 0DECn – All axes set deceleration value Error [#]: DEC? – Read operation with missing axis number [27] nDEC – Missing deceleration parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Deceleration ? – Read deceleration value
Parameter Range:	n – 0 to 99 x – 0.001 to AMX
Related Commands:	ACC, AMX, VEL, SAV
Example:	`2DEC100` | Axis 2, Set deceleration to 100 mm/s^2 [or degrees/s^2] – `7DEC?` | Axis 7, Read deceleration value

DEF

Restore Factory Defaults

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓				✓

Command Description:	This command restores the factory default parameters.
Returns:	A read operation is not available with this command.
Syntax:	nDEF – Standard syntax Error [#]: DEF – Missing axis number [30]
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 1 to 99
Related Commands:	SAV
Example:	`1DEF` | Axis 1, Set default parameters]

EAD

Set Encoder Type

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to specify whether the encoder signal for a specified axis is analog or digital. The SAV command can be used to save the encoder type setting to flash.
Returns:	A read operation returns the following encoder mode values for the specified axis: 0 – Digital 1 – Analog 2 – Absolute
Syntax:	nEADx – Standard syntax nEAD? – Read encoder mode value 0EADx – All axes set encoder value Error [#]: xEAD – Missing encoder mode parameter [28] EAD? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number x[int] – Encoder mode ? – Read encoder mode value
Parameter Range:	n – 0 to 99 x – 0 for digital incremental, 1 for analog incremental, 2 for absolute (BiSS-C)
Related Commands:	ENC, SAV
Example:	`9EAD0` | Axis 9, Set encoder to digital input

EFF

Set Encoder Filter

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to set the available encoder filter on the NanoDrive. The filter operates as a running average, helping reduce the encoder noise through software. The size of encoder samples used in the running average can be set using this command. The SAV command can be used to save the encoder filter settings to flash.
Returns:	A read operation returns the size of the encoder filter when idle and when moving.
Syntax:	nEFFx1,x2 – Standard syntax nEFF? – Read encoder filter values 0EFFx1,x2 – All axes set encoder filter values Error [#]: xEFF – Missing EFF parameters [28] EFF? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number x1[int] – Encoder filter size during motion x2[int] – Encoder filter size when idle ? – Read back EMF constant
Parameter Range:	n – 0 to 99 x1 – 1 to 100 [samples] x2 – 1 to 100 [samples]
Related Commands:	EAD, ENC, SAV
Example:	`9EFF1,10` | Axis 9, Set filter size to 1 when moving and to 10 when idle

EMF

Set Back EMF Constant

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to set the back EMF constant of the stepper motor. The command is not applicable with other motor types. The SAV command can be used to save the back EMF setting to flash.
Returns:	A read operation returns the back EMF constant in V/m/s for the specified axis.
Syntax:	nEMFx – Standard syntax nEMF? – Read back EMF value 0EMFx – All axes set back EMF value Error [#]: xEMF – Missing back EMF parameter [28] EMF? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number x[int] – Back EMF constant ? – Read back EMF constant
Parameter Range:	n – 0 to 99 x – 0.001V to 1000.0 V
Related Commands:	FSR, MCS, MTV, UST, EMF
Example:	`9EMF10` | Axis 9, Set back EMF to 10V/m/s

ENC

Set Encoder Resolution

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to set the desired encoder resolution for the specified axis. When a digital encoder is connected, encoder resolution is determined by the encoder itself and the ENC setting will need to reflect this value. The SAV command can be used to save the encoder resolution settings to flash.
Returns:	A read operation returns the encoder resolution value for the specified axis.
Syntax:	nENCx – Standard syntax nENC? – Read encoder resolution value 0ENCx – All axes execute encoder resolution value Error [#]: ENC? – Read operation with missing axis number [27] nENC – Missing encoder resolution parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Encoder resolution ? – Read encoder resolution value
Parameter Range:	n – 0 to 99 x – 0.0000001 to 9.999999 µm/count (milli-degrees/count)
Related Commands:	EAD, EPL, SAV
Example:	`2ENC10` | Axis 2, Set encoder resolution to 10 microns/count (10 milli- degrees/count)

END

End Program Recording

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓		✓		

Command Description:	This command is used to exit out of program recording mode, which is initiated by the PGM command. The END command must be placed separately on the last line of the program sequence and preceded by the % sign to be considered a part of the current program. The resulting program is saved upon exit for later use.
Returns:	A read operation is not available with this command.
Syntax:	%nEND – Standard syntax Error [#]: END – Missing axis number [30] 1END – Missing "%", will not be stored in program.
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 1 to 99
Related Commands:	EXC, PGM
Example:	`1PGM` | Axis 1, Begin program recording `%1VEL1;1ACC.5` | Axis 1, Set velocity value to 1 mm/s; Axis 1, Set acceleration value to 0.5 mm/s^2 [degrees/s^2] `%1END` | Axis 1, End program recording

EPL

Encoder Polarity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to switch the encoder signal polarity for the specified axis. If the controller doesn't seem to be recording encoder position correctly, the polarity of the encoder signals could be reversed. Use this command to switch from the default setting (normal operation, n=0). The SAV command can be used to save the encoder polarity setting to flash.
Returns:	A read operation returns the following encoder polarity values for the specified axis: 0 – Normal operation 1 – Reverse operation
Syntax:	nEPLx – Standard syntax nEPL? – Read encoder polarity value 0EPLx – All axes execute encoder polarity value Error [#]: EPL? – Read operation with missing axis number [27] nEPL – Missing encoder polarity parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Encoder polarity ? – Read encoder polarity value
Parameter Range:	n – 0 to 99 x – 0 for normal operation, 1 for reverse operation
Related Commands:	DBD
Example:	`13EPL0` | Axis 13, Set encoder polarity to normal operation `6EPL1` | Axis 6, Set encoder polarity to reverse operation

ERA

Erase Program

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓				✓

Command Description:	This command is used to erase a specified program from an axis. Before recording a program, use the LST command to see which program numbers are available for that axis. There are 16 program numbers available allowing up to 16 programs to be stored. An existing program cannot be overwritten and must be erased first. Therefore, use this command to erase the specified program and make space for a new one.
Returns:	A read operation is not available with this command.
Syntax:	nERAx – Standard syntax Error [#]: ERAx – Missing axis number [30] nERA – Missing program number parameter [28]
Parameter Description:	n[int] – Axis number x[int] – Program number to be erased
Parameter Range:	n – 1 to 99 x – 1 to 16
Related Commands:	LST
Example:	`5ERA4` | Axis 5, Erase program 4

ERR

Read and Clear Errors

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓		✓			

Command Description:	This command is used to read and clear any pending error messages. If no errors are present on the NanoDrive, it will respond with a "#No Error" message
Returns:	A read operation returns a list of error messages for the specified axis in the following format. "AAA" signifies the specific command name that the error corresponds to. Error Number – Description [AAA]
Syntax:	nERR? – Standard syntax Error [#]: ERR? – Read operation with missing axis number [123]
Parameter Description:	n[int] – Axis number ? – Read error messages
Parameter Range:	n – 1 to 99
Related Commands:	None
Example:	`3ERR?` | Axis 3, Read error messages

EST

Emergency Stop

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to stop a specific axis or all connected axes simultaneously in case of an emergency. The controller executes the largest possible deceleration.
Returns:	A read operation is not available with this command.
Syntax:	nEST – Standard syntax 0EST – All axes execute emergency stop
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 0 to 99
Related Commands:	STP
Example:	`8EST` | Axis 8, Emergency stop – `0EST` | All axes, Emergency stop

EXC

Execute Program

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓				✓

Command Description:	This command is used to execute a specified program for one or multiple axes. If executing a program globally, all connected axes should have individual programs stored under the specified program number prior to execution.
Returns:	A read operation is not available with this command.
Syntax:	nEXCx – Standard syntax 0EXCx – All axes execute program Error [#]: nEXC – Missing program number parameter [123]
Parameter Description:	n[int] – Axis number x[float] – Program number to be executed
Parameter Range:	n – 0 to 99 x – 1 to 32
Related Commands:	PGM
Example:	`4EXC5` | Axis 4, Execute program 5 – `0EXC2` | All axes, Execute program 2

FBK

Set Open or Closed Loop Mode

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	

Command Description:	This command is used to select the feedback mode of the controller. See section 4.2 for more details. The SAV command can be used to save the feedback mode setting to flash.
Returns:	A read operation returns the following loop mode values for the specified axis: 0 – Open Loop [default] 2 – Open Loop during Trajectory, Closed Loop when In Position 3 – Closed Loop
Syntax:	nFBKx – Standard syntax nFBK? – Read encoder mode value Error [#]: FBKx – Missing axis number [30] FBK? – Read operation with missing axis number [27] nFBK – Missing closed/open loop parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Open/closed loop mode ? – Read encoder mode value
Parameter Range:	n – 1 to 99 x – 0 for open loop mode, 2 for open loop with closed loop deceleration, 3 closed loop
Related Commands:	ENC, EAD, EPL, DBD, SAV
Example:	`2FBK3` | Axis 2, Set closed loop mode

FFP

Feed Forward Parameter

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the feedforward parameter of the controller. This is used when the controller is operating in feedback mode 2 or 3. The SAV command can be used to save the feed forward parameter to flash.
Returns:	A read operation returned the current feed forward parameter
Syntax:	nFFPx – Standard syntax nFFP? – Read encoder mode value Error [#]: FFPx – Missing axis number [30] FFP? – Read operation with missing axis number [27] nFFP – Missing closed/open loop parameter [28]
Parameter Description:	n[int] – Axis number x[int] – Feed forward parameter ? – Read encoder mode value
Parameter Range:	n – 1 to 99 x – 0 to 99,999
Related Commands:	PID, FBK, SAV
Example:	`2FFP1000` | Axis 2, Set feed forward parameter

FSR

Full Steps Per Revolution

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the number of full steps per single revolution of the motor shaft. It is determined by the motor. The SAV command can be used to save the full steps per revolution setting to flash.
Returns:	A read operation returns the Full Steps Per Revolution for the specified axis.
Syntax:	nFSRx – Standard syntax nFSR? – Read encoder mode value 0FSRx – All axes set Full Steps Per Revolution to x Error [#]: FSRx – Missing axis number [30] FSR? – Read operation with missing axis number [27] nFSR – Missing full step per revolution parameter [28]
Parameter Description:	n[int] – Axis number x[int] – Full steps per revolution ? – Read full steps per revolution value
Parameter Range:	n – 1 to 99 x – 0 to 10000
Related Commands:	LSP, SAV
Example:	`1FSR200` | Axis 1, Set 200 Full Steps Per Rev

MICRONIX USA, LLC
Fountain Valley, California
www.micronixusa.com

HAC

Home Acceleration

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired value for the home acceleration and deceleration for a specified axis. The controller will not allow for HAC values that are greater than AMX. The SAV command can be used to save the home acceleration setting to flash.
Returns:	A read operation returns the home acceleration and deceleration value in mm/s^2 for the specified axis.
Syntax:	nHACx – Standard syntax 0HACx – All axes set home acceleration/deceleration nHAC? – Read home acceleration/deceleration Error [#]: HAC? – Read operation with missing axis number [27] nHAC – Missing direction setting [28]
Parameter Description:	n[int] – Axis number x[float] – Acceleration
Parameter Range:	n – 0 to 99 x – 0.001 to AMX ? – Read acceleration value
Related Commands:	ACC, DEC, HOM, AMX, SAV
Example:	`4HAC0.1` | Axis 4, Set home acceleration & deceleration to 0.1 mm/s^2 [degrees/s^2]

HCG Home Configuration

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to select the direction of motion when the Home [HOM] command is initialized. The SAV command can be used to save the home configuration setting to flash.
Returns:	A read operation returns the current direction setting: 0 – Home starts in the direction of the negative limit 1 – Home starts in the direction of the positive limit
Syntax:	nHCGx – Standard syntax 0HCGx – All axes set direction nHCG? – Read direction setting Error [#]: HCG? – Read operation with missing axis number [27] nHCG – Missing direction setting [28]
Parameter Description:	n[int] – Axis number x [int] – Set direction of motion.
Parameter Range:	n – 0 to 99 x – 0 for setting motion in the direction of the negative limit 1 for setting motion in the direction of the positive limit
Related Commands:	HOM, SAV
Example:	`3HCG0` | Axis 3, Set initial direction of Home command towards the negative limit – `0HCG1` | All Axes, Set initial direction of Home command towards the positive limit

HOM Home

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to find the home (reference) position for a specified axis. An error will occur if there is no encoder signal at the time of execution. Initial home direction is configured using the HCG command. This command will jog the stage till it reaches the limit configured by the HCG command. It will then acquire the reference position by looking for the index.
Returns:	A read parameter returns the following calibration values for the specified axis: 0 – Not calibrated to home position 1 – Calibrated to home position
Syntax:	nHOM – Standard syntax nHOM? – Returns 1 if homed since last startup otherwise returns 0 0HOM – All axes execute home position Error [#]: HOM? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 0 to 99
Related Commands:	HCG
Example:	`1HOM` | Axis 1, Move to home position

HST

Hard Stop Detection

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to enable hard stop detection, which stops motion when the stage reaches a hard stop. The SAV command can be used to save the hard stop detection setting to flash.
Returns:	A read operation returns the following hard stop detection off/on values for the specified axis: 0 – Hard Stop Detection is off 1 – Hard Stop Detection is on
Syntax:	nHSTx – Standard syntax nHST? – Read Hard Stop Detection off/on value 0HSTx – All axes set Hard Stop Detection value Error [#]: HST? – Read operation with missing axis number [27] xHST – Missing motor off/on parameter [28]
Parameter Description:	n[int] – Axis number
Parameter Range:	N – 0 to 99
Related Commands:	STP, SAV
Example:	`8HST1` | Axis 8, Hard Stop Detection on – `0HST0` | All axes, Hard Stop Detection off

HVL

Home Velocity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired home velocity for the specified axis. The home velocity value should be lower than the maximum allowable velocity [VMX] for the command to be accepted. The SAV command can be used to save the home velocity setting to flash.
Returns:	A read operation returns the home velocity value in mm/s for the specified axis.
Syntax:	nHVLx – Standard syntax nHVL? – Read velocity value 0HVLx – Missing axis number, all axes set velocity Error [#]: HVL? – Read operation with missing axis number [27] nHVL – Missing velocity parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Velocity value ? – Read velocity value
Parameter Range:	n – 0 to 99 x – 0.001 to VMX
Related Commands:	VMX, HOM, SAV
Example:	`1HVL.25` | Axis 1, Set home velocity to 0.25mm/s – `5HVL?` | Axis 5, Read home velocity value

IDN

Identification Name

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓	✓	✓	✓			✓

Command Description:	An identification name can be stored in each NanoDrive, allowing each axis to have a unique descriptor. The SAV command can be used to save the identification name to flash.
Returns:	A read operation returns the ASCII representation of the stored identification name
Syntax:	nIDNx1 – Standard syntax 0IDNx1 – All axes set identification name Error [#]: IDN – Missing axis number [27]
Parameter Description:	n[int] – Axis number x1[string] – Identification name
Parameter Range:	n – 0 to 99 x1 – 50 character maximum
Related Commands:	SAV
Example:	`4INDaxis4` | Axis 4, set ID name to "axis4"

INI

Initialize Phasing

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓		✓		✓

Command Description:	This command initializes the phase angle of the connected BLDC motor. The routine may result in slight motion (<5mm) in either direction, so the motor needs to be clear of any obstructions. Phase initialization needs to be done on power up when an incremental encoder is being used. The phase angle will not be retained when the NanoDrive is powered off.
Returns:	A read operation returns the following initialization completed values for the specified axis: 0 – Initialization has not been completed 1 – Initialization has been completed
Syntax:	nINI – Standard syntax 0INI – All axes execute acceleration value Error [#]: INI? – Read operation with missing axis number [27] INI – Missing axis number [27]
Parameter Description:	n[int] – Axis number ? – Read initialization status
Parameter Range:	n – 0 to 99
Related Commands:	MOT, MTC
Example:	`4INI` | Axis 4, start phase angle initialization routine

INP

In Position

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓			✓

Command Description:	This command is used to set the in-position range and time requirements. If both the in-position range and time are met by the controller, it is "in position" and bit 3 of the STA register will be set. The in-position range refers to the maximum number of encoder counts (±) from the target that is considered acceptable. The in-position time refers to the number of seconds the stage must be within the in-position range. The in-position feature is applicable in feedback modes FBK2 and FBK3. If either parameter of in-position is set to zero, the deadband setting (DBD) will be used to determine bit 3 of the STA register. The SAV command can be used to save the in-position settings to flash.
Returns:	A read operation returns the values assigned to the In-position range and time parameters
Syntax:	nINPx1,x2 – Standard syntax 0INPx1,x2 – All axes execute acceleration value nINP? – Read in-position range and time values Error [#]: INP? – Read operation with missing axis number [27] nINP – Missing in-position range and timeout parameter values [28]
Parameter Description:	n[int] – Axis number x1[int] – In-position Range x2[int] – In-position time ? - Read in-position range and timeout values
Parameter Range:	n – 0 to 99 x1 – Encoder dependent, 0 to disable in-position, Encoder Counts x2 – Encoder dependent, Seconds, 0 to disable in-position, Seconds
Related Commands:	ENC, DBD, FBK, STA, SAV
Example:	`4INP5,1.5` | Axis 4, set in-position range to ±5 encoder counts. Controller must stay in-position for 1.5 seconds to be in-position

IOD

Set IO Definition

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
		✓					

Command Description:	This command is used to select Input or Output for one of the I/O pins on the 12-Pin Gecko connector. The SAV command can be used to save the IO direction setting to flash.
Returns:	A read operation can return the direction of a specified I/O or combined direction assignment for all I/O
Syntax:	nIODx1,x2 – Standard syntax nIOD? – Read I/O definition of all I/O pins nIODx1? – Return I/O direction for specified I/O Error [#]: IODx1,x2 – Missing axis number [30] IOD? – Read operation with missing axis number [27] nIOD – Missing closed/open loop parameter [28]
Parameter Description:	n[int] – Axis number x1[int] – I/O Pin x2[int] – Input/ Output ? – Read I/O definition of each I/O pins
Parameter Range:	n – 1 to 99 x1 – 1 – IO1 2 – IO2 3 – IO3 4 – IO4 x2 – 0 – Output 1 – Input
Related Commands:	IOF, IOP, IOS, SAV
Example:	`2IOD2,1` | Axis 2, Set IO2 to an Input

IOF

Set IO Function

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
		✓	✓	✓	✓	✓	✓

Command Description:	This command is used to select the function of an I/O pin. The SAV command can be used to save the IO function settings to flash.
Returns:	A read operation is not available with this command.
Syntax:	nIOFx1,x2 – Standard syntax nIOF? – Read the I/O function assignment for all I/O nIOFx1? – Read the I/O function assignment for the specified I/O Error [#]: IOFx1,x2 – Missing axis number [30] IOF? – Read operation with missing axis number [27] nIOF – Missing closed/open loop parameter [28]
Parameter Description:	n[int] – Axis number x1[int] – I/O Pin x2[int] – I/O Function ? – Read encoder mode value
Parameter Range:	n – 1 to 99 x1 – 1 – IO1 2 – IO2 3 – IO3 4 – IO4 x2 – 0 – No function 1 – Trace data acquisition on trigger (TRA) 2 – Output pulse trigger when in position 3 – Output level when motion ends 4 – Output pulse trigger at constant velocity (CVP) 5 – Output pulse when position is reached (PTP) 6 – Output pulse at regular intervals (PIP)
Related Commands:	IOF, TRA, CVP, PTP, PIP, IOS, SAV
Example:	`2IOF2,1` | Axis 2, Set IO2 to data logging trigger

IOP

Set IO Polarity

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
	✓	✓	✓				

Command Description:	This command is used to select the polarity of an I/O pin. The SAV command can be used to save the IO polarity setting to flash.
Returns:	A read operation returns the polarity values assigned to the I/O Pins.
Syntax:	nIOPx1,x2 – Standard syntax nIOP? – Read polarity value for all I/O nIOPx1? – Read polarity value for the specified IO Error [#]: IOPx1,x2 – Missing axis number [30] IOP? – Read operation with missing axis number [27]
Parameter Description:	n [int] – Axis number x1[int] – I/O Pin x2[int] – I/O Polarity ? – Read all I/O pins polarity value
Parameter Range:	n – 1 to 99 x1 – 1 – IO1 2 – IO2 3 – IO3 4 – IO4 x2 – 0 – Active Low Pulses (5 to 0V pulses) 1 – Active High Pulses (0 to 5V pulses)
Related Commands:	IOF, TRA, CVP, PTP, PIP, IOS, SAV
Example:	`2IOP2,1` | Axis 2, Set IO2 to Active High

IOS

Set IO Status

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
	✓	✓	✓				

Command Description:	This command is used to select the output or read input of an I/O pin. *This command supersedes all I/O commands (IOF, IOP, CVP, PTP, and PIP)
Returns:	A read operation returns the output values assigned to the I/O Pins.
Syntax:	nIOSx1,x2 – Standard syntax nIOS? – Read input/output value for all I/O nIOSx1? – Read input/output value for the specified I/O Error [#]: IOSx1,x2 – Missing axis number [30] IOS? – Read operation with missing axis number [27]
Parameter Description:	n [int] – Axis number x1[int] – I/O Pin x2[int] – I/O Status ? – Read input/output value
Parameter Range:	n – 1 to 99 x1 – 1 – IO1 2 – IO2 3 – IO3 4 – IO4 x2 – 0 – Output Off 1 – Output On
Related Commands:	IOF, IOP
Example:	`2IOS2,1` | Axis 2, Set IO2 to output "on"

IWL

Integrator Windup Limit

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets the integrator windup limit during feedback mode 3. This setting will effectively limit the max allowable error for the integrator term of the PID feedback loop. The SAV command can be used to save the integrator windup limit to flash.
Returns:	A read operation will return the current integrator windup limit.
Syntax:	nIWLx1 – Standard syntax nIWL? – Read output value Error [#]: IWLx1 – Missing axis number [30] IWL? – Read operation with missing axis number [27]
Parameter Description:	n [int] – Axis number x1 [int] – Integrator Windup Value ? – Read output value
Parameter Range:	n – 1 to 99 x1 – 0 to 100,000,000
Related Commands:	FBK, PID, IWL, SAV
Example:	`2IWL1000` | Axis 2, Set integrator windup value

JAC

Jog Acceleration and Deceleration

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired value for the jog acceleration and deceleration for a specified axis. The controller will not allow for JAC values that are greater than AMX. The SAV command can be used to save the jog acceleration and deceleration setting to flash.
Returns:	A read operation returns the jog acceleration and deceleration value in mm/s^2 for the specified axis.
Syntax:	nJACx – Standard syntax 0JACx – All axes execute acceleration value nJAC? – Read acceleration value Error [#]: JAC? – Read operation with missing axis number [27] nJAC – Missing acceleration parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Acceleration ? – Read acceleration value
Parameter Range:	n – 0 to 99 x – 0.001 to AMX
Related Commands:	ACC, DEC, AMX, JOG, SAV
Example:	`4JAC0.1` | Axis 4, Set jog acceleration & deceleration to 0.1 mm/s^2 [degrees/s^2]

JOG

Jog Mode

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to jog a specific axis, or move continuously in a direction with no target position. The jog velocity is a percentage of the maximum velocity and may be changed on-the-fly by sending another JOG command during motion.
Returns:	A read operation returns the jog speed value in mm/s for the specified axis.
Syntax:	nJOGx – Standard syntax Error [#]: JOG? – Read operation with missing axis number [27] JOGx – Missing axis number [30] nJOG – Missing velocity parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Velocity
Parameter Range:	n – 1 to 99 x – 000.001% to ±100.000 % (of maximum velocity [VMX])
Related Commands:	JAC, VMX
Example:	`4JOG10` | Axis 4, Jog at 10% maximum velocity

LCG

Limit Configuration

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command selects whether NanoDrive will use soft limits or external limit switches to define the end of travel locations. The SAV command can be used to save the limit configuration setting to flash.
Returns:	A read operation is not available with this command.
Syntax:	nLCGx – Standard syntax Error(s): LCGx – Missing axis number [30] nLCG – Missing program number parameter [28]
Parameter Description:	n[int] – Axis number x[int] – 0 – ignore [default] 1– active
Parameter Range:	n – 1 to 99 x – 0 – ignore [default] 1 – Soft Limits Only 2 – Limit Switches Only 3 – Limit Switched and Soft Limits enabled
Related Commands:	LPL, LDR, TLN, TLP, SAV
Example:	`1LCG2` | Axis 1, set limit switches active

LDP

Load Parameters

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓				✓

Command Description:	This command is used to restore the set of most recently saved settings for the specified axis. This allows the user to return to a user set predefined state.
Returns:	A read operation cannot be used with this command.
Syntax:	nLDP – Standard syntax 0LDP – All axes restore saved settings
Parameter Description:	n[int] – Axis number
Parameter Range:	N – 0 to 99
Related Commands:	None
Example:	`1LDP` | Axis 1, restore saved settings

LDR

Positive/ Negative Limit Location

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	Determines orientation of positive limit and negative limit. The SAV command can be used to save the limit location setting to flash.
Returns:	A read operation returns the following limit direction values for the specified axis: 0 – Normal orientation 1 – Reverse orientation
Syntax:	nLDRx – Standard syntax nLDR? – Read velocity value 0LDRx – All axes set limit direction Error [#]: LDR? – Read operation with missing axis number [27] nLDR – Missing limit parameter [28]
Parameter Description:	n[int] – Axis number x[int] – limit direction value ? – Read limit direction value
Parameter Range:	n – 0 to 99 x – 0 or 1
Related Commands:	LPL, LCG, SAV
Example:	`1LDR1` | Axis 1, set to reverse orientation – `5LDR?` | Axis 5, Read limit switch orientation

LPF

Low Pass Filter

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets can enable and configure a low pass filter on the PID output when the axis is in closed loop mode (FBK3). The SAV command can be used to save the low pass filter settings to flash.
Returns:	A read operation returns the low pass filter parameters
Syntax:	nLPFx1,x2 – Standard syntax nLPF? – Read low pass filter parameters 0LPFx1,x2 – All axis set low pass filter parameters Error(s): LPFx – Missing axis number [30] nLPF – Missing program number parameter [28]
Parameter Description:	n[int] – Axis number x1 – 0 – Disable Low Pass Filter 1 – Enable Low Pass Filter x2[int] – Cutoff frequency
Parameter Range:	n – 1 to 99 x1 – 0 – enable 1 – disable X2 – 0 to 999,999,999 Hz
Related Commands:	FBK, PID, SAV
Example:	`6LPF1,3000` | Axis 6, enable low pass filter at 3kHz

LPL

Limit Switch Polarity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets whether the limit switch inputs are active high[1] or low[0]. The SAV command can be used to save the limit polarity setting to flash.
Returns:	A read operation returns the program table for the specified axis.
Syntax:	nLPLx – Standard syntax Error(s): LPLx – Missing axis number [30] nLPL – Missing program number parameter [28]
Parameter Description:	n[int] – Axis number x – 0 – Active Low 1 – Active High
Parameter Range:	n – 1 to 99 x – 0 – active low [default] 1 – active high
Related Commands:	LCG, LDR, SAV
Example:	`6LPL1` | Axis 6, limit switches set to active high

LRB

Limit Rebound Amount

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓	✓	✓	✓	✓	✓	✓

Command Description:	This command sets the amount the motor will move back off a limit once the limit is triggered during motion. This applies to both hard stop limits and limit switches. If the limit rebound is set to zero the motor will immediately stop once the limit condition is met. The motor will remain at the limit position until another move is issued. The SAV command can be used to save the limit rebound amount setting to flash.
Returns:	A read operation returns the limit rebound amount for the specified axis.
Syntax:	nLRBx – Standard syntax Error(s): LRBx – Missing axis number [30] nLRB – Missing limit rebound parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Limit rebound amount
Parameter Range:	n – 1 to 99 x – 0.000 to ± 999.999999 mm (degrees)
Related Commands:	LCG, MLN, MLP, SAV
Example:	`1LRB0.5` | Axis 1, set limit rebound amount to 0.5mm

LSP

Lead Screw Pitch

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets the lead screw pitch. The SAV command can be used to save the leadscrew pitch setting to flash.
Returns:	A read operation returns the lead screw pitch value in mm for the specified axis.
Syntax:	nLSPx – Standard syntax nLSP? – Read lead screw pitch value 0LSPx – All axes set lead screw pitch to x Error [#]: LSP? – Read operation with missing axis number [27] nLSP – Missing lead screw pitch parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Lead Screw Pitch ? – Read Lead Screw Pitch
Parameter Range:	n – 1 to 99 x – 0.001 to 999.999mm
Related Commands:	FSR, UST, SAV
Example:	`1LSP0.25` | Axis 1, Set lead screw pitch to 0.25mm – `5LSP?` | Axis 5, Read lead screw pitch

LST

Program List

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
			✓			

Command Description:	This command is used to display the stored commands that make up a given internal program.
Returns:	A list of the commands in order as executed when the given program is run.
Syntax:	nLSTx – Standard syntax Error [#]: 1LST? – Read Not Available for this Command [38]
Parameter Description:	n[int] – Axis number x[int] – Program# to be read
Parameter Range:	n – 1 to 99 x – 1 to 16
Related Commands:	PGM
Example:	`6LST1` | Axis 6, return program 1 list of commands

MCM

Motor Current Maximum

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to read the maximum output current to the motor as determined by hardware. The MCS setting cannot exceed this value. The SAV command can be used to save the maximum motor current setting to flash.
Returns:	A read operation returns the max motor current value in Amps for the specified axis
Syntax:	nMCMx – Standard syntax nMCM? – Read motor current settings Error [#]: MCM – Missing axis number [30] nMCM – Missing motor current settings [28]
Parameter Description:	n[int] – Axis number x1[float] – Maximum allowable current setting
Parameter Range:	n – 0 to 99 x – 0 to 10.00 [A]
Related Commands:	MCS, MCR, SAV
Example:	`3MCM?` | Axis 3, Read max motor current value

MCR

Motor Current Reduction

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets the percentage the motor current reduces when the motor is idle. Reducing the motor current when idle will reduce the temperature of the motor while also reducing the holding force/torque. The motor current reduction setting is only applicable to stepper motor operation. The SAV command can be used to save the motor current reduction setting to flash.
Returns:	A read operation returns the motor current reduction value in percentage for the specified axis.
Syntax:	nMCRx – Standard syntax 0MCRx – All axes set the motor current reduction value Error [#]: MCRx – Missing axis number [30]
Parameter Description:	n[int] – Axis number x[int] – Motor current reduction value
Parameter Range:	n – 0 to 99 x – 0 to 100
Related Commands:	MCS, SAV
Example:	`8MCR50` | Axis 8, Set motor current reduction to 50%

MCS

Motor Current Settings

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets the desired current output level during different stages of motion. The parameters of the motor current settings will vary depending on the motor type. For stepper motors, the run current setting [x1] will define the current output to the motor when moving at constant speed. The I^2T setting [x2] will define how long the NanoDrive can output current above the run current setting[x1]. For 3-phase brushless motors and brushed DC motors, the run current setting [x1] will define the current output to the motor when moving at constant speed. The surge current setting [x2] will define the maximum current output during phases of motion that require additional force. The I^2T setting [x3] will define how long the NanoDrive can output current above the run current setting[x1]. If the I^2T setting to exceeded, the motor will disable and enter an error state. The SAV command can be used to save the motor current settings to flash. Piezo operation does not use the MCS command.
Returns:	A read operation returns the motor run and surge currents in amps and I^2T in A^2sec
Syntax:	nMCSx1,x2 – Standard syntax (Stepper motors) nMCSx1,x2,x3 – Standard syntax (BLDC & DC motors) nMCS? – Read motor current settings Error [#]: MCSx1,x2,x3 – Missing axis number [30] nMCS – Missing motor current settings [28]
Parameter Description:	n[int] – Axis number x1[float] – Motor run current setting x2[float] – Motor surge current setting x3[int] – I^2T setting
Parameter Range:	BLDC & DC Motors N – 0 to 99 x1 – 0 to MCM [A] x2 – 0 to MCM [A] x3 – 0 to 32-bit integer [A^2sec] Stepper Motors N – 0 to 99 x1 – 0 to MCM [A] x2 – 0 to 32-bit integer [A^2sec]
Related Commands:	MCM, MCR, SAV
Example:	`3MCS2.1,4,1000` | Axis 3, Set BLDC motor current setting to 2.1A run current, 4.0A surge current, and 1000 A^2sec I^2T setting

MFE

Maximum Following Error

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets the maximum allowable position error between the theoretical position and encoder positions. If the error is reached the motor will either stop motion or disable, depending on the following error action setting. The SAV command can be used to save the maximum following error to flash.
Returns:	A read operation returns the maximum following error and the following error action.
Syntax:	nMFEx1,x2,x3 – Standard syntax 0MFEx1,x2,x3 – All axes set the motor current reduction value nMFE? – Read max following error settings Error [#]: MFEx1,x2,x3 – Missing axis number [30]
Parameter Description:	n[int] – Axis number x1[float] – Maximum following error x2[float] – Maximum following error timeout x3[int] – Maximum following error action
Parameter Range:	n – 0 to 99 x1 – 0 to 999.999999 mm x2 – 0 for immediate, Seconds x3 – 0 to stop motion 1 to disable motor
Related Commands:	FBK, ENC, SAV
Example:	`1MFE1.0,1.0, 1` | Axis 1, Set max following error to 1mm and timeout to 1 second. Motor will disable if max following error is exceeded

MLN

Move to Negative Limit

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓		✓		✓

Command Description:	This command initiates a move to the negative limit position. Upon reaching the negative hard limit the controller will then move the stage back from the hard limit and stop. An error will occur if there is no encoder signal at the time of execution.
Returns:	A read operation is not available with this command.
Syntax:	nMLN – Standard syntax 0MLN – All axes execute move to negative limit position Error [#]: MLN – Missing axis number [30]
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 0 to 99
Related Commands:	MLP
Example:	`8MLN` | Axis 8, Move to negative limit position – `0MLN` | All Axes, Move to negative limit position

MLP

Move to Positive Limit

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓		✓		✓

Command Description:	This command initiates a move to the positive limit position. Upon reaching the positive hard limit the controller will then move the stage back from the hard limit and stop. An error will occur if there is no encoder signal at the time of execution.
Returns:	A read operation is not available with this command.
Syntax:	nMLP – Standard syntax 0MLP – All axes execute move to positive limit position Error [#]: MLP – Missing axis number [30]
Parameter Description:	n[int] – Axis number
Parameter Range:	N – 0 to 99
Related Commands:	MLN
Example:	`1MLP` | Axis 1, Move to positive limit position – `0MLP` | All Axes, Move to positive limit position

MOT

Toggle Motor Off/On

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to turn the motor current flow "Off" or "On" for a specified axis. Turning the motor current off will cause the piezo to relax and the stage will shift slightly.
Returns:	A read operation returns the following motor current off/on values for the specified axis: 0 – Motor current is off 1 – Motor current is on
Syntax:	nMOTx – Standard syntax nMOT? – Read motor current off/on value 0MOTx – All axes set motor value Error [#]: MOT? – Read operation with missing axis number [27] xMOT – Missing motor off/on parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Motor current off/on ? – Read motor current off/on value
Parameter Range:	n – 0 to 99 x – 0 for motor current off 1 for motor current on
Related Commands:	None
Example:	`1MOT0` | Axis1, Set motor current to off

MPL

Toggle Motor Polarity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command sets the motor polarity for the specified axis. If the theoretical positive direction is away from the motor, changing this setting will make the theoretical positive direction towards the motor. The SAV command can be used to save the motor polarity setting to flash.
Returns:	A read operation returns the current motor polarity setting for the specified axis.
Syntax:	nMPLx – Standard syntax nMPL? – Read motor current off/on value 0MPLx – All axes set motor value Error [#]: MPL? – Read operation with missing axis number [27] nMPL – Missing motor off/on parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Motor Polarity setting ? – Read motor polarity value
Parameter Range:	n – 0 to 99 x – 0 Normal 1 Reverse
Related Commands:	MVA, MVR, JOG, SAV
Example:	`1MPL0` | Axis1, To normal Polarity

MSA

Synchronous Move - Absolute

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to set up a synchronous move using the absolute position of the axes involved. This command is most useful when coordinating motion to an absolute position between 2 or more axes and requires a RUN command on a separate line to execute the synchronous move. It is recommended to run multiple MSA commands on the same command line, as they are executed closer together than on separate lines. An error will occur if the commanded position is outside of the soft limits.
Returns:	A read operation is not available with this command.
Syntax:	nMSAx – Standard syntax 0MSAx – All axes execute synchronous move Error [#]: nMSA – Missing absolute position parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Absolute position
Parameter Range:	n – 0 to 99 x – 0.000000 to 999.999999 mm (degrees)
Related Commands:	RUN, MSR
Example:	`1MSA10;2MSA10` | Axis 1, Move to absolute position 10 mm[°]; Axis 2, Move to absolute position: 10 mm[°] `0RUN` | All axes, Execute synchronous move - `0MSA5` | All axes, Move to absolute position: 5 mm[°] `0RUN` | All axes, Execute synchronous move

MICRONIX USA, LLC
Fountain Valley, California
www.micronixusa.com

MSR

Synchronous Move – Relative

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to set up a relative move using the relative position of the axes involved. This command is most useful when coordinating relative positions between 2 or more axes and requires a RUN command on a separate line to execute the synchronous move. It is recommended to use multiple MSR commands on the same command line, as they are executed closer together than on separate lines. An error will occur if the commanded increment will cause the stage to travel outside of the set soft limits.
Returns:	A read operation is not available with this command.
Syntax:	nMSRx – Standard syntax 0MSAx – All axes execute synchronous move Error [#]: nMSR – Missing relative position parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Relative position
Parameter Range:	n – 0 to 99 x – 0.000000 to 999.999999 mm (degrees)
Related Commands:	RUN, MSA
Example:	`4MSR.1;5MSR.5` | Axis 4, Move 0.1 mm [degrees]; Axis 5, Move 0.5 mm[°] `0RUN` | Execute synchronous move – `0MSR0.01` | All axes, Move 0.01 mm[°] `0RUN` | All axes, execute synchronous move

MTC Motor Control Type

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the type of motor being operated by the NanoDrive controller (piezo, stepper, 3-phase brushless, or DC motor). The SAV command can be used to save the motor control type setting to flash.
Returns:	A read operation returns the motor control type setting.
Syntax:	nMTCx – Standard syntax nMTC? – Read motor control type 0MTCx – All axes set the motor control type Error [#]: MTC? – Read operation with missing axis number [27] nMTC – Missing motor type parameter [28]
Parameter Description:	n[int] – Axis number x[int] – Motor control type
Parameter Range:	n – 0 to 99 x – 1 for piezo motor 2 for stepper motor 3 for 3-phase brushless motor 4 for DC motor
Related Commands:	SAV
Example:	`4MTC2` | Axis 4, Set motor control type to stepper motor

MTP Motor Pitch

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the magnetic pitch setting. It is only applicable to 3-phase brushless motors. The SAV command can be used to save the motor pitch setting to flash.
Returns:	A read operation returns the magnetic pitch setting.
Syntax:	nMTPx – Standard syntax nMTP? – Read motor pitch setting 0MTPx – All axes set the motor pitch setting Error [#]: MTP? – Read operation with missing axis number [27] nMTP – Missing motor pitch parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Motor pitch
Parameter Range:	n – 0 to 99 x – 1.0 to 999.999 mm
Related Commands:	MTC, MTV, SAV
Example:	`1MTP12` | Axis 1, Set magnetic pitch to 12mm

MTV

Motor Voltage

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the motor voltage setting. It is only applicable to stepper and 3-phase brushless motor types. The SAV command can be used to save the motor voltage setting to flash.
Returns:	A read operation returns the motor voltage setting.
Syntax:	nMTVx – Standard syntax nMTV? – Read motor voltage setting 0MTVx – All axes set the motor voltage setting Error [#]: MTV? – Read operation with missing axis number [27] nMTV – Missing motor voltage parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Motor voltage
Parameter Range:	n – 0 to 99 x – 1.0 to 60.0V
Related Commands:	MCS, MTC, SAV
Example:	`1MTV5` | Axis 1, Set motor voltage to 5V

MVA Move Absolute

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to initiate an instantaneous move to an absolute position for a specified axis. An error will occur if the commanded position is outside of the soft limits.
Returns:	A read operation is not available with this command.
Syntax:	nMVAx – Standard syntax 0MVAx – All axes execute instantaneous move Error(s): nMVA – Missing absolute position parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Absolute position
Parameter Range:	n – 0 to 99 x – 0.000000 to ± 999.999999 mm (degrees)
Related Commands:	MVR
Example:	`4MVA14.5` | Axis 4, Move to absolute position: 14.5 mm [°] – `0MVA5.5` | All axes, Move to absolute position: 5.5 mm [°]

MVR Move Relative

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to initiate an instantaneous move to a relative position for a specified axis. An error will occur if the commanded increment will cause the stage to travel outside of the set soft limits.
Returns:	A read operation is not available with this command.
Syntax:	nMVRx – Standard syntax 0MVRx – All axes execute command. Error(s): nMVR – Missing relative position parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Relative position
Parameter Range:	n – 0 to 99 x – 0.000000 to ± 999.999999 mm [°]
Related Commands:	MVA
Example:	`6MVR10` | Axis 6, Move 10 mm [°] – `0MVR.89` | All axes, Move 0.89 mm [°]

PGL

Loop Program

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to change the program loop setting. If the program loop flag is set, any program that is executed will run in a continuous loop. It can be combined with the PGS command to run a program continuously on startup. A looping program can be stopped at any time by sending a STP command to the controller
Returns:	A read operation returns the program loop setting for the specified axis.
Syntax:	nPGLx1,x2 – Standard syntax nPGLx1? – Read Error(s): PGLx1,x2 – Missing axis number [30] nPGL – Missing program number parameter [28]
Parameter Description:	n[int] – Axis number x1[int] – Program to be looped x2[int] – Number of repetitions
Parameter Range:	N – 1 to 99 x1 – 1 to 32 x2 – 0 for infinite repetition 1 to 99
Related Commands:	PGS, STP
Example:	`1PGL1,0` | Axis 1, Program 1 will run continuously `8PGL5,3` | Axis 8, Program 5 will run 3 times `2PGL3?` | Axis 2, read number of loops of program 3

PGM

Begin Program Recording

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	

Command Description:	This command is used to enter program recording mode for a specified axis. The program being recorded must use a unique program number or else the program will be ignored. Use the LST command to check program number availability and use the ERA command to erase any previously recorded programs. Each program has a size limit of 4Kb.
Returns:	A read operation returns the program table for the specified axis.
Syntax:	nPGMx – Standard syntax nPGM? – Read a binary representation of written program numbers If programs 1 and 2 are written it will return 3 If programs 1 and 4 are written it will return 9 If only program 1 is written it will return 1 If only program 3 is written it will return 4 Error(s): PGMx – Missing axis number [30] nPGM – Missing program number parameter [28]
Parameter Description:	n[int] – Axis number x[int] – Program number to be recorded
Parameter Range:	n – 1 to 99 x – 1 to 32
Related Commands:	END, EXC, LST, ERA
Example:	`1PGM3` | Axis 1, Begin recording program. Save program as program 3

PGS

Run Program At Start-Up

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓		✓	✓

Command Description:	This command is used to set a program to run immediately on start-up. Only one program per axis can run on start up. NOTE: The PGS value must be saved using the SAV command prior to power down for a program to run on startup.
Returns:	A read operation returns a value for the specified axis in the format below: 0 – No program set to run 1-32 – Program set to run on start-up
Syntax:	nPGSx – Standard syntax 0PGSx – Missing axis number, all axes set program to run on start-up nPGS? – Read program(s) set to run on start-up Error [#]: PGS? – Read operation with missing axis number [27] nPGS – Missing program set to run on start-up parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Program set to run on start-up ? – Read encoder mode value
Parameter Range:	n – 0 to 99 x – 0 – No Program 1 to 32 – Specific program set to run on start-up
Related Commands:	LST, PGM
Example:	`6PGS5` | Axis 6, set program 5 to run on start-up – `0PGS16` | All axes, set program 16 to run on start-up – `3PGS?` | Axis 3, Read program to run on start-up – `3PGS0` | Axis 3, Set no program to run on start-up

PID

Set Feedback Constants

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the encoder feedback constants for a specified controller. The SAV command can be used to save the closed-loop feedback constants to flash.
Returns:	A read operation returns the encoder feedback constant values for the specified axis.
Syntax:	nPIDx – Standard syntax nPID? – Read encoder feedback constant values Error(s): PIDx – Missing axis number [30] PID? – Read operation with missing axis number [27] nPID – Missing encoder feedback constant parameters [28]
Parameter Description:	n[int] – Axis number x1[float] – K_p (proportional constant) x2[float] – K_i (integral constant) x3[float] – K_d (derivative constant) ? – Read encoder feedback constants and values
Parameter Range:	n – 1 to 99 x1 – 0.000 to 100.000 x2 – 0.000 to 100.000 x3 – 0.000 to 100.000
Related Commands:	FBK, ENC, POS, FFP, IWL, SAV
Example:	`5PID.02,1.5,1` | Axis 5, Set K_p to 0.02, K_i to 1.5, K_d to 1 `3PID?` | Axis 3, Read PID settings

PIP

Pulse at Regular Intervals

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
	✓	✓	✓			✓	

Command Description:

This command is used in conjunction with IOD, IOF, and IOP to command the controller to emit a pulse at specified regular intervals when given a start point, an interval distance, and an end point.

PIP operation is a one-time use and will end after finishing all pulse triggers.

Returns:

A read operation returns the values assigned to the start point, interval distance, and end point.

Syntax:

nPIPx1,x2,x3 – Standard syntax
0 PIPx1,x2,x3 – All axes execute position and interval value
nPIP? – Read interval pulse values

Error [#]:
1PIP1,-0.3,2 Invalid Input Parameter (interval does not have the same sign as end position minus start position).

Parameter Description:

n[int] – Axis number
x1[float] – Beginning position for PIP command
x2[float] – Desired pulse interval
x3[float] – Ending position for PIP command

Parameter Range:

n – 0 to 99
x1 – -400.000 to 400.000 mm
x2 – -400.000 to 400.000 mm
x3 – -400.000 to 400.000 mm

Related Commands: IOD, IOF, IOP

Example:

```
4IOP1,0          | Axis 4, Set IO Pin 1's pulse as active low
4IOF1,6          | Axis 4, Set IO Pin 1 for PIP function
4PIP1,0.2,5      | Axis 4, Pulse triggers at IO pins with PIP every
                   0.2mm from 1mm to 5mm.

2IOP4,1          | Axis 2, Set IO Pin 4's pulse as active high
2IOF4,6          | Axis 2, Set IO Pin 4 for PIP function
2PIP-2,-0.5,-6   | Axis 2, Pulse triggers at IO pin with PIP every
                   0.5mm from -2mm to -6mm.
```

POS

Position

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓		✓		✓	

Command Description:	This command is used to read the position information from the specified axis controller
Returns:	A read operation returns the position values in mm for the specified axis in the following format: [Theoretical position in mm, Encoder position in mm] [Theoretical position in degrees, Encoder position in degrees]
Syntax:	nPOS? – Standard syntax Error(s): POS? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number ? – Read position values
Parameter Range:	n – 1 to 99
Related Commands:	MVR
Example:	`4POS?` | Axis 4, Read position values

PTP

Pulse at Target Position

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
	✓	✓	✓			✓	

Command Description:	This command is used in conjunction with IOD, IOF, and IOP to command the controller to emit a pulse at a specified position, from a specified direction (or from both directions).
Returns:	A read operation returns the values assigned to the trigger position and trigger direction.
Syntax:	nPTPx1,x2 – Standard syntax 0 PTPx1,x2 – All axes execute position and pulse value nPTP? – Read interval pulse values Error [#]: 1PTP1,3 – Invalid Input Parameter ("3" is not an accepted value for "trigger direction").
Parameter Description:	n[int] – Axis number x1[float] – Trigger Position x2[int] – Trigger direction
Parameter Range:	n – 0 to 99 x1 – -400.000 to 400.000 mm x2 – 0 triggers when moving negative 1 triggers when moving positive 2 triggers from both sides
Related Commands:	IOD, IOF, IOP
Example:	`4IOP4,1` | Axis 4, Set IO Pin 4's pulse as active high `4IOF4,5` | Axis 4, Set IO Pin 4 for PIP function `4PTP5.73,2` | Axis 4, Pulse triggers at IO pins with PTP upon reaching 5.73mm from either direction.

PWM

PWM Frequency

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the internal PWM frequency when operating a stepper, 3-phase brushless, or DC motor. The PWM setting can be set from 20kHz to 160kHz in multiples of 20kHz. The SAV command can be used to save the PWM setting to flash.
Returns:	A read operation returns the PWM frequency in units of Hz.
Syntax:	nPWMx – Standard syntax nPWM? – Read PWM frequency value Error(s): PWM? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number x[int] – PWM frequency (kHz) ? – Read PWM frequency value
Parameter Range:	n – 1 to 99 x – 20, 40, 60, 80, 100, 120, 140, or 160kHz
Related Commands:	MTC, SAV
Example:	`2PWM40` | Axis 2, Set PWM frequency to 40kHz

REZ

Set Resolution

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the DAC (digital to analog converter) steps per micron resolution for the specified axis. This command is only applicable to piezo motor types. The SAV command can be used to save the piezo resolution setting to flash.
Returns:	A read operation returns the resolution value in steps per micron for the specified axis.
Syntax:	nREZx – Standard syntax nREZ? – Read steps per micron resolution value Error(s): REZ? – Read operation with missing axis number [27] REZx – Missing axis number [30] nREZ – Missing steps per micron resolution parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Steps per micron resolution (steps/mili-degree) (default is 8000) ? – Read steps per micron resolution value (steps/milli-degrees)
Parameter Range:	n – 1 to 99 x – 0 to 99999 DAC steps per micron (steps/milli-degrees)
Related Commands:	MTC, SAV
Example:	`9REZ4000` | Axis 9, Set resolution to 4000 steps/micron [steps/milli-degrees] – `3REZ?` | Axis 3, Read steps/micron [steps/degrees] resolution value

MICRONIX USA, LLC
Fountain Valley, California
www.micronixusa.com

RUN

Start Synchronous Move

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓		✓		✓

Command Description:	This command is used to start a global synchronous move previously set up by using the MSA or MSR commands.
Returns:	A read operation cannot be used with this command.
Syntax:	nRUN – Standard syntax
Parameter Description:	-
Parameter Range:	-
Related Commands:	MSA, MSR
Example:	`3MSR5;4MSR5` | Axis 3, setup 5 mm[degrees] move; Axis 4, setup 5 mm [degrees] move `0RUN` | All axes, Execute synchronous moves

SAV

Save Axis Settings

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓				✓

Command Description:	This command is used to save all settings for the specified axis. This allows an axis to be configured on power up.
Returns:	A read operation cannot be used with this command.
Syntax:	nSAV – Standard syntax 0SAV – All axes save settings
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 0 to 99
Related Commands:	None
Example:	`16SAV` | Axis 16, save settings

SCM

Stepper Control Method

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to determine how a stepper motor is controlled, either current or voltage control. This setting is only applicable in stepper control mode The SAV command can be used to save the stepper control method to flash.
Returns:	A read operation returns the stepper control method 0 – Current control 1 – Voltage control
Syntax:	nSCMx – Standard syntax nSCM? – Read stepper control method 0SCMx – All axes set the stepper control method Error [#]: SCM? – Read operation with missing axis number [27] nSCM – Missing stepper control parameter [28]
Parameter Description:	n[int] – Axis number x[int] – Stepper control method
Parameter Range:	n – 0 to 99 x – 0 for current control 1 for voltage control
Related Commands:	MTC, SAV
Example:	`4SCM1` | Axis 4, Set stepper control method to voltage control

STA

Status Byte

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓		✓		✓	

Command Description: This command is used to check the status register for a specified axis.

Returns:

A read operation will return an integer from 0 to 255 describing the status of the axis. The byte must be decoded in binary to determine the value of each bit.

Bit	7	6	5	4	3	2	1	0
Name	ERR	ACC	CNST	DEC	STP	PGM	PLS	NLS

Bit 7: 1 – One or more errors have occurred. Use ERR? Or CER to clear.
0 – No Errors have occurred.

Bit 6: 1 – Currently in Acceleration phase of motion.
0 – Not in Acceleration phase of motion.

Bit 5: 1 – Currently in Constant Velocity phase of motion.
0 – Not in Constant Velocity phase of motion.

Bit 4: 1 – Currently in Deceleration phase of motion.
0 – Not in Deceleration phase of motion.

Bit 3: 1 – Stage has stopped. (In Closed Loop Stage, is in the deadband) OR in position
0 – Stage is moving. (In Closed Loop, Stage is out of deadband) OR not in position

Bit 2: 1 – A Program is currently running
0 – No program is running

Bit 1: 1 – Positive Switch is Activated
0 – Positive Switch is not Activated

Bit 0: 1 – Negative Switch is Activated
0 – Negative Switch is not Activated

Syntax: nSTA? – Standard syntax

Error(s):
STA? – Read operation with missing axis number [27]
nSTA – Missing read operation parameter [28]

Parameter Description:
n[int] – Axis number
? – Read status register

Parameter Range: n – 1 to 99

Related Commands: INP, CER, ERR

Example:
```
6STA?
```
| Axis 6, Read status register

STP

Stop Motion

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓		✓		✓		✓

Command Description:	This command is used to stop motion for a specified axis.
Returns:	A read operation cannot be used with this command.
Syntax:	nSTP – Standard syntax 0STP – All axes execute stop
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 0 to 99
Related Commands:	EST, DEC
Example:	`8STP` | Axis 8, execute stop

SVP

Save Startup Position

During Motion		Real-time		Program		Global	
Set	Read	Set	Read	Set	Read	Set	Read
	✓	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the startup position. Default is 0. This setting does not require the SAV command to save it into memory. If the stage is using an absolute encoder, the stage will move to the startup position on power. The startup position feature and be enabled/disabled with the first parameter of this command.
Returns:	A read operation returns the Startup position settings for the specified axis.
Syntax:	nSVPx1,x2 – Standard syntax nSVP? – Read startup position parameters Error[#] SVP – Missing axis number
Parameter Description:	n[int] – Axis number x1[int] – 0 disable 1 enabled x2[float] – Startup Position mm [degree] ? – Read Startup Position
Parameter Range:	n – 0 to 99 x – TLN (-999.999999mm) to TLP(999.999999mm)
Related Commands:	None
Example:	`4SVP` | Set current position to Startup position `2SVP2.3` | Set startup position to 2.3mm

SYN

Sync

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
				✓		✓

Command Description:	This command is used in a program together with the wait for sync [WSY] command in order to synchronize motion between multiple axes. Command must be preceded by a % to be considered part of the current program.
Returns:	A read operation cannot be used with this command.
Syntax:	%nSYN – Standard syntax 0SYN – Missing axis number, command accepted as standard syntax
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 0 to 99
Related Commands:	WSY
Example:	`4SYN` | Send sync to axis 4

TLN

Negative Soft Limit Position

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired negative soft limit position, using absolute position, for the specified axis. The negative soft limit position value must be less than the positive soft limit position value [TLP] for the command to be accepted. The SAV command can be used to save the negative soft limit position to flash.
Returns:	A read operation returns the negative soft limit position value.
Syntax:	nTLNx – Standard syntax nTLN? – Read negative soft limit position value 0TLNx – All axes set limit position value nTLN – Set current position to negative limit Error(s): TLN? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number x[float] – Negative soft limit position ? – Read negative soft limit position
Parameter Range:	n – 0 to 99 x – -9999.999999 to TLP, mm [degrees]
Related Commands:	LCG, TLP, SAV
Example:	`2TLN0.005` | Axis 2, Set negative soft limit position to 0.005 mm [degrees] `_` `6TLN?` | Axis 6, Read negative soft limit position value

TLP

Positive Soft limit Position

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired positive soft limit position, using absolute position, for the specified axis. The positive soft limit position value must be greater than the negative soft limit position value [TLN] for the command to be accepted. The SAV command can be used to save the positive soft limit position to flash.
Returns:	A read operation returns the positive soft limit position value for the specified axis.
Syntax:	nTLPx – Standard syntax nTLP? – Read positive soft limit position value 0TLPx – All axes set limit position value nTLP – Set current position to negative limit Error(s): TLP? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number x[float] – Positive soft limit position ? – Read positive soft limit position
Parameter Range:	n – 0 to 99 x – TLN to + 9999.999999 mm [degrees]
Related Commands:	LCG, TLN, SAV
Example:	`4TLP10.005` | Axis 4, Set positive soft limit position to 10.005 mm [degrees] _ `9TLP?` | Axis 9, Read positive soft limit position value

TRA

Perform Trace

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓			✓

Command Description:	This command is used to execute a trace of the specified axis.
Returns:	A read operation returns the position samples taken for the specified axis.
Syntax:	nTRAx1,x2,x3 – Standard syntax nTRA? – Read position values 0TLPx1,x2,x3 – All axes execute trace Error(s): TRA? – Read operation with missing axis number [27] nTRA – Missing parameters [28]
Parameter Description:	n[int] – Axis number x1[int] – Number of samples taken (default is 1000) x2[int] – 10kHz /Sampling frequency (default is 1) x3[float] – Trace starting position (default is immediate) ? – Read position
Parameter Range:	n – 0 to 99 x1 – 1 to 2500 x2 – 1 to 1000 Servo clocks per cycle x3 – -999.999999 to 999.999999 mm [degrees]
Related Commands:	DAT
Example:	`5TRA5,10,1` | Axis 5, execute trace with 5 samples at a sampling frequency of 1kHz starting at a position of 1 mm [degrees] `3TRA2000,,` | Axis 3, execute trace with 2000 samples at a sampling frequency of 10kHz starting at the current position

VEL

Velocity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
✓	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to set the desired velocity for the specified axis. The velocity may be changed on-the-fly by sending another VEL command during motion. The velocity value should be lower than the maximum allowable velocity [VMX] for the command to be accepted. The SAV command can be used to save the velocity to flash.
Returns:	A read operation returns the velocity value in mm/s for the specified axis.
Syntax:	nVELx – Standard syntax nVEL? – Read velocity value 0VELx – Missing axis number, all axes set velocity Error [#]: VEL? – Read operation with missing axis number [27] nVEL – Missing velocity parameter [28]
Parameter Description:	n[int] – Axis number x[float] – Velocity value ? – Read velocity value
Parameter Range:	n – 0 to 99 x – 0.001 to VMX
Related Commands:	VMX, REZ, SAV
Example:	`1VEL.25` | Axis 1, Set velocity to 0.25mm/s [degrees/s] – `5VEL?` | Axis 5, Read velocity value

VER

Firmware Version

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓		✓		✓	

Command Description:	This command is used to check the firmware version for the specified axis.
Returns:	A read operation returns the firmware version for the specified axis.
Syntax:	nVER? – Standard syntax Error(s): VER? – Read operation with missing axis number [27] nVER – Missing read operation parameter [28]
Parameter Description:	n[int] – Axis number ? – Read firmware version
Parameter Range:	n – 1 to 99
Related Commands:	None
Example:	`11VER?` | Axis 11, Read firmware version

VMX

Maximum Allowable Velocity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓	✓	✓	✓	✓	✓

Command Description:	This command is used to read the maximum allowable velocity for a specific axis.
Returns:	A read operation returns the maximum allowable velocity value in mm/s for the specified axis.
Syntax:	nVMX? – Read maximum allowable velocity value Error [#]: VMX? – Read operation with missing axis number [27] nVMX – Missing read operation parameter [123]
Parameter Description:	n[int] – Axis number ? – Read maximum allowable velocity value
Parameter Range:	n – 1 to 2000
Related Commands:	REZ, VEL
Example:	`4VMX?` | Axis 4, Read maximum allowable velocity value

MICRONIX USA, LLC
Fountain Valley, California
www.micronixusa.com

VRT

Encoder Velocity

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
	✓		✓		✓	

Command Description:	This command returns the actual velocity calculated from the encoder.
Returns:	A read operation returns the encoder velocity in mm/s.
Syntax:	nVRT? – Standard syntax Error [#]: VRT? – Read operation with missing axis number [27]
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 1 to 99
Related Commands:	POS
Example:	`5VRT?` | Axis 5, Read encoder velocity

WST

Wait For Stop

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
				✓		

Command Description:	This command is used in a program to wait until motion is completed to begin executing the next command. Command must be preceded by a % to be considered part of the current program.
Returns:	A read operation cannot be used with this command.
Syntax:	%nWST – Standard syntax WST – Missing axis number, command accepted as standard syntax
Parameter Description:	n[int] – Axis number
Parameter Range:	N – 1 to 99
Related Commands:	PGM
Example:	`%7WST` | Axis 7, Wait for motion to stop before executing next command

WSY

Wait For Sync

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
				✓		

Command Description:	This command is used in a program together with the sync [SYN] command in order to synchronize motion between multiple axes. Command must be preceded by a % to be considered part of the current program.
Returns:	A read operation cannot be used with this command.
Syntax:	%nWSY – Standard syntax WSY – Missing axis number, command accepted as standard syntax
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 1 to 99
Related Commands:	SYN
Example:	`%1WSY` | Axis 1, Wait until sync command is received before executing next command

WTM

Wait For Time Period

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
				✓		

Command Description:	This command is used in a program to wait for a specified period of time before executing the next command. Command must be preceded by a % to be considered part of the current program.
Returns:	A read operation cannot be used with this command.
Syntax:	%nWTMx – Standard syntax WSTx – Missing axis number, command accepted as standard syntax
Parameter Description:	n[int] – Axis number x[int] – Time
Parameter Range:	n – 1 to 99 x – 0 to 999999 milliseconds
Related Commands:	PGM
Example:	`%2WTM42` | Axis 2, Wait for 42 milliseconds before executing next command

ZRO

Zero Position

During Motion		Idle		Program		Global
Set	Read	Set	Read	Set	Read	Set
		✓		✓		✓

Command Description:	This command is used to set the absolute zero position for the specified axis.
Returns:	A read operation cannot be used with this command.
Syntax:	nZRO – Standard syntax Error [#]: ZRO – Missing axis number [123]
Parameter Description:	n[int] – Axis number
Parameter Range:	n – 1 to 99
Related Commands:	None
Example:	`1ZRO` | Axis 1, set current position as absolute zero

5.9 Error Messages

Error Number	Name	Description
10	Receive Buffer Overrun	The Receive Buffer has reached or exceeded maximum capacity.
11	Motor Disabled	The command that triggered this error was trying to move the servo while it was disabled.
12	No Encoder Detected	The command that triggered this error was trying to access encoder data when no encoder was attached.
13	Index Not Found	The controller moved across the full range of motion and did not find an index.
14	Home Requires Encoder	The HOM command requires an encoder signal.
15	Move Limit Requires Encoder	The MLN and MLP commands require an encoder signal.
20	Command is Read Only	The command that triggered this error only supports read operations. The command must be followed by a question mark to be accepted. Ex: XXX?
21	One Read Operation Per Line	Multiple read operations on the same command line. Only one read operation is allowed per line, even if addressed to separate axes.
22	Too Many Commands On Line	The maximum number of allowed commands per command line has been exceeded. No more than 8 commands are allowed on a single command line.
23	Line Character Limit Exceeded	The maximum number of characters per command line has been exceeded. Each line has an 80 character limit.
24	Missing Axis Number	The controller could not find an axis number or the beginning of an instruction. Check the beginning of the command for erroneous characters.
25	Malformed Command	The controller could not find a 3-letter instruction in the input. Check to ensure that each instruction in the line has exactly 3 letters referring to a command.

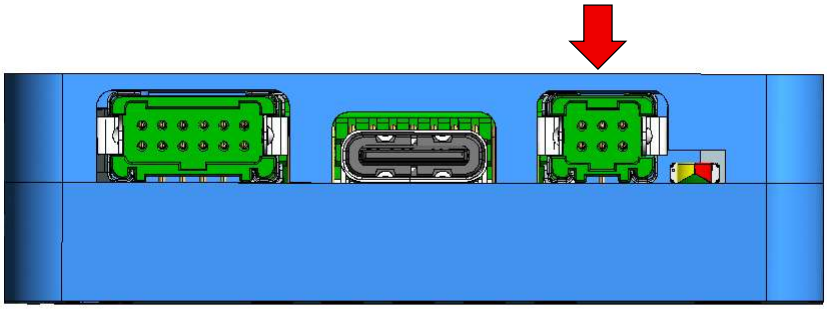
26	Invalid Command	The 3-letter instruction entered is not a valid command. Ensure that the 3-letter instruction is a recognizable command.
27	Global Read Operation Request	A read request for a command was entered without an axis number. A read request cannot be used in a global context.
28	Invalid Parameter Type	1. The parameter entered does not correspond to the type of number that the instruction requires. For example, the command may expect an integer value, therefore sending a floating point value will trigger this error. 2. The allowable precision for a parameter has been exceeded. For example, velocity can be specified with a precision of 0.001 mm/sec. If a more precise velocity value of 0.0001 mm/sec is entered, this error will be triggered. Refer to the command pages for the type of parameter that each command expects.
29	Invalid Character in Parameter	There is an alpha character in a parameter that should be a numeric character.
30	Command Cannot Be Used In Global Context	The command entered must be addressed to a specific axis number. Not all commands can be used in a global context. Check the specific command page or the table of commands for more info.
31	Parameter Out Of Bounds	The parameter is out of bounds. The current state of the controller will not allow this parameter to be used. Check the command page for more information.
32	Incorrect Jog Velocity Request	The jog velocity can only be changed during motion by using a new JOG command. If the VEL command is used to change the velocity, this error will be triggered. The VEL command can only be used to change velocity during motion initiated by the move commands [MVR, MVA, MSR, MSA].
33	Not In Jog Mode	Sending a JOG command during motion initiated by a move command will trigger this error. To initiate Jog Mode, the controller should be at stand-still. To change velocity during a move, use the VEL command.
34	Trace Already In Progress	This error is triggered when a new trace command is received after a trace is already in progress. Trace settings may be modified only if the trace hasn't started recording data. Otherwise, wait until the trace has finished before modifying the trace settings.
35	Trace Did Not Complete	An error occurred while recording trace data. Try the operation again.

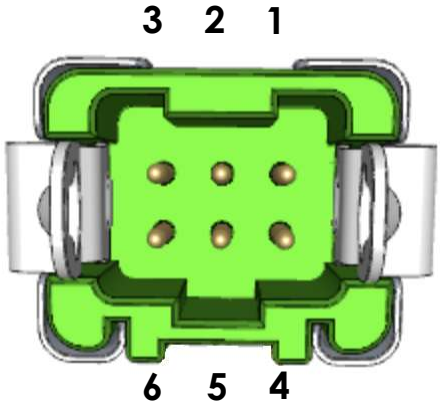
36	Command Cannot Be Executed During Motion	Only certain commands can be executed when motion is in progress. Check the command pages for information on individual commands.
37	Move Outside Soft Limits	If a requested move will take the controller outside of the preset travel limits, then the command will not be executed.
38	Read Not Available For This Command	This error is triggered by a read request from a command that does not support a read operation.
39	Program Number Out of Range	The number entered for the program number was either less than 1 or greater than 16.
40	Program Size Limit Exceeded	The program has exceeded the character limit of 4 Kb.
41	Program failed to Record	Error in recording program. Erase program and try operation again.
42	End Command Must Be on its Own Line	The End command used to end a program must be on a separate line from all other instructions.
43	Failed to Read Program	An error occurred while trying to read a program. Try the Operation again.
44	Command Only Valid Within Program	The command that triggered this error is only suitable for use within a program.
45	Program Already Exists	A program already exists for the indicated program parameter. The program must be erased with the ERA command before being written again.
46	Program Doesn't Exist	The indicated program does not exist. This error can occur when you try to execute a program number that has not had a program assigned to it.
47	Read Operations Not Allowed Inside Program	Read Operations are not permitted in programs.
48	Command Not Allowed While Program in Progress	The command that triggered this error was given while a program was executing.
50	Limit Activated	Motion in the direction of the activated limit switch is disallowed if limit switches are enabled.

51	End of Travel Limit	The requested move will take the controller outside of its valid travel range, therefore the move is disallowed.
52	Home In Progress	A Home or a Move To Limit Procedure is in progress. Motion commands are disallowed during this time. A STP or EST command can be used to terminate the Home, and then a motion command can be sent.
53	IO Function Already In Use	The I/O Function in question is already assigned to another I/O pin. Some Functions can only be assigned to one pin at a time. See the documentation for each function for more details.
55	Limits Are Not Configured Properly	Both Limit Switches are active, so motion is disallowed in both directions. Most likely the LPL (Limit Polarity command) setting should be switched.
80	Command Not Available in this Version	The command entered is not supported in this version of the firmware.
81	Analog Encoder Not Available In this Version	The current version of firmware installed does not support Analog Encoders.
-	No Error	Return value when the ERR? Is queried without an active error.

6. Appendix

6.1 Power Input Pin-out



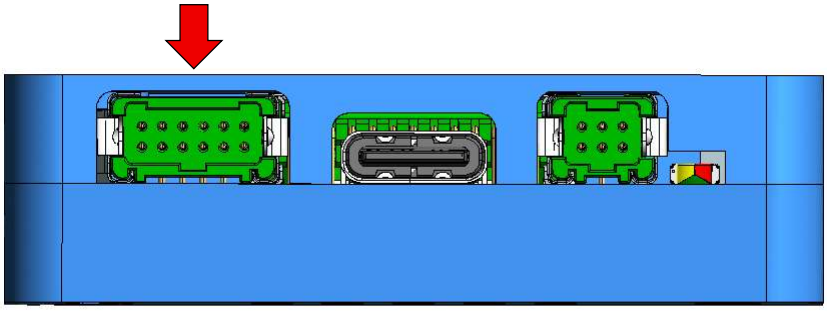


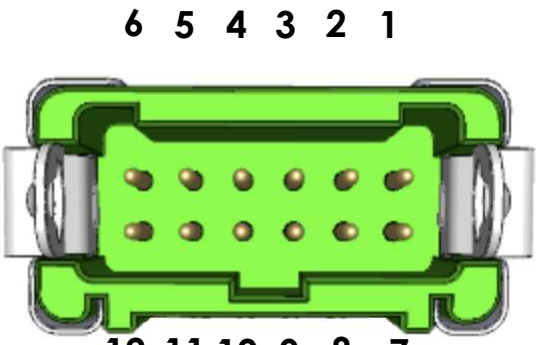
Description	Pin		Description
Power (+12 to 60VDC)	1	4	Ground
Power (+12 to 60VDC)	2	5	Ground
Power (+12 to 60VDC)	3	6	Ground

Mating Connector:
Manufacturer: Harwin
Part Number: G125-2040696L0
Contacts: G125-0010005 (for 26 AWG) or G125-0020005 (for 28-32 AWG)

Recommended Cable Assembly:
Harwin Gecko's G125-FC10605L0-xxxxL (xxxx is cable length in mm)

6.2 GPIO and Communication Pin-out



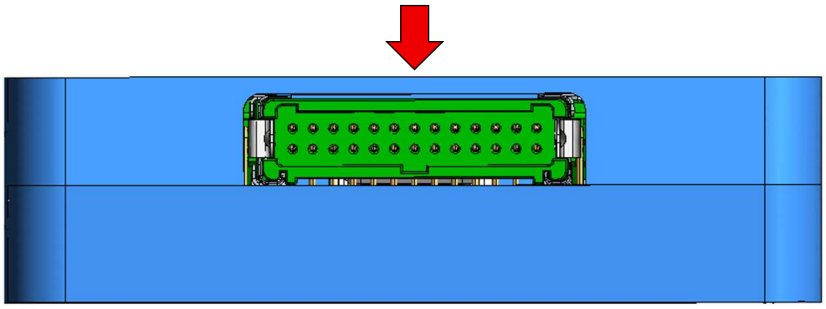


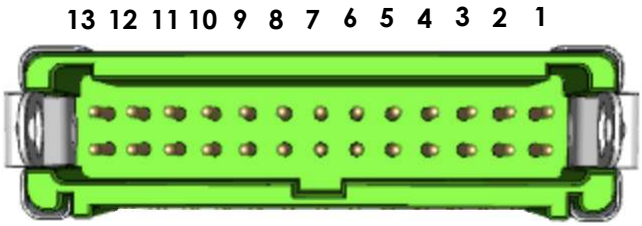
Description	Pin		Description
GPIO 1	1	7	GPIO 2
GPIO 3	2	8	GPIO 4
Ground	3	9	Ground
RS485A	4	10	RS485B
Reserved	5	11	Reserved
Address In	6	12	Address Out

Mating Connector:
Manufacturer: Harwin
Part Number: G125-2041296L0
Contacts: G125-0010005 (for 26 AWG) or G125-0020005 (for 28-32 AWG)

Recommended Cable Assembly:
Harwin Gecko's G125-FC11205LO-xxxxL (xxxx is cable length in mm)

6.3 Motor & Encoder Pin-out





Description	Pin		Description
Absolute SLI+	1	14	Absolute SLI-
Absolute MA+	2	15	Absolute MA-
Absolute SLO+	3	16	Absolute SLO-
Incremental Index +	4	17	Incremental Index-
Incremental B+/Sin+	5	18	Incremental B-/Sin-
Incremental A+/Cos+	6	19	Incremental A-/Cos-
+5VDC Output	7	20	Ground
Reserved	8	21	Reserved
Limit Positive	9	22	Limit Negative
Motor Phase 3	10	23	Motor Ground
Motor Phase 3	11	24	Motor Phase 4
Motor Phase 1	12	25	Motor Phase 2
Motor Phase 1	13	26	Motor Phase 2

Mating Connector:
Manufacturer: Harwin
Part Number: G125-2042696L0
Crimps: G125-0010005 (for 26 AWG) or G125-0020005 (for 28-32 AWG)

Recommended Cable Assembly:
Harwin Gecko's G125-FC12605LO-xxxxL (xxxx is cable length in mm)

6.4 Firmware

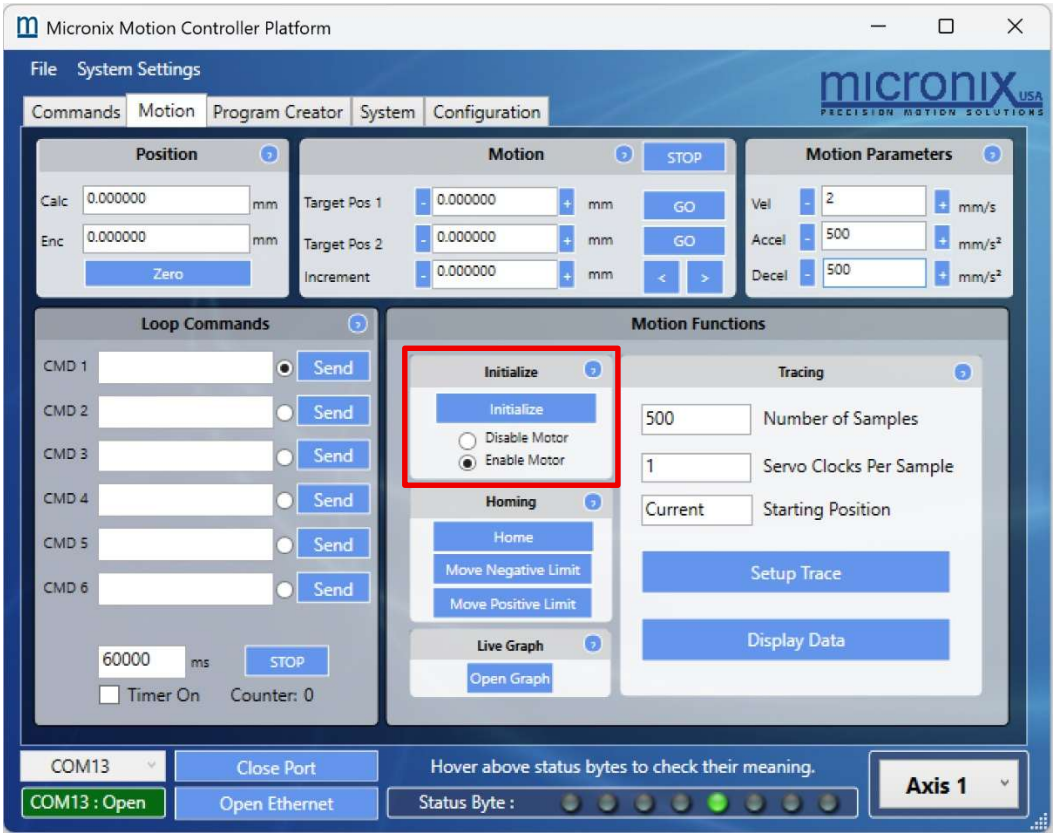
The firmware on the NanoDrive can be updated without returning the controller to manufacturer. Contact Micronix USA support for questions regarding the current firmware release and updating the firmware of your NanoDrive. Please be sure to check the firmware version by using the VER command.

6.5 3-Phase Brushless Motor Initialization

To operate correctly, 3-phase brushless motor requires an initialization sequence. The initialization routine can be started with the INI command and performs an automated motion to find the starting phase angle of the motor. This routine takes approximately 10 seconds to complete. Once the initialization routine is complete, the motor is automatically enabled and ready for motion.

Motors that use a digital incremental encoder will need to perform the phase initialization routine after every power cycle. If an absolute encoder is used, the initialization routine only needs to be done during the initial setup of the NanoDrive. The relationship between the encoder position and phase angle is preserved while the NanoDrive is powered off.

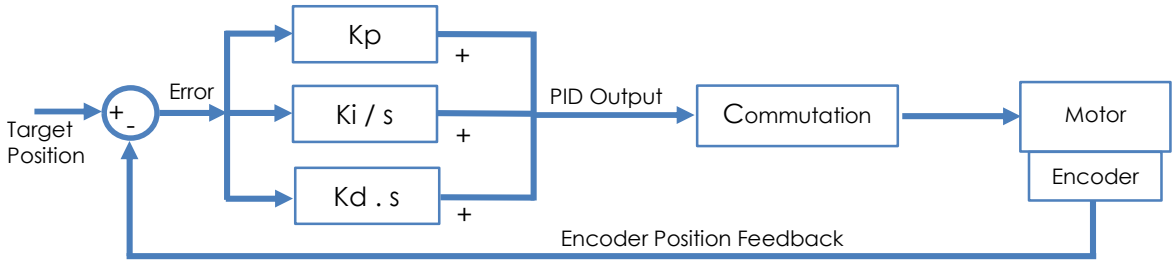
Initialization can be performed when using the Micronix Motion Controller Platform GUI. After connecting to the NanoDrive, an initialization button is on the Motion tab, as shown below.



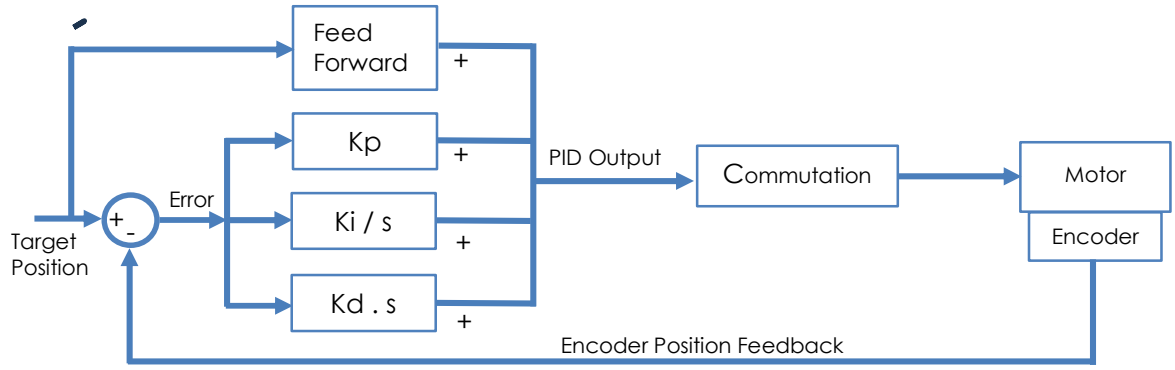
Stepper motors, piezo motors, and DC motors do not require the phase initialization routine.

6.6 Closed Loop Operation

The feedback loop for closed loop operation will vary depending on motor type. Below is the servo loop block diagram for 3-phase brushless motors (nMTC3) and DC motors (nMTC4).



Servo loop control for piezo motors (nMTC1) and stepper motors (nMTC2) will incorporate a feed forward parameter. Below is the servo loop block diagram.



In each servo cycle, the calculated target position and the encoder feedback position will be used to determine the error. The summation of the proportional, integral, and derivative terms will determine the error adjustment for the given servo cycle. A feed forward parameter will be included for stepper and piezo motors. The proportional (Kp), integral (Ki), and derivative (Kd) gains can be adjusted through the PID setting (PID command). The feed forward parameter can be set by the feed forward setting (FFP).

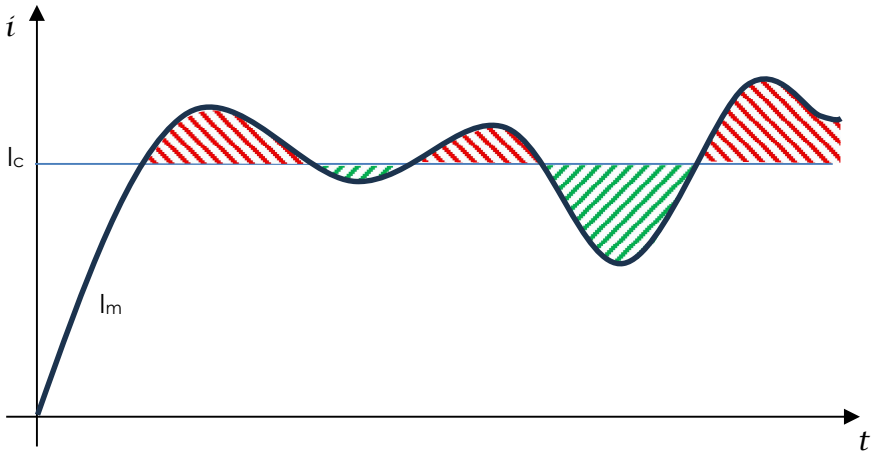
Feedback mode 3 is available for piezo, 3-phase brushless, and DC motors. Both 3-phase brushless and DC motors can only operate in feedback mode 3.

6.7 Motor Current Operation

Motor current settings will vary depending on the type of motor driven by the NanoDrive. Note that piezo motors do not use any motor current settings for operation.

For 3-phase brushless and brushed DC motors, motor current is defined by three parameters, the continuous current, surge current, and I^2T parameter. These parameters are defined using the MCS command. The continuous current is the nominal current of the motor. The surge current, or the acceleration current, is the maximum allowable motor current. Both the continuous and surge current parameters are defined in the motor datasheet.

The I^2T parameter is meant to protect the motor from overheating and defines the amount of time the NanoDrive can output motor current greater than the continuous motor current setting. The I^2T is calculated by the integration of the measured current, designated as I_m, and continuous current, designated as I_c, in the graph below.



Once the integration of the measured and continuous current exceeds the I^2T parameter defined by the MCS command, the NanoDrive will flag, error, and disable the motor.

For stepper motor operation, motor current is defined by two parameters, continuous current and I^2T. The continuous current and I^2T settings for the stepper motor have the same behavior as the 3-phase brushless motor current operation.